Isocontours

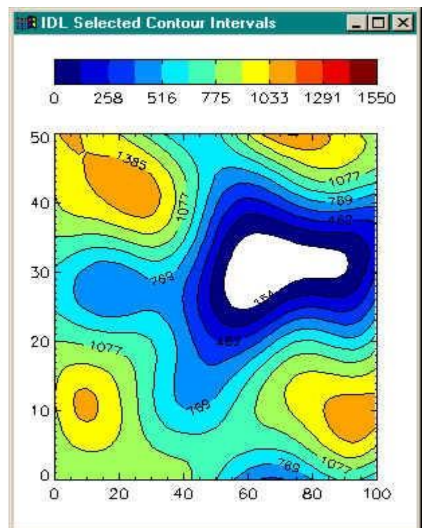
Marching Cubes

Courtesy of Prof. Han-Wei Shen
(CSE 5544, The Ohio State University)

What is an iso-contour?

- Points in a scalar field (pressure, temperature, etc.) that have a constant value
  - 2D: isoline
  - 3D: isosurface
- It is also called a level set

$$L_f(c) = \{x \mid f(x) = c\}$$



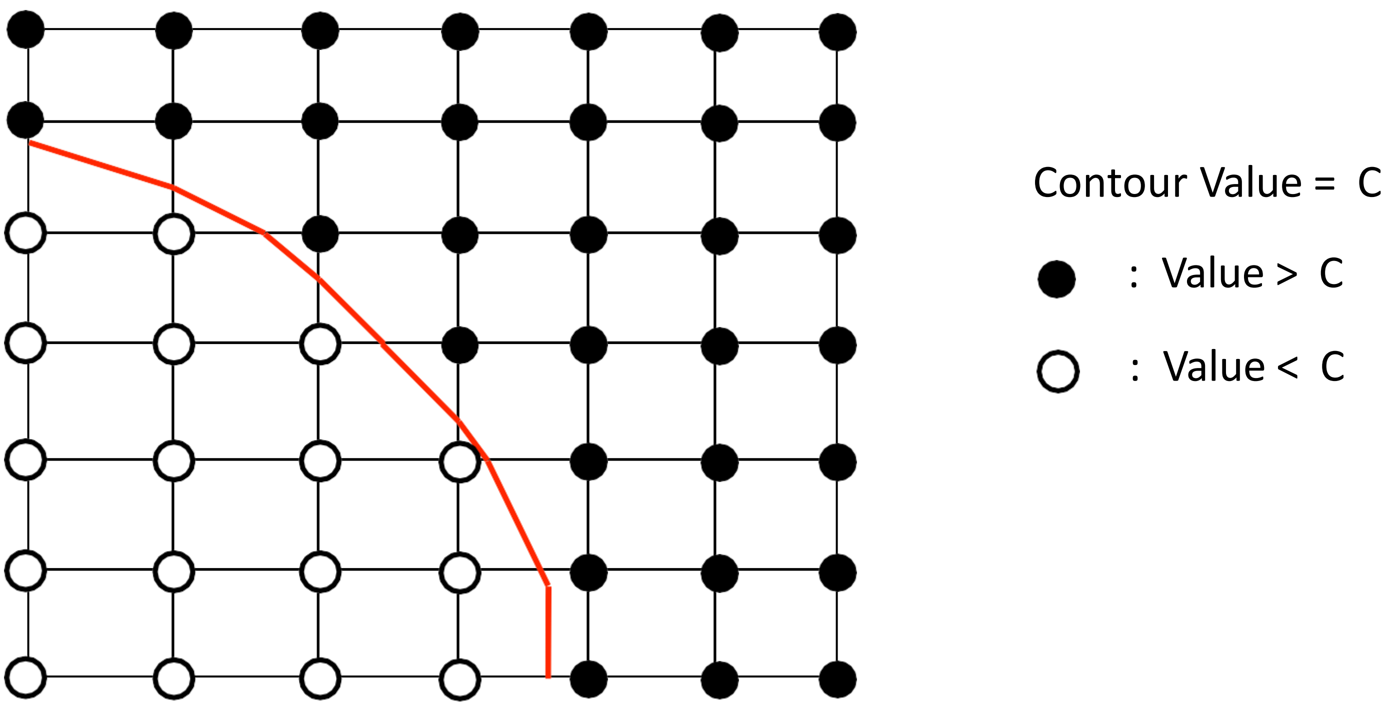
isolines

Isosurfaces

S05-01

2D Isocontour

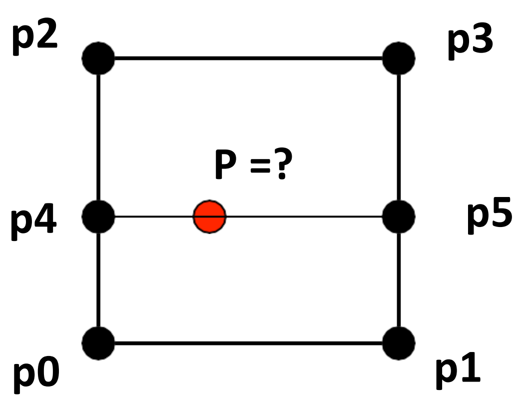
- Given a 2D scalar field, computing a 2D isocontour can be achieved cell by cell



