Information Visualization

W11: Scalar Data Visualization 1 - Isosurface Extraction

Graduation School of System Informatics

Department of Computational Science

Naohisa Sakamoto, Akira Kageyama

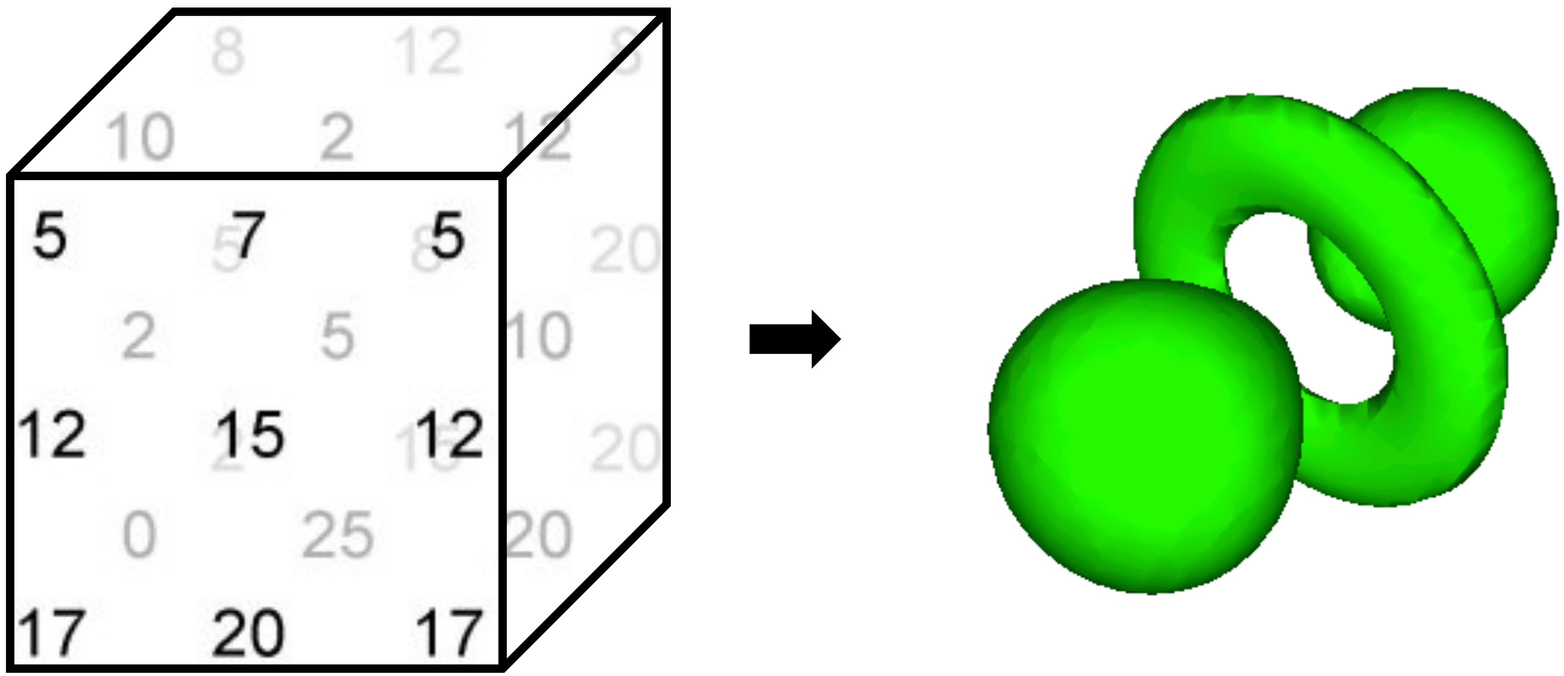
May.23, 2017

Schedule

- W01 4/11 Guidance
- W02 4/12 Setup
- W03 4/18 Introduction to Data Visualization
- W04 4/19 CG Programming
- W05 4/25 Rendering Pipeline
- W06 4/26 Coordinate Systems and Transformations
- W07 5/09 Shading
- W08 5/10 Shader Programming
- W09 5/16 Visualization Pipeline
- W10 5/17 Data Model and Transfer Function
- W11 5/23 Scalar Data Visualization 1 (Isosurface Extraction)
- W12 5/24 Implementation of Isosurface Extraction
- W13 5/30 Scalar Data Visualization 2 (Volume Rendering)
- W14 5/31 Implementation of Volume Rendering
- W15 6/06 Student Presentations

Isosurface Extraction

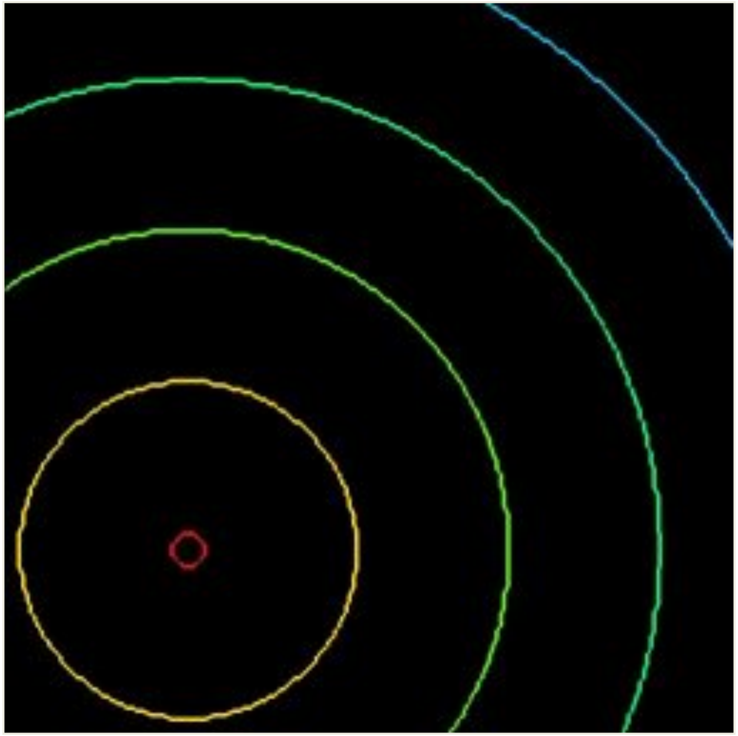
- Isosurfaces
  - Marching Cubes



Isosurface Extraction

- Isolines (Contour lines)
  - Marching Squares

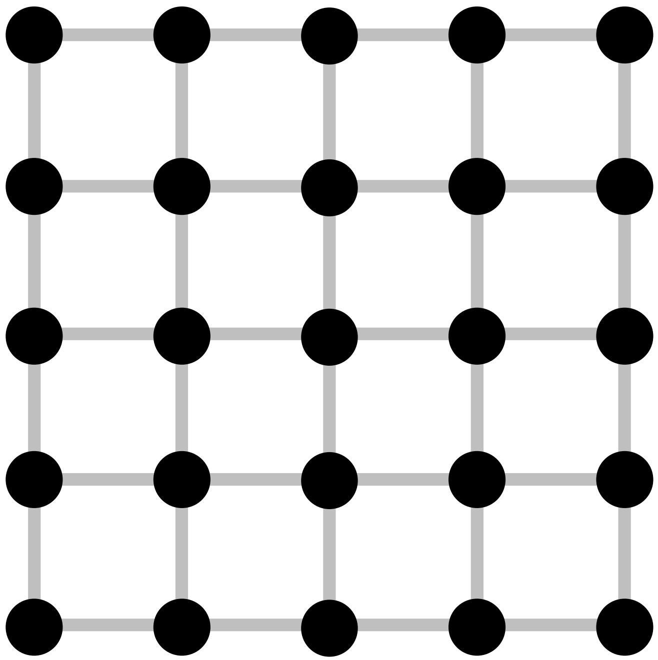
5	7	5	2	0
12	15	12	9	2
17	20	17	12	5
20	25	20	15	7
17	20	17	12	5



Contour Lines

- A set of points with the specified value S

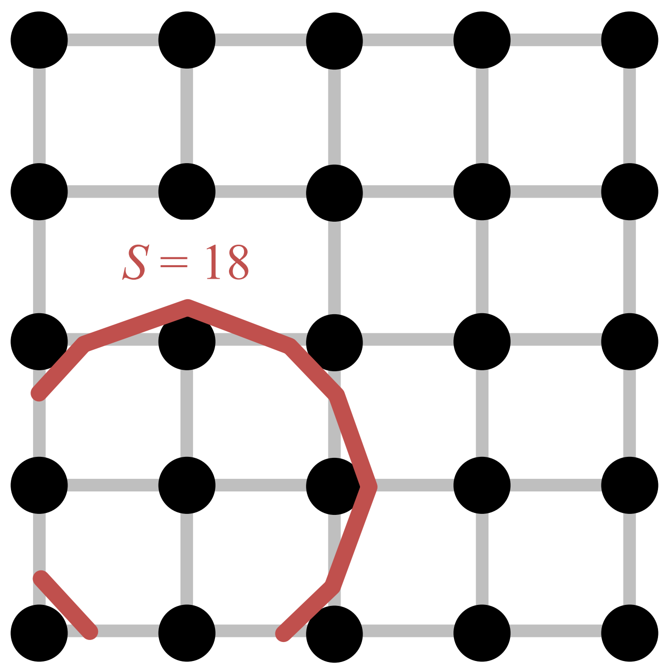
5	7	5	2	0
12	15	12	9	2
17	20	17	12	5
20	25	20	15	7
17	20	17	12	5



