Representation of 2D/3D Curves and Surfaces

AP Georgy Gimel’farb
ggim001@cs.auckland.ac.nz
1 Discretisation

2 Curves / surfaces

3 Structured grids

4 Unstructured grids

Recommended reading:

- M. Shaefer, Computational Engineering - Introduction to Numerical Methods, Springer, 2006: Chapter 3 and 8
Spatial Discretisation

- Discrete 1D, 2D, or 3D spatial domain: a grid (or lattice)
- Most practical objects have very complex geometry
  - Spatial discretisation of the problem domain is also called grid generation
  - May be a very time-consuming task!

Main interdependent problems:
- Efficient generation of the grid
- Grid’s impact onto the accuracy and the efficiency of the subsequent computation
A Practical Example of Discrete 3D Surfaces

http://tetruss.larc.nasa.gov/usm3d_v52/TESTCASES.html
A Practical Example of Discrete 3D Surfaces

http://tetruss.larc.nasa.gov/usm3d_v52/TESTCASES.html
Geometric Description

**Volume modelling:**
- A number of chosen simple objects (e.g., cubes, cylinders, spheres, etc.) combined by Boolean algebra operations

\[ \square - \circ = \bigcirc \]

**Most common approach:** **Boundary modelling:**
- Specifying and composing curvilinear boundary curves (2D case) or surfaces (3D case)
Numerical Grids

- Discretisation of the problem domain:
  - Grid structure – defined by grid cells
  - 2D cells – defined by grid lines
  - 3D cells – defined by grid surfaces defined again by grid lines

- The more regular the grid, the more efficient the computational algorithms

- But the regular grid is less flexible in modelling complex geometry of a problem domain
Beziers Curves and Surfaces

Usual description of curves and curvilinear surfaces:

1. **Bezier curves**
2. **B-splines** *(basis splines)* generalising the Bezier curves
   - Any spline is represented as a linear combination of B-splines

**Bezier curve** \( \mathbf{x} = \mathbf{x}(s) \) of degree \( n \) over the range \( a \leq s \leq b \):

\[
\mathbf{x}(s) = \sum_{i=0}^{n} b_i B^n_i(s) \quad \text{with} \quad n + 1 \text{ control points} \quad b_i; \quad i = 0, \ldots, n
\]

**Bezier surface** \( \mathbf{x} = \mathbf{x}(s, t) \); \( a_1 \leq s \leq b_1; \ a_2 \leq t \leq b_2 \):

\[
\mathbf{x}(s, t) = \sum_{i=0}^{n} \sum_{k=0}^{m} b_{i,k} B^n_i(s) B^m_k(t)
\]
Bernstein Polynomials

Control points for Bezier curves and surfaces are combined with the Bernstein polynomials:

\[ B_i^n(s) = \frac{1}{(b-a)^n} \binom{n}{i} (s-a)^i (b-s)^{n-i} \]

where \( \binom{n}{i} \) are the binomial coefficients:

\[ \binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \leq i \leq n \\ 0 & \text{otherwise} \end{cases} \]

<table>
<thead>
<tr>
<th>( n )</th>
<th>( B_0^n(s) )</th>
<th>( B_1^n(s) )</th>
<th>( B_2^n(s) )</th>
<th>( B_3^n(s) )</th>
<th>( B_4^n(s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 1 - s )</td>
<td>( s )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( (1 - s)^2 )</td>
<td>( 2s(1 - s) )</td>
<td>( s^2 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( (1 - s)^3 )</td>
<td>( 3s(1 - s)^2 )</td>
<td>( 3s^2(1 - s) )</td>
<td>( s^3 )</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( (1 - s)^4 )</td>
<td>( 4s(1 - s)^3 )</td>
<td>( 6s^2(1 - s)^2 )</td>
<td>( 4s^3(1 - s) )</td>
<td>( s^4 )</td>
</tr>
</tbody>
</table>
Bernstein Polynomials: \(0 \leq s \leq 1\)

**Example**: Cubic polynomials

\[
\begin{align*}
B_0^3(s) &= (1 - s)^3; \\
B_1^3(s) &= 3s(1 - s)^2; \\
B_2^3(s) &= 3s^2(1 - s), \text{ and } B_3^3(s) &= s^3
\end{align*}
\]

Properties of the polynomials

\[
B_i^n(s) = \binom{n}{i} s^i (1 - s)^{n-i}
\]