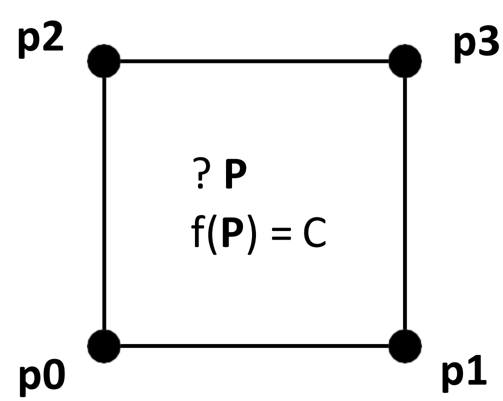
Isocontouring

- Isocontouring in a cell is an inverse problem of value interpolation

Interpolation

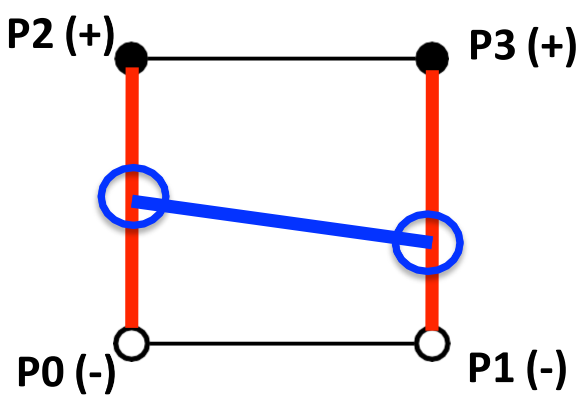


Isocontouring



Isocontouring by Linear Interpolation

- We can compute isocontour within a cell based on linear interpolation



(1) Identify edges that are 'zero crossing'

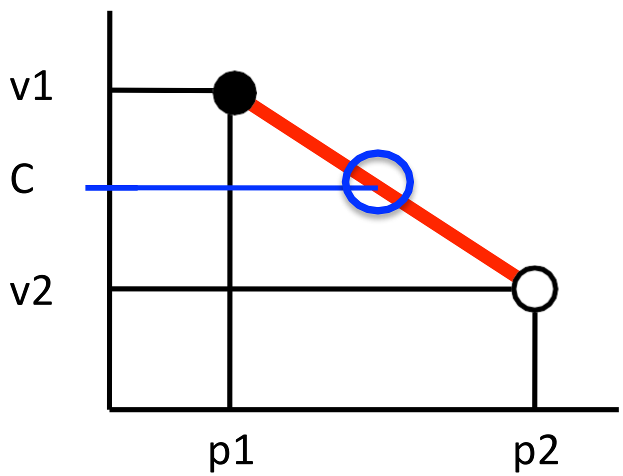
- Values at the two end points are greater (+) and smaller (-) than the contour value

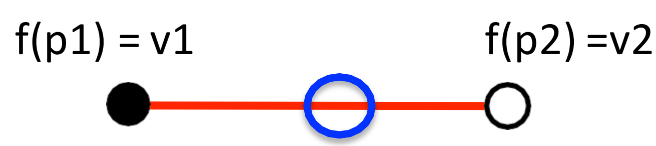
(2) Calculate the positions of **P** in those edges

(3) Connect the points with lines

Step 1: Identify Edges

- Edges that have values greater (+) and less (-) than the contour values must contain a point P that has f(p) = c
  - This is based on the assumption that values vary linearly and continuously across the edge



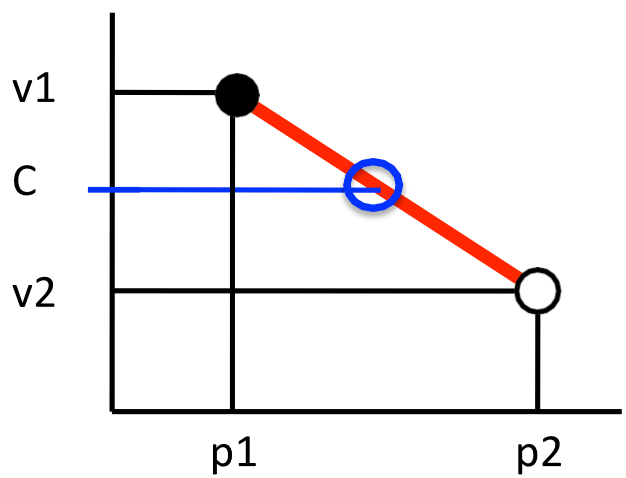


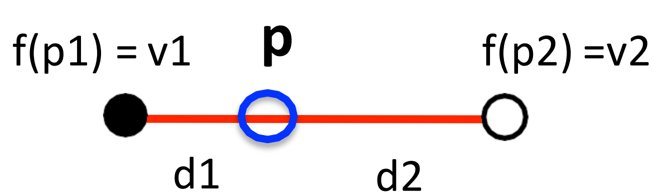
Step 2: Compute Intersection

- The intersection point **f(p) = c** on the edge can be computed by linear interpolation

$$d1/d2 = (v1-C) / (C-v2) \Rightarrow (\mathbf{p} - \mathbf{p}1)/(\mathbf{p}2-\mathbf{p}1) = (v1-C) / (v1-v2)$$

