Scientific APL2 Computing: Bezier, BSPLINES, NURBS (Curves & Surfaces)

Jean-Claude Duguet

Pierre Bezier, graduated by the High Schools "ARTS et METIERS" and "Ecole Superieure D'electricite", also Doctor in Mathematics. He invented the Machines-Transfert, exported by Renault world-wide, revolutionizing the automation for the sequential tooling of motor-blocs etc.

As Director of Renault General Management, he became in 1965 the world pioneer for Computer Manufacturing Aided Design(CFAO) by the software UNISURF (FORTRAN), exported world-wide. He was also Vice-President of the "Societe des Ingenieurs Civils De France" and President of the mathematical Cercle Pierre De Jumieges, and died at the end of November, 1999. His idea was that "all objects should be represented by little numbers, an interactive method of mathematical computing permitting to create and draw 2/3D-Curves". Reminds their definition: geometric location of successive barycenters for a giving number of fixed poles, their sum of masses remaining equal to one. 3D-Surfaces are generated by moving 2D-Curves along the third dimension and parametric modified along the x and y axis in function of the z value and, if animation, of the time.

Usual BEZIER-Curves (2/3D)

The module B_DEMO computes, on lines 5/6, the binomial components and the parameter-values for the Curve then the matrix product of BERNSTEIN matrix with the matrix of Poles Coordinates. The iterator(þ), combined with nested arrays, speeds up the computation at a level analogous to compilers but they are useful against piracy. Line 7 computes polynomial adjusted ordinates; after 7, the successive first derivatives on the curve, by means of the first derivatives of the components from Bernstein matrix. The manner is similar by the successive derivatives The curves do not change if the References change. You can verify this in the algorithms which uses a double contracted (matrix) product. If you permute the transposed arguments, the transposed result is the same as the initial matrix-product. This property remains similar by the extended cases: BSPLINES and NURBS. Below module B_DEMO:

```
∇M←N B_DEMO P;B;n;t;u;v;w;⎕IO
[1] ⍝ M:Matrx(x y..)/Curve.
      P:Matrx Poles(X Y..)
[2] ⍝ N+1:Nbr of points computed on
      Curve. Deflt:N=10.
[3] ⍝ 2=≡P:PolynlAdjustdCrve/d∘n.
      Compute 1st derivatve
[4]  t←⌽0,(⍳N)÷N←'10' ⎕EA↑'N' ⎕IO←↑1
     u←1=≡P ⍝ Initialze
[5]  B←⊂(!n)÷(!v)×!w←⌽v←0,⍳n←¯1+↑ρ↑
     P M←(N+1),↑⌽ρP←⊃P
[6]  →u/0 M←Mρ⍎2⍕,(⊃B←⌽B×¨(t*⊂v)×
     (1-t)*⊂w)+.×P ⍝ Polynl
[7]  M←M,⍎2⍕,(M[;2]⌹M[;1]∘.*v)+.×
     ⊃[1]M[;1]*¨⊂v ⍝ Adjust
[8]  u←100,↑1 B←⊃(1↓¯1↓B)×¨⌽¨((⊂v)÷
     ¨t)-(⊂w)÷¨1-t←1↓¯1↓t
[9]  v←B∧.=↑0 B←(((n+1)↑n,-n),[1]B,
     [1](-n+1)↑n,-n)+.×⌽P
[10] v←1∊ρ↑t u←(⊂'÷/'),¨⍕¨⊂[2]↑B
     B[t;]←(⌽2,ρt←v/⍳N+1)ρu
[11] ⍎v/↑'u[t]←⊂(×u[t-1])×100 ¯100'
     u←⍎2⍕(⊂'100')⎕EA¨u
[12] M←M,↑u u[v]←100×-×B[v←(100≡¨u)/
     ⍳ρu;1]  ⍝ (c) α⎕ω
   ∇
```

Ex:4 Poles, polynomial adjusted, plus 1st derivatives. CPU-Time: 0 milliseconds

```
⍉B_DEMO⊂4 2ρ0 0 1 1 0 1 1 0
```

⍝ Turn-Point on the Middle

```
0 0.24 0.39 0.47  0.5 0.5 0.5 0.53
0.61  0.76 1
0 0.27 0.48 0.63 0.72 0.75 0.72 0.63
0.48  0.27 0
¯0.05 0.43 0.58 0.61 0.61 0.61  0.61
0.61  0.58  0.43 ¯0.05
1 1.25 1.67 2.5 5 100 ¯100  ¯5 ¯2.5 ¯
1.67 ¯1.25 ¯1
```

