Isocontour/surface Extractions

2D Isocontour 3D Isosurface

Isocontour (0)
Remember bi-linear interpolation

To know the value of P, we can first compute p4 and p5 and then linearly interpolate P

Isocontour (1)
Consider a simple case: one cell data set
The problem of extracting an isocontour is an inverse of value interpolation. That is:

Gieven f(p0)=v0, f(p1)=v1, f(p2)=v2, f(p3)=v3
Find the point(s) P within the cell that have values F(P) = C

Isocontour (2)
We can solve the problem based on linear interpolation

(1) Identify edges that contain points P that have value f(P) = C
(2) Calculate the positions of P
(3) Connect the points with lines

Isocontouring – Step 1
(1) Identify edges that contain points P that have value f(P) = C

If v1 < C < v2 then the edge contains such a point

Isocontouring – Step 2
(2) Calculate the position of P

Use linear interpolation:

P = P1 + (C-v1)/(v2-v1) * (P2-P1)
Isocontouring - Step 3

Connect the points with line(s)

Based on the principle of linear variation, all the points on the line have values equal to \( C \).

Inside or Outside?

Just a naming convention

1. If a value is smaller than the isovalue, we call it “Inside”
2. If a value is greater than the isovalue, we call it “Outside”

Isocontour cases

How many cases can an isocontour intersect a cell?

When comparing the value of \( P_i \) with the isovalue \( C \), there can be two cases:

\[ V_i \geq C \text{ or } V_i < C \]

So there can be \( 2 \times 2 \times 2 \times 2 = 16 \) cases!

How many cases again?

In fact, there are only 4 unique topological cases

1. Complete outside (inside)
2. One inside (outside), 3 outside (inside)
3. Two inside (outside), two outside (inside)
4. Two contours pass through

Put it all together

Divide-and-conquer algorithm

1. Look at one cell at a time
2. Compare the values at 4 vertices with the isovalue \( C \)
3. Linear interpolate along the edges
4. Connects the interpolated points together

3D Isocontour (Isosurface)
Isosurface Extraction

Extend the same divide-and-conquer algorithm to three dimension

- 3D cells
- Look at one cell at a time
- Let’s only focus on voxel

Divide-and-Conquer

How many cases?

Now we have 8 vertices
So it is: $2^8 = 256$

How many unique topological cases?

Case Reduction (1)

Value Symmetry

Case Reduction (2)

Rotation Symmetry

By inspection, we can reduce 256 $\rightarrow$ 14

Isosurface Cases

Total number of cases: 14
Marching Cubes Algorithm

A Divide-and-Conquer Algorithm

Vi is ‘1’ or ‘0’ (one bit)
1: > C; 0: <C (C = sovalue)

Each cell has an index mapped to a value ranged [0,255]

Index = v8 v7 v6 v5 v4 v3 v2 v1

Marching Cubes (2)

Given the index for each cell, a table lookup is performed to identify the edges that have intersections with the isosurface

Marching Cubes (3)

+ Perform linear interpolations at the edges to calculate the intersection points
  + Connect the points

Why is it called marching cubes?

Linear search through cells
Row by row, layer by layer
Reuse the interpolated points for adjacent cells