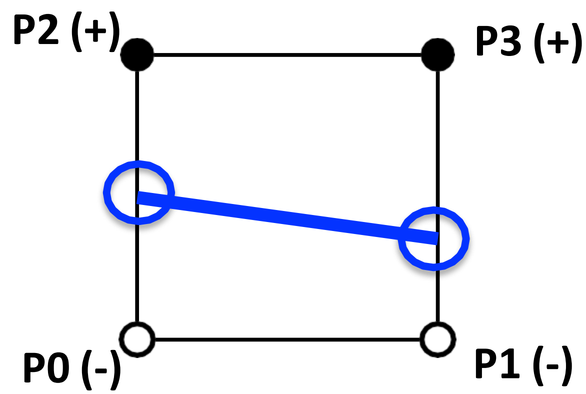
$$\mathbf{p} = (v1-C)/(v1-v2) * (\mathbf{p}2-\mathbf{p}1) + \mathbf{p}1$$





Step 3: Connect the Dots

- Based on the principle of linear interpolation, all points along the line $\mathbf{p}_4\mathbf{p}_5$ have values equal to C



Repeat Step1 – Step 3 for all cells

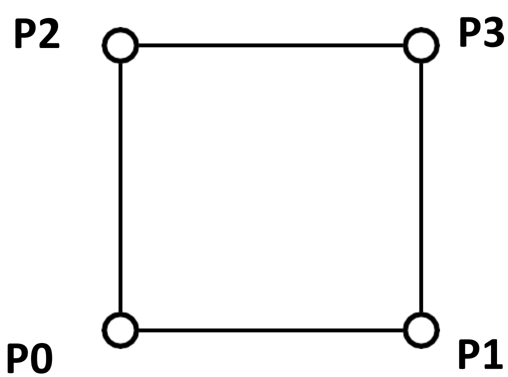
S05-02

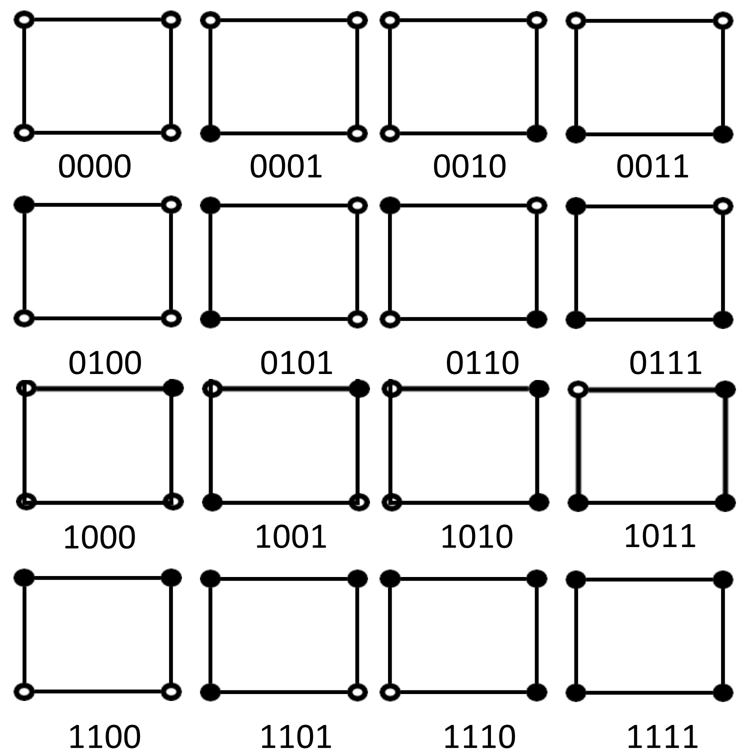
Isocontour Cases

- How many way can an isocontour intersect a linear rectangular cell?

P4	P3	P2	P1

1: > C
0: < C

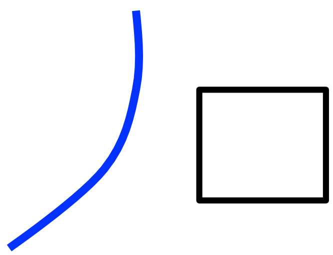




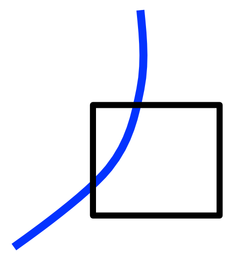
- The value at each vertex can be either greater or less than the contour value
- So there are 2 x 2 x 2 x 2 = 16 cases

Unique Topological Cases

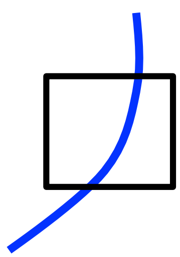
- There are only four unique topological cases



