Hermite approximation for offset curve computation

Victor Ostromoukhov
Peripheral Systems Laboratory
Swiss Federal Institute of Technology
IN-F Ecublens, CH-1015 Lausanne, Switzerland

Abstract. The present paper proposes a new method for calculating the G^1-continuous offset curve to a cubic Bézier curve, based on the Hermite approximation technique. This method is improved by preliminary curvature estimation and is intended for use in cpu-time sensitive CAGD applications.

Keywords: Bézier curve, Hermite approximation, offset curve, maximum curvature, computer-aided geometric design (CAGD).

1. Introduction

Cubic Bézier curves are widely used in CAGD applications such as page-description languages (PDLs) and font design tools. Very often a standard set of primitives for such systems consists of points, straight line segments, and cubic Bézier curves. All other types of element (e.g. circular segments) are expressed in terms of basic primitives. This is the case in the PostScript page-description language [Adobe '85], the Metafont font generation system [Knuth '79] and the RastWare font manipulation system [Hersch '87]. Scan-conversion of a shape outlined by a set of standard basic elements has been described in detail (see [Rogers '85], [Hersch '88]). As regards systems which can draw with a finite pen, the situation is a little more complicated: it is a well-known fact that, in general, the offset curve to a cubic Bézier curve is not a cubic Bézier itself. Some systems introduce a simple shortcut: they do not calculate the outline shape of the offset curve analytically; instead, they simply fill a pen shape moving along a *generator* (we use the term *generator* to indicate the trajectory of the center of a circular pen, see [Knuth '86], [Hobby '89]). This approach is not suitable in applications where strokes may intersect clipping shapes.

The simplest analytical solution for finding the offset curve is a linear approximation: a generator is subdivided into a set of small straight lines; then two parallel lines are found at distance $\pm d$ from them. Although it is relatively simple, this approach generates an offset curve with a cumbersome large set of segments, especially at high resolution.

It is natural to look for an approximation to the offset curve in the same class of functions as the generator. This approach is developed in [Hoschek '85]. The least-squares method with iterative reparametrisation used in [Hoschek '88] is very precise, but very time-consuming. In the present paper, we propose in section 1 another alternative for finding the analytic offset curve based on the Hermite approximation technique. It is a fast method, but it only works correctly on regular smooth curves which have convex control polygons with non-parallel lateral edges. Consequently, a preliminary analysis of the generator is needed.

In section 2, we describe a method for preliminary maximum curvature estimation, in order to locate and eliminate from our approximations all cusp and return points of the offset curve. Such non-smooth subsegments are computed by a linear approximation.

2. Hermite approximation for offset curves

In the general case, a basic parametric curve of degree 3 has an arbitrary control polygon leading to 4 main classes of curves: arch (regular), cusp, 1-inflection-point curve and 2-inflection-point curve [De Rose, Stone '89].

Let us consider a simplified task: finding the offset curve of a regular parametric curve which have a convex control polygon.

A generator curve $\mathbf{G}(t)$ with a control polygon having control vertices at $\mathbf{V}_0, \mathbf{V}_1, \mathbf{V}_2$ and $\mathbf{V}_3$, is given by

$$\mathbf{G}(u) = \sum_{i=0}^{3} B_i^3(u)\mathbf{V}_i$$

where $B_i^3(u)$ are the Bernshteĭn polynomials of degree 3.

The corresponding offset curve $\mathbf{G_d}(u)$ can be expressed as

$$\mathbf{G_d}(u) = \mathbf{G}(u) + d\mathbf{n}(u)$$

where d is a diameter of the pen and $\mathbf{n}(u)$ is a unit vector normal to curve $\mathbf{G}(u)$ at point t.

We call $\mathbf{Q}(t)$ a cubic Bézier approximation to the $\mathbf{G_d}(u)$ curve. We will look for $\mathbf{Q}(t)$ in the G^1-continuous class:

$$\mathbf{Q}'(t) = \lambda\mathbf{G}'(u), \qquad t = 0, 1 \quad u = 0, 1. \tag{1.1}$$

It is natural to expect $\mathbf{Q}(t)$ to be as close to $\mathbf{G_d}(u)$ as possible. The method described in [Hoschek '89], using a least-squares method, is rather precise, but very cpu-expensive because it needs to reparametrize the obtained curve iteratively to estimate an error.

