Curve Representation

- All forms of geometric modeling require the ability to define curves.

Curves in Modeling

- Curves produced in modeling systems:
  - Straight lines
  - Conic sections
  - Free-form parametric curves (B-Splines, NURBS)
  - Curves of intersection between surfaces defined during model construction

Curve Use in Design

Engineering design requires ability to express complex curve shapes.
- Some examples
  - Turbine blades
  - Ship hulls
  - Automotive body panels

Complex Curves

- A curve can be described by a finite number of short straight segments.
  - However, on close inspection this is only an approximation.
  - To get a better approximation we can use more segments per unit length.
  - This increases the amount of data required to store the curve and makes it difficult to manipulate.
  - Need a way of representing these curves in a more mathematical fashion. Ideally, our descriptions will be:

Curves

Ideally, a curve description should be:
- Reproducible
  - The representation should give the same curve every time;
- Computationally Quick
- Easy to manipulate and edit
  - Especially important for design purposes.
- Easy to combine with other curves

Types of Curve Equations

- Explicit (non-parametric)
  \[ Y = f(X), Z = g(X) \]
- Implicit (non-parametric)
  \[ f(X,Y,Z) = 0 \]
- Parametric
  \[ X = X(t), Y = Y(t), Z = Z(t) \]

The explicit and implicit formats have serious disadvantages for use in computer-based modeling.
Curve Equation Examples

- **Explicit**  \( y = mx + b \)
- **Implicit**  \( x^2 + y^2 - R^2 = 0 \)
- **Parametric**  \( x = x_0 + ut ; y = y_0 + ut \)

Explicit Curves

- \( y = ax^3 + bx^2 + cx + d \), where \( a, b, c, d \) are constants
- For a given range of \( x \), will define a curve
- However, cannot turn back upon itself and cannot represent a vertical line

Implicit Curves

- \( ax^2 + by^2 + cxy + d = 0 \)
- Can produce self intersecting curves
- Easy to determine whether a given point lies on the line (set membership)
- Can be difficult to transform
- Packages may use implicit forms and translate them to parametric when needed.

Parametric:

\[ X = X(t) \],  \( Y = Y(t) \), \( Z = Z(t) \);
\[ 0 \leq t \leq 1 \] (typ)

- Substituting a value for \( t \) gives a corresponding position along curve
- Overcomes problems associated with implicit and explicit methods
- Most commonly used representation scheme in modelers

Parametric example

Linear (1st order) Curve

Parametric representation of a line. The parameter \( u \), is varied from 0 to 1 to define all points along the line.
\[ X = X(u) \quad Y = Y(u) \]
Parametric Line

• Line in then defined in terms of its endpoints
• Positions along the line are based upon the parameter value
  – For example, the midpoint of a line occurs at \( u = 0.5 \)

Scalar Formulation

• This means a parametric line can be defined by:
  \[ L(u) = [x(u), y(u), z(u)] = A + (B - A)u \]
  where \( A \) and \( B \) are the line endpoints.
  e.g. A line from point \( A = (2,4,1) \) to point \( B = (7,5,5) \) can be represented as:
  \[
  \begin{align*}
  x(u) & = 2 + (7-2)u = 2 + 5u \\
  y(u) & = 4 + (5-4)u = 4 + u \\
  z(u) & = 1 + (5-1)u = 1 + 4u
  \end{align*}
  \]

Another basic example would be that of a conic (circle)

• Two parameter curves are:
  \[ X = \cos(u) \]
  \[ Y = \sin(u) \]
  • with range \(-\pi/4 \leq u \leq \pi/4\)

Graph of \( X = \cos(u) \)

Graph of \( Y = \sin(u) \)

Combined curve is a quarter circle
Controls for circular arc

- Shape (based upon parametric equation)
- Location (based upon center point)
- Size
  - arc (based upon parameter range)
  - radius (a coefficient to unit value)
- Similar list could be formed for other conics

Curve Tangent Vectors

- Curves are defined by parametric equation
- Position along the curve is defined by the equation
- At any point along the curve there exists a vector defining the curve “direction”
- This is the tangent vector. It is defined by the first derivative of the parametric curve equation
- For a straight line this derivative will equal a constant

Curvature

- Parametric equation defines position along the curve
- The 1st derivative of the parametric equation defines direction or the “tangent vector”.
- The 2nd derivative defines the rate of change of direction; this is curvature.