In practical cases, use below module BEZIER for composite Curves:

```
  ∇  M←N BEZIER P;i;n;t;⎕IO
[1] ⍝ M:Mtrx points-coord P:Mtrxs
      Poles-coord(X Y..)(..)
[2] ⍝ N+1:Nbr of points on each Arc.
      Deflt:N=10  (c) α⎕ω
[3] i←0,⍳n←¯1+↑⍴↑P←⍎(1<≡P)↓↑'⊂P' t←0,
      (⍳N)÷N←'10' ⎕EA↑'N' ⎕IO←1
[4] M←⍎¨2⍕¨⊃⍉¨,/⍉¨(⊂⊃(⊂(!n)÷(!i)×!⌽i)
      ×¨(t*⊂i)×(⌽t)*⊂⌽i)+.×¨P
∇
```

N.B. If 3 Poles, the Curve applies efficiently to a parabolic Arc. If aligned Poles, it applies to a straight-line Segment

BSPLIN-Curves (Based Splines).

SPLINES, "lattes" in French, relate to an old method used for modeling the prow of ships etc. In the present case, the word has no mnemonic connotation. The module BSPLIN below, deducted by BEZIER, uses, for 3 Poles, the same middle Pole and the other two symmetric from the middle related to the two other Poles, computes the Riesenfeld Matrix in place of Bernstein, that applies efficiently to arcs of any order, for ex. to cubic arcs by 4 Poles. The junctions of adjacent arcs have identical first derivatives, from where their name "uniform curves".

```
  ∇ M←N BSPLIN P;i;n;t;⎕IO
[1] ⍝ M:Coord(x y..)/Crve. Mtrxs
      Poles-coord(X Y..)(..)
[2] ⍝ N+1:Nbr of points on each Arc.
      Dflt:N=10  (c) α⎕ω
[3] t←0,(⍳N)÷N←'10' ⎕EA↑'N' P←⍎
      ((⎕IO←1)<≡P)↓'⊂P'
[4] M←(n+1)×(t+¨⊂(⌽i)-N←(-i)↓¨
      ⊂i←0,⍳n)*n←¯1+↑⍴↑P
[5] M←⍎¨2⍕¨⊃⍉¨,/⍉¨(⊂+/¨⊃M×⊂(¯1*¨N)÷
      (!¨N)×!¨n+1-N)+.×¨P
   ∇
```

Ex: Identity BEZIER and BSPLIN by corresponding 3 Poles.

```
(BEZIER 3 2⍴0 0 1 1 2 0)≡BSPLIN 3 2⍴¯
1 ¯1 1 1 3 ¯1
   1
```

XNURBS And Engine Extended NURBS

Computing All Curves and Surfaces

NURBS is the acronym of non uniform rational BSPLINES. The poles are those of BEZIER or BSPLIN, the curves of which are specified by vector parameters not all null. If all 1, the results are those of BEZIER/BSPLIN. To distinguish that set argument N decimal by BSPLIN. You can change the Referential. For ex, you move the origin onto X & Y by the statement: X Y+[2]M. To rotate axis applies the rotated tensor onto íM. This software engine accepts all cases, usable by all APL2-implementations. You display that using AP207 with the Point-Co-ord. which are generally portable. The self-booting diskette UNISOFT displays any cases. Module XNURBS below computes this at extreme high speed by iterator(þ)

```
  ∇ M←N XNURBS P;i;n;p;t;⎕IO
[1] ⍝ M:Mtx points(x y z) P:Mtx
      Poles(X,pX)(Y,pY)(Z,pZ)
[2] ⍝ with ponderators pX.. N:intgr
      BEZIER;decml/BSPLIN
[3] ⍝ 1+⌊N:Nbr of points on the
      Curve. Dflt:N=10/BEZIER
[4]  N←N≠⌊↑N P←2↑¨P,¨~↑t←0,(⍳⌊N)÷⌊
       N←'10' ⎕EA↑'|↑N' ⎕IO←1
[5]  →N/↑BSP i←0,⍳n←¯1+↑⍴P←×/¨↑P
       P[;2]←↑(n p)←⊂[1]⊃P[;2]
[6] BEZ:→↑END M←(⊂(!n)÷(!i)×!⌽i)×¨
       (t*⊂i)×(⌽t)*⊂⌽i
[7] BSP:M←((t+⊂(⌽i)-N)*n)×⊂(¯1*N)÷
       (!N)×!n+1-N←(-i)↓¨⊂i
[8]  M←(n+1)×⊂[2]+/¨⊃M ⍝ BSP:Compute
       RIESENFELD-Mtx
[9] END:M←⊃⍎¨⊂[2]2⍕(⊃M÷+/¨M←(⊂p)×¨M)
       +.×P ⍝ (c) α⎕ω
    ∇
```