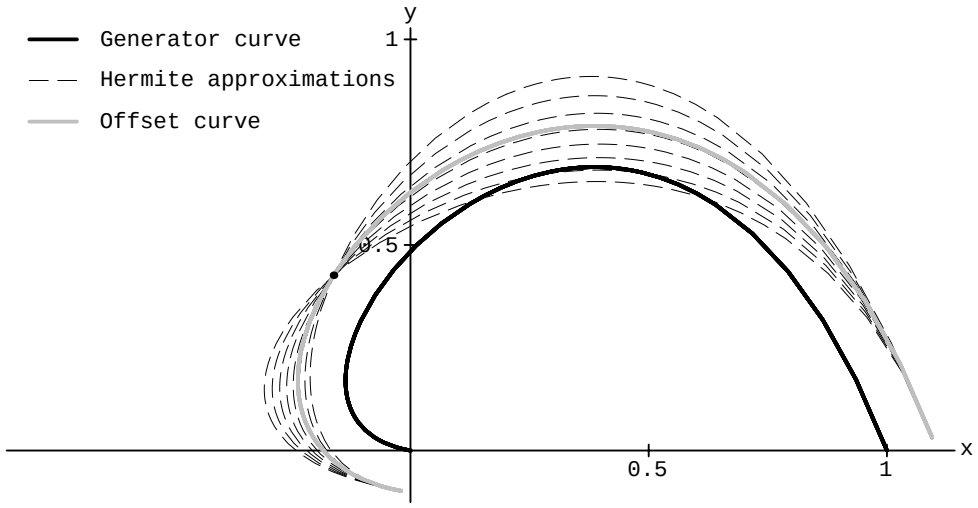


Fig. 1 *A family of Hermite approximations passing through* $\mathbf{G}_{\mathbf{d}}(\frac{1}{3})$.

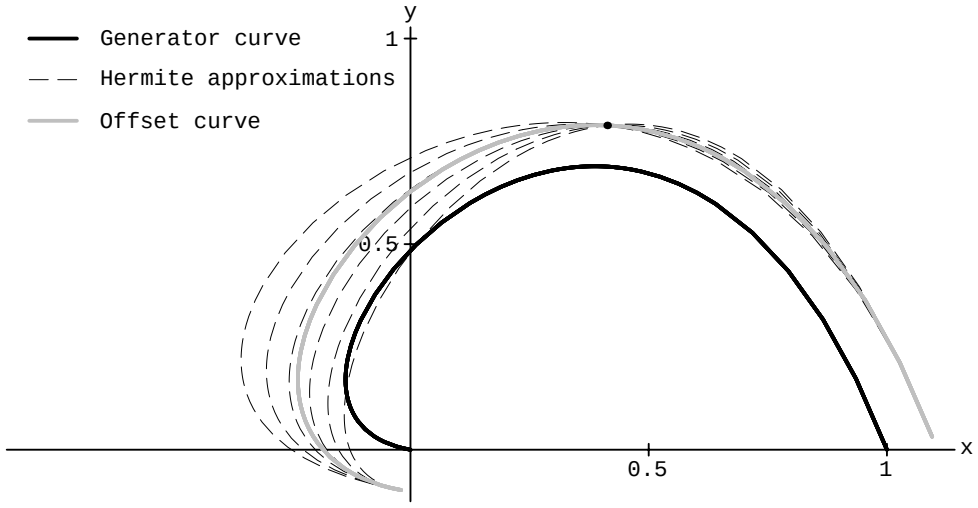


Fig. 2 *A family of Hermite approximations passing through* $\mathbf{G}_{\mathbf{d}}(\frac{2}{3})$.

The main idea of the present method is to use a Hermite approximation, i.e. to find a cubic approximation curve $\mathbf{Q}(t)$ such that its derivatives $\mathbf{Q}'(0)$ and $\mathbf{Q}'(1)$ are parallel to the derivatives of the generator, $\mathbf{G}'(0)$ and $\mathbf{G}'(1)$ respectively. By choosing the appropriate absolute values of $|\mathbf{Q}'(0)|$ and $|\mathbf{Q}'(1)|$, $\mathbf{Q}(t)$ can be forced to pass through 2 points of the exact offset curve $\mathbf{G_d}(u)$:

