CAM Part II: Intersections

In the previous part, we looked at the computations of derivatives of B-Splines and NURBS. The first derivatives are of interest, since they are used in computing the tangents (and thereby, the normals), to generate offset points for tool path generation. The second derivative gives curvature values, that are related to the tool sizes that can cut the surface without gouging.

Up till now, we have been able to derive most entities of interest using exact analytical forms. However, the topic of Surface-Surface-Intersection (SSI) and its special cases all require approximation solutions, and we shall heavily rely upon numerical techniques for solving equations – for example the Newton-Raphson scheme. Why numerical solutions ? Since we would like to develop a uniform algorithm that can handle all the surface-types that may occur in a design. We can easily solve a conic-plane intersection analytically; however, the intersecting of two parametric polynomial vector functions is not easy, in general – especially since the output may be a point, a 3D curve, or even a 3D surface; in pathological cases, it may be an arbitrary combination of all these types. But we shall ignore strange cases, and concentrate of the general case, where the two surfaces are well behaved, and their intersection, if it is not null, is one or more bounded 3D curves.

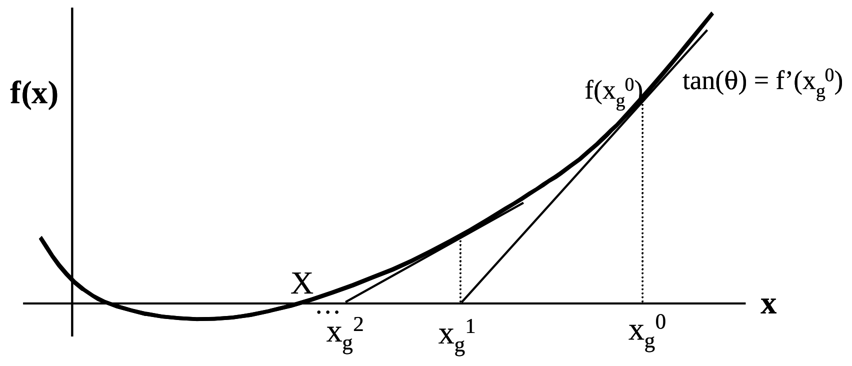
The Newton-Raphson Method

The basic step in solving for intersections is a technique for finding the roots of an equation for the form $f(x) = 0$, or $f(x, y) = 0$. The is easy if the function can be factorized, but we only know how to get exact solutions for roots of polynomials up to degree 4; beyond that, we must use numerical methods. Newton-Raphson is the easiest to implement.

Newton Raphson for single-variable functions:

```
Step 1. Make a starting guess: $x_g$
Step 2. Compute $f(x_g)$
Step 3. If  difference( $f(x_g), 0.0$) > tolerance
                Step 3.1. Compute $f'(x_g)$
                Step 3.2. $x_g = x_g – f(x_g)/f'(x_g)$
                Step 3.3. Go to Step 2.
        Else return $x_g$.
```

The figure below shows a graphical illustration of how the method converges.



Of course, when doing surface-surface intersections, we need to solve for roots of equations involving more than one parameter – typically, we deal with parametric equations of the form ***s***(u, v). For this, we use some numerically stable extension of Newton-Raphson to multivariate cases. We look at the general bi-variate case, find the roots of two equations: $F(x, y) = G(x, y) = 0$.

