Implementation of Kernel Operations

Milind Sohoni
http://www.cse.iitb.ac.in/~sohoni

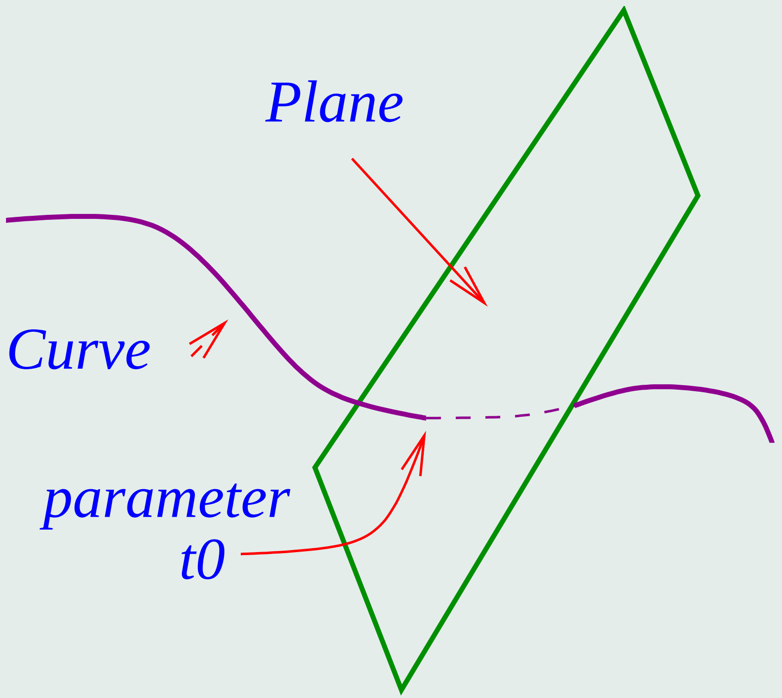
Overview

In this sequence of two talks we will outline algorithms for implementing typical kernel operations. We will discuss:

- Curve-Plane intersection.
- Curve-Curve intersection in 2d.
- Curve-Surface intersection.
- Point projection on Surface.
- Extrude surface creation.
- Blend constructions.

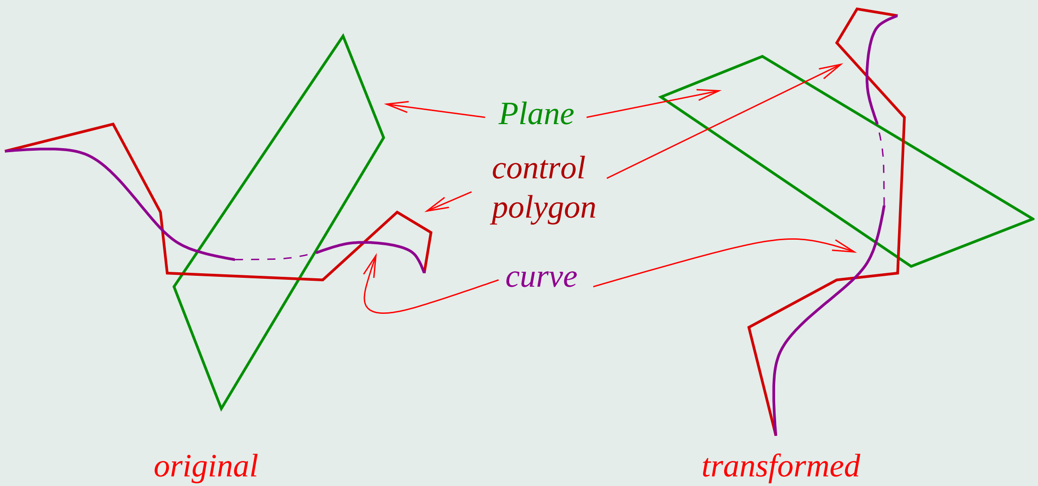
Curve-Plane Intersection

Suppose C is given as $C(t) = (x(t), y(t), z(t))$, and say that the plane is given by $ax + by + cz + d = 0$.