Contour Lines

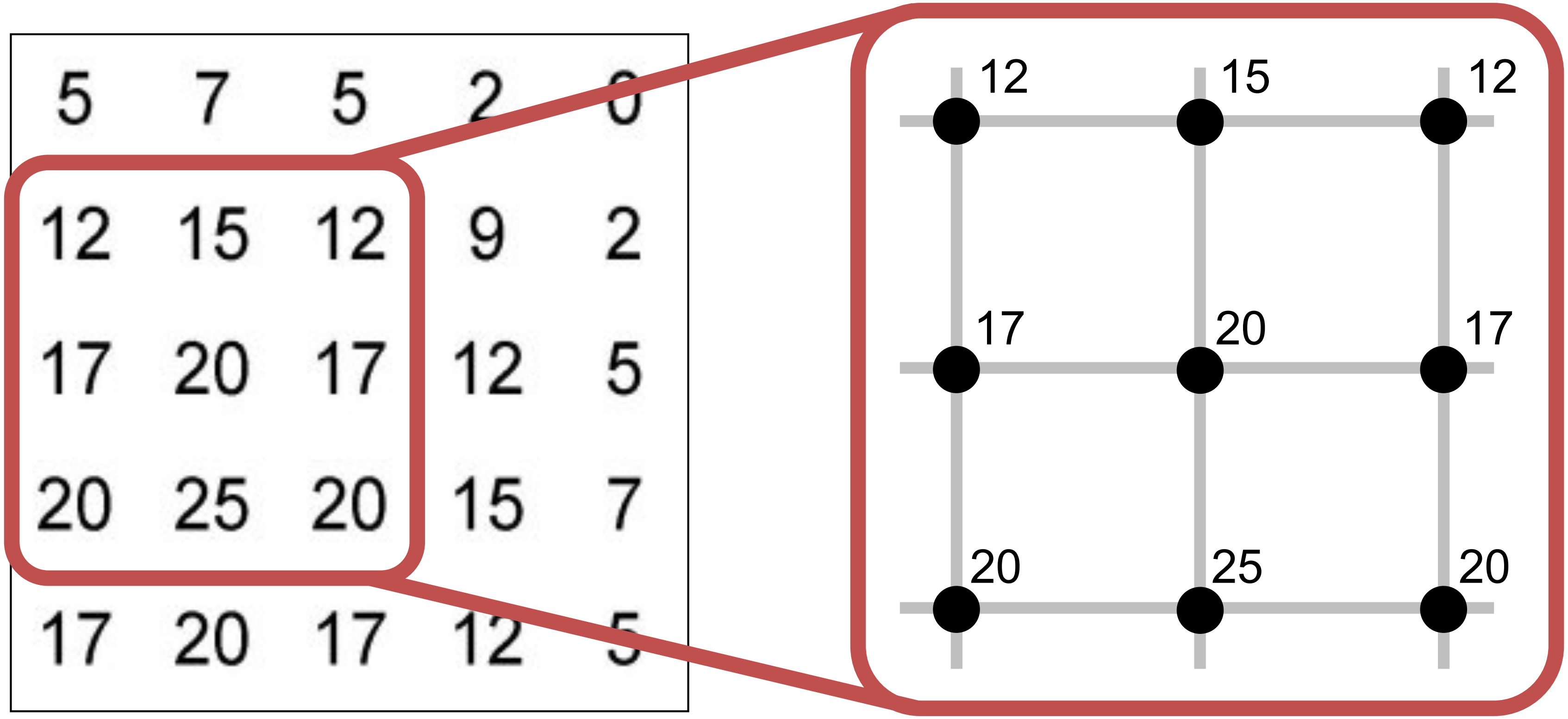
- Marching Squares
  - $S = 18$

5	7	5	2	0
12	15	12	9	2
17	20	17	12	5
20	25	20	15	7
17	20	17	12	5



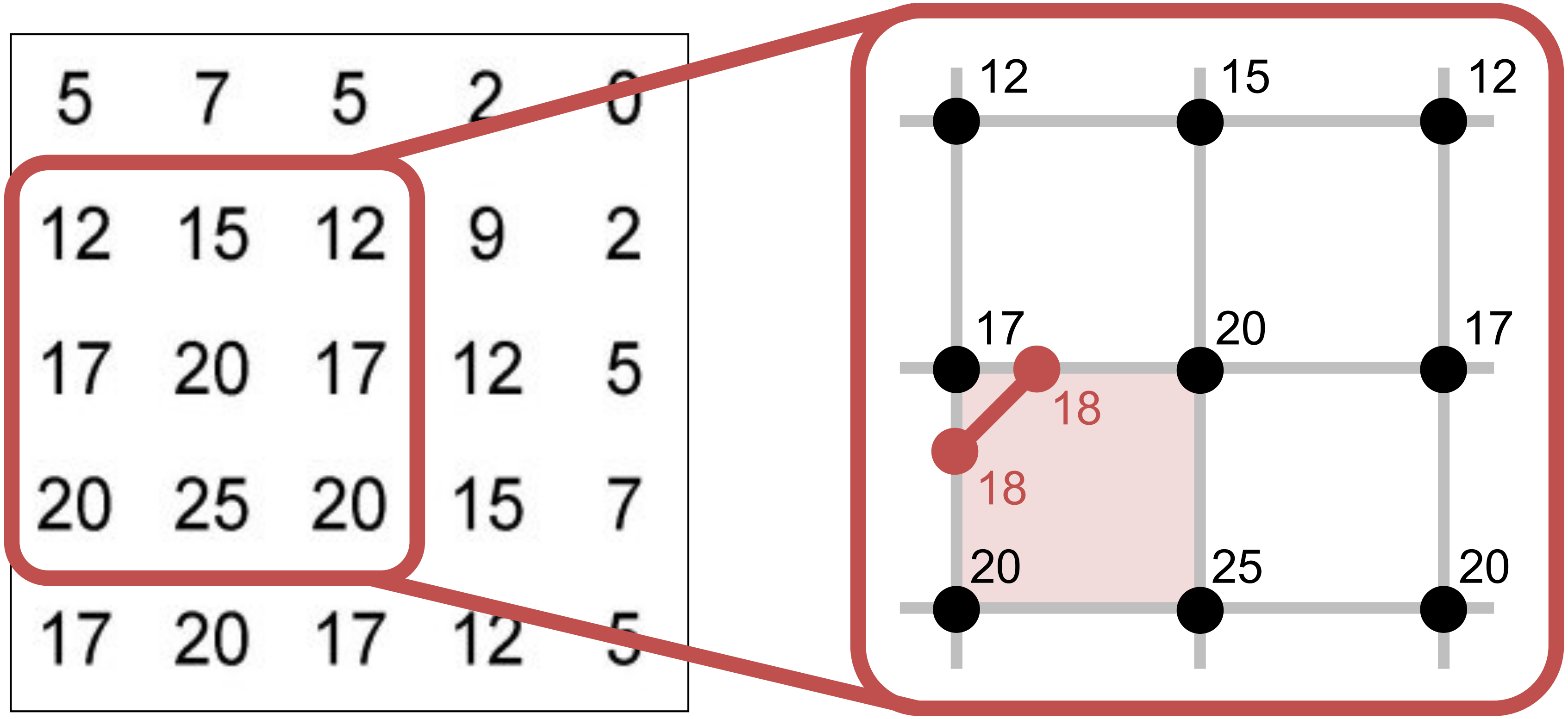
Contour Lines

- Marching Squares
  - $S = 18$



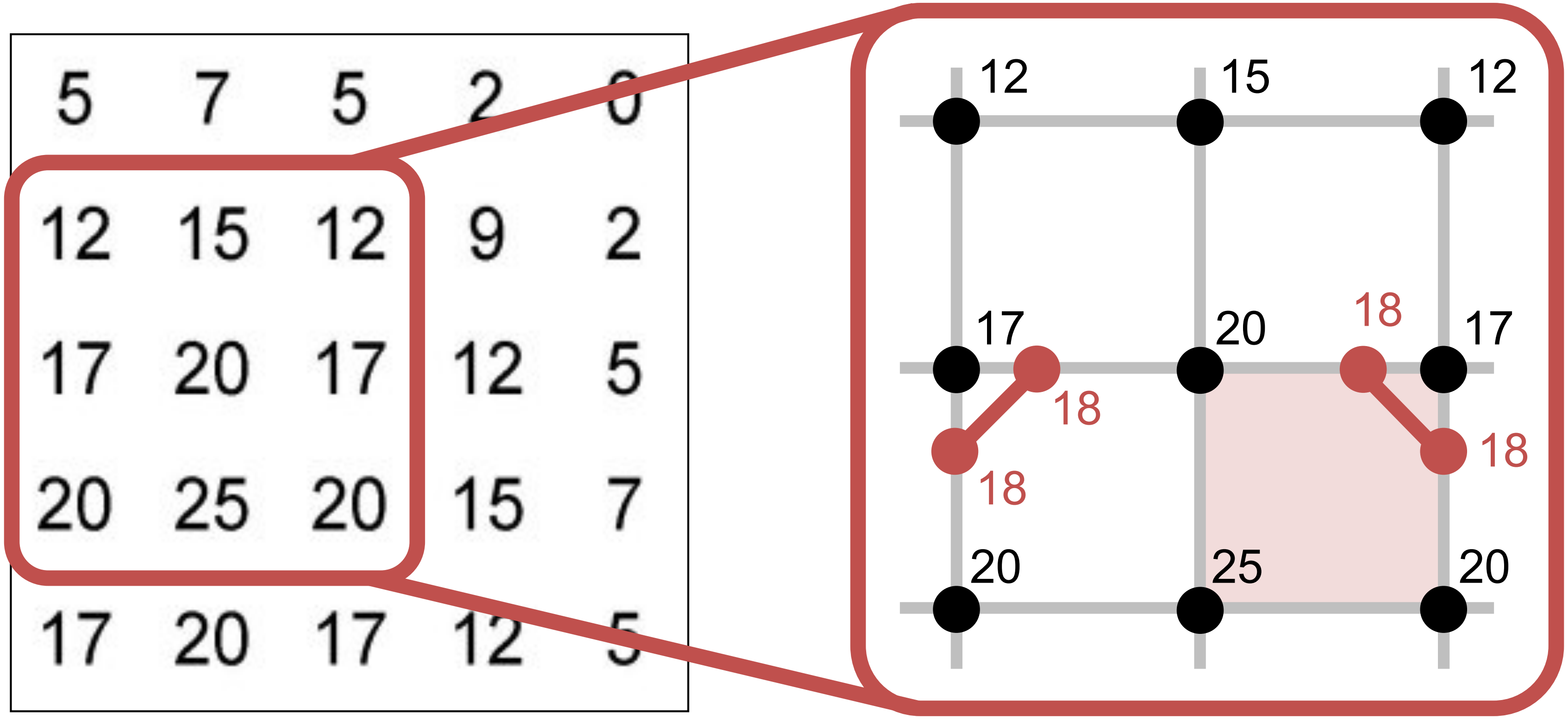
Contour Lines

- Marching Squares
  - $S = 18$



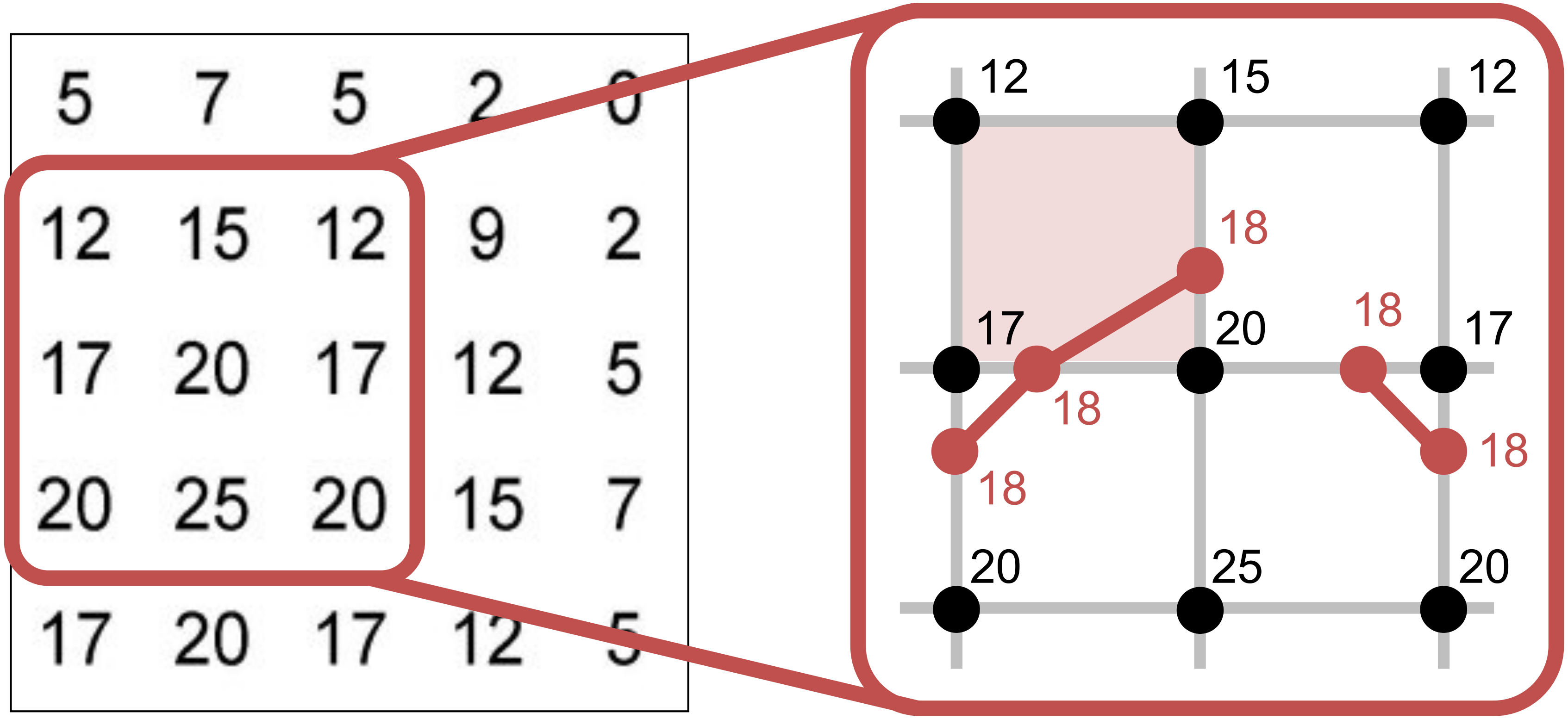
Contour Lines

- Marching Squares
  - $S = 18$



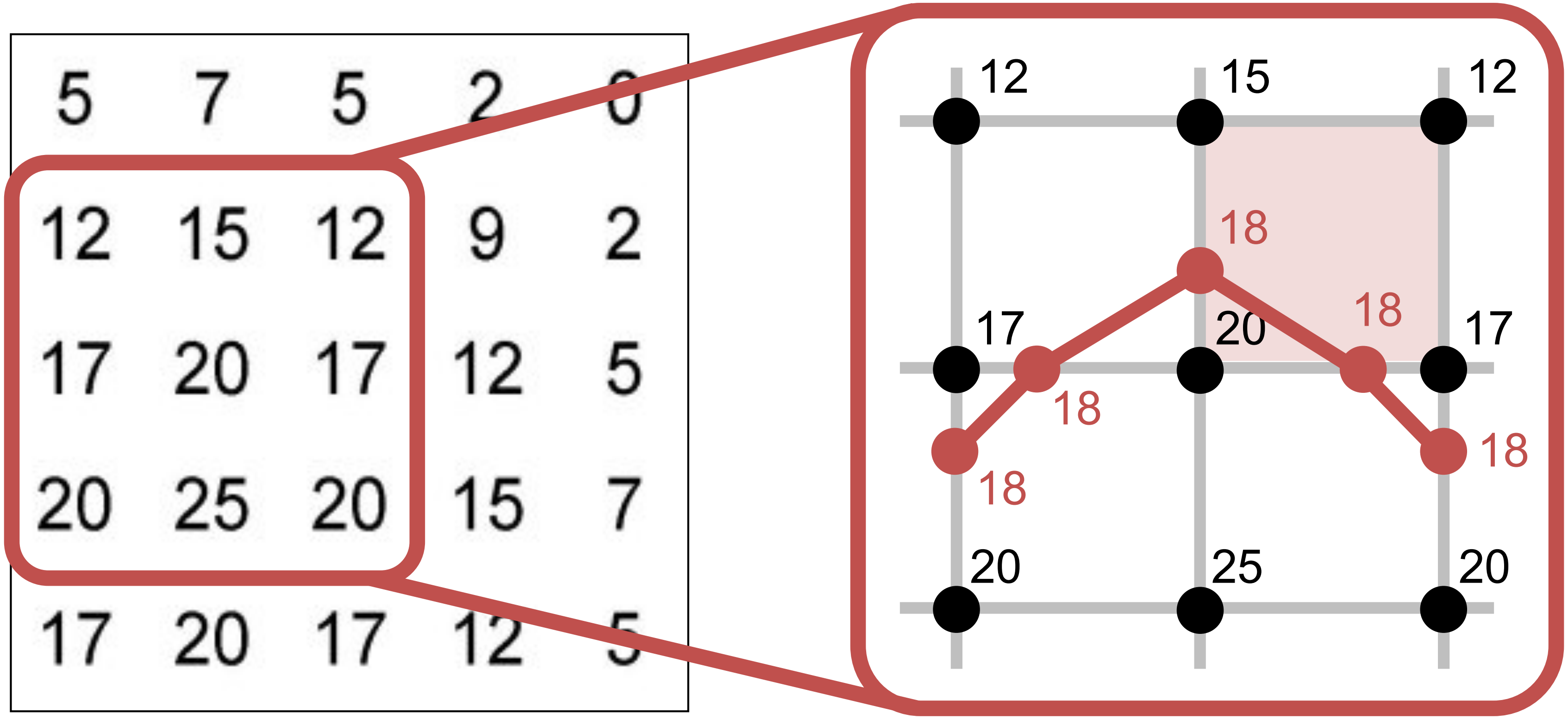
Contour Lines

- Marching Squares
  - $S = 18$



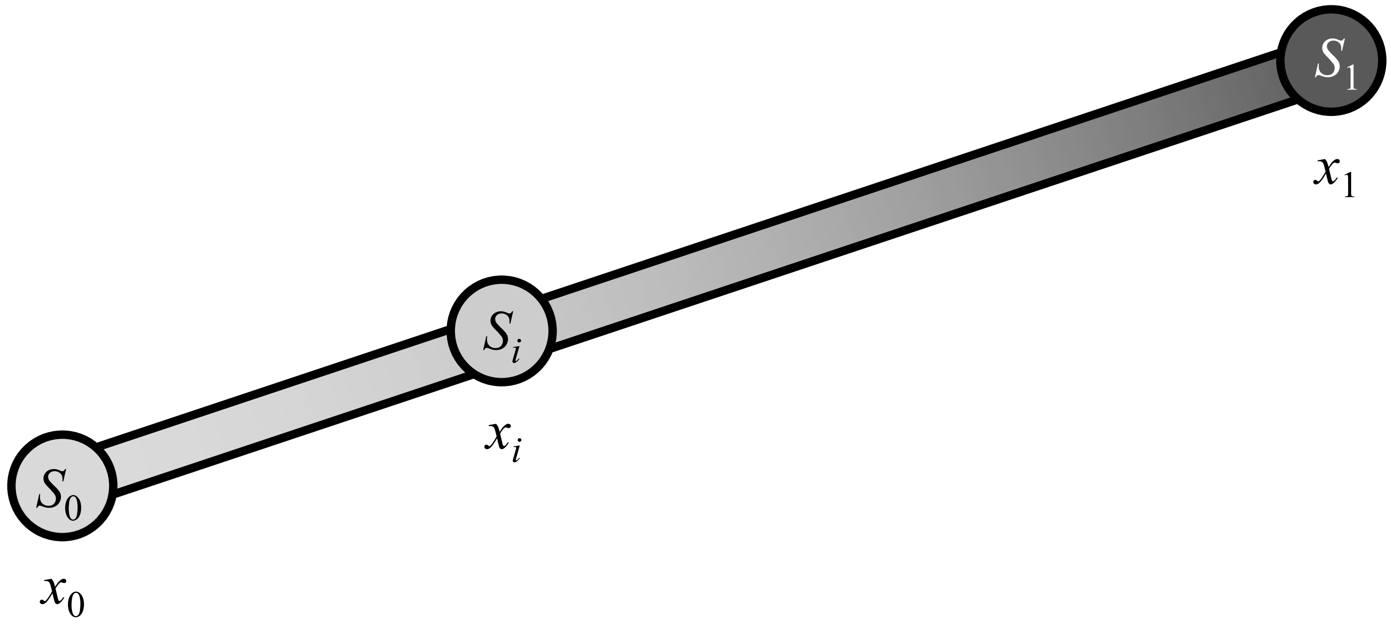
Contour Lines

- Marching Squares
  - $S = 18$



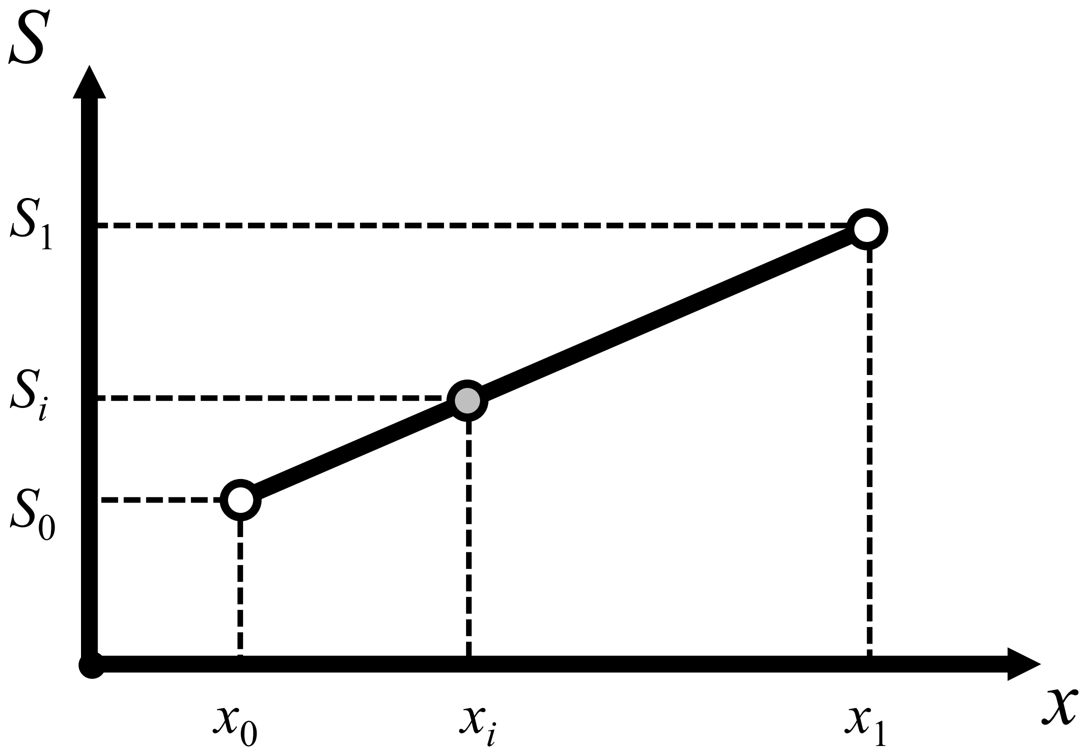
Interpolation

- Scalar value S_i at x_i ?



Interpolation

- Scalar value S_i at x_i ?
  - Linear interpolation