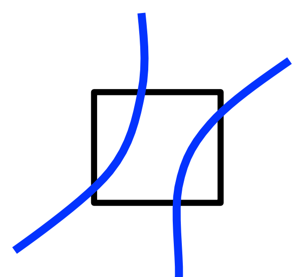
(1) No intersection



(2) Intersect with two adjacent edges



(3) Intersect with two opposite cases

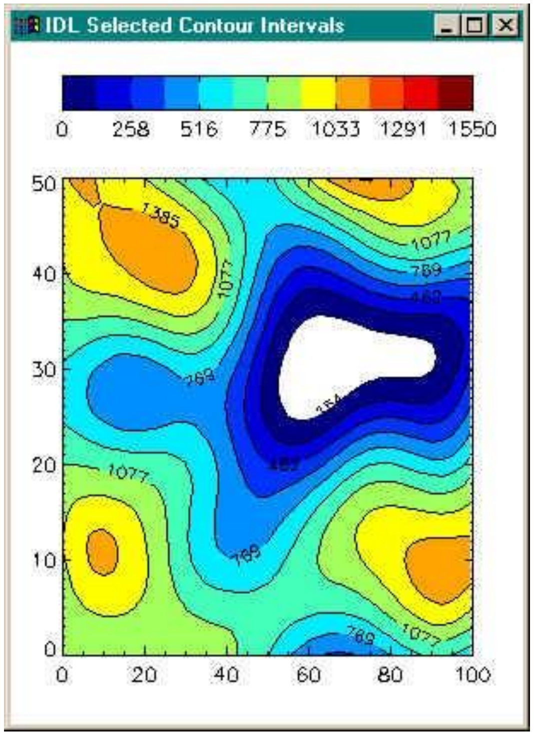


(4) Two contours pass through the cell

What are the possible bit strings for each of the cases here?

Put It All Together

- 2D Isocontouring algorithm:

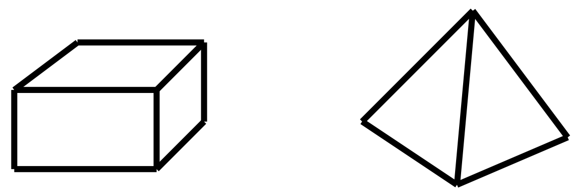


1. Process one cell at a time
2. Compare the values at 4 vertices with the contour value C and identify intersected edges
3. Linear interpolate along the intersected edges
4. Connects the interpolated points together

3D Isocontour (Isosurface)

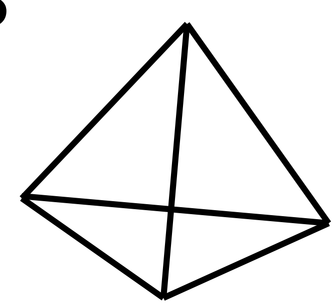
Isosurface Extraction

- Extend the 2D algorithm presented above to three dimensions
  - 3D cells



  - Linear interpolation along edges in active cells
    - Active cells: cells that intersect with the isosurface
  - Compute surface patches within each cell
    - Depend on which edges are intersected by the isosurface

Tetrahedral Cells

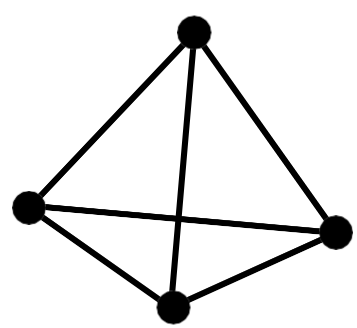


- Active cells: min < C < Max
- Mark cell vertices that are greater than the contour value as “+”, and smaller than the contour value as “-”
- Each cell has four vertices
  - Each cell can have a value greater (+) or smaller (-) than the contour value
  - A total of 2 x 2 x 2 x 2 = 16 combinations, but there are only three unique topological cases

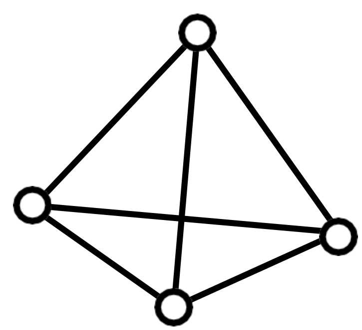
S05-03

Case 1: No Intersection (all outside or inside)

- Values at all cell vertices are greater or smaller than the contour values
  - If we call cell values greater than the contour value as ‘outside’ and smaller as ‘inside’, then all cell vertices are either completely inside or outside of the isosurface



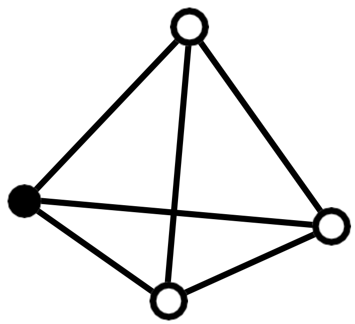
All Vertices Outside



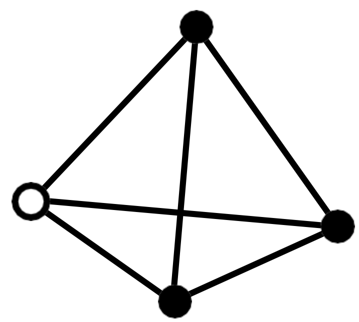
All Vertices Inside

Case 2: One Outside (or Inside)