Statement 6 are the immediate APL2-transcriptn of algorithm computing items of binomial low "(t+(1-t))*n" usually called Bernstein-Polynomial. Same for statements 7/8, computing the Riesenfeld-polynomial (BSPLINES).

By composite Crvs set "⊃⍉¨,/⍉¨N XNURBS¨P", where P= Vector Matrix-Poles

Example: Identity Bezier and XNURBS if specifications are all = 1.

```
(BEZIER 3 2ρ1 0 2 1 3 0)≡XNURBS 3 2ρ1
 0 2 1 3 0,¨1
   1
```

Briefly, this paper gives one overview for APL2-Prototyping of BEZIER Curves & Surfaces. They formula are familiar to Mathematicians, which must convert that in APL-Notation, computes himself, looks at the result, optimizes at low cost, in a short time, proves the feasibility of this approach. Remember here the merit of our friend Pagnon who had beyond 1986 programmed any Bezier curves, using the VSAPL Release, not yet nested. Note also the conference Tollet-Harrison/APL-1992 "Bezier curves built with straight-lines segments implemented in APL2 for Corel-Draw".

The book "Courbes BEZIER-BSPLINES-NURBS" (Editor ELLIPSES/1998/PARIS) from MM. Demengel and Pouget provides the algorithms for computing and concludes: This mathematics book discloses the black box of the mathematics embedded in CAD software. APL2 goes 1 step ahead by providing a suitable framework and low-cost prototyping, "without relying on black box software". The future is here based on programmable mathematics and APL2 is an acclaimed tool in this domain.

BIBLIOGRAPHY

1. APL2 At A Glance (J. Brown, S. Pakin, R. Polivka) Prentice-Hall/1988

2. APL2 Programming Language Reference Manual, IBM Corporation

3. Conference Of Pierre Bezier - Cercle Pierre De Jumieges - June 1998

4. Les Courbes De Bezier / Bsplines / NURBS Ellipses/Paris 1998

Two Views of The 2001 APL2000 User Conference

David Kehoe

If you are an APL programmer, or even an APL wannabe, you're crazy if you don't visit this exciting village APL2000 creates at its annual User Conference each fall. How often can you get together with rooms full of people that know that "Quadwee" is not some obscure region in Asia? Or have an in-depth conversation about nested arrays while picking up seashells on the beach?

My company, ComSi, has been doing custom software development for small businesses for over 20 years. We have yet to find a platform for rapid and flexible development that even comes close to APL, specifically APL+WIN. However, we have always struggled with the fact that APL is not well known (which we ALL trying to change), that creates problems selling our services and limits our access to assistance and resources that a typical VB, C or HTML programmer has. Hence, the APL2000 Conference is the ultimate for folks like us; much like a school reunion, we go to this village for 4 days of intense sharing, learning, communicating, supporting, and socializing. I always come away from it with new ideas, friends, code, business prospects, and excited confidence that Cognos and APL2000 keep moving the ball forward to make APL+WIN a strong and viable platform.

This year, the village was built on the beach in Naples, Florida…the negative forces of September 11th and the threat of a hurricane were indeed present, but not consuming. When Eric Baelen (APL2000) is present, there's always a positive excitement in the air. As usual, the content presented at the Conference was nearly overwhelming…something for everyone. I always spend the next few months after the conference reviewing the proceedings, trying out the code snippets, and integrating the new features into our applications. The main tracks this year were Internet integration, database access, ways to use APL with other applications, tricks for APL developers, and of course, new features in APL+WIN. The 'breakout session' format allows you to tailor the conference to topics you find most pertinent or interesting.