11. Procedural Methods

Overview

- Particle Systems
- Marching Squares
- Reading: ANG. Ch.11

Particle Systems

Introduction

- Most important of procedural methods
- Used to model
    - Natural phenomena
        - Clouds
        - Terrain
        - Plants
    - Crowd Scenes
    - Real physical processes

Newtonian Particle

- Particle system is a set of particles
- Each particle is an ideal point mass
- Six degrees of freedom
  - Position
  - Velocity
- Each particle obeys Newtons' law

$$f = ma$$

Particle Equations

$\mathbf{p}_i = (x_i, y_i\ z_i)$

$\mathbf{v}_i = d\mathbf{p}_i /dt = \mathbf{p}_i' = (dx_i /dt, dy_i /dt , z_i /dt)$

$m\ \mathbf{v}_i' = \mathbf{f}_i$

Hard part is defining force vector

Force Vector

- Independent Particles
  - Gravity
  - Wind forces
  - O(n) calulation
- Coupled Particles O(n)
  - Meshes
  - Spring-Mass Systems
- Coupled Particles $O(n^2)$
  - Attractive and repulsive forces

Solution of Particle Systems

```
float time, delta state[6n], force[3n];
state = initial_state();
for(time = t0; time<final_time, time+=delta) {
    force =  force_function(state, time);
    state = ode(force, state, time, delta);
    render(state, time)
}
```

Simple Forces

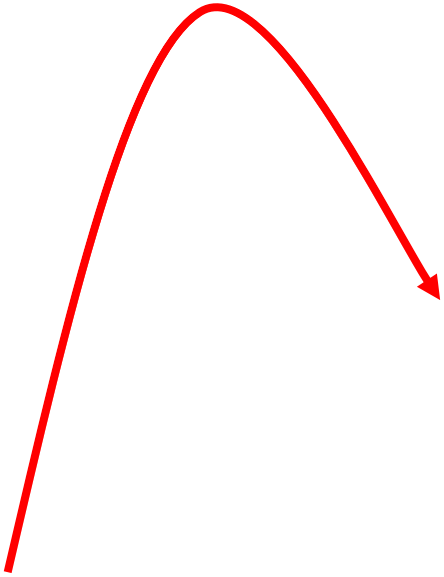
- Consider force on particle i

$$\mathbf{f}_i = \mathbf{f}_i(\mathbf{p}_i, \mathbf{v}_i)$$

- Gravity $\mathbf{f}_i = \mathbf{g}$

$$\mathbf{g}_i = (0, -g, 0)$$

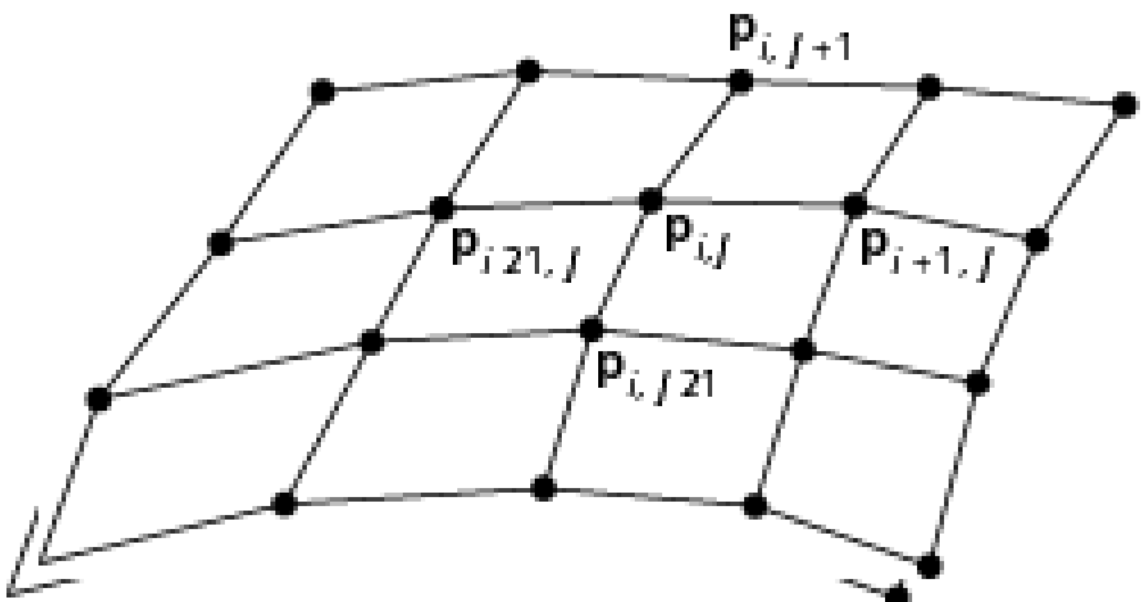
- Wind forces
- Drag



$\mathbf{p}_i(t_0), \mathbf{v}_i(t_0)$

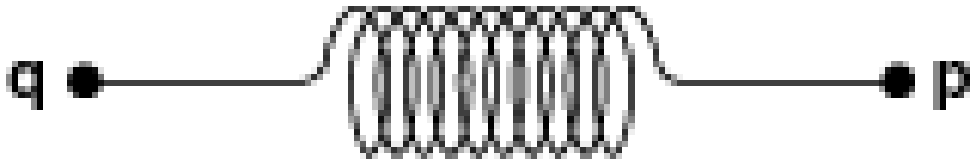
Meshes

- Connect each particle to its closest neighbors
  - O(n) force calculation
- Use spring-mass system



Spring Forces

- Assume each particle has unit mass and is connected to its neighbor(s) by a spring
- Hooke's law: force proportional to distance ($d = \|\mathbf{p} - \mathbf{q}\|$) between the points



Hooke's Law

- Let s be the distance when there is no force

$$\mathbf{f} = -k_s(|\mathbf{d}| - s)\,\mathbf{d}/|\mathbf{d}|$$

k_s is the spring constant

$\mathbf{d}/|\mathbf{d}|$ is a unit vector pointed from $\mathbf{p}$ to $\mathbf{q}$

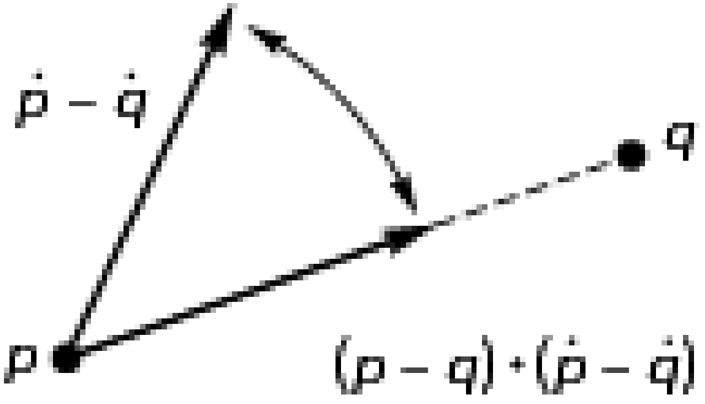
- Each interior point in mesh has four forces applied to it

Spring Damping

- A pure spring-mass will oscillate forever
- Must add a damping term

$$\mathbf{f} = -(k_s(|\mathbf{d}| - s) + k_d\, \dot{\mathbf{d}}\cdot\mathbf{d}/|\mathbf{d}|)\mathbf{d}/|\mathbf{d}|$$

- Must project velocity



Attraction and Repulsion

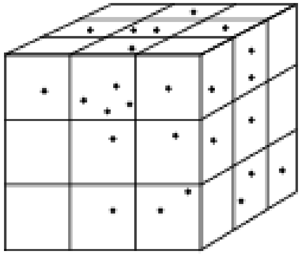
- Inverse square law

$$\mathbf{f} = -k_r\mathbf{d}/|\mathbf{d}|^3$$

- General case requires $O(n^2)$ calculation
- In most problems, the drop off is such that not many particles contribute to the forces on any given particle
- Sorting problem: is it O(n log n)?