$$S_i = (1 - t)S_0 + tS_1$$

where

$$t = \frac{x_i - x_0}{x_1 - x_0}$$

Interpolation Function

- Coordinate transformation from global coordinates x to local coordinates p
  - Interpolate the coordinate values
  - Transform using Newton-Raphson method
- Interpolation function $N_i(p_j)$
  - Defined for each node

$$N_i(p_j) = \begin{cases} 1.0 & (i = j) \\ 0.0 & (i \neq j) \end{cases}$$

Interpolation: Line segment

- The point p in local coordinates for a point x on a line segment in global coordinates

Interpolation: Line segment

- The scalar value S for the point p can be calculated using interpolation functions N_0 and N_1

$$S(p) = \sum_{k=0}^{1} N_k(p) S_k$$

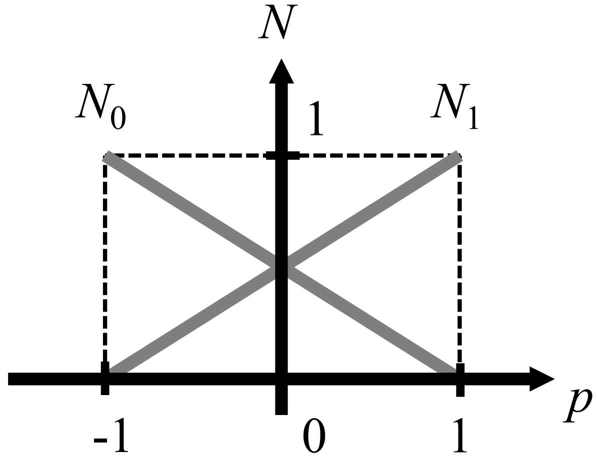
subject to

$$N_0(-1) = 1, \quad N_0(1) = 0$$