For a plane curve (C), the curvature at a given point P has a magnitude equal to the reciprocal of the radius of an osculating circle and is a vector pointing in the direction of that circle’s center.

Curvature

- The osculating circle meets the curve at a tangent point
- An arc or circle has a constant curvature.
- A straight line has a curvature of zero.
  - When a circular fillet is tangent to a straight-line segment, the curvature falls abruptly from a constant value to zero.

Curvature

For a plane curve given parametrically as \( f(t) = (x(t), y(t)) \) the curvature is

\[
\kappa = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{3/2}}
\]

where the dots denote differentiation with respect to \( t \).
Osculating Plane

In the case of a space curve, the osculating circle lies in a plane defined by the curve’s normal and tangent vectors.

Moving Triad

• The curve normal vector points in the direction of the radius of curvature.
• The cross product of the normal and tangent vectors yields the bi-normal vector.
• A unique set of these vectors exists at each point along the curve.
• This set is referred to as the “moving triad”

Moving Triad

• The moving triad provides vital information of the characteristics of a moving object
  – Motion is in the direction of the tangent vector,
  – “up” is in the direction of the bi-normal vector
  – The rate of turning and turning direction are given by the curvature and the direction of the normal vector, respectively.
  – That is, as you move along the curve this triad always provides you with the forward, up and turning directions.

Representing Complex Curves

• typically represented as…
  – a series of simpler curves (each defined by a single equation) pieced together at their endpoints (piecewise construction).
  – These simpler curves may be linear or polynomial
  – simpler curves are based upon control points (data pts. used to define the curve)

Use of control points

• General curve shape may be generated using methods of:
  – Interpolation
    curve will pass though user defined control points
  – Approximation
    curve will pass near control points, curve may interpolate the start and end points

Control Points Defining Curves

• The following example shows an:
  Interpolating
  (passes through control points)
  Piecewise linear curve
  curve defined by multiple segments, in this case linear
Interpolating Curve: Piecewise linear

- Linear segments used to approximate smooth shape
- Segments joints known as KNOTS (see knot point slide)
- Requires too many datapoints for most shape approximations
- Representation not flexible enough to editing (for example, translating the curve is cumbersome)

Interpolating Curve: Piecewise linear

Piecewise polynomial (composite curves)

- Segments defined by polynomial functions
- Again, segments join at KNOTS
- Most common polynomial used is cubic (3rd order)*
- Segment shape controlled by two or more adjacent control points.
  * see curve order slide

Piecewise polynomial (composite curves)

Knot points

- Locations where segments join are referred to as knots
- Knots may or may not coincide with control points in interpolating curves, typically they DO NOT coincide.

Curve continuity

- concern is continuity at knots (where curve segments join)
- continuity conditions:
  - point continuity (no slope or curvature restriction)
  - tangent continuity (same slope at knot)
  - curvature continuity (same slope and curvature at knot)
Curve continuity

- Continuity is symbolically represented by capital "C" with a superscript representing level.
  - $C^0$ continuity is continuity of endpoint only, or continuity of position
  - $C^1$ continuity is tangent continuity or first derivative of position
  - $C^2$ continuity is curvature continuity or second derivative of position

Composite curves: continuity

- Point continuity
- Tangent continuity
- Curvature continuity

Curve continuity

- Continuity is also an issue at the intersection points of different user-defined curves
- When curves are used to define surfaces and solids, poor curve intersections can produce non-smooth geometry (creases, corners)

Interpolation curves

- Interpolating piecewise polynomial curve
- Typically possess curvature continuity
- Shape defined by:
  - endpoint and control point location
  - tangent vectors at knots*
  - curvature at knots*
* typically calculated internally by default

Interpolation curves

- User may be able to define:
  - Curve slope
  - Curvature

  At the interpolated control points.

Approximation techniques:

- Developed to permit greater design flexibility in the generation of freeform curves.
- Two very common methods in modern CAD systems, Bezier and B-Spline.
Approximation techniques

- employ control points (set of vertices that approximate the curve)
- curves do not pass directly through points (except possibly at start and end)
- intermediate points affect shape as if exerting a "pull" on the curve
- allow user to set shape by "pulling" out curve using control point location

Bezier curve & control points

Effect of curve order

- The higher the order of a curve, the "stiffer" the curve (less dramatic curvature changes)
- Maximum curve order dependent upon the number of control points
  order = one less than the number of control pts
- High order curves can exhibit irregularities

B-Spline curves

Most modern CAD systems use the NURB curve representation scheme.