- Only one of the vertices are outside or inside of the isosurface



One Outside

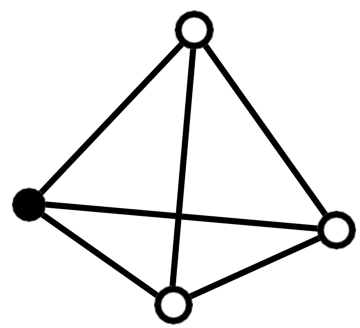


One Inside

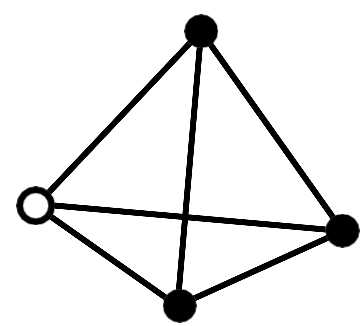
- Isosurface only intersects with edges that have ‘+’ and ‘–’ vertices at two ends

Case 2: One Outside (or Inside)

- Only one of the vertices are outside or inside of the isosurface



One Outside

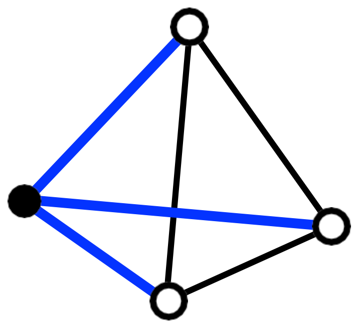


One Inside

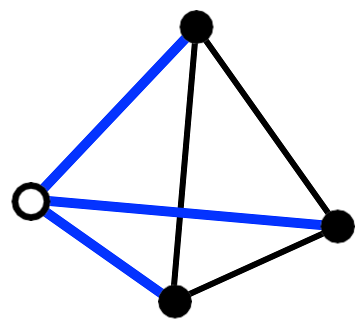
- Isosurface only intersects with edges that have '+' and '–' vertices at two ends

Case 2: One Outside (or Inside)

- Only one of the vertices are outside or inside of the isosurface



One Outside

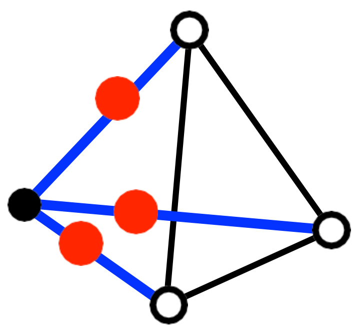


One Inside

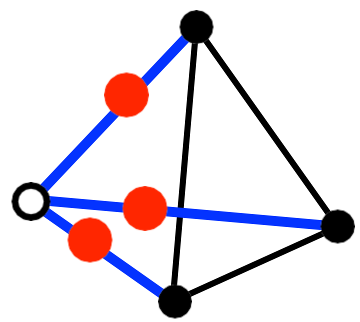
- Isosurface only intersects with edges that have ‘+’ and ‘–’ vertices at two ends

Case 2: Triangulation

- Compute the intersection points on the active edges



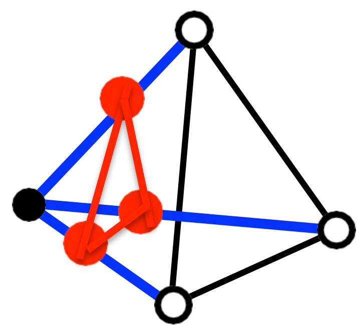
One Outside



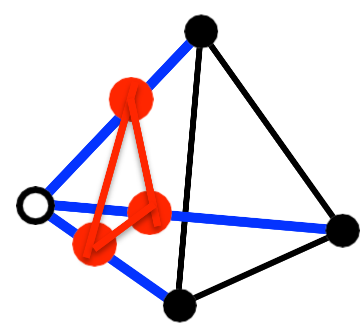
One Inside

Case 2: Triangulation

- Compute the intersection points on the active edges



One Outside

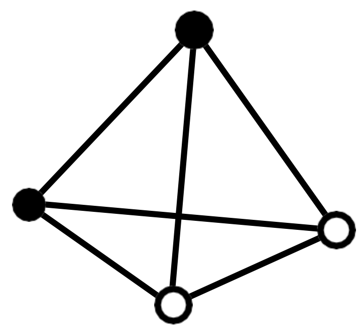


One Inside

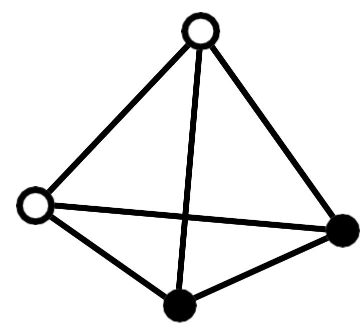
- Connect the intersection points into a single triangle

Case 3: Two Outside (or Inside)