Boxes

- Spatial subdivision technique
- Divide space into boxes
- Particle can only interact with particles in its box or the neighboring boxes
- Must update which box a particle belongs to after each time step



Linked Lists

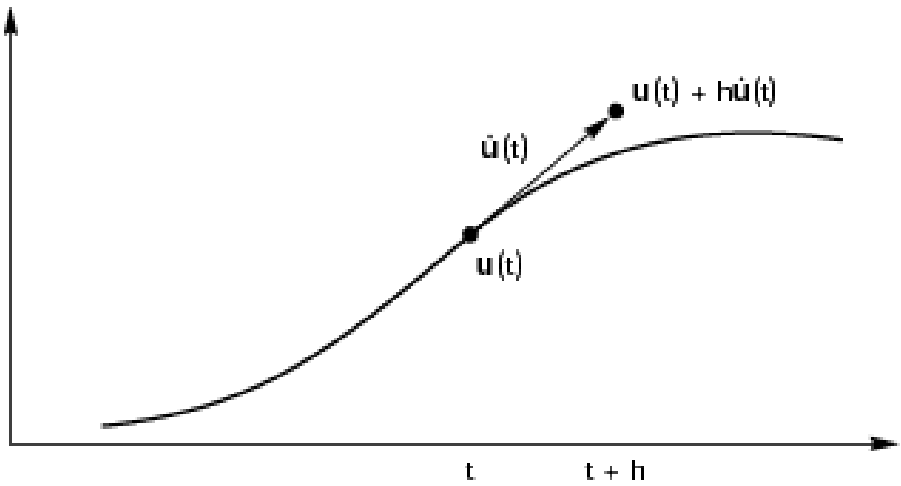
- Each particle maintains a linked list of its neighbors
- Update data structure at each time step
- Must amortize cost of building the data structures initially

Particle Field Calculations

- Consider simple gravity
- We don’t compute forces due to sun, moon, and other large bodies
- Rather we use the gravitational field
- Usually we can group particles into equivalent point masses

Solution of ODEs

- Particle system has 6n ordinary differential equations
- Write set as $d\mathbf{u}/dt = g(\mathbf{u},t)$
- Solve by approximations using Taylor's Thm



Euler's Method

$$\mathbf{u}(t + h) \approx \mathbf{u}(t) + h\, d\mathbf{u}/dt = \mathbf{u}(t) + h\mathbf{g}(\mathbf{u}, t)$$

Per step error is $O(h^2)$

Require one force evaluation per time step

Problem is numerical instability

depends on step size

Improved Euler

$$\mathbf{u}(t + h) \approx \mathbf{u}(t) + h/2(\mathbf{g}(\mathbf{u}, t) + \mathbf{g}(\mathbf{u}, t+h))$$

Per step error is $O(h^3)$

Also allows for larger step sizes

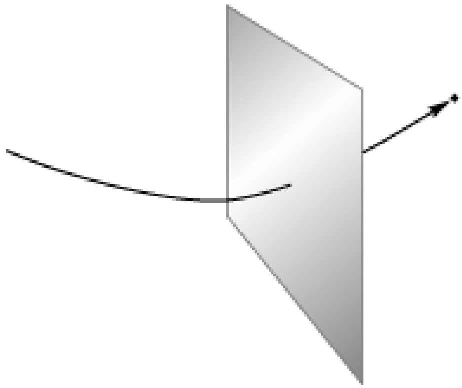
But requires two function evaluations per step

Also known as Runge-Kutta method of order 2

Contraints

- Easy in computer graphics to ignore physical reality
- Surfaces are virtual
- Must detect collisions separately if we want exact solution
- Can approximate with

repulsive forces



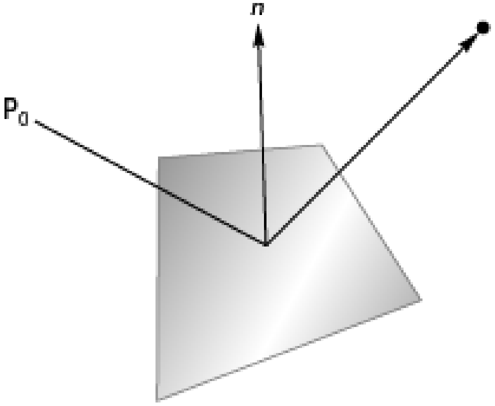
Collisions

Once we detect a collision, we can calculate new path

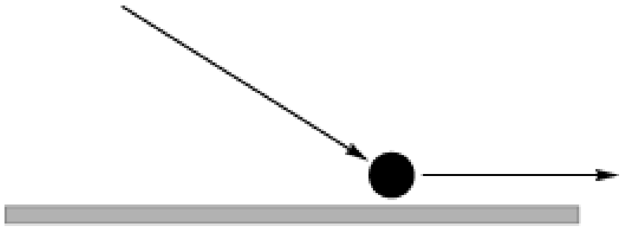
Use coefficient of resititution

Reflect vertical component

May have to use partial time step



Contact Forces



Marching Squares

Objectives

- Nontrivial two-dimensional application
- Important method for
  - Contour plots
  - Implicit function visualization
- Extends to important method for volume visualization
- This lecture is optional
- Material not needed to continue to Chapter 3

Displaying Implicit Functions

- Consider the implicit function

$$g(x,y)=0$$

- Given an x, we cannot in general find a corresponding y
- Given an x and a y, we can test if they are on the curve

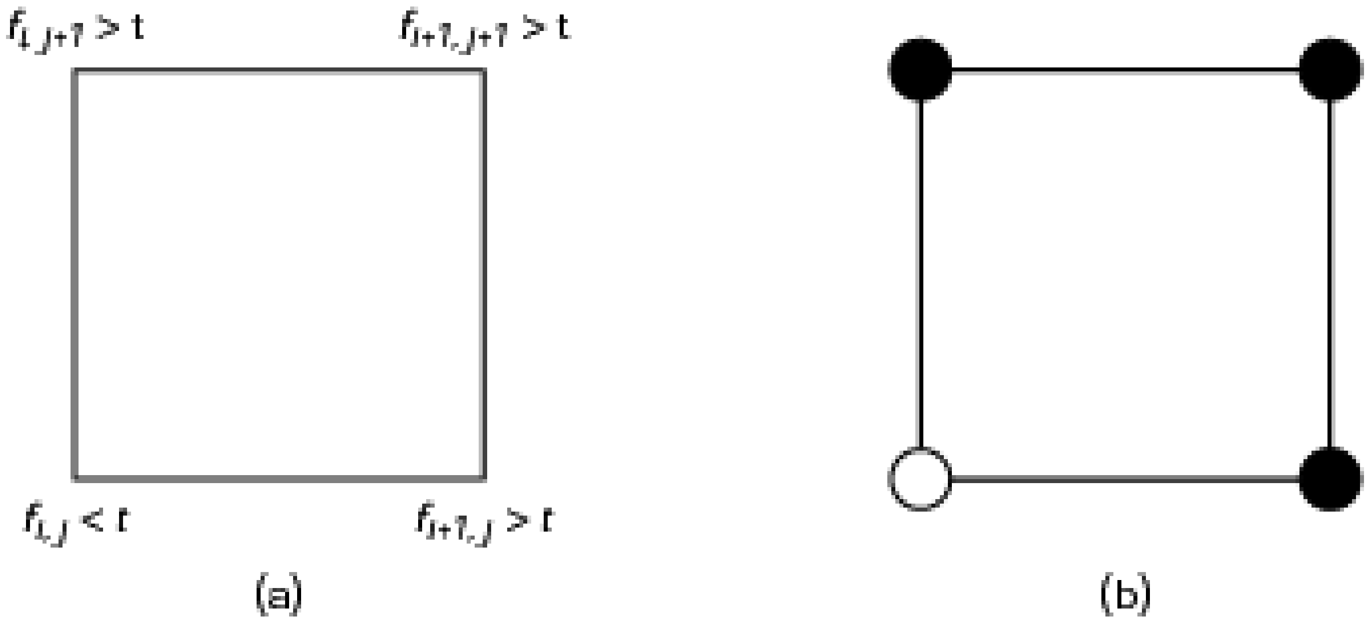
Height Fields and Contours

- In many applications, we have the heights given by a function of the form $z=f(x,y)$
- To find all the points that have a given height c, we have to solve the implicit equation $g(x,y)=f(x,y)-c=0$
- Such a function determines the isosurfaces or contours of f for the isosurface value c