$$N_1(-1) = 0, \quad N_1(1) = 1$$

Interpolation: Line segment

- Interpolation Function: $N_0\,,\,N_1$

$$N_0(-1) = 1, \quad N_0(1) = 0$$
$$N_1(-1) = 0, \quad N_1(1) = 1$$



$$\Rightarrow \quad N_0 = \frac{1}{2}\left(1 - p\right), \quad N_1 = \frac{1}{2}\left(1 + p\right)$$

Interpolation: Line segment

- Data interpolation

$$\begin{aligned} S(p) &= N_0(p)S_0 + N_1(p)S_1 \\ &= \frac{1}{2}(S_0 + S_1) + \frac{1}{2}(S_1 - S_0)p \end{aligned}$$

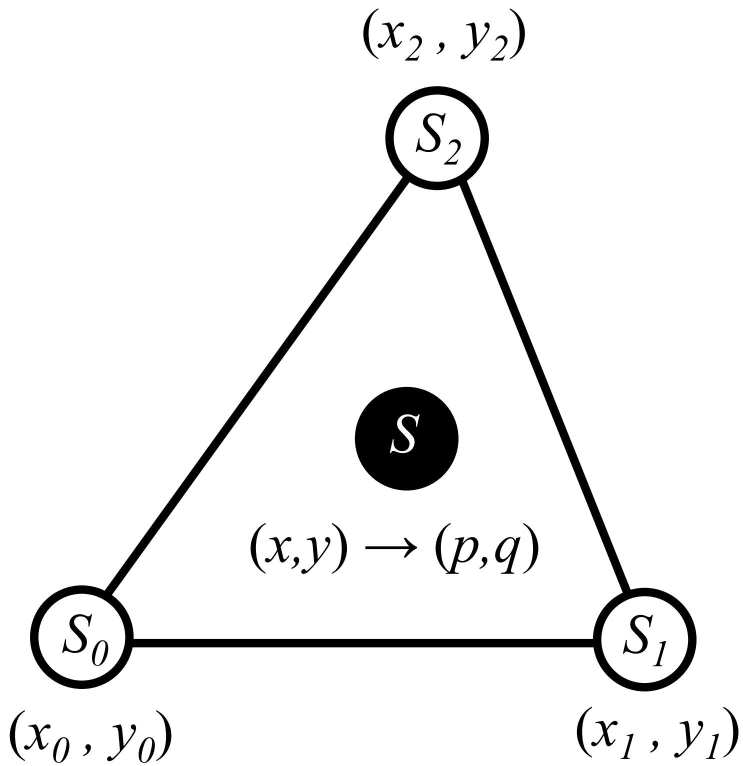
- Coordinate interpolation

$$x(p) = \frac{1}{2}(x_0 + x_1) + \frac{1}{2}(x_1 - x_0)p$$

$$\Rightarrow p = \frac{2x - x_0 - x_1}{x_1 - x_0}$$

Interpolation: Triangle

- The point (p, q) in local coordinates for a point (x, y) within a triangle in global coordinates