- Two of the vertices are outside or inside of the isosurface



Two outside

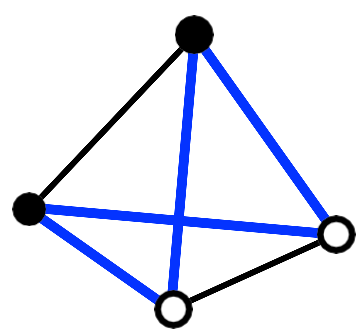


Two Inside

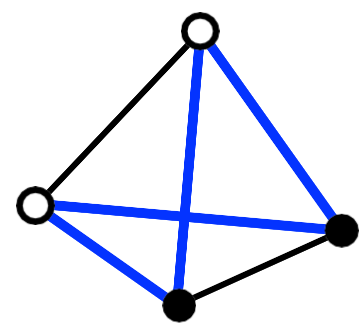
- Isosurface only intersects with edges that have ‘+’ and ‘–’ vertices at two ends

Case 3: Two Outside (or Inside)

- Two of the vertices are outside or inside of the isosurface



Two outside

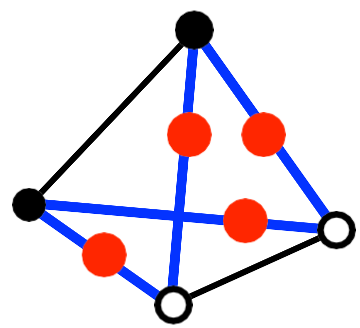


Two inside

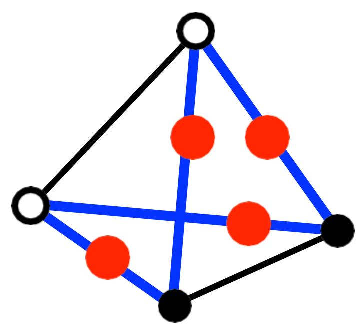
- Isosurface only intersects with edges that have ‘+’ and ‘–’ vertices at two ends

Case 3: Two Outside (or Inside)

- Two of the vertices are outside or inside of the isosurface



Two outside

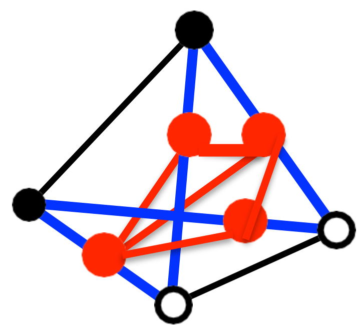


Two inside

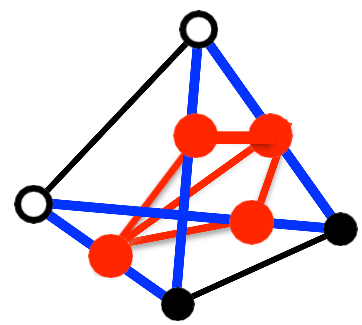
- Isosurface only intersects with edges that have '+' and '–' vertices at two ends

Case 3: Two Outside (or Inside)

- Two of the vertices are outside or inside of the isosurface



TWO Outside



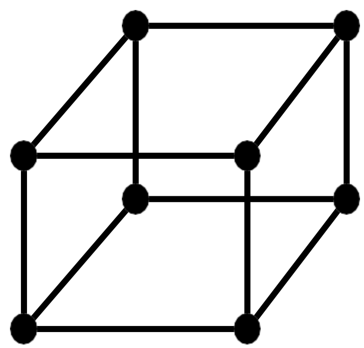
Two Inside

- Isosurface only intersects with edges that have ‘+’ and ‘–’ vertices at two ends

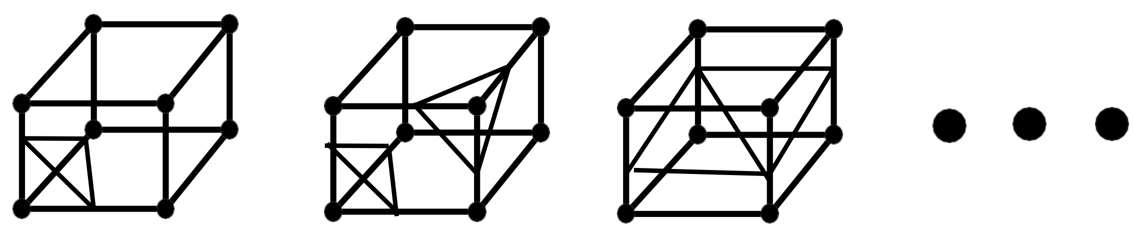
Put It All Together

1. Iterate through all tetrahedral cells
   a) Compare the values at the four corners of each cell
   b) Determine the edges that intersect with the isosurface if any
   c) Compute the surface-edge intersection points using linear interpolation
   d) Triangulate the intersection points based on the cases discussed above