- Recursive definition: \(B^0_0(s) = 1\)

\[
\begin{align*}
B_i^n(s) &= (1 - s)B_{i-1}^{n-1}(s) + sB_i^{n-1}(s) \\
B_i^n(s) &= 0 \text{ for } j < 0 \text{ and } j > n
\end{align*}
\]

- Partition of unity:

\[
\sum_{i=0}^{n} B_i^n(s) = 1
\]

- Bezier curve of degree 1 \(\Rightarrow\) linear interpolation between the end points:

\[
\begin{align*}
B_0^1(s) &= 1 - s \\
B_1^1(s) &= s
\end{align*}
\]

\[
\mathbf{x}(s) = \mathbf{b}_0(1 - s) + \mathbf{b}_1 s
\]
Beziers Curve of Degree 3; $0 \leq s \leq 1$: An Example

\[ x(s) = \sum_{i=0}^{3} b_i \binom{3}{i} s^i (1 - s)^i \]

\[ = b_0 (1 - s)^3 + 3b_1 s(1 - s)^2 + 3b_2 s^2(1 - s) + b_3 s^3 \]

\[ = b_0 + 3s(b_1 - b_0) + 3s^2(b_2 - 2b_1 + b_0) + s^3(b_3 - 3b_2 + 3b_1 - b_0) \]

- Movement of one control point affects the whole curve
- $B_i^n(s)$ has only one maximum at $s = \frac{i}{n}$ in $[0, 1]$, so the movement of $b_i$ mostly affects the curve around the parameter value $s = \frac{i}{n}$
Bezjèr Curve of Degree 3; \(0 \leq s \leq 1\): An Example (cont.)

Movement of one control point \((b_1)\):

\[
\begin{align*}
&y \quad \downarrow \\
&s = 0 \quad b_0 \\
&s = 1 \quad b_3
\end{align*}
\]
Bezier Curve of Degree 3; $0 \leq s \leq 1$: An Example (cont.)

Movement of one control point ($b_2$):

![Bezier Curve Diagram](image-url)
Problems in Boundary Modelling

- Gaps, discontinuities, overlaps between neighbouring surfaces or curves
- Bezier surfaces provide a good basis for corrections
  - Simple conditions for the control points near the boundary for smooth transitions between the neighbouring surface pieces
  - E.g., continuity of first and second derivatives
Structured Grids

- Generating a structured grid – a unique mapping

\[(x, y) = (x(i, j), y(i, j)) \quad \text{or} \quad (i, j) = (i(x, y), j(x, y))\]

- \(i = 0, 1, \ldots, N\) and \(j = 0, 1, \ldots, M\) – discrete values in the regular logical or computational domain
- \((x, y)\) – continuous coordinates in the (generally) irregular physical or problem domain
Structured Grids

- Properties of a structured grid are controlled by the components of the Jacobi matrix (or Jacobian) $\mathbf{J}$ of the mappings $(i, j) \rightarrow (x, y)$ or $(x, y) \rightarrow (i, j)$:

  $$
  \mathbf{J} = \begin{bmatrix}
  \frac{\partial i}{\partial x} & \frac{\partial i}{\partial y} \\
  \frac{\partial j}{\partial x} & \frac{\partial j}{\partial y}
  \end{bmatrix}
  $$

- These components are called metrics of the grid
- For the uniqueness of the mapping – the non-zero determinant of $\mathbf{J}$:

  $$
  \det(\mathbf{J}) = \frac{\partial i}{\partial x} \frac{\partial j}{\partial y} - \frac{\partial j}{\partial x} \frac{\partial i}{\partial y} \neq 0
  $$
**Initialisation**: Prescribe grid points at the boundary of the problem domain (in physical coordinates $\mathbf{x} = [x, y]$):

$$
\begin{align*}
\mathbf{x}(i, 0) &= \mathbf{x}_s(i); & \mathbf{x}(i, M) &= \mathbf{x}_n(i) & \text{for } i = 0, \ldots, N \\
\mathbf{x}(0, j) &= \mathbf{x}_w(j); & \mathbf{x}(N, j) &= \mathbf{x}_e(j) & \text{for } j = 0, \ldots, M
\end{align*}
$$