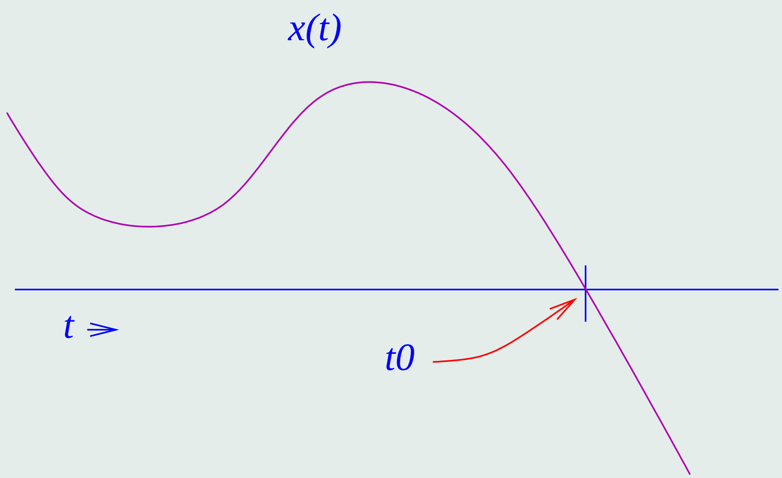
Nice Fact

If we have a linear transformation on the space which transforms $C(t)$ to $C'(t)$, and we have the control points P of $C(t)$ then those of $C'(t)$ are obtained by performing the linear operation on P.



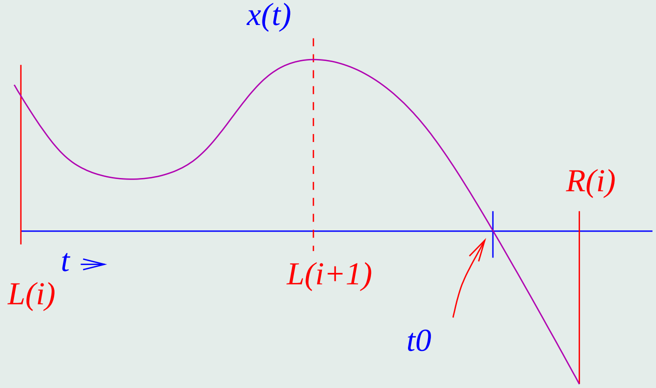
Thus...

We may assume that the plane is given by $X = 0$. In other words, we need to solve $x(t) = 0$ and get the parameter t.



Bisection Method

Input: Interval $[a, b]$ known to contain a zero[a].
Output: Either $[a, (a+b)/2]$ or $[(a+b)/2, b]$ with the same guarantee.



Stop: When interval is small enough.
Speed: linear in precsion.

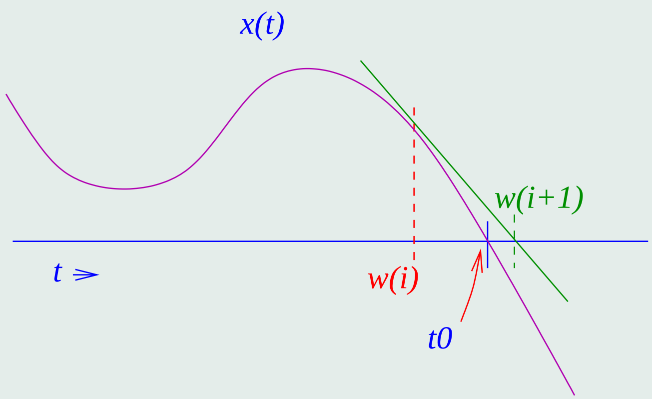
---

[a] How is one to check this?

Newton-Raphson Method

Input: Current Point w_i.
Method: Draw a tangent at $(w_i, f(w_i))$ and compute zero. Thus next point is:

$$w_{i+1} = w_i - \frac{f(w_i)}{f'(w_i)}$$



Stop: When $f(w_i)$ is small.
Speed: Very fast, $O(n^2)$, but very sensitive.