Rectangular Cells



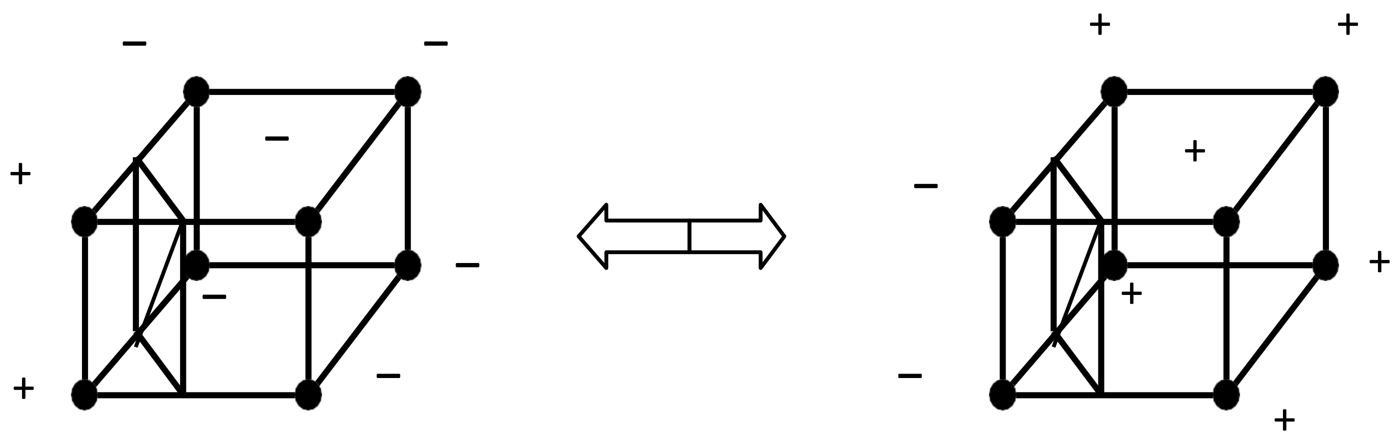
With 8 vertices in a cell, each having a value greater or smaller than the contour value, there can be $2^8 = 256$ possible cases