Marching Squares

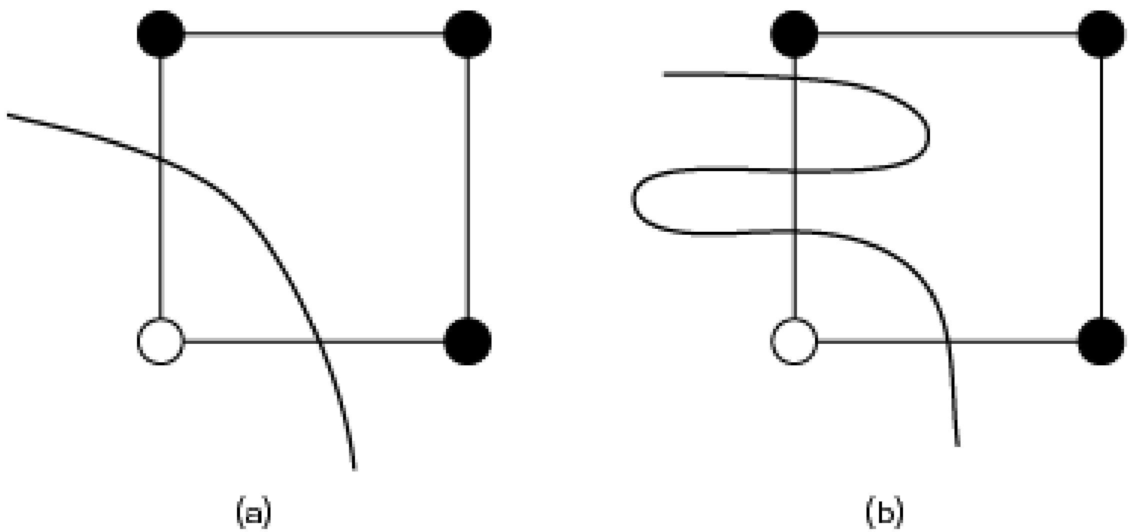
- Displays isocurves or contours for functions $f(x,y) = t$
- Sample $f(x,y)$ on a regular grid yielding samples $\{f_{ij}(x,y)\}$
- These samples can be greater than, less than, or equal to t
- Consider four samples $f_{ij}(x,y)$, $f_{i+1,j}(x,y)$, $f_{i+1,j+1}(x,y)$, $f_{i,j+1}(x,y)$
- These samples correspond to the corners of a cell
- Color the corners by whether than exceed or are less than the contour value t

Cells and Coloring

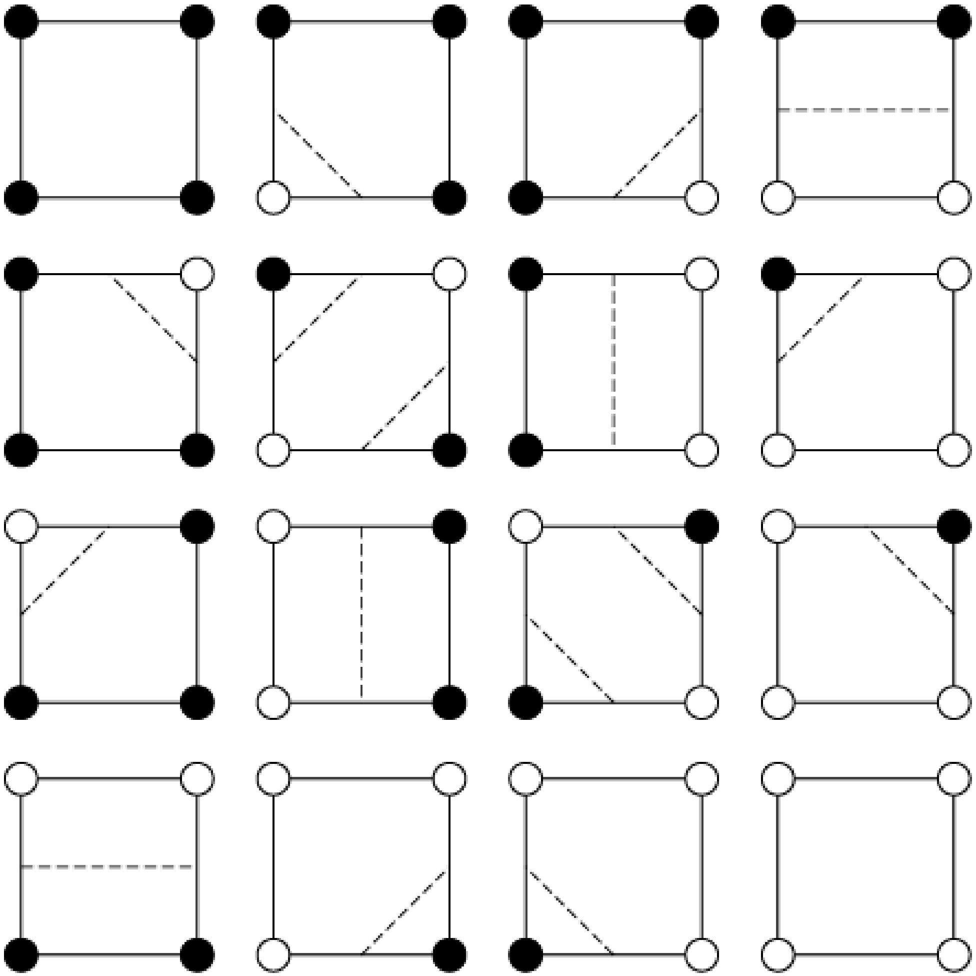


Occum's Razor

- Contour must intersect edge between a black and white vertex an odd number of times
- Pick simplest interpretation: one crossing

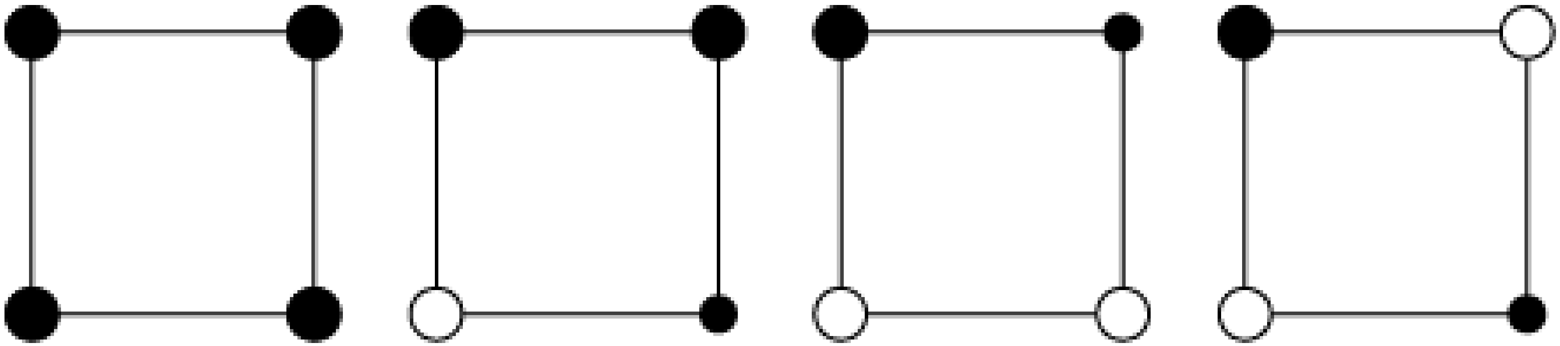


16 Cases



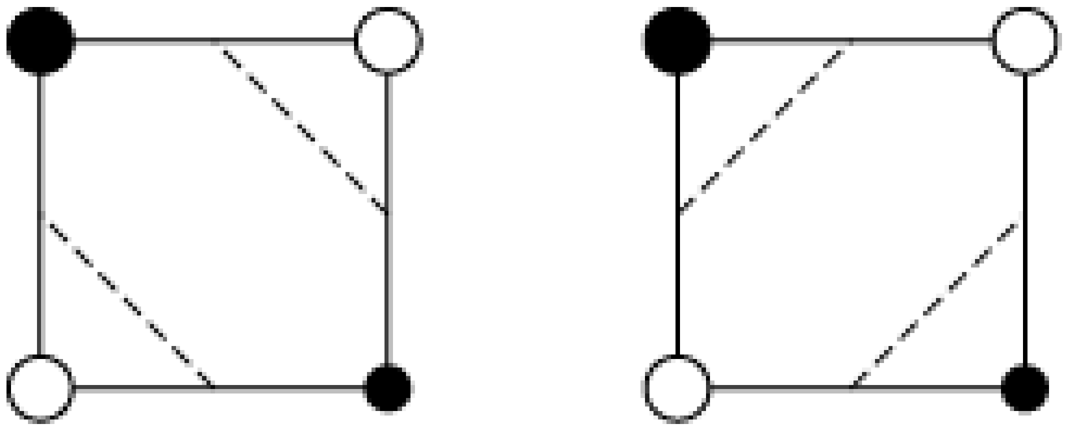
Unique Cases

- Taking out rotational and color swapping symmetries leaves four unique cases
- First three have a simple interpretation