- Locations of the prescribed grid points – on the corresponding boundary curves
- Compatibility conditions for the corner points:

$$
\begin{align*}
\mathbf{x}_s(0) &= \mathbf{x}_w(0); & \mathbf{x}_s(N) &= \mathbf{x}_e(0) \\
\mathbf{x}_n(0) &= \mathbf{x}_w(M); & \mathbf{x}_n(N) &= \mathbf{x}_e(M)
\end{align*}
$$
**Algebraic Grid Generation**

- **Linear (or transfinite) interpolation** to determine interior points of the domain from the boundary grid points (below \( \alpha_i = \frac{i}{N} \) and \( \beta_j = \frac{j}{M} \)):

\[
\mathbf{x}(i, j) = (1 - \beta_j) \mathbf{x}_s(i) + \beta_j \mathbf{x}_n(i) + (1 - \alpha_i) \mathbf{x}_w(j) + \alpha_i \mathbf{x}_e(j)
- \alpha_i (\beta_j \mathbf{x}_n(N) + (1 - \beta_j) \mathbf{x}_s(N))
- (1 - \alpha_i) (\beta_j \mathbf{x}_n(0) + (1 - \beta_j) \mathbf{x}_s(0))
\]
Algebraic Grid Generation: An Example

Boundary points (n – “north”; s – “south”; e – “east”; w – “west”):

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_n(i)$</td>
<td>(0.5, 1.5)</td>
<td>(1.3, 0.52)</td>
<td>(2.03, 1.1)</td>
<td>(2.65, 0.475)</td>
<td>(3, 0)</td>
</tr>
<tr>
<td>$x_s(i)$</td>
<td>(0, 0)</td>
<td>(0.35, -0.15)</td>
<td>(0.65, -0.35)</td>
<td>(0.85, -0.7)</td>
<td>(1, -1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_w(j)$</td>
<td>(0, 0)</td>
<td>(0.285, 0.2)</td>
<td>(0.5, 0.5)</td>
<td>(0.6, 1)</td>
<td>(0.5, 1.5)</td>
</tr>
<tr>
<td>$x_e(j)$</td>
<td>(1, -1)</td>
<td>(1.4, -0.55)</td>
<td>(1.85, -0.1)</td>
<td>(2.4, 0.125)</td>
<td>(3, 0)</td>
</tr>
</tbody>
</table>

Transfinite interpolation of the points $x(i, j)$:

<table>
<thead>
<tr>
<th>$j/i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0, 0)</td>
<td>(0.35, -0.15)</td>
<td>(0.65, -0.35)</td>
<td>(0.85, -0.7)</td>
<td>(1, -1)</td>
</tr>
<tr>
<td>1</td>
<td>(0.285, 0.2)</td>
<td>(0.682, 0.186)</td>
<td>(1.025, 0.025)</td>
<td>(1.265, -0.3)</td>
<td>(1.4, -0.55)</td>
</tr>
<tr>
<td>2</td>
<td>(0.5, 0.5)</td>
<td>(0.975, 0.598)</td>
<td>(1.39, 0.45)</td>
<td>(1.7, 0.125)</td>
<td>(1.85, -0.1)</td>
</tr>
<tr>
<td>3</td>
<td>(0.6, 1)</td>
<td>(1.206, 1.103)</td>
<td>(1.748, 0.863)</td>
<td>(2.181, 0.431)</td>
<td>(2.4, 0.125)</td>
</tr>
<tr>
<td>4</td>
<td>(0.5, 1.5)</td>
<td>(1.3, 1.52)</td>
<td>(2.03, 1.1)</td>
<td>(2.65, 0.475)</td>
<td>(3, 0)</td>
</tr>
</tbody>
</table>
Algebraic Grid Generation: An Example

\[
\begin{bmatrix}
    x(2,1) \\
    y(2,1)
\end{bmatrix}
= 0.75 \begin{bmatrix}
    0.65 \\
    -0.35
\end{bmatrix} + 0.25 \begin{bmatrix}
    2.03 \\
    1.10
\end{bmatrix} + 0.5 \begin{bmatrix}
    0.285 \\
    0.2
\end{bmatrix} + 0.5 \begin{bmatrix}
    1.4 \\
    -0.55
\end{bmatrix}

- 0.5 \left( 0.25 \begin{bmatrix}
    3 \\
    0
\end{bmatrix} + 0.75 \begin{bmatrix}
    1 \\
    -1
\end{bmatrix} \right) - 0.5 \left( 0.25 \begin{bmatrix}
    0.5 \\
    1.5
\end{bmatrix} + 0.75 \begin{bmatrix}
    0 \\
    0
\end{bmatrix} \right)