But the total number of unique topological cases is much smaller than 256

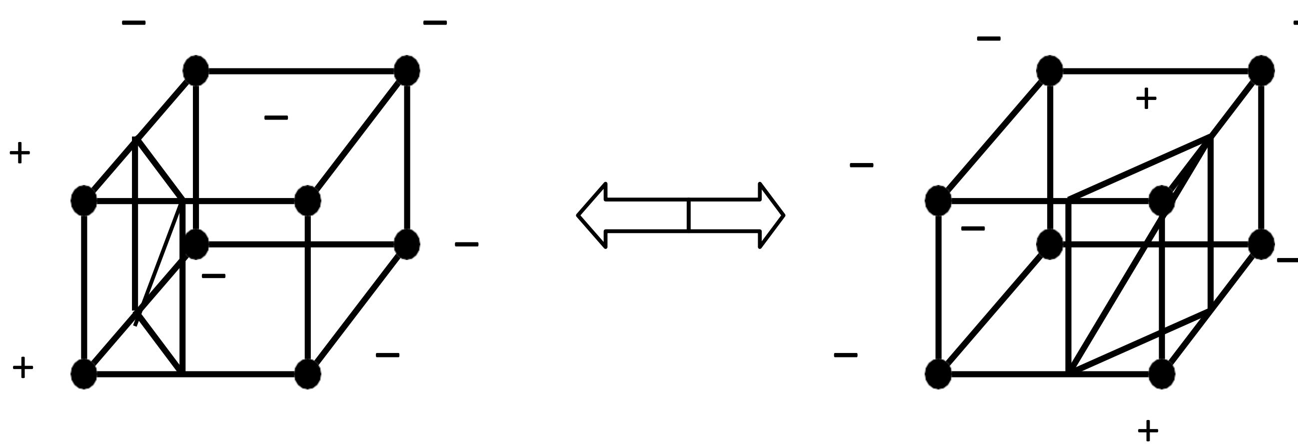
Example of Case Reduction

Value Symmetry



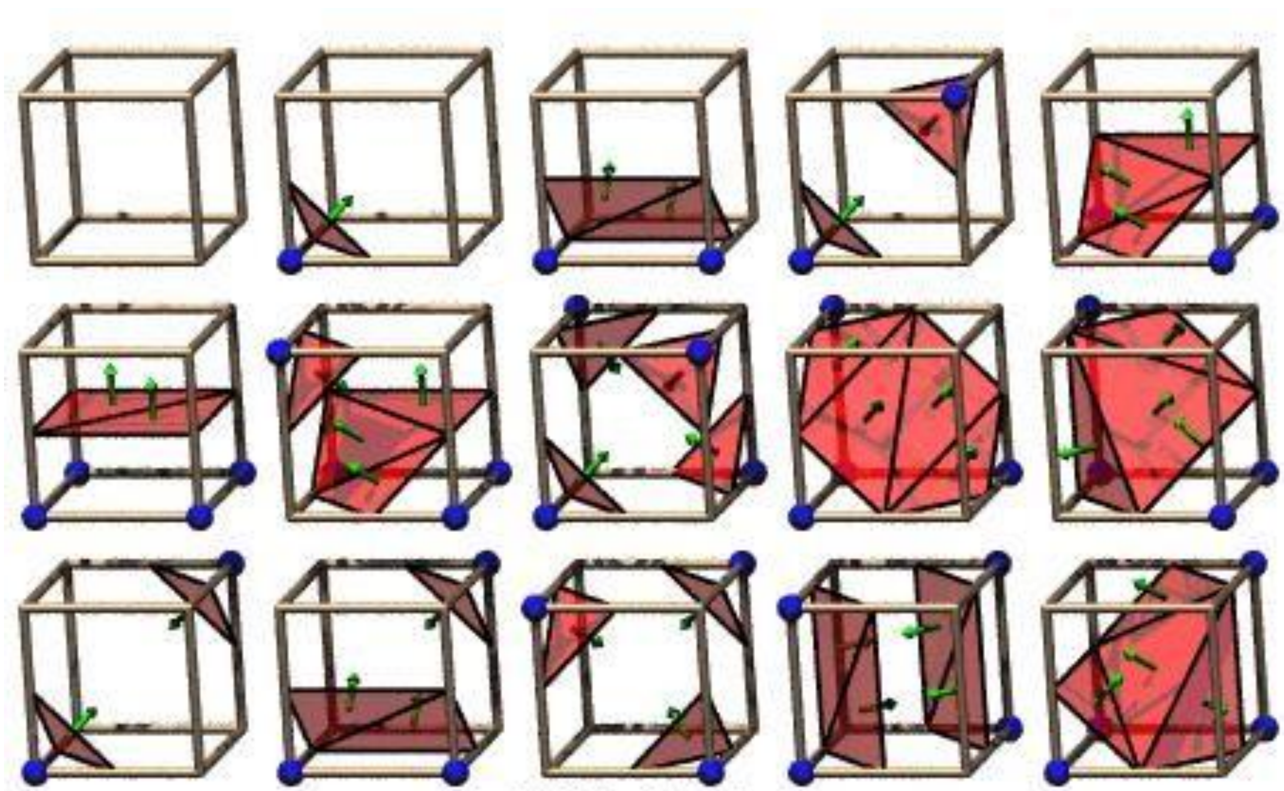
Example of Case Reduction

Rotation Symmetry



By inspection, we can reduce the number of cases from 256 to 15

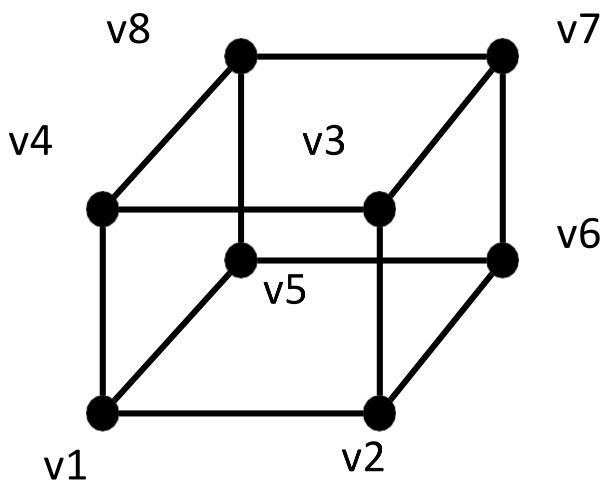
Isosurface Cases



The 15 Cube Combinations

The Marching Cubes Algorithm

- By Lorensen and Cline in 1987



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

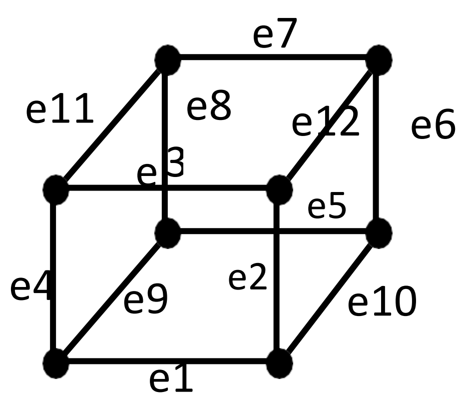
Each cell has an index mapped to a value ranged [0,255]

Index =

v8	v7	v6	v5	v4	v3	v2	v1

Marching Cubes Table Lookup

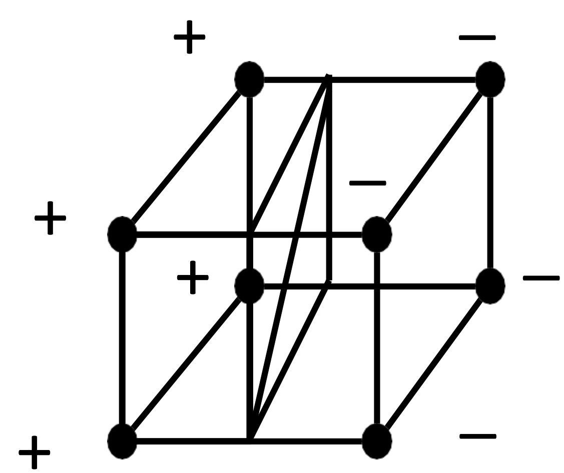
- Based on the values at the vertices, map the cell to one of the 15 cases
- Perform a table lookup to see what edges have intersections



Index	intersection edges
0	e1, e3, e5
1	…
2	
3	
	• • •
14	

Compute Intersections

- Perform linear interpolation to compute the intersection points at the edges
- Connect the points by looking up the marching cubes table



Marching the Cubes

- Sequentially scan through the cells – row by row, layer by layer
- You can re-use the intersection points for neighboring cells