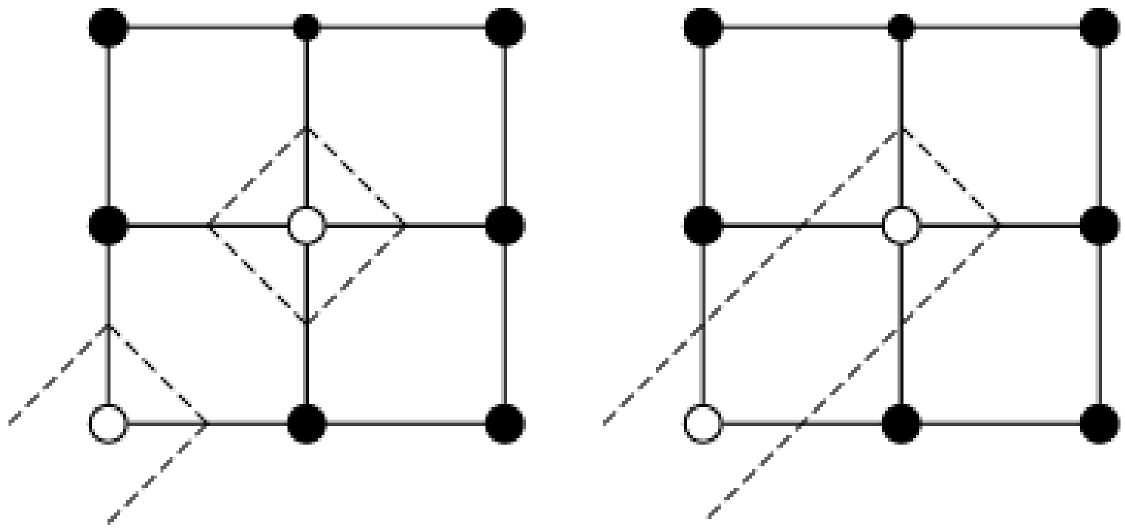
Ambiguity Problem

- Diagonally opposite cases have two equally simple possible interpretations

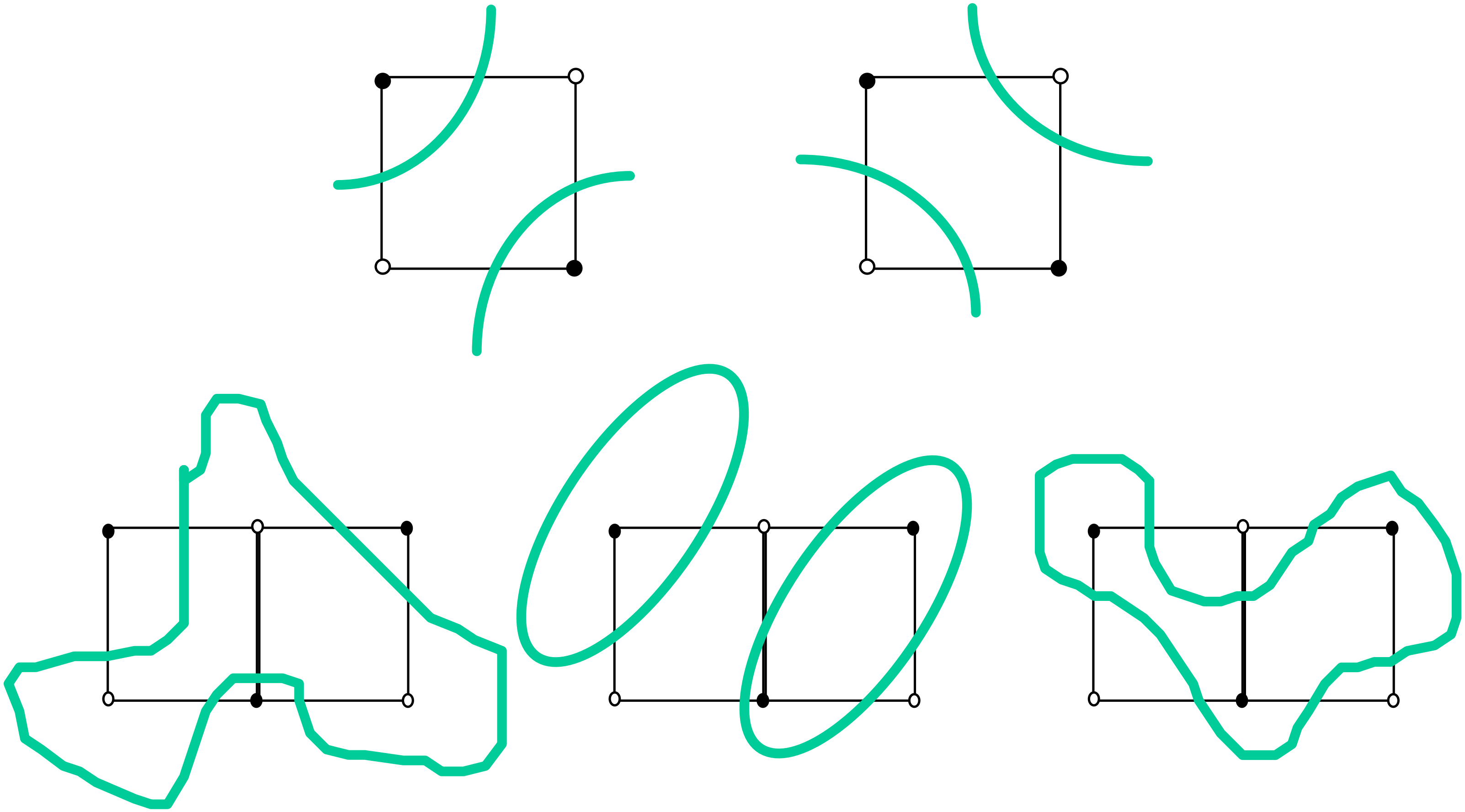


Ambiguity Example

- Two different possibilities below
- More possibilities on next slide



Ambiguity Problem

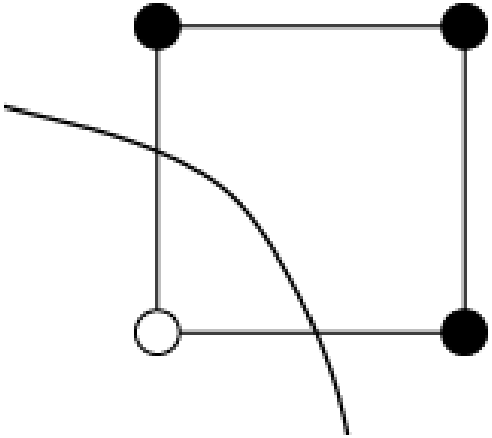


Is Problem Resolvable?

- Problem is a sampling problem
  - Not enough samples to know the local detail
  - No solution in a mathematical sense without extra information
- More of a problem with volume extension (marching cubes) where selecting “wrong” interpretation can leave a hole in a surface
- Multiple methods in literature to give better appearance
  - Supersampling
  - Look at larger area before deciding

Interpolating Edges

- We can compute where contour intersects edge in multiple ways
  - Halfway between vertics
  - Interpolated based on difference between contour value and value at vertices