$$\mathbf{Q_H}(t_1^*) = \mathbf{G_d}(u_1) \tag{1.2}$$

$$\mathbf{Q_H}(t_2^*) = \mathbf{G_d}(u_2)$$

where $\mathbf{Q_H}(t)$ is the Hermite representation of $\mathbf{Q}(t)$ (see [Farin '90, p.75]):

$$\mathbf{Q_H}(t) = \mathbf{p}_0 H_0^3(t) + \mathbf{m}_0 H_1^3(t) + \mathbf{m}_1 H_2^3(t) + \mathbf{p}_1 H_3^3(t) \tag{1.3}$$

where $\mathbf{p}_0 = \mathbf{G_d}(0)$, $\mathbf{p}_1 = \mathbf{G_d}(1)$, $\mathbf{m}_0 = \mathbf{G'_d}(0)$, $\mathbf{m}_1 = \mathbf{G'_d}(1)$.

In general, $t_1^* \neq u_1$, and $t_2^* \neq u_2$. For u_1 and u_2, it is possible to choose any values between 0 and 1, e.g., $u_1 = 1/3, u_2 = 2/3$.

From (1.1) and (1.3), we can reformulate the problem as follows:

$$\mathbf{p}_0 H_0^3(t_1^*) + \lambda_1 \mathbf{G}'(0) H_1^3(t_1^*) + \lambda_2 \mathbf{G}'(1) H_2^3(t_1^*) + \mathbf{p}_1 H_3^3(t_1^*) = \mathbf{G_d}(1/3) \tag{1.4a}$$

$$\mathbf{p}_0 H_0^3(t_2^*) + \lambda_1 \mathbf{G}'(0) H_1^3(t_2^*) + \lambda_2 \mathbf{G}'(1) H_2^3(t_2^*) + \mathbf{p}_1 H_3^3(t_2^*) = \mathbf{G_d}(2/3) \tag{1.4b}$$

It is a (non-linear) system of 4 equations with 4 unknowns: $\lambda_1, \lambda_2, t_1^*$ and t_2^*.

Let us try to develop an algorithm to solve (1.4) based on simple geometric considerations. First, we take only part (1.4a) of system (1.4). If we fix t_1, then all $H_i^3(t_1)$ are known and (1.4a) is a linear system of 2 equations with 2 unknowns λ_1 and λ_2. As mentioned above, λ_1 and λ_2 determine the absolute value of $|\mathbf{Q}'(0)|$ and $|\mathbf{Q}'(1)|$, and, therefore, the Bézier control polygon for $\mathbf{Q}(t)$ becomes defined. The solution of (1.4a) with fixed t_1 gives a cubic curve, passing exactly through point $\mathbf{G_d}(1/3)$. Fixing t_1 to another value, gives another cubic curve passing through point $\mathbf{G_d}(1/3)$. A family of such curves is presented in Fig. 1.

Similar considerations are applicable to (1.4b) using a set of t_2: see Fig. 2. It is clear that from these 2 families of cubic curves we can select 2 curves with t_1^* and t_2^* satisfying (1.2).

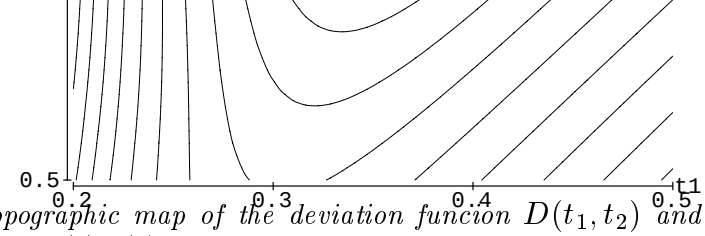


Fig. 3 *Topographic map of the deviation funcion $D(t_1, t_2)$ and a typical path from the starting point $(t_1^{(0)}, t_2^{(0)})$ to the solution (t_1^*, t_2^*).*

Therefore, the algorithm will operate separately on $(1.4a)$ and $(1.4b)$ performing alternately "left-step" around point $\mathbf{G_d}(1/3)$ and "right-step" around point $\mathbf{G_d}(2/3)$. At the beginning, during the initial set-up, t_1 is assigned to $1/3$ and t_2 to $2/3$. At each step, new values of t_1 and t_2 are sought, thus minimizing the deviation function $D(t_1, t_2)$ defined as

$$D(t_1, t_2) = D_L(t_1, t_2) + D_R(t_1, t_2),$$

where $D_L(t_1, t_2)$ is a distance between points $\mathbf{G_d}(2/3)$ and $\mathbf{Q_H}(t_2)$ during "left-step" with fixed t_1, and $D_R(t_1, t_2)$ is a distance between points $\mathbf{G_d}(1/3)$ and $\mathbf{Q_H}(t_1)$ during "right-step" with fixed t_2.