Home Page

Title Page

Contents

◀◀ ▶▶

◀ ▶

Page 8 of 25

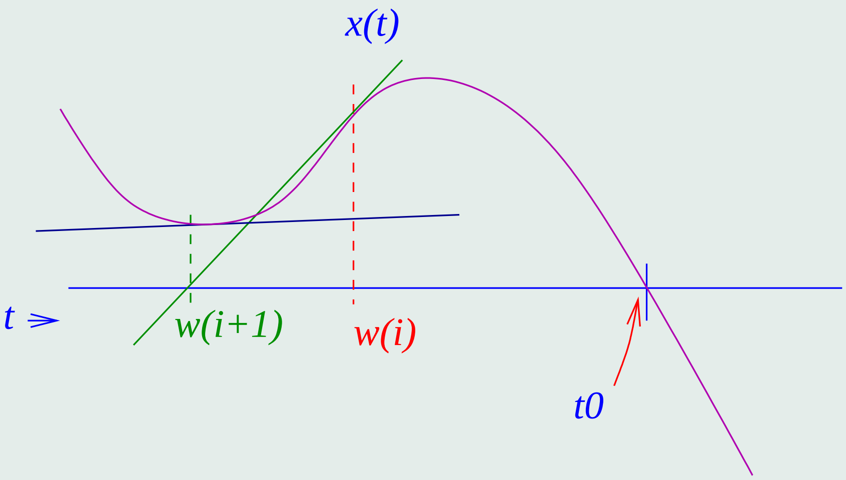
Go Back

Full Screen

Close

Quit

A Bad Case



Thus, NR is fast in (i) the neighborhood of a zero AND (ii) when the zero is simple.

General Procedure

1. Refine the Control polygon to locate a zero.
2. If zero is not simple, use special procedure.
3. For simple zero, use the Newton-Raphson method.

This shows the importance of:

- Differentiability of the curve.
- The Use of Control Polygon.
- Procedures (Subdivision, Knot-Insertion) to refine a control polygon of a curve.

Also note that one does NOT need the form of the function f, but just an evaluator.

Curve-Curve Intersection in 2D

Suppose $C = (x_1(t), y_1(t))$ and $D = (x_2(u), y_2(u))$ are two curves. The intersection is given by:

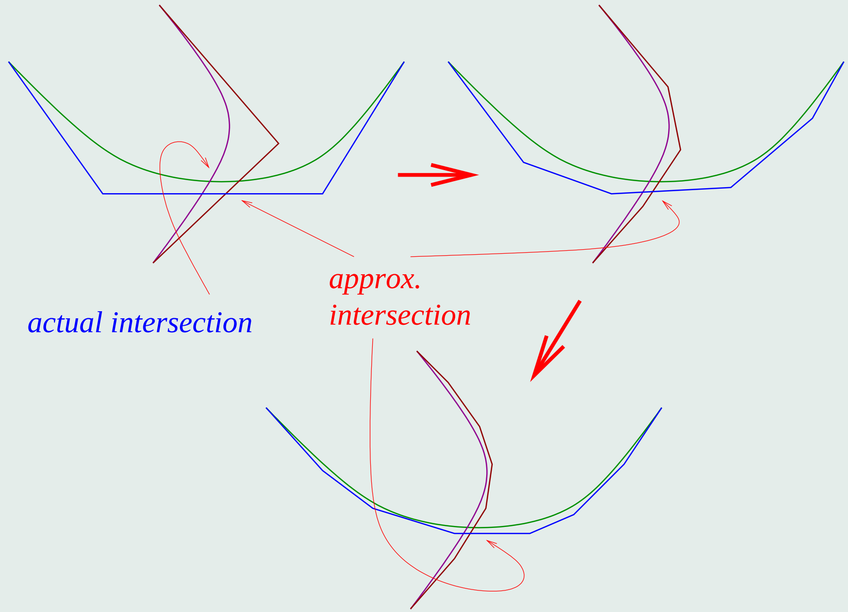
$$\begin{aligned} x_1(t) - x_2(u) &= 0 \\ y_1(t) - y_2(u) &= 0 \end{aligned}$$