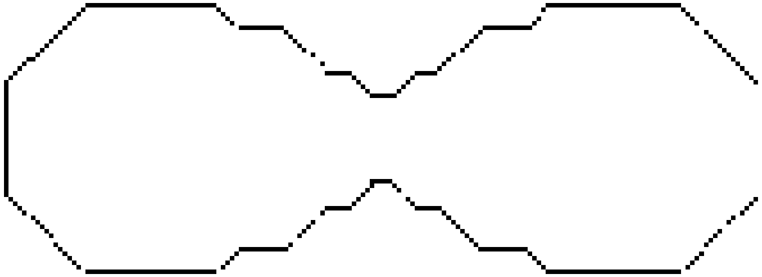


Example: Oval of Cassini

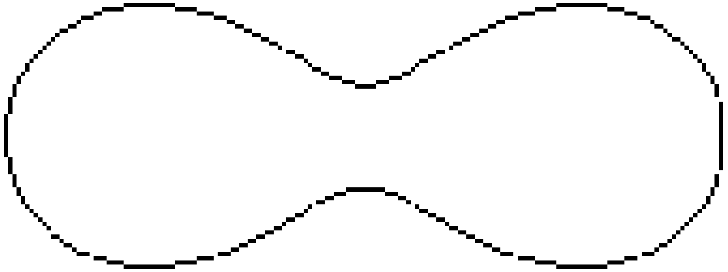
$$f(x,y)=(x^2+y^2+a^2)^2-4a^2x^2-b^4$$

Depending on a and b we can have 0, 1, or 2 curves

midpoint intersections

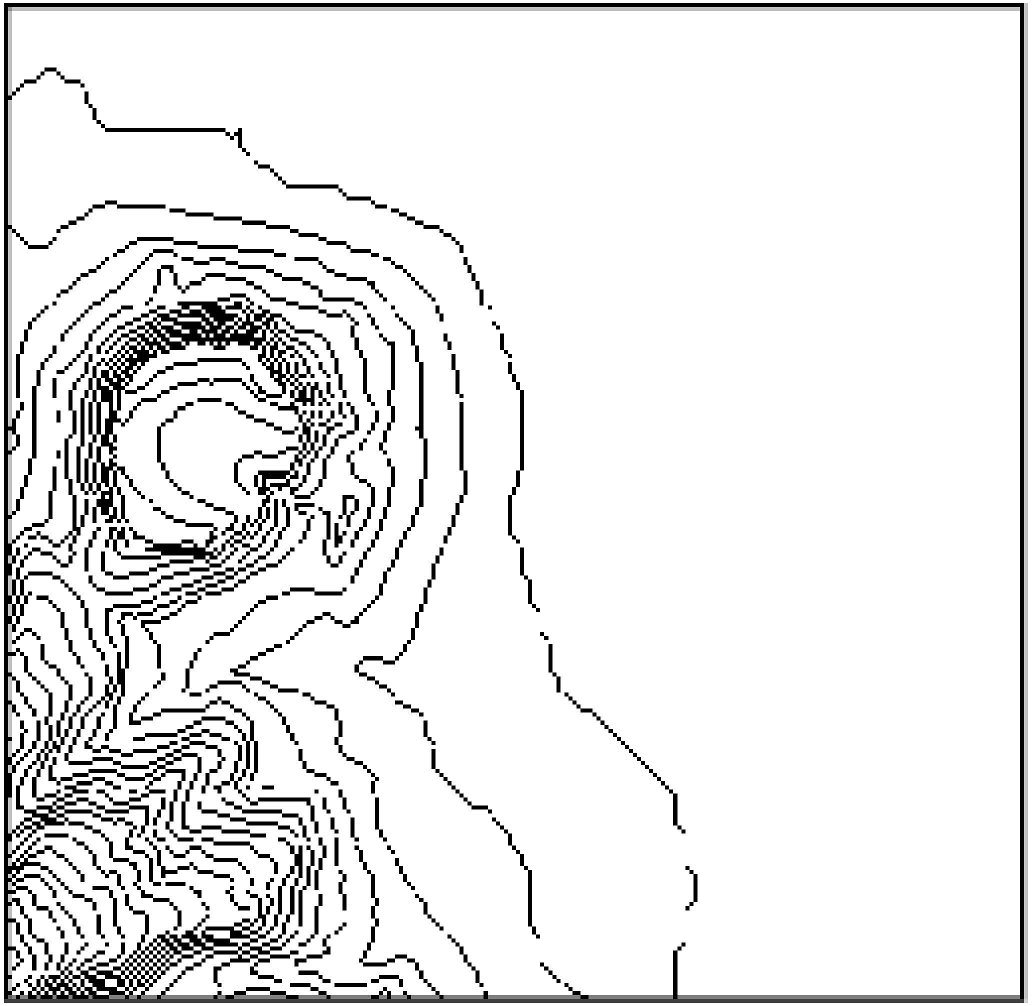


interpolating intersections



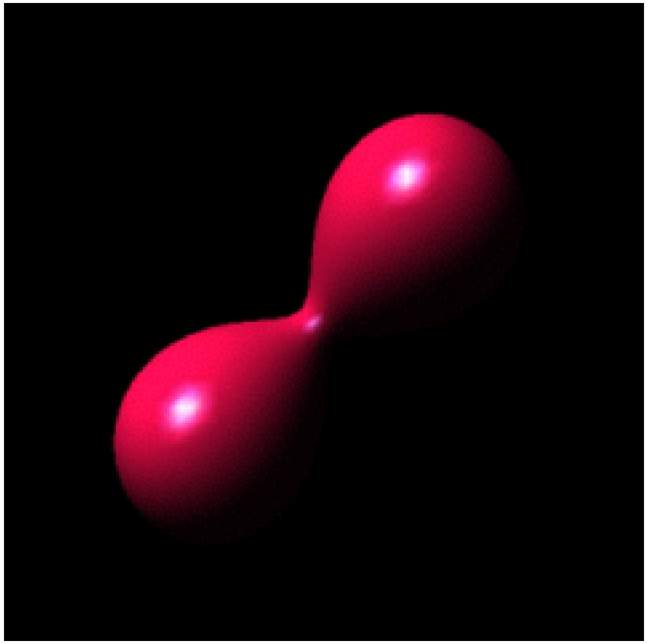
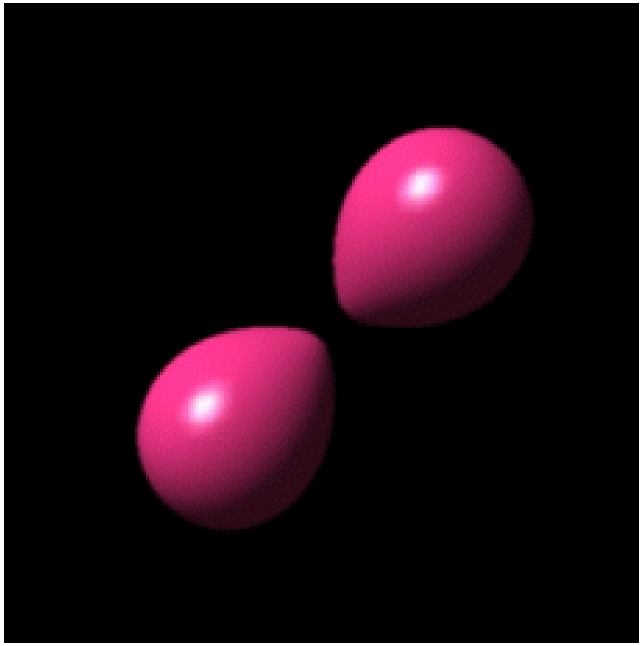
Contour Map

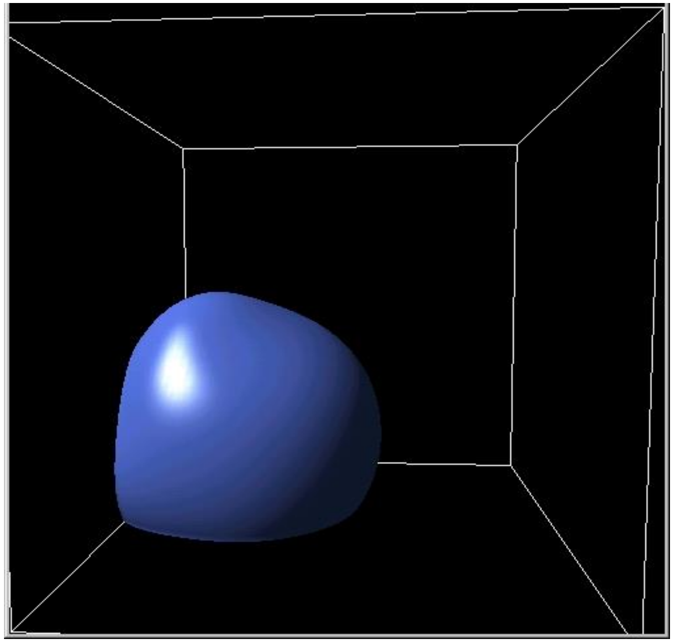
- Diamond Head, Oahu Hawaii
- Shows contours for many contour values