Fig. 3 shows a typical topographic map of such a deviation function and a typical path from $(t_1^{(0)}, t_2^{(0)})$ to (t_1^*, t_2^*). The solution will converge to (t_1^*, t_2^*) when starting not far from the minimum point (the choice of $(1/3, 2/3)$ as a starting point seems reasonable). Our method is based on hypothesis that the function $D_L(t_1, t_2)$ is linear on t_1 and $D_R(t_1, t_2)$ is linear on t_2, which is not always true; the degree of linearity of these functions will determine convergence speed. A detailed description of the algorithm is given in appendix A. Fig. 4 gives an illustration of the iterative Hermite approximation process.

As mentioned above, this method gives us a G^1-approximation passing through 2 points $\mathbf{G_d}(1/3)$ and $\mathbf{G_d}(2/3)$; its behaviour is not guaranteed in the middle segment of the 3 segments $(\mathbf{G_d}(0), \mathbf{G_d}(1/3))$, $(\mathbf{G_d}(1/3), \mathbf{G_d}(2/3))$ and $(\mathbf{G_d}(2/3), \mathbf{G_d}(1))$. We use the sum of the distances between $\mathbf{G_d}(t)$ and $\mathbf{Q_H}(t)$ (only their normal component) at the 3 midpoints to estimate the error in the approximation process. If the error is greater than the admitted tolerance, generator curve $\mathbf{G}(t)$ must be subdivided and our method applied to both subsegments.

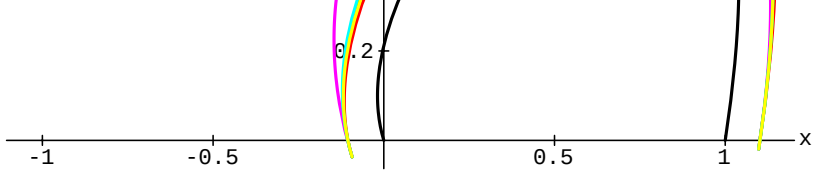


Fig. 4 *Successive Hermite approximations.*

3. Curvature analysis

Farouki has shown in [Farouki '89] that the offset curve of a regular plane curve of degree 3 has the same number of special points as a generator curve *plus* all the points corresponding to the points which have curvature $\kappa = -1/d$ on the generator, where d is the diameter of the pen. These latter points will produce cusps and return points on the offset curve (see Fig. 5).

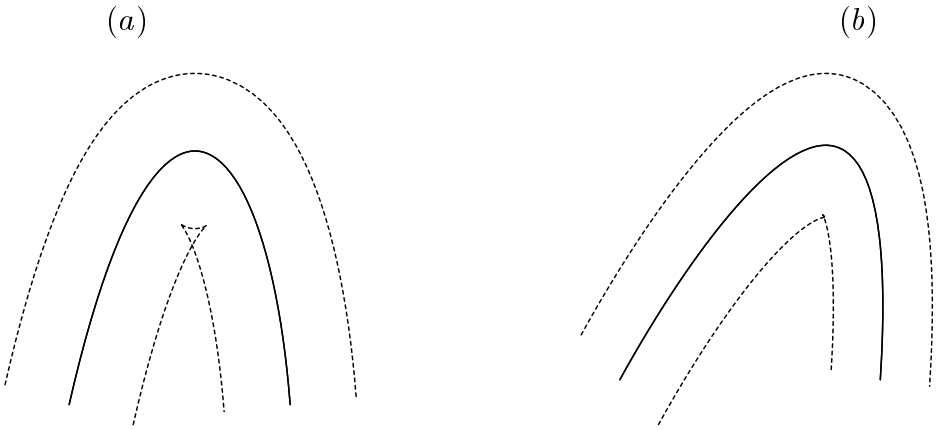


Fig. 5 *Offset curves having 2 return points (a) and a cusp (b).*

Our approximation algorithm of an offset curve by Bézier curves works only on rather regular, *smooth* segments. Consequently, we have to exclude from our approximation process all non-smooth points corresponding to points which have curvature $\kappa = -1/d$. One possible solution is to subdivide the generator curve $\mathbf{Q}(t)$ *exactly* at points where $\kappa = -1/d$. Unfortunately, it is not possible to solve the following equation analytically