Or in other words simultaneous solution of two equations in two variables:

$$f(t,u) = 0 \quad g(t,u) = 0$$

Again, there is the robust-but-slow polygon-subdivision based scheme, and the fast-but-sensitive multi-dimensional Newton-Raphson scheme.
Also note that the robust schemes usually work in model-space while the fast schemes work in parameter space.

A Sample Polygon-Based Intersection



Notice that the by the Bezier-Bernstein theorem, approximate intersection point gets closer to the actual intersection point.
Although not shown, many solvers will loaclize the intersection to smaller segments using sub-division.

The Multi-dimesional Newton-Raphson

Recall, we need to solve:

$$f(t,u) = 0 \quad g(t,u) = 0$$

If we have an initial guess (t_0, u_0), then we use:

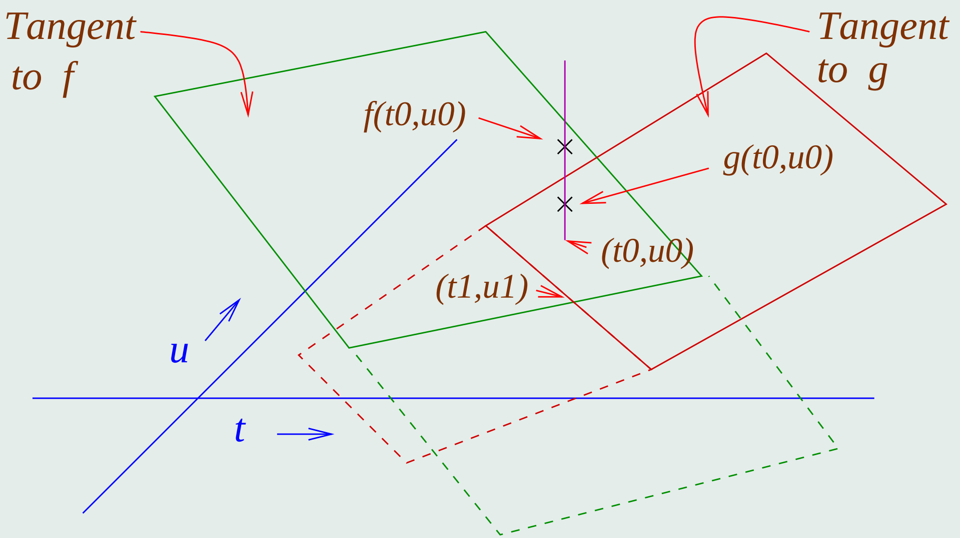
$$\begin{aligned} f(t,u) &\approx f(t_0,u_0) + \frac{\partial f}{\partial t}(t_0,u_0)[t-t_0] + \frac{\partial f}{\partial u}(t_0,u_0)[u-u_0] \\ g(t,u) &\approx g(t_0,u_0) + \frac{\partial g}{\partial t}(t_0,u_0)[t-t_0] + \frac{\partial g}{\partial u}(t_0,u_0)[u-u_0] \end{aligned}$$

Now these taylor approximations are linear and may be solved:

$$\begin{bmatrix} \frac{\partial f}{\partial t}(t_0,u_0) & \frac{\partial f}{\partial u}(t_0,u_0) \\ \frac{\partial g}{\partial t}(t_0,u_0) & \frac{\partial g}{\partial u}(t_0,u_0) \end{bmatrix} \begin{bmatrix} t \\ u \end{bmatrix} = \begin{bmatrix} f(t_0,u_0) - t_0\frac{\partial f}{\partial t}(t_0,u_0) - u_0\frac{\partial f}{\partial u}(t_0,u_0) \\ g(t_0,u_0) - t_0\frac{\partial g}{\partial t}(t_0,u_0) - u_0\frac{\partial g}{\partial u}(t_0,u_0) \end{bmatrix}$$

This give us the next iterant (t_1, u_1).