Interpolation: Triangle

- The scalar value S for the point (p, q) can be calculated using interpolation functions N_0, N_1 and N_2

$$S(p,q)=\sum_{k=0}^{2}N_k(p,q)S_k$$

subject to

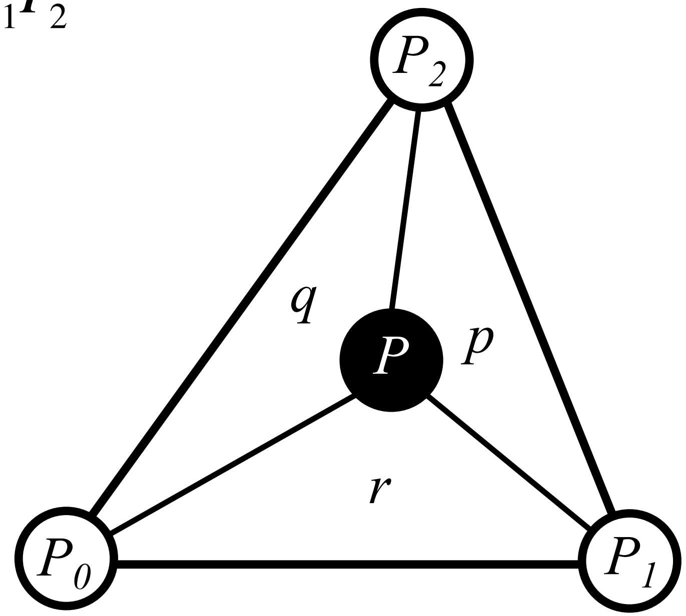
$$N_0(1,0)=1, \quad N_0(0,1)=0, \quad N_0(0,0)=0$$
$$N_1(1,0)=0, \quad N_1(0,1)=1, \quad N_1(0,0)=0$$
$$N_2(1,0)=0, \quad N_2(0,1)=0, \quad N_2(0,0)=1$$

Interpolation: Triangle

- Interpolation Function: N_0 , N_1 , N_2
  - Local coordinate = Area coordinate (p, q, r)

$$p = \frac{\Delta P P_1 P_2}{\Delta P_0 P_1 P_2}, \quad q = \frac{\Delta P P_2 P_0}{\Delta P_0 P_1 P_2}, \quad r = \frac{\Delta P P_0 P_1}{\Delta P_0 P_1 P_2}$$

$$\begin{aligned} & N_0(p,q) && = && p \\ \Rightarrow \quad & N_1(p,q) && = && q \\ & N_2(p,q) && = && r && = && 1 - p - q \end{aligned}$$



Interpolation: Trangle

- Data interpolation

$$\begin{aligned} S(p,q) &= N_0(p,q)S_0 + N_1(p,q)S_1 + N_2(p,q)S_2 \\ &= (S_0 - S_2)p + (S_1 - S_2)q + S_2 \end{aligned}$$

- Coordinate interpolation

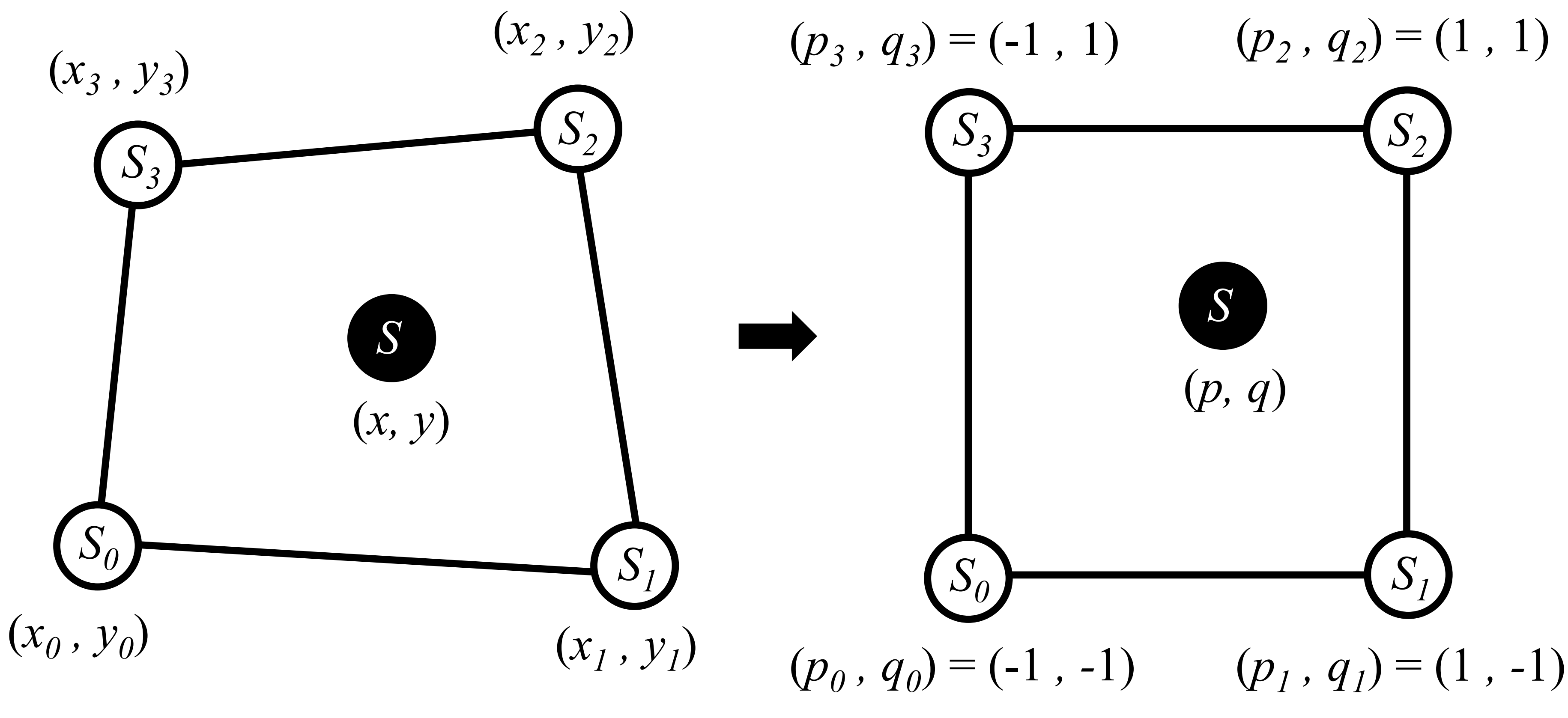
$$x(p,q) = (x_0 - x_2)p + (x_1 - x_2)q + x_2$$

$$y(p,q) = (y_0 - y_2)p + (y_1 - y_2)q + y_2$$

$$\Rightarrow \begin{aligned} p &= \frac{(y_1 - y_2)(x - x_2) - (x_1 - x_2)(y - y_2)}{(y_1 - y_2)(x_0 - x_2) - (x_1 - x_2)(y_0 - y_2)} \\ q &= \frac{(x_0 - x_2)(y - y_2) - (y_0 - y_2)(x - x_2)}{(y_1 - y_2)(x_0 - x_2) - (x_1 - x_2)(y_0 - y_2)} \end{aligned}$$

Interpolation: Quadrangle

- The point (p, q) in local coordinates for a point (x, y) within a quadrangle in global coordinates