Let the exact root be at a point, (X, Y), and our first guess of the root be (x_g, y_g), such that $h = X – x_g$, and $k = Y – y_g$.
Then, using Taylor's expansion,

$0 = F(X, Y) = F(x_g, y_g) + h\, F_x(x_g, y_g) + k\, F_y(x_g, y_g) + \dots$ higher order terms of h and k
$0 = G(X, Y) = G(x_g, y_g) + h\, G_x(x_g, y_g) + k\, G_y(x_g, y_g) + \dots$ higher order terms of h and k

As the approximation approaches the true roots, h and k approach zero, and therefore the higher order terms may be ignored, giving (approximate) values, h_g, k_g:

$$\begin{bmatrix} h_g \\ k_g \end{bmatrix} = \frac{1}{D}\begin{bmatrix} G_y & -F_y \\ -G_x & F_x \end{bmatrix}\begin{bmatrix} -F \\ -G \end{bmatrix}, \qquad D = \det\begin{bmatrix} F_x & F_y \\ G_x & G_y \end{bmatrix}$$

Note:
(1) D is the determinant of the Jacobian matrix of F and G
(2) F, G, and the partial derivatives F_x, F_y, G_x, G_y are computed at (x_g, y_g).

Using these values, we improve our guess solution to: $x_{g+1} = x_g + h_g$; $y_{g+1} = y_g + k_g$.
This sets up the iterative procedure for computing the roots of the simultaneous non-linear equations $F(x, y) = G(x, y) = 0$.

Surfaces-Surface Intersection (SSI)

For two surfaces, $\boldsymbol{s}_1(u_1, v_1)$ and $\boldsymbol{s}_2(u_2, v_2)$, the intersection is the set of points satisfying simultaneously the equation: $\boldsymbol{s}_1(u_1, v_1) - \boldsymbol{s}_2(u_2, v_2) = \boldsymbol{0}$.
These are three scalar equations in four variables, so there is, in general, one degree of freedom – which implies that the intersection is a curve in 3D space (recall that a curve is a function with a single degree of freedom).

In general, solving for the resulting equation is difficult, so what we do is find a series of points on the intersection curve, and join these points to form the (approximate) result. The joining procedure can be linear, circular, or spline interpolation.

To find the series of points, a simple way would be to intersect the equation of the curve with a series of surfaces – for instance, a series of planes parallel to the XY plane, each separated by a small distance δ. Of course, you can see that this choice will potentially be unstable – some points will be far apart if the intersection curve itself is nearly parallel to the XY plane.

So we generalize: each additional constraint may be represented as a function, the step constraint function, $g(\mathbf{r}) = \mathbf{0}$. The substitution of $\mathbf{r} = \mathbf{r}_1(u, v)$ into $g(\mathbf{r}) = \mathbf{0}$ gives us the constraint $G_1(u, v) = 0$. In other words, we solve, at each step, for u_1, v_1, u_2 and v_2 from:

$$\mathbf{r}_1(u_1, v_1) - \mathbf{r}_2(u_2, v_2) = \mathbf{0}$$

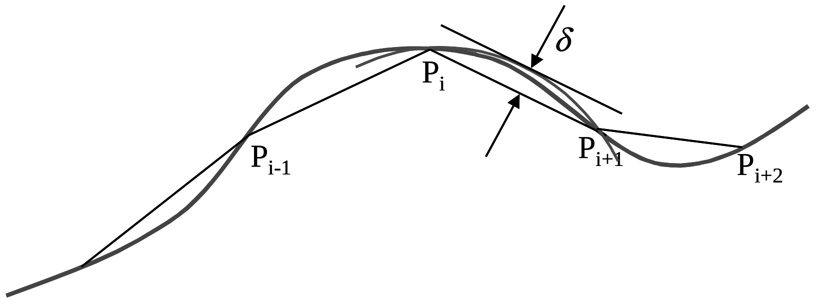
$$\lambda\, G_1(u_1, v_1) + \mu\, G_2(u_2, v_2) = 0$$

Where λ and μ are chosen at our will to simplify the constraint equation (for example, to cancel a higher order term between G_1 and G_2). We may use the Newton-Raphson method for solving this system.

Problem: How do we know a good set of surfaces, $g(\mathbf{r})$?

Typically, we would like to select the simplest surface for intersections, that is, planes. How do we know whether the separation between each subsequent pair of planes is small enough ? In other words, can we guarantee that the error between, say, a linear interpolation between two intersection points, and the real intersection curve, is less than some pre-determined tolerance level ? Let's look at the simplest analysis.