A Picture of the 2D Newton-Raphson



The tangent planes are shown, while the functions are not.

The convergence depends on order-2 constants which are curvatures.

A Numerical

Let

$$\begin{aligned} f(t,u) &= tu + t + u \\ g(t,u) &= t^2 + u \end{aligned}$$

Let $(t_0, u_0) = (1, 1)$. We evaluate various quantities:

$$\begin{bmatrix} \frac{\partial f}{\partial t} & \frac{\partial f}{\partial u} \\ \frac{\partial g}{\partial t} & \frac{\partial g}{\partial u} \end{bmatrix} = \begin{bmatrix} u+1 & t+1 \\ 2t & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 & 1 \end{bmatrix}$$

Since $f(1,1) = 3$ and $g(1,1) = 2$, we get the the equations:

$$\begin{aligned} 3 + 2(t-1) + 2(u-1) &= 0 \\ 2 + 2(t-1) + (u-1) &= 0 \end{aligned}$$

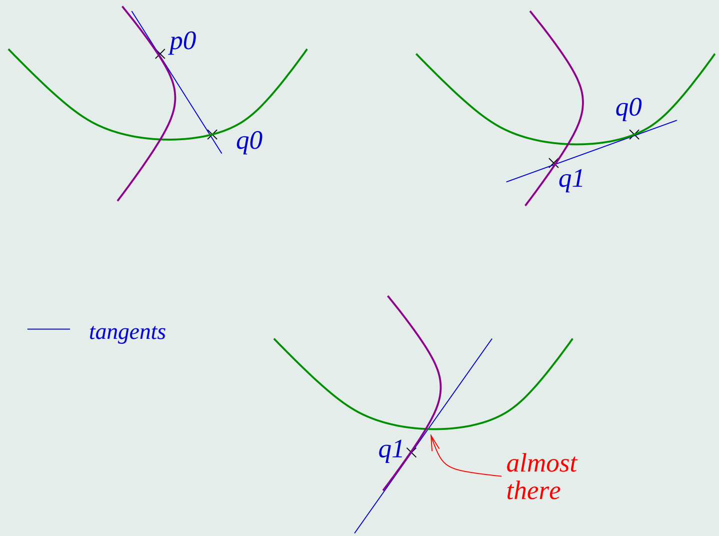
Solving this, we get $(t_1, u_1) = (0.5, 0)$. Note that

$$f(0.5, 0) = 0.5 \quad g(0.5, 0) = 0.25$$

This is better than the point $(1, 1)$ closer to the actual zero of $(0, 0)$.

Mixed Mode

There are also some mixed mode methods, which are of Newton-Raphson type but which act in the model space. These are even more sensitive, and of course, faster.



Outlined above is such a method. It constructs a sequence (p_i) on the first curve and (q_i) on the second, alternately, using tangents. This makes the method $O(n^2)$.

Curve-Surface Intersection

We easily set up the equations. Let $S = (x_1(u,v), y_1(u,v), z_1(u,v))$ and $C = (x_2(t), y_2(t), z_2(t))$. We get:

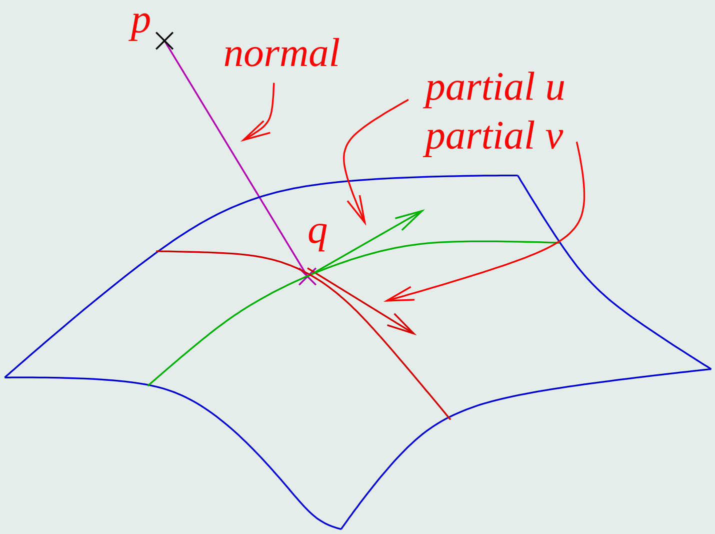
$$\begin{aligned} x_1(u,v) - x_2(t) &= 0 \\ y_1(u,v) - y_2(t) &= 0 \\ z_1(u,v) - z_2(t) &= 0 \end{aligned}$$

Thus we have a similar situation, viz., m equations in m unknowns. Again there are sub-division robust techniques which are used to localize the problem, and finally Newton-Raphson to finish off the job.

This theme repeats: one tries to cast a geometric problem into this formulation.

Point Projection

Let p be a point and S a surface. we wish to find the closest point $q \in S$ to p.



This is formulated by the condition that $q-p$ is perpendicular to the tangents $\frac{\partial}{\partial u}$ and $\frac{\partial}{\partial v}$ at q.

Details

Let $p = (x_0, y_0, z_0)$ and S be given by $x(u,v), y(u,v), z(u,v))$. The partials are given by:

$$\begin{aligned} \frac{\partial}{\partial u} &= \left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right) \\ \frac{\partial}{\partial v} &= \left(\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v}\right) \end{aligned}$$

We thus get the equation:

$$\begin{bmatrix} \frac{\partial x(u,v)}{\partial u} & \frac{\partial y(u,v)}{\partial u} & \frac{\partial z(u,v)}{\partial u} \\ \frac{\partial x(u,v)}{\partial v} & \frac{\partial y(u,v)}{\partial v} & \frac{\partial z(u,v)}{\partial v} \end{bmatrix} \begin{bmatrix} x(u,v) - x_0 \\ y(u,v) - y_0 \\ z(u,v) - z_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

How is this to be solved?

Thus, we get two equations in two unknowns. Note that even the evaluation of the defining equations requires partial derivatives.
Let us call $f(u, v)$ as:

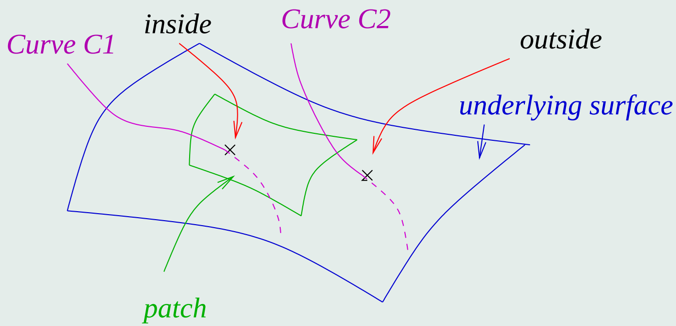
$$(x(u,v) - x_0)\frac{\partial x(u,v)}{\partial u} + (y(u,v) - y_0)\frac{\partial y(u,v)}{\partial u} + (z(u,v) - z_0)\frac{\partial z(u,v)}{\partial u}$$

$g(u, v)$ is similarly defined. We note that in applying the Newton-Raphson, we need not only $f(u_0, v_0)$ but $\frac{\partial f}{\partial u}$ and $\frac{\partial f}{\partial v}$ as well.
Thus, in applying the N-R technique, we will need f to be differentiable, i.e., $x(u, v)$ to be doubly differentiable.

Consequently, the surface must be doubly-differentiable.

Surrounding Logic

There are many more things to this than just the core solver. A simple example is say, the curve-surface intersection.
Note that the solver disregards the trim-curves and the domain of definition, but just considers the parametrization function $(x_1(u,v), y_1(u,v), z_1(u,v))$ of the surface, while solving.