Interpolation: Quadrangle

- The scalar value S for the point (p, q) can be calculated using interpolation functions N_0, N_1, N_2 and N_3

$$S(p,q) = \sum_{k=0}^{3} N_k(p,q) S_k$$

Interpolation: Quadrangle

- Interpolation Function: N_0, N_1, N_2, N_3

$$N_0(p,q) = \frac{1}{4}(1-p)(1-q) = \frac{1}{4}(1+p_0p)(1+q_0q)$$
$$N_1(p,q) = \frac{1}{4}(1+p)(1-q) = \frac{1}{4}(1+p_1p)(1+q_1q)$$
$$N_2(p,q) = \frac{1}{4}(1+p)(1+q) = \frac{1}{4}(1+p_2p)(1+q_2q)$$
$$N_3(p,q) = \frac{1}{4}(1-p)(1+q) = \frac{1}{4}(1+p_3p)(1+q_3q)$$

$$\Rightarrow N_k(p,q) = \frac{1}{4}(1+p_kp)(1+q_kq)\ , k = 0,1,2,3$$

Interpolation: Quadrangle

- Data interpolation

$$S(p,q) = \sum_{k=0}^{3} N_k(p,q) S_k$$

$$= \sum_{k=0}^{3} \frac{1}{4}(1 + p_k p)(1 + q_k q) S_k$$

Interpolation: Quadrangle

- Transformation from (p,q) to (x,y)

$$x(p,q) = \sum_{k=0}^{3} N_k(p,q)x_k$$

$$= \sum_{k=0}^{3} \frac{1}{4}(1 + p_k p)(1 + q_k q)x_k$$

$$y(p,q) = \sum_{k=0}^{3} N_k(p,q)y_k$$

$$= \sum_{k=0}^{3} \frac{1}{4}(1 + p_k p)(1 + q_k q)y_k$$

Interpolation: Quadrangle

- Transformation from (x,y) to (p,q)
  - Newton-Raphson method
    1. Start from an intial value (p0,q0)
    2. Calculate (xn, yn) from (pn,qn)

$$\begin{cases} x_n = x(p_n, q_n) \\ y_n = y(p_n, q_n) \end{cases}$$

Interpolation: Quadrangle

- Transformation from (x,y) to (p,q)
  - Newton-Raphson method
    3. Calculate a difference between a point (x,y) obtained by Taylor expansion and (xn,yn) in global coordinates.

$$\begin{cases} x = x(p_n + \Delta p, q_n + \Delta q) \\ y = y(p_n + \Delta p, q_n + \Delta q) \end{cases}$$

$$x - x_n = \frac{\partial x}{\partial p}\Delta p + \frac{\partial x}{\partial q}\Delta q$$
$$y - y_n = \frac{\partial y}{\partial p}\Delta p + \frac{\partial y}{\partial q}\Delta q$$

Interpolation: Quadrangle

- Transformation from (x,y) to (p,q)
  - Newton-Raphson method
    4. Calculate the difference (dp,dq) in local coordinates as follows:

$$\begin{pmatrix} \Delta p \\ \Delta q \end{pmatrix} = \mathbf{J}^{-1} \begin{pmatrix} x - x_n \\ y - y_n \end{pmatrix}$$

where, J is a Jacobian matrix

$$\mathbf{J} = \begin{pmatrix} \dfrac{\partial x}{\partial p} & \dfrac{\partial x}{\partial q} \\ \dfrac{\partial y}{\partial p} & \dfrac{\partial y}{\partial q} \end{pmatrix} = \sum_{i=0}^{3} \begin{pmatrix} \dfrac{\partial N_i(p,q)}{\partial p} x_i & \dfrac{\partial N_i(p,q)}{\partial q} x_i \\ \dfrac{\partial N_i(p,q)}{\partial p} y_i & \dfrac{\partial N_i(p,q)}{\partial q} y_i \end{pmatrix}$$

Interpolation: Quadrangle

- Transformation from (x,y) to (p,q)
  - Newton-Raphson method
    5. Calculate a new point (pn+1,qn+1) by adding (dp,dq) to the current point (pn,qn) in local coordinates.

$$\begin{pmatrix} p_{n+1} \\ q_{n+1} \end{pmatrix} = \begin{pmatrix} p_n \\ q_n \end{pmatrix} + \begin{pmatrix} \Delta p \\ \Delta q \end{pmatrix}$$

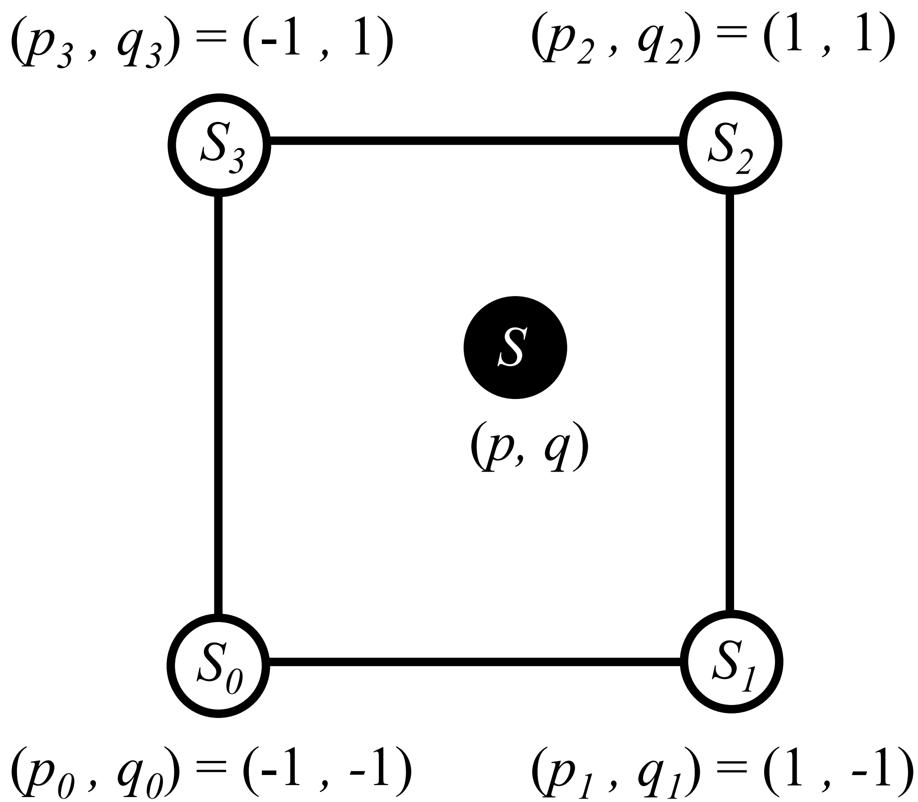
    6. Terminate if the difference (dp,dq) can be assumed as approximately zero, otherwise increment n and return to the step 2.

Interpolation: Quadrangle

- Transformation from (x,y) to (p,q)
  - In case of a rectangle

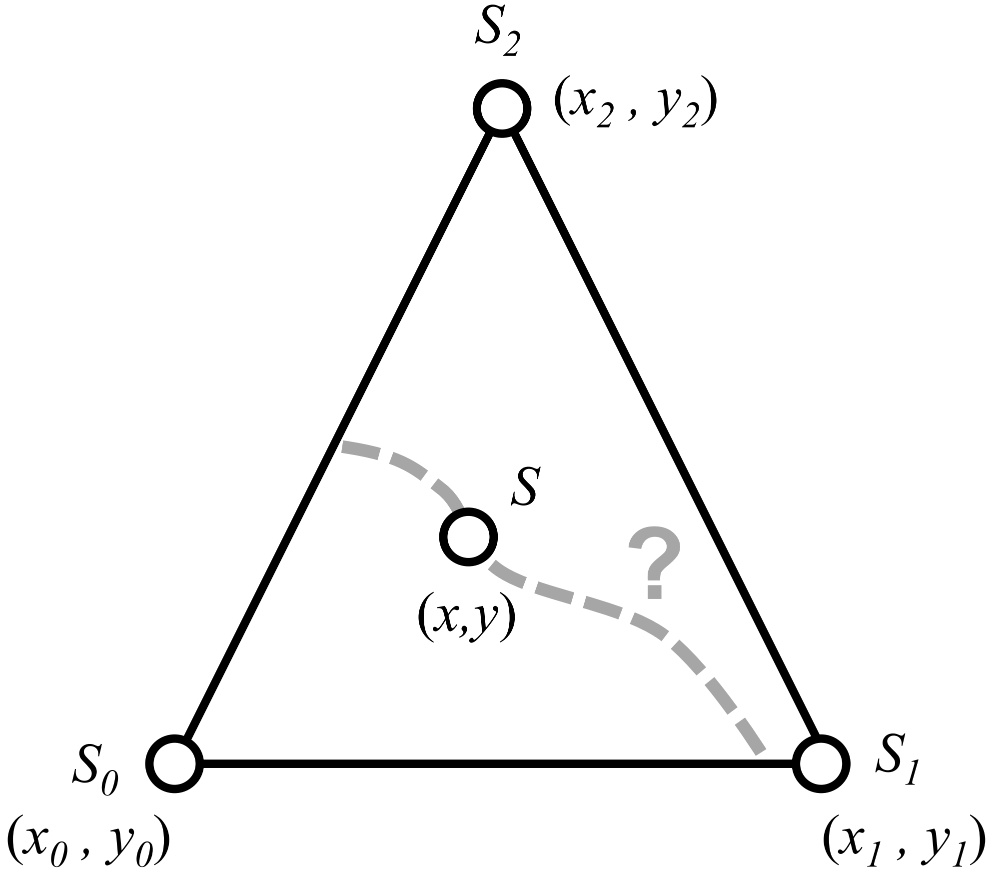
$$p = 2\frac{x - x_0}{x_1 - x_0} - 1$$

$$q = 2\frac{y - y_0}{y_1 - y_0} - 1$$



Isolines on a triangle

- A set of points have a scalar value S



Isolines on a triangle

- Interpolated scalar value S at (p,q)

$$S(p,q) = \sum_{k=0}^{2} N_k(p,q)S_k = (S_0 - S_2)p + (S_1 - S_2)q + S_2$$

(p,q) can be calculated as follows:

$$p = a_0 + b_0 x + c_0 y$$
$$q = a_1 + b_1 x + c_1 y$$

where,

$$a_0 = (x_1 y_2 - y_1 x_2)/2\Delta \qquad a_1 = (x_2 y_0 - y_2 x_0)/2\Delta$$
$$b_0 = (y_1 - y_2)/2\Delta \qquad b_1 = (y_2 - y_0)/2\Delta$$
$$c_0 = (x_2 - x_1)/2\Delta \qquad c_1 = (x_0 - x_2)/2\Delta$$