$$\kappa(\mathbf{Q}(t)) = \frac{\dot{\mathbf{Q}}(t) \times \ddot{\mathbf{Q}}(t)}{|\dot{\mathbf{Q}}(t)|^3} = -\frac{1}{d} \tag{2.1}$$

for a cubic Bézier curve $\mathbf{Q}(t)$.

Standard numerical methods for finding roots are particularly complicated in this case, because as shown in Fig. 6, the generator curve $\mathbf{G}(t)$ can have 0, 1, 2, 3 or 4 points t^* such that $\kappa(\mathbf{G}(t^*)) = -1/d$. Although we tried to estimate at least the number of roots from the Bézier control polygon, we did not find any straightforward relationships between the shape of the control polygon and the number of roots.

For practical purposes, we have adopted a mixed technique: all smooth segments having curvature $|\kappa| < 1/d$ are approximated by the technique described in the

previous chapter; all segments whose maximum curvature exceeds the value of $|\kappa| = 1/d$, are separated into smooth and non-smooth parts ($|\kappa| \ll \frac{1}{d}$ and $|\kappa| \gtrsim \frac{1}{d}$), and the approximation is carried out separately. The smooth subsegments are approximated just like smooth segments; the non-smooth subsegments are approximated linearly.

4. Conclusions

The method for offset curve computation introduced here can be used efficiently in applications where it is desirable to express strokes and offset curves using the same class of curves as the generator. Preliminary maximum curvature estimation is required in order to avoid incorrect results in special cases when cusps and return points appear on an offset curve. In such cases, the offset curve must be described by a combination of Bézier curves and polysegments. Such a description provides a reasonable trade-off between quality and efficiency.

References

1. Adobe Systems Inc. *PostScript language reference manual.* Addison-Wesley, 1985.
2. G. Farin. *Curves and surfaces for computer aided geometric design.* Second edition. Academic Press, 1990.
3. R.T. Farouki. Hierarchical segmentation of algebaic curves and some applications. *Mathematical methods in CAGD,* Academic press, pp. 239-248, 1989.
4. R.D. Hersch. Character generation under grid constraints. *SIGGRAPH'87, ACM Computer Graphics,* 21(4), 1987.
5. R.D. Hersch. Vertical scan-conversion for filling purposes. *Proceedings CGI'88,* Springer Verlag, pp. 318-327, 1988.
6. J.D. Hobby. Rasterizing curved lines of constant width. *J. ACM,* 36(2), 1989.
7. J. Hoschek. Offset curves in the plane. *Computer Aided Design,* 17(2), 1985.
8. J. Hoschek. Spline approximation of offset curves. *Computer Aided Geometric Design,* 5(1), 1988.
9. D.E. Knuth. *TEX and Metafont. New directions in typesetting.* AMS and Digital Press, 1979.
10. D.E. Knuth. *Computers and typesetting. Vol. C, D.* Addison-Wesley, 1986.
11. D. Rogers. *Procedural elements for computer graphics.* McGraw-Hill, 1985.

12. M.C. Stone, T.D. DeRose. A geometric characterization of parametric cubic curves. *ACM Transactions on graphics,* 8(3), 1989.

Appendix A. Algorithm Find_Hermite_Offset.

Globals: $t_1^{prev}, t_1^{current}, t_2^{prev}, t_2^{current}$.

Initial setting: $t_1^{prev} = \frac{1}{3}; t_1^{current} = \frac{1}{3} + \delta; t_2^{prev} = \frac{2}{3}; t_2^{current} = \frac{2}{3} + \delta;$

where δ is a small number (for example, $\frac{1}{20}$).