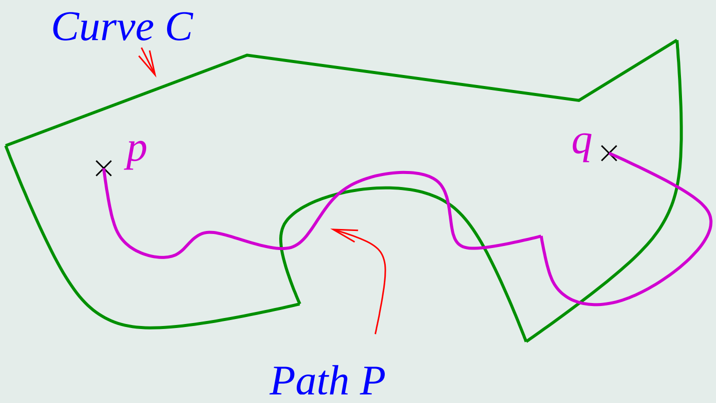


We see above, for the two curves, the solver will return (u_0, v_0) which is inside, and another (u_1, v_1) which is outside.

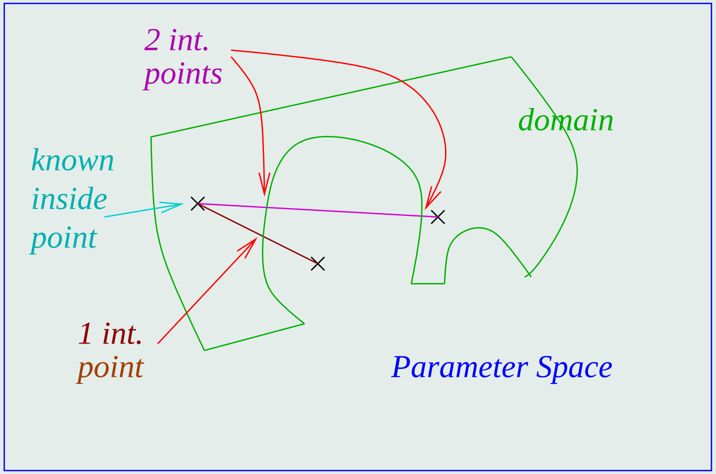
The Jordan Curve Theorem

Thus, once the (u_0, v_0) parameters of the intersection point have been determined, we must ascertain that (u_0, v_0) belongs to the domain. This is done by the Jordon Curve Theorem.

If C is a closed curve, and p is a point inside C. If P is a path from p to q which meets C transversally, then q is inside C iff the number of intersection points of C and P are even.



How does it apply



For each patch, record initially a (u_*, v_*) as a known point inside the domain. For any other point (u_0, v_0), its membership can be determined by counting the intersection points of the line joining these points and the bounding curves of the domain.

In Summary

Requirements:

- Continuous and highly differentiable function definitions.
- Definitions should extend beyond the domains of curves and surfaces.
- Evaluators: explicit definitions not required.

The basic paradigm:

- A solver for m equations in m unknowns. This is numerically stable.
- A formulation of the problem as an instance of above.
- An iterator whose fixed point is the solution.

Wait: What about Surface-Surface Intersections

Notice that our basic paradigm is m-equations and m-unknowns. Thus the solution set is necessarily a *finite* collection of points.

Surface-Surface intersection will create curves, i.e., a *continuum* of points. Clearly, a representation of this can only be done through finitely many points.

This brings in the need of a Constructor which will bring these higher dimensional entities into existence through a clever choice of points on it.

Things Not Covered– MANY

- Bounding Box methods and Polygon approximators.
- Polygon Calculus and solvers.
- Gradient Methods.
- Degeneracy solvers.

Exercises: Convert typical queries into solver problems.

- Is point p on the surface S?
- Locate on S the point of maximum z-coordinate.
- Do two $curves$ in space intersect?