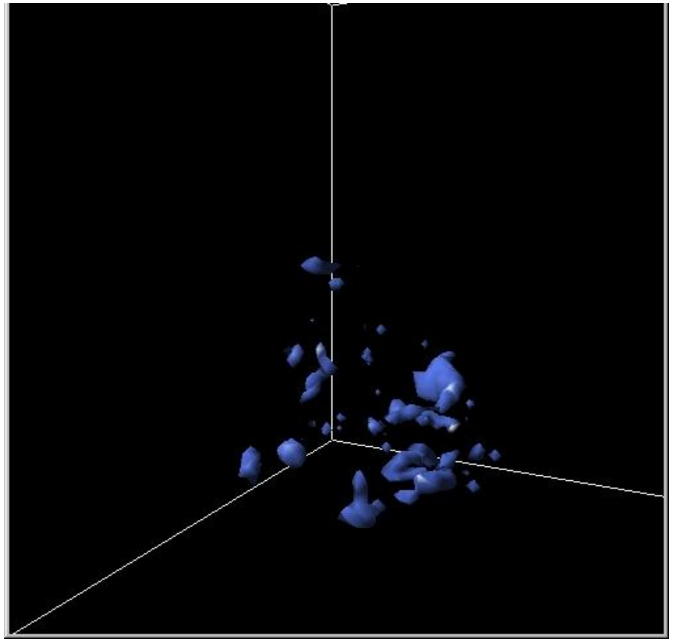


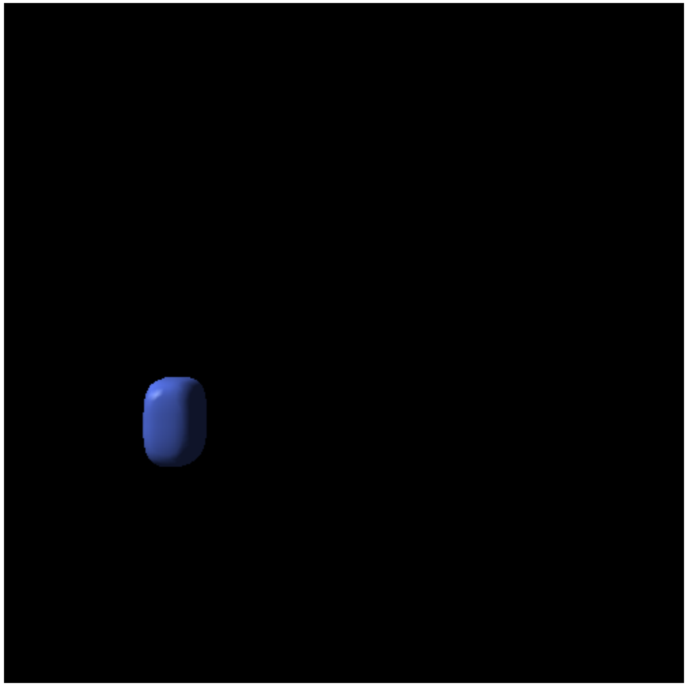
Marching Cubes

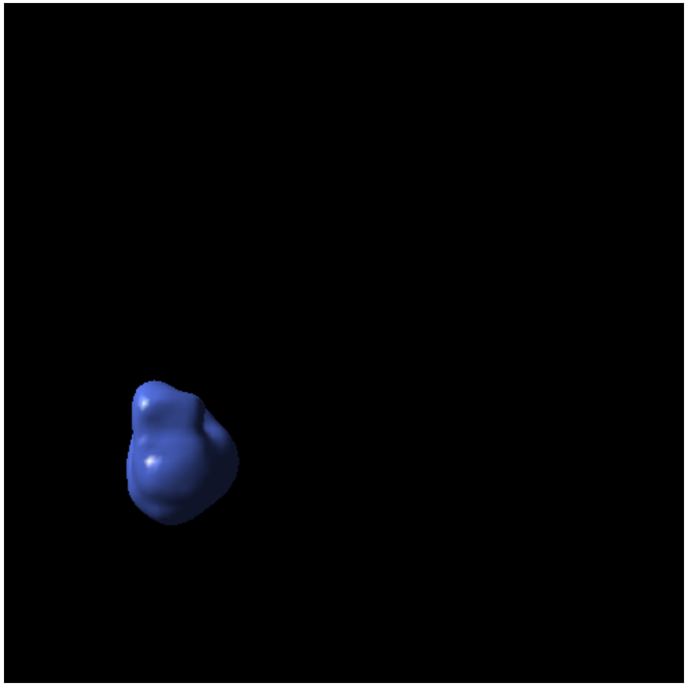
- Isosurface: solution of g(x,y,z)=c
- Same argument to derive method but use cubic cell (8 vertices, 256 colorings)
- Standard method of volume visualization
- Suggested by Lorensen and Kline before marching squares