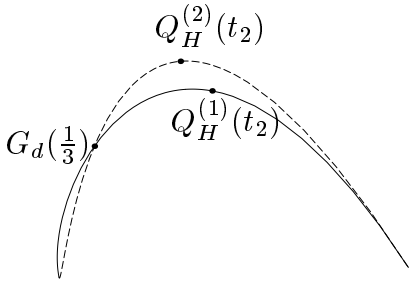
Procedure Left_step

Given: $t_1^{(1)} = t_1^{prev}, t_1^{(2)} = t_1^{current}, t_2 = t_2^{current}, G_d(\frac{1}{3}), G_d(\frac{2}{3})$

• Find $Q_H^{(1)}(t)$ and $Q_H^{(2)}(t)$: solve equations (1.2)

$$Q_H^{(1)}(t_1^{(1)}) = G_d(\frac{1}{3})$$

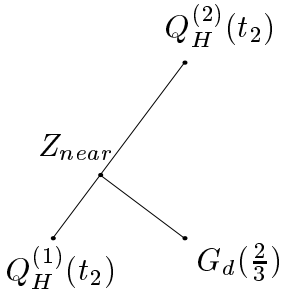
$$Q_H^{(2)}(t_1^{(2)}) = G_d(\frac{1}{3})$$



• Find the point Z_{near} on straight line $(Q_H^{(1)}(t_2), Q_H^{(2)}(t_2))$, nearest to $G_d(\frac{1}{3})$.

• Find the next approximation for t_1: t_1' (supposition of linearity):

$$\frac{t_1' - t_1^{(1)}}{t_1^{(2)} - t_1^{(1)}} = \frac{|Z_{near} - Q_H^{(1)}(t_2)|}{|Q_H^{(2)}(t_2) - Q_H^{(1)}(t_2)|}$$



• Re-assign the variables $t_1^{current}, t_1^{prev}$:

$$t_1^{current} = t_1'; t_1^{prev} = t_1^{(2)}$$

• Estimation of error during left_step:

$$Err\_left\_step = |t_1^{current} - t_1^{prev}|$$

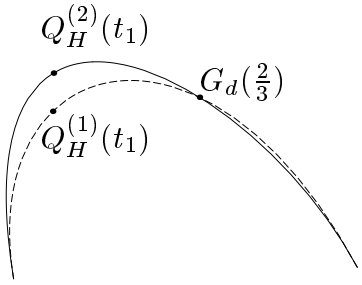
Procedure Right_step

Given: $t_2^{(1)} = t_2^{prev}, t_2^{(2)} = t_2^{current}, t_1 = t_1^{current}, G_d(\frac{1}{3}), G_d(\frac{2}{3})$

- Find $Q_H^{(1)}(t)$ and $Q_H^{(2)}(t)$: solve equations (1.2)

$$Q_H^{(1)}(t_2^{(1)}) = G_d(\frac{2}{3})$$

$$Q_H^{(2)}(t_2^{(2)}) = G_d(\frac{2}{3})$$



- Find the point Z_{near} on straight line $(Q_H^{(1)}(t_2), Q_H^{(2)}(t_2))$, nearest to $G_d(\frac{1}{3})$.
- Find the next approximation for t_2: t_2' (supposition of linearity):

$$\frac{t_2' - t_2^{(1)}}{t_2^{(2)} - t_2^{(1)}} = \frac{|Z_{near} - Q_H^{(1)}(t_1)|}{|Q_H^{(2)}(t_1) - Q_H^{(1)}(t_1)|}$$

- Re-assign the variables $t_2^{current}, t_2^{prev}$:

$$t_2^{current} = t_2'; t_2^{prev} = t_2^{(2)}$$

- Estimation of error during right_step:

$$Err\_right\_step = |t_2^{current} - t_2^{prev}|$$

Procedure Calc_Error

- Error = OrtoDist$(\frac{1}{6}, \frac{t_1^{cur}}{2})$+ OrtoDist$(\frac{1}{2}, \frac{t_1^{cur}+t_2^{cur}}{2})$+ OrtoDist$(\frac{5}{6}, 1 - \frac{1-t_2^{cur}}{2})$,

where OrtoDist(u, v) is the component of the vector $(Q_H(v) - G_d(u))$, perpendicular to the vector $G(u)$

Procedure Find_Hermite_Offset.

- Initial setting
- Iterations

  DO

    Left_step

    Right_step

  WHILE (Err_left_step + Err_right_step ¿ t_tolerance)
- Calc_Error
- IF (Error ¿ Q_tolerance) THEN

  Subdivide_And_Find_Hermite_Offset.