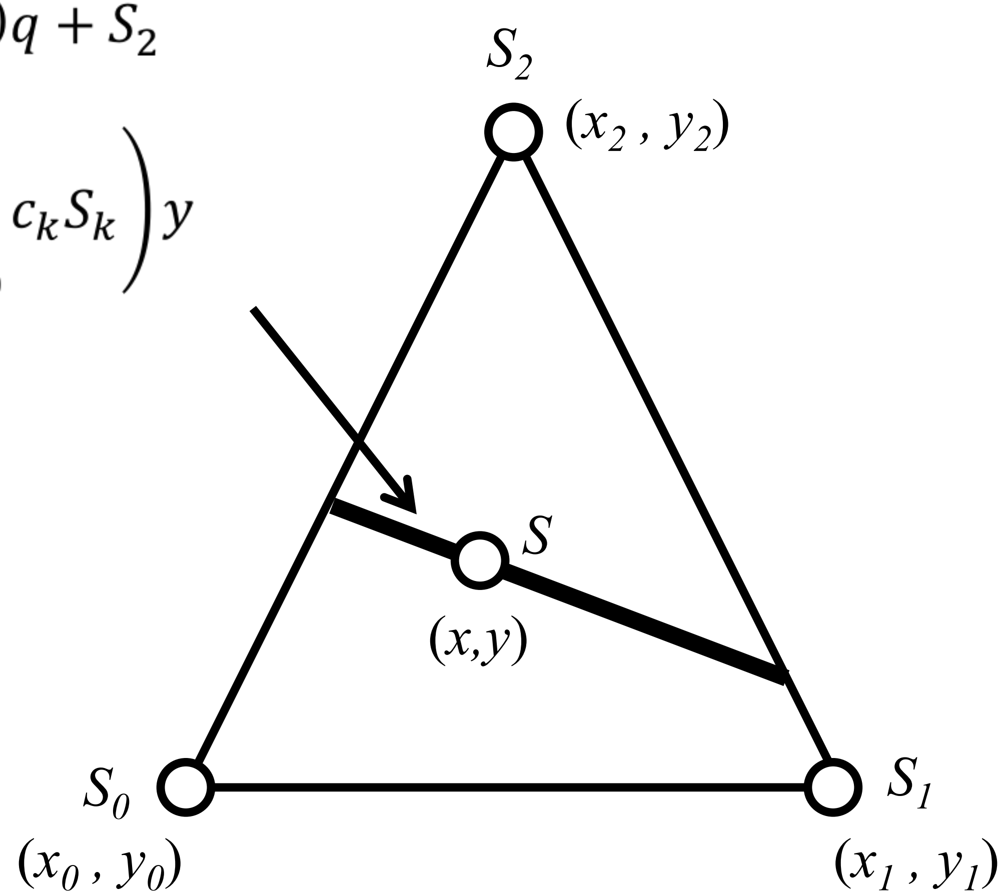
$$2\Delta = x_1 y_2 - x_2 y_1 - x_0 y_2 + x_2 y_0 + x_0 y_1 - x_1 y_0$$

Isolines on a triangle

- Isoline for S

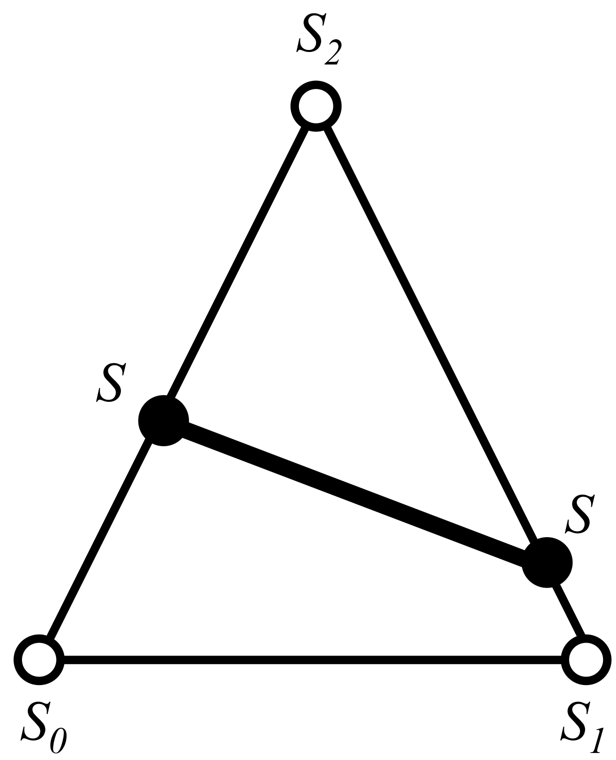
$$S(p,q) = (S_0 - S_2)p + (S_1 - S_2)q + S_2$$

$$= \sum_{k=0}^{2} a_k S_k + \left(\sum_{k=0}^{2} b_k S_k\right)x + \left(\sum_{k=0}^{2} c_k S_k\right)y$$



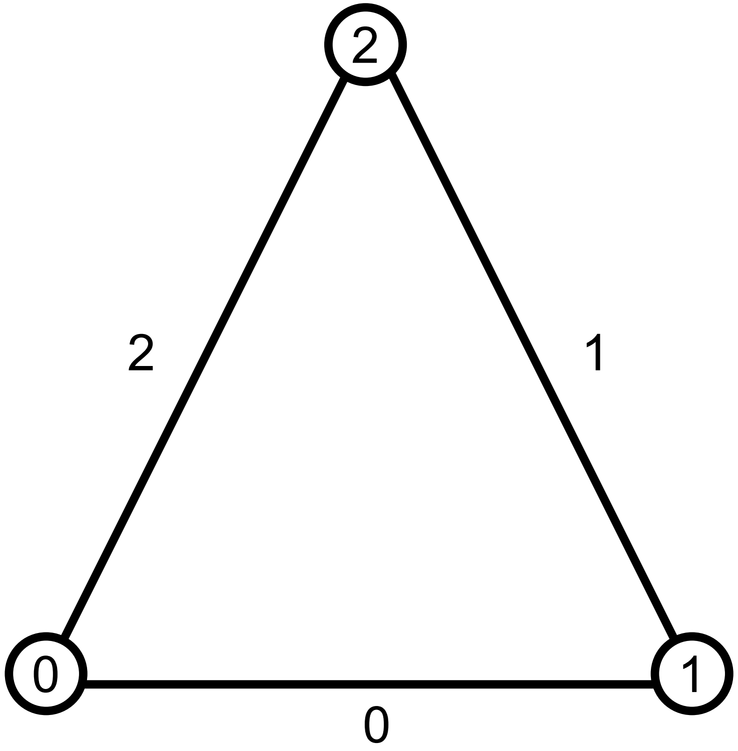
Marching Triangles

- Isolines for a triangle
  - Straight line
  - Intersections with edges
- Procedure
  1. Compare a threshold S with three scalar values defined on each vertex
  2. No isolines if all of the scalar values are greater or less than S
  3. Otherwise, calculate the intersection points with edges.