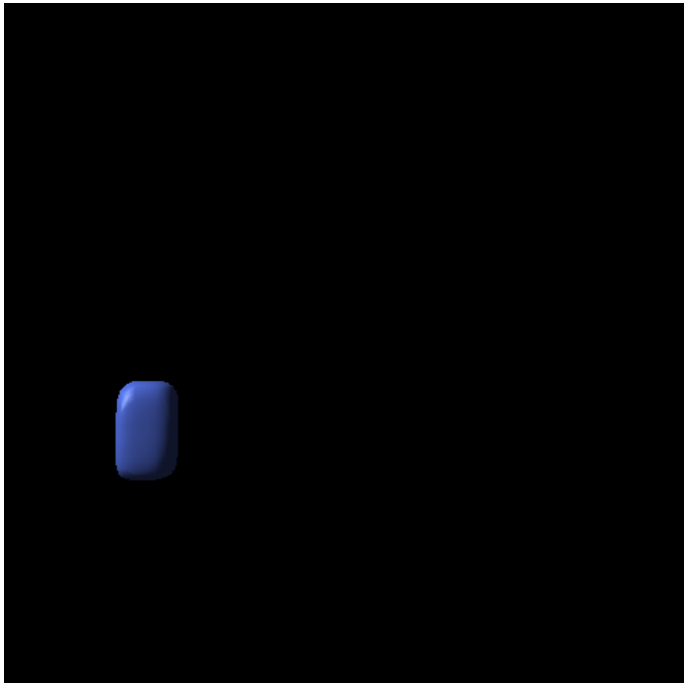
 

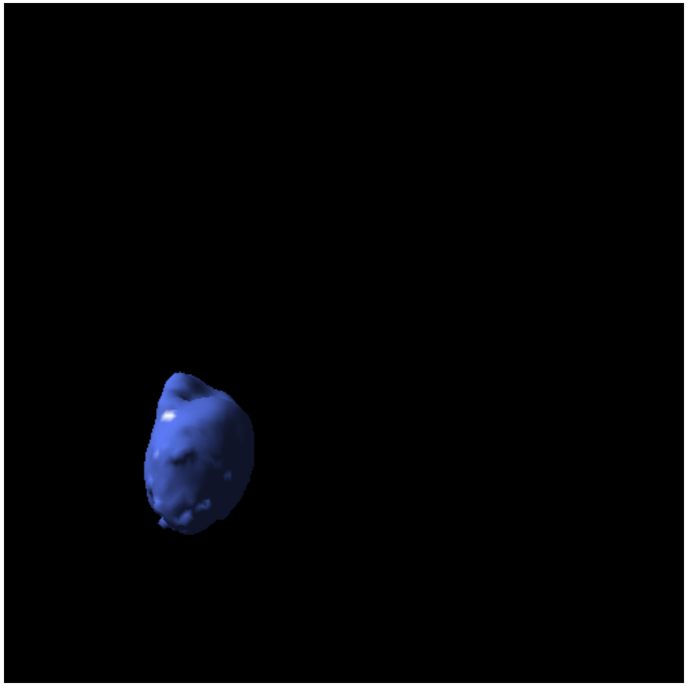


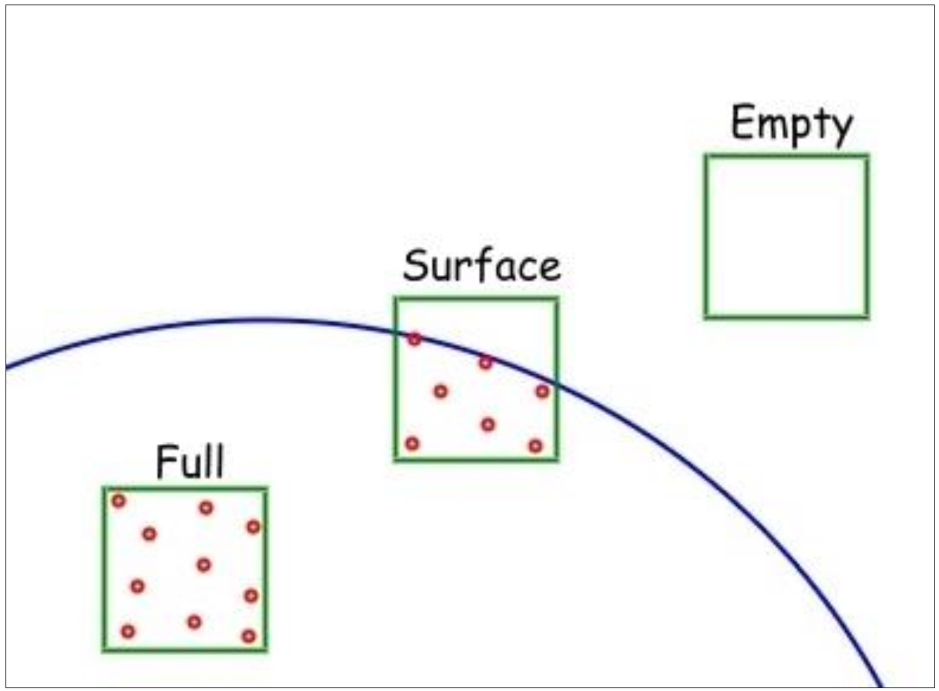




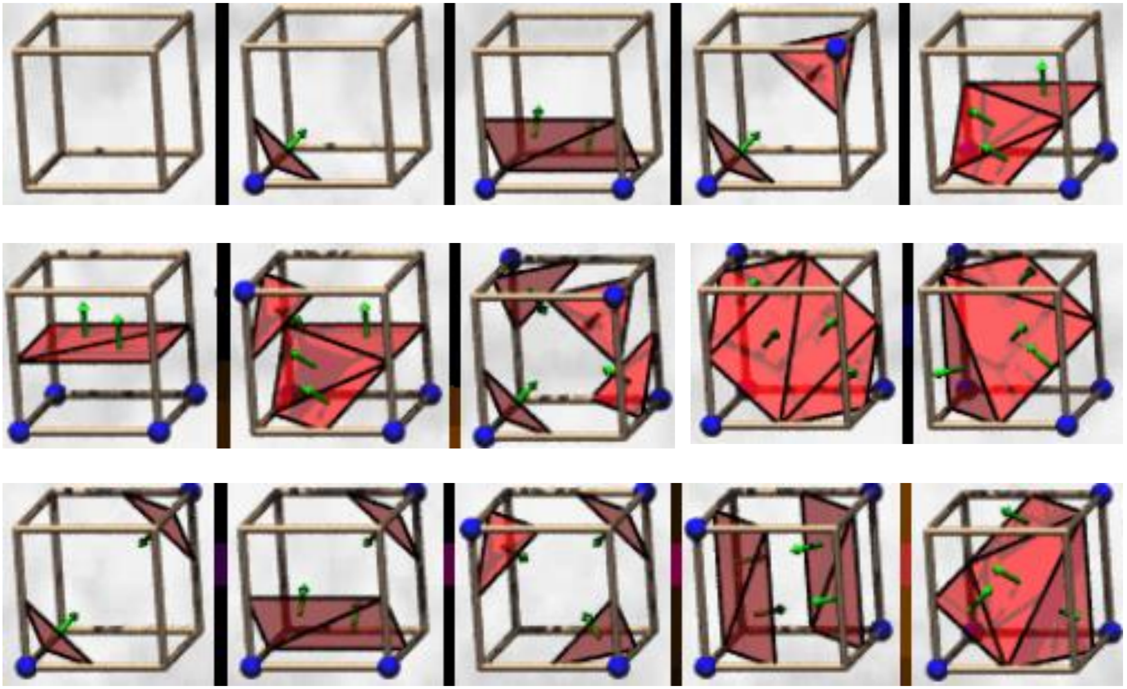


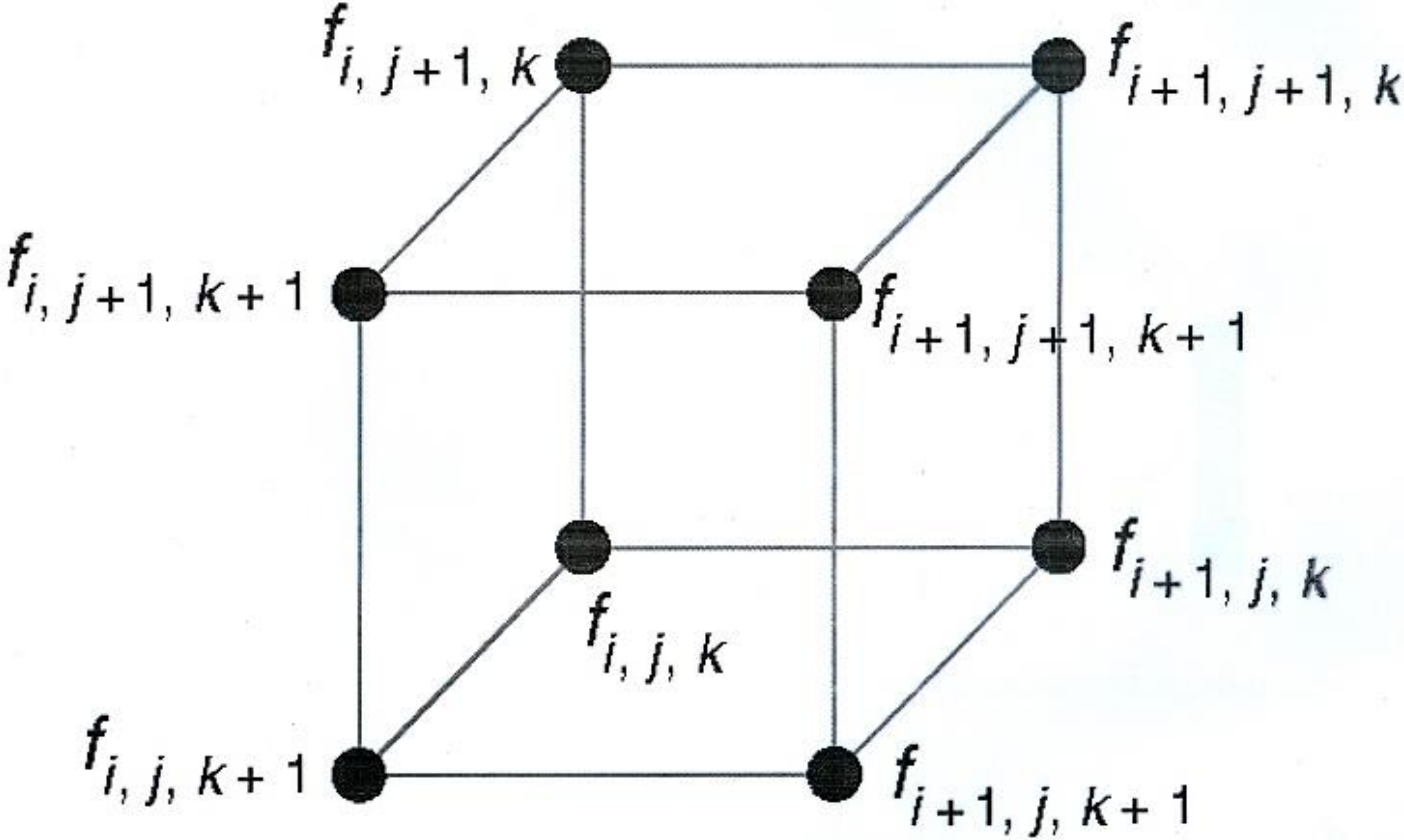


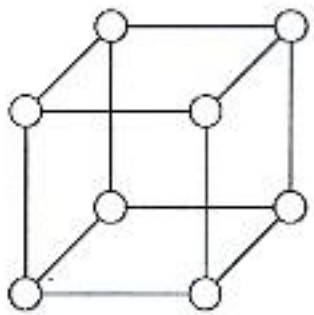



Empty
Surface
Full

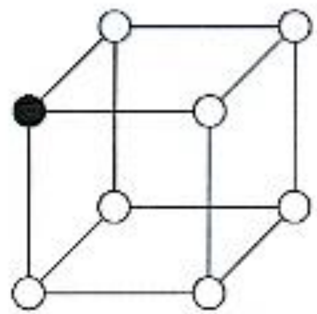
Marching-Cubes Methods



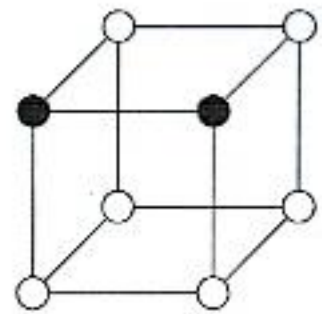

$f_{i,\, j+1,\, k}$
$f_{i+1,\, j+1,\, k}$
$f_{i,\, j+1,\, k+1}$
$f_{i+1,\, j+1,\, k+1}$
$f_{i,\, j,\, k}$
$f_{i+1,\, j,\, k}$
$f_{i,\, j,\, k+1}$
$f_{i+1,\, j,\, k+1}$