Step Length Determination



The figure above shows the approximations in the process. At each step, we need to know if the error between a linear approximation of the int-curve (shown by black line

segments) and the actual int-curve (blue line) is less than some given tolerance, TOL. If the line segments are small, we may approximate the int-curve for that region by a circular arc. A simple method to do so would be the following:
Step 1. Determine point $\boldsymbol{P_i}$;
Step 2. Determine next point $\boldsymbol{P_{i+1}}$;
Step 3. Determine the curvature, κ_i, of the intersection curve at $\boldsymbol{P_i}$.
Step 4. Fit a circle of radius $R_i = 1/\kappa_i$ between $\boldsymbol{P_i}$ and $\boldsymbol{P_{i+1}}$
Step 5. Error $\delta = R_i - sqrt(R_i^2 - |\boldsymbol{P_i P_{i+1}}|^2/4)$
Step 6. If $\delta > TOL$, then discard P_{i+1} and find a point closer to P_i by selecting a new $g(\boldsymbol{r})$.

Notes:
The formula for Step 5 can be easily derived by fitting a circular arc between the two points, $\boldsymbol{P_i}$ and $\boldsymbol{P_{i+1}}$.
The only problem that remains is to find the curvature κ of the intersection curve at $\boldsymbol{P_i}$.
Let the int-curve have tangent, normal and bi-normal vectors T, N and B. since the tangent is perpendicular to both surfaces,

$$\boldsymbol{T} = \boldsymbol{n_1} X \boldsymbol{n_2} / | \boldsymbol{n_1} X \boldsymbol{n_2} |$$

$$\kappa \boldsymbol{B} = -\kappa \boldsymbol{N} X \boldsymbol{T} = \kappa \boldsymbol{N} X (\boldsymbol{n_1} X \boldsymbol{n_2})/ | \boldsymbol{n_1} X \boldsymbol{n_2} |$$

and using the definitions of normal curvatures for the surfaces, we get:

$$\kappa \boldsymbol{B} = (\kappa_{n1} \boldsymbol{n_2} - \kappa_{n2} \boldsymbol{n_1}) / | \boldsymbol{n_1} X \boldsymbol{n_2} |$$

If the angle between the surface normals is q, then we get:

$$\kappa^2 = (\kappa_{n1}^2 - 2 \kappa_{n1} \kappa_{n2} \cos\theta + \kappa_{n1}^2)/ \sin^2\theta$$

All quantities on the RHS of this equation are computable, since we can compute the normal vectors at the surface, and also the normal curvature values.

There is one troubling assumption in the expression above – we don't really know a point on the surface, Pi. We only know some point close to it (since it is computed using a numerical method). However, there are techniques to determine a point on the surface closest to a given point (this is called a Jacobian inversion).

In all the above, we assume that the surfaces are well behaved, and the step size we select is reasonable. In reality, the above process is rather unstable and not as robust as we like. The popular variations of that will make the procedure more robust are described in some relatively recent literature, particularly the works of Lee & Fredericks, Barnhill et al, and Pratt & Geisow.

References

Most of the above notes are from Faux and Pratt. Other interesting references follow.

Faux, I. D., Pratt, M. J., *Computational Geometry for Design & Manufacture*, Ellis-Horwood

Choi, B. K., *Surface Modeling for CAD/CAM*, Elsevier

Pratt, M. J., Geisow, A. D., Surface/Surface Intersection Problems, *The Maths of Surfaces*, ed. J. A. Gregory

Barnhill, R. E., Farin, G., Jordan, M., Piper, B. R., Surface/Surface Intersection, *Computer-Aided Geometric Design*, 4 (1987), 3-16

Lee, R. B., Fredericks, D. A., Intersection of Parametric Surfaces and a Plane, *IEEE Computer Graphics and Applications*, 4, 8, (1984), 48-51