- NURB stands for Non-Uniform, Rational, B-spline.
- Uniformity deals with the spacing of control points.
- Rational functions include a weighting value at each control point for effect of control point.

NURBs

- very popular due to their flexibility in curve generation.
- Permit the mathematical form to be used to represent entire families of curves including:
  - Bezier
  - B-Spline
  - conics
Wireframe modeling systems

• Definition
• Database information
• Uses/Disadvantages
• Curve representation

Wireframe

• name is taken from the model appearance
• only the edges of a geometry are displayed.

Wireframe model construction

• methods typically very straightforward
• uses the same commands and techniques as 2D construction.
• entities are the same as those for 2D graphics, with inclusion of some extended database information, (Z-coordinate data)

Basic database information

(if we restrict discussion to linear edge models)

• Vertices
  – Coordinate values for vertices
• Edges
  – Vertices associated with edges (endpoints)
• Faces
  – Loops (faces) formed by edges

Wireframe Tetrahedron

Brief discussion of basic database structures

• Relational
  – A set of lists, uses arrays for storage
• Hierarchical
  – A trees structure, think of a company’s executive structure
• Network
  – Uses data pointers to relate data elements
Relational Database

<table>
<thead>
<tr>
<th>Vertex List</th>
<th>Edge List</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1 (0,0,0)</td>
<td>E1 [V1, V2]</td>
</tr>
<tr>
<td>v2 (1,0,0)</td>
<td>E2 [V2, V3]</td>
</tr>
<tr>
<td>v3 (0,1,0)</td>
<td>E3 [V3, V1]</td>
</tr>
<tr>
<td>v4 (0,0,1)</td>
<td>E4 [V2, V4]</td>
</tr>
<tr>
<td></td>
<td>E5 [V4, V3]</td>
</tr>
<tr>
<td></td>
<td>E6 [V1, V4]</td>
</tr>
</tbody>
</table>

Hierarchical Database

Object level
- Object (root)

Surface level
- Surface1
- Surface2

Edge
- Edge1
- Edge4
- Edge6

Vertex
- Vertex1
- Vertex4

Coord
- x1
- y1

“Network” Database

Bounded face

Introduction to model validity criteria
(example restricted to linear edges)

- Each vertex has 3 coordinate values
- Each edge delimited by two vertices
- Minimum of 3 edges must intersect at each vertex
- Minimum of 3 edges required to define a loop (face)
**Wireframe Cylinder**
(structured to conform to validity criteria)

- In the example shown, the model definition (edges, vertices) was chosen to meet a validity criteria set.
  - Three edges intersect at a vertex
  - Edges delimited by two vertices

---

**Model Information Stored vs. Model Usefulness**

- The information available from a model database, and hence its usefulness, is dependent upon the data stored.
- User chooses model definition based upon application

---

**Examples of information available from wireframe database.**

- Distance and length calculations.
  - Distance between vertices, length of edges
- Basic entity connectivity
  - Which vertices delimit an edge.
  - Which edges share vertices.
  - Which edges form a loop (or face).
- A basic visual representation of the object being modeled.

---

**Visual Deficiencies of Wireframes**

- ambiguity
  - complex models difficult to interpret
- does not allow for use of photorealistic rendering tools*
  *some software capable of hidden line removal on limited basis.

---

**Ambiguous wireframe model**

What does this object look like?
Other Limitations of Wireframe Model Structure

- No ability to determine computationally information on mass properties such as volume, mass, moments of inertia, etc.
- No guarantee that the model definition is correct, complete or manufacturable.

Other Limitations of Wireframe Model Structure

- No ability to determine computationally information such as the line of intersect between two faces of intersecting models.
- Limited ability for checking interference between mating parts (typically visual only).

Wireframe Uses:

- Geometry display by modeling systems
- Visualization of motion (simple animations)
- Modeling of geometries such as projected profiles and revolutions.
- 2D drafting

Wireframe Uses

- Wireframe models of limited value for manufacturing and analysis.
  - Manufacturing: profile geometry only, 2D operations such as turning, profile milling
  - Analysis: can be used as input into FEA packages, especially in the case of truss and frame geometries.