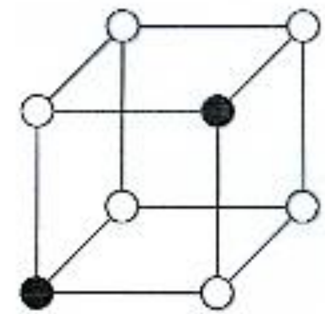
0



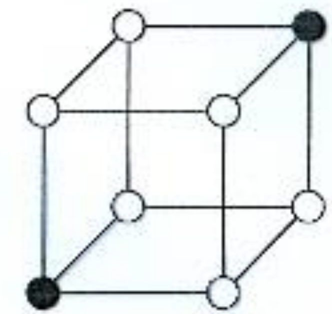
1



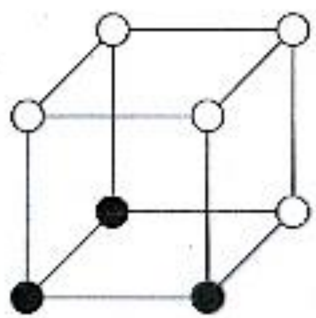
2



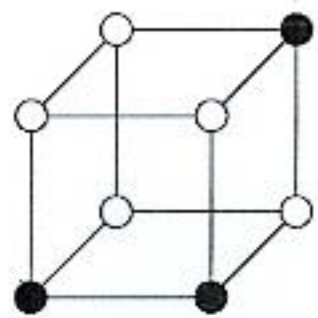
3



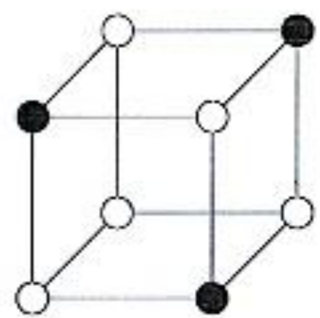
4



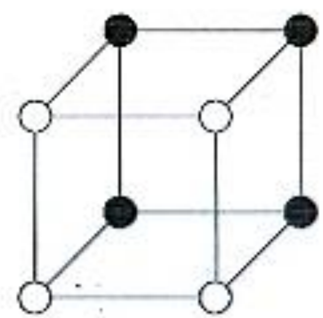
5



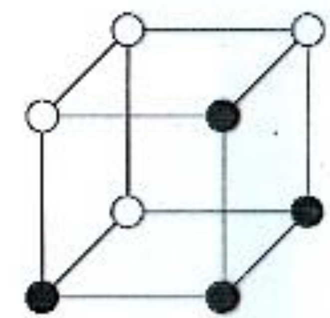
6



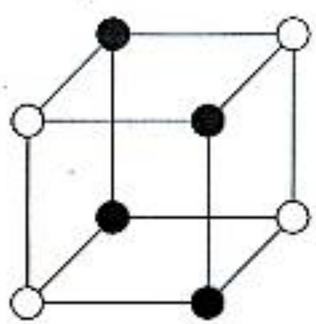
7



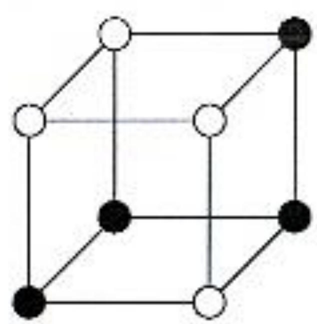
8



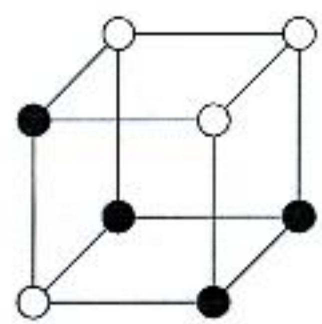
9



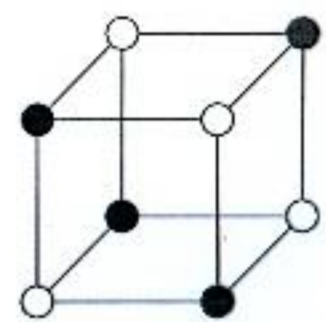
10



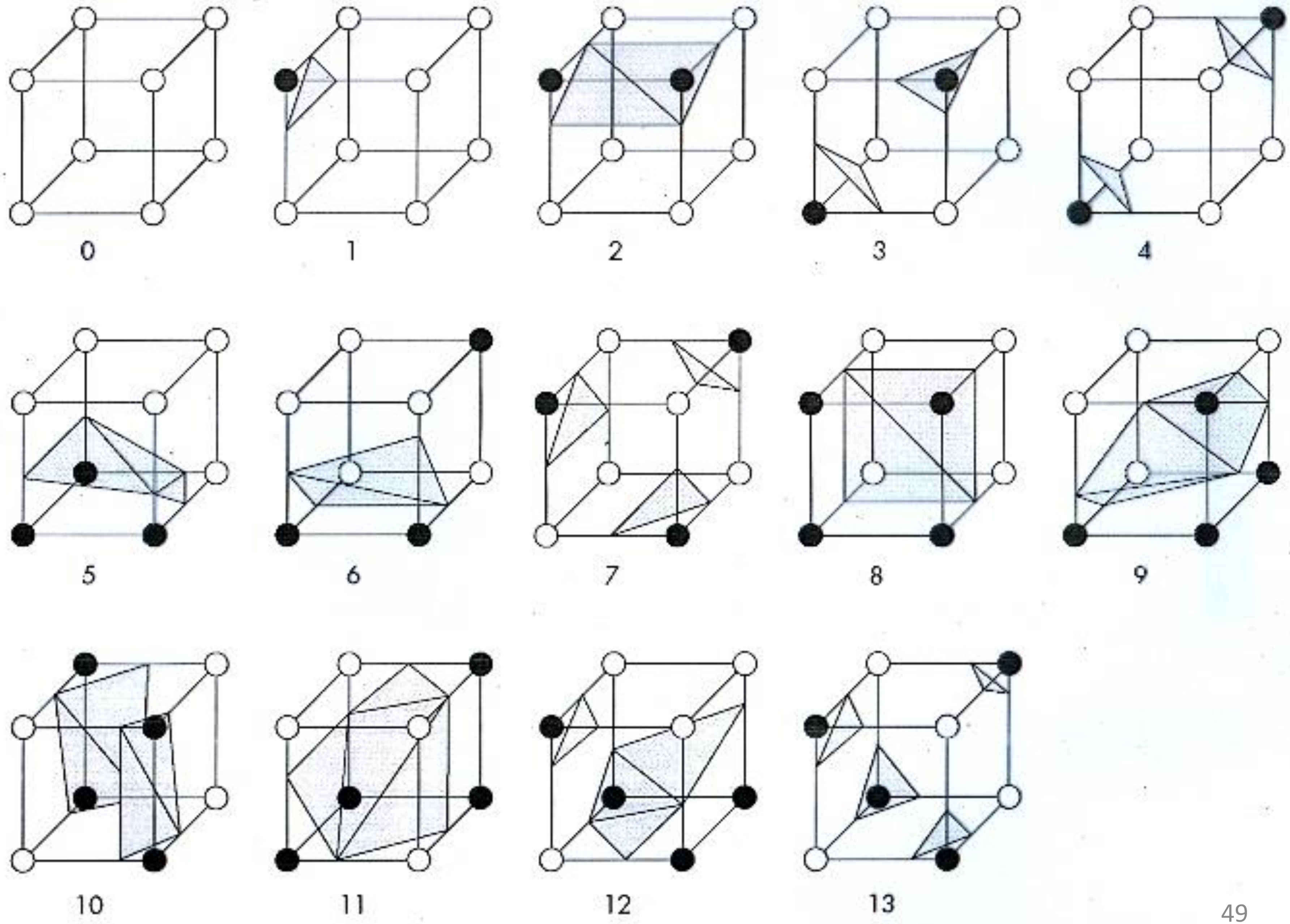
11

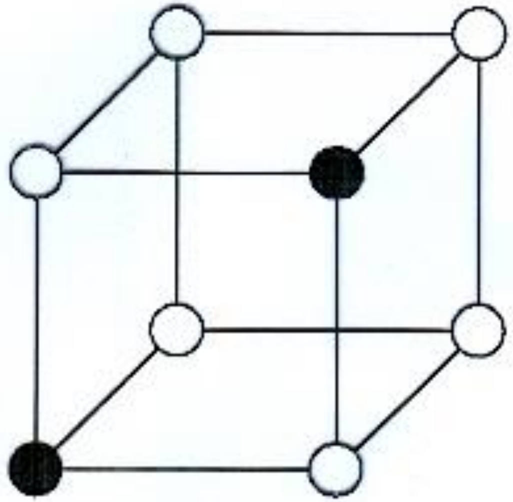


12

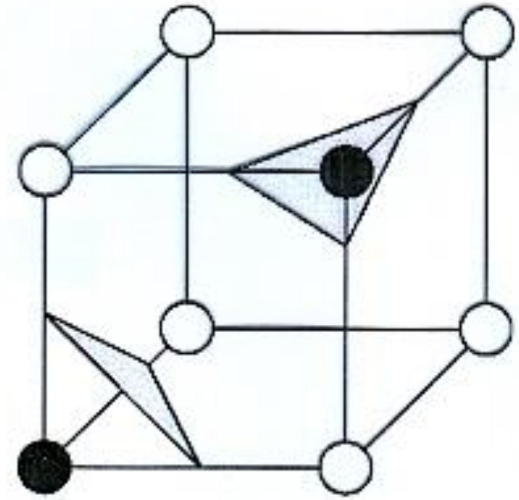


13

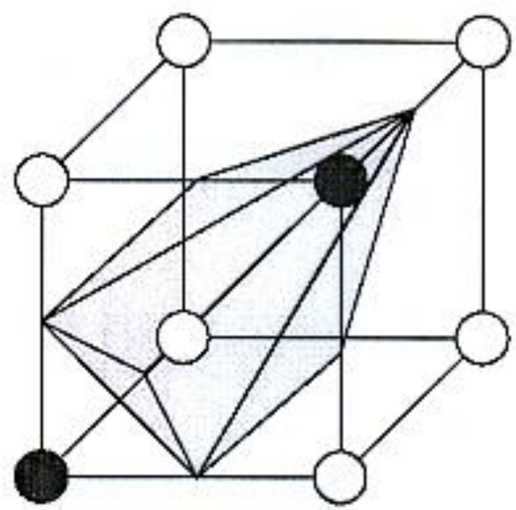

0
1
2
3
4
5
6
7
8
9
10
11
12
13



(a)



(b)



(c)