= \begin{bmatrix}
0.4875 + 0.5075 + 0.1425 + 0.7 - 0.75 - 0.0625 \\
-0.2625 + 0.275 + 0.1 - 0.275 + 0.375 - 0.1875
\end{bmatrix} = \begin{bmatrix}
1.025 \\
0.025
\end{bmatrix}
\]
Algebraic Grid Generation

- Generalisation of the transfinite interpolation
  - Prior problem domain partitioning into different sub-domains
  - Usage of higher-order interpolation rules
- Higher discretisation accuracy – clustering grid lines in regions with high gradients of the variables
  - **Stretching functions** concentrate grid points along a curve
    - Simple stretching function to concentrate grid points \( x_i; \ i = 1, \ldots, N - 1 \), at the right end of the interval \([x_0, x_N]\):

\[
x_i = x_0 + \frac{\gamma^i - 1}{\gamma^N - 1} (x_N - x_0)
\]

- Control parameter (expansion factor) \( \gamma; 0 < \gamma < 1 \): the closer \( \gamma \) is to zero, the denser the grid points are near \( x_N \):

\[
\begin{align*}
\gamma &= 0.5 \\
\gamma &= 0.7 \\
\gamma &= 0.9 \\
\gamma &= 1.0
\end{align*}
\]
Unstructured Grids: Delaunay Triangulation

• Unique triangulation of a given set of grid points fulfilling certain properties

• **Bowyer-Watson algorithm**
  - Based on the property that the circle through three points (circumcircle) of an arbitrary triangle contains no other points
  - The circumcircle of a triangle is uniquely determined: its midpoint is at the intersection point of the perpendicular bisectors of the triangle sides

• **Initialisation**: a very coarse problem domain triangulation

• **Successive addition of points**:  
  - Deleting all triangles containing the new point
  - Connecting the new point with the corner points of the polygon which results from the deletion of the triangles
Delauney Triangulation

Inserting a new point in a Delauney triangulation with the Bowyer-Watson algorithm:

- This triangulation can be started from scratch by defining a super-triangle fully containing the problem domain
- After having inserted all points, all triangles which contain one or more vertices of the super-triangle are removed
If locations at which the grid points should be added are prescribed, the Bowyer-Watson algorithm is carried out as long as all points are inserted.

Alternatively, it can be combined with a procedure for an automatic generation of grid points:

- One possible approach – a priority list on the basis of the diameter of the circumcircle
- The triangle with the highest priority is checked whether the diameter of its circumcircle is larger than a given threshold
- If this criterion is fulfilled, the midpoint of the circumcircle is chosen as a new grid point
- Then the obtained new triangles are inserted into the grid and added to the priority list
- Termination when no more triangle fulfils the above criterion
Advancing Front Method

For a prescribed distribution of the interior grid points

1. **Initialisation**: Number successively all edges along the outer boundary of the problem domain clockwise and the inner boundaries (if exist) counter-clockwise to define the advancing front by the vector $a$ of the full numbering.

2. For the last edge in $a$: search all grid points on or within the advancing front; select the point with the smallest sum of the distances to the two grid points of the last edge, and form a new triangle with the selected and the two points of the last edge.

3. Delete from $a$ the edges of the new triangle, which are contained in $a$; adjust the numbering of the remaining edges, and add the edges of the new triangle, which are not contained in $a$, at the end of $a$.

4. Repeat Steps 2 and 3 until all edges in $a$ are deleted.

The interior points can be automatically selected rather than prescribed.
Advancing Front Method

Grid point distribution

Steps of unstructured grid generation: the blue thick lines represent the respective actual advancing fronts
Pros and Cons of Grid Generation Methods

• Advancing front methods:
  + Simple automatic generation of the interior grid points ensuring “good” quality of the triangles
  + Boundary of the problem domain is represented by grid lines
    • Boundary discretisation defines the starting point of the method and is not modified during the process
    – Relatively high computational effort for checking if the points generated are admissible and have tolerable distances

• Delauney triangulations
  + Less computationally intensive than advancing front methods
    • No complex checking of intersections and minimal distances
    – No boundary integrity is guaranteed for non-convex problem domains
      • Triangles can be located (at least partially) outside the problem domain
      • Difficulties for generalisations for the 3D case