Marching Triangles

- Vertex and edge ID



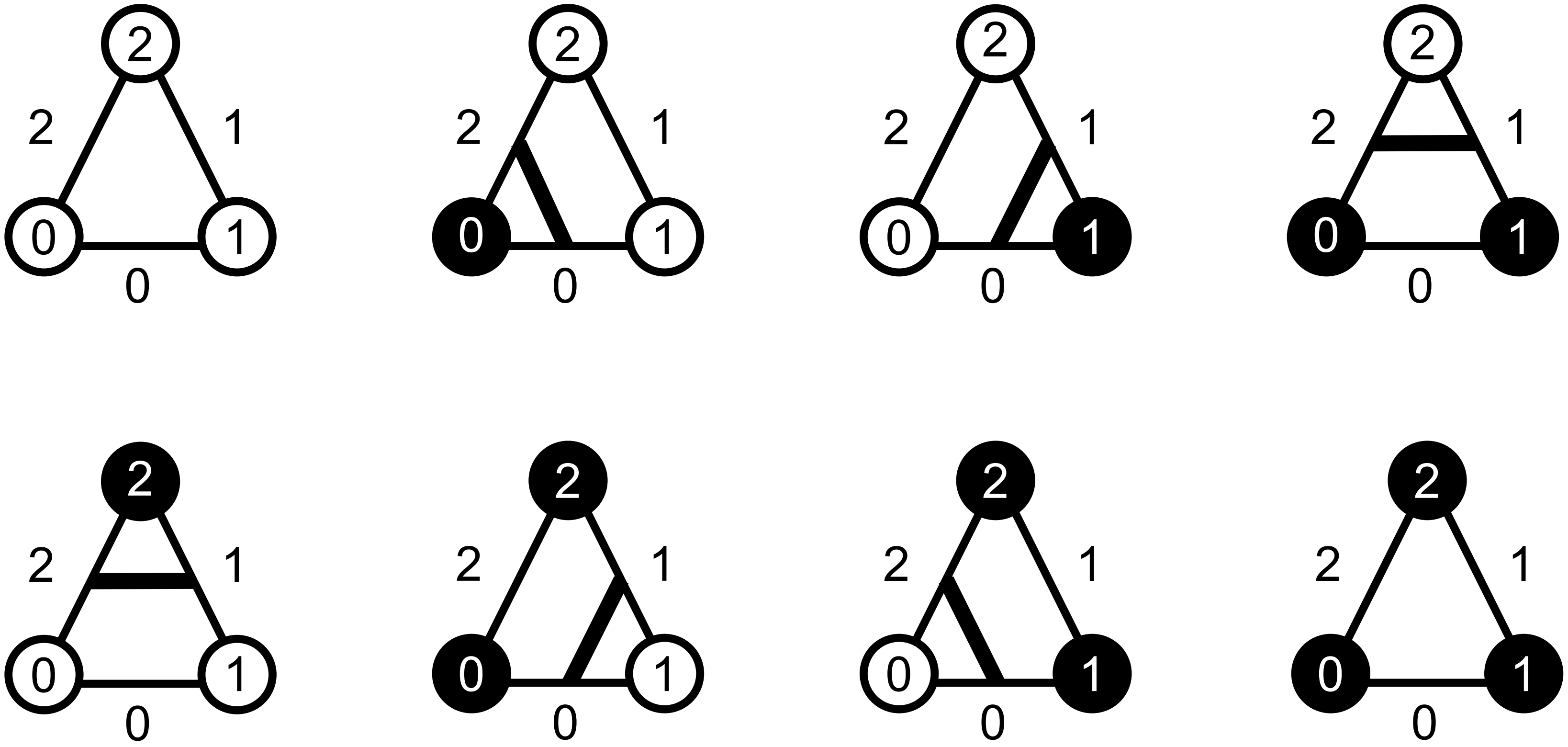
Vertex ID table

Edge ID	Vert. ID1	Vert. ID2
0	0	1
1	1	2
2	2	0

Marching Triangles

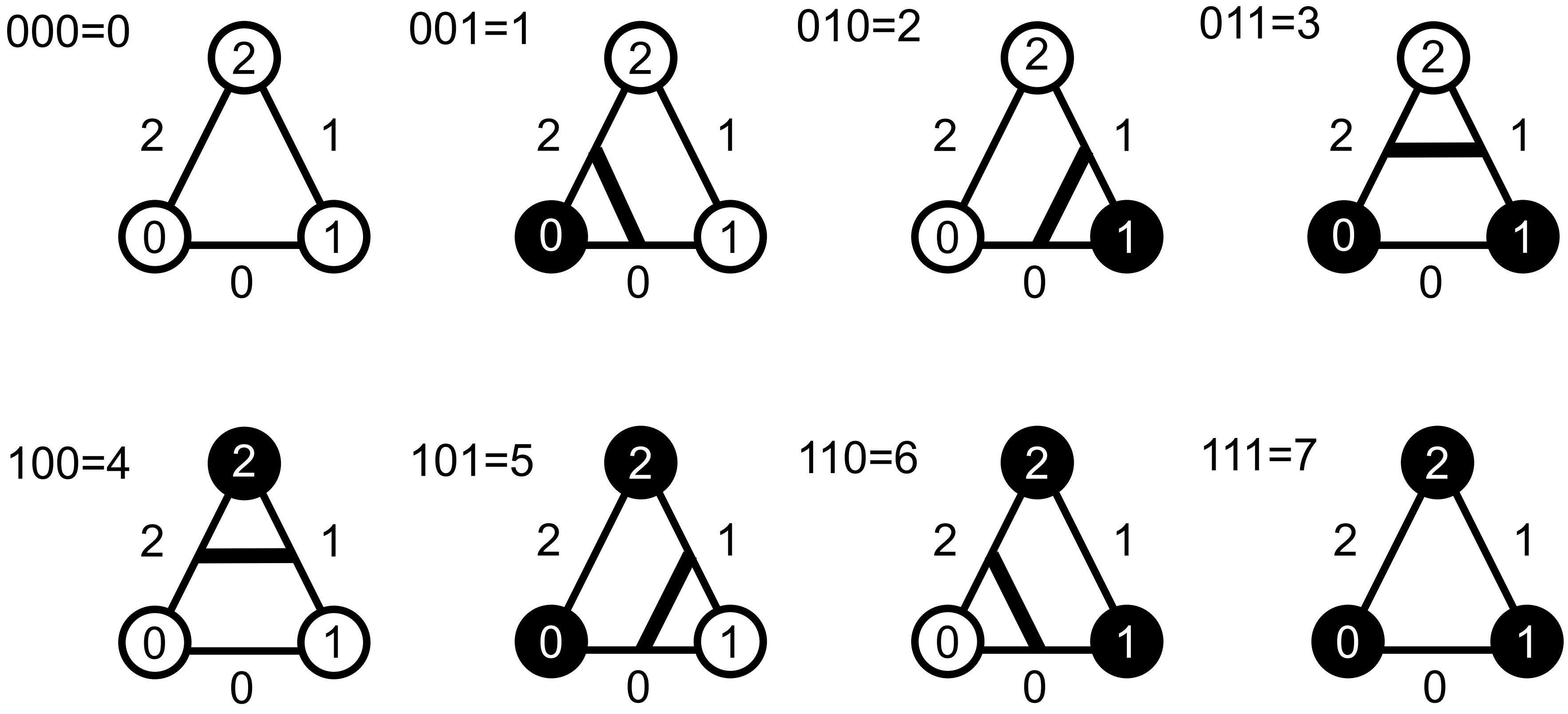
- Intersection patterns
  - $2^3 = 8$ patterns

○ : $S > S_i$
● : $S \leqq S_i$



Marching Triangles

- Intersection pattern table
  - Index for the table is represented in 3 bits



Marching Triangles

- Index calculation
  - logical sum of the scalar value and the threshold.

```
var index = 0;                         // 0 = 0000
if ( S0 > S ) { index |= 1; } // 1 = 0001
if ( S1 > S ) { index |= 2; } // 2 = 0010
if ( S2 > S ) { index |= 4; } // 3 = 0100
```

Marching Triangles

- Intersection pattern table

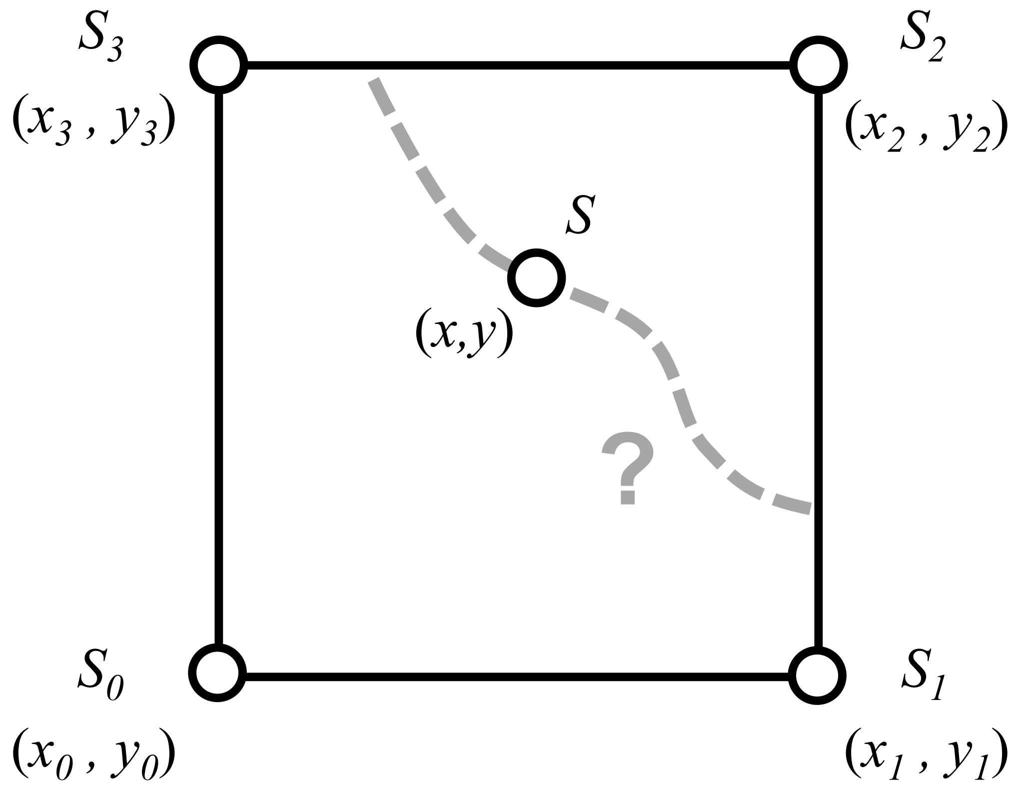
Edge ID table

Index	Edge ID1	Edge ID2
000 = 0	-1	-1
001 = 1	0	2
010 = 2	0	1
011 = 3	1	2
100 = 4	2	1
101 = 5	1	0
110 = 6	2	0
111 = 7	-1	-1

※ ”-1”: No intersection point on the edge

Isolines on a rectangle

- A set of points have a scalar value S



Isolines on a rectangle

- Interpolated scalar value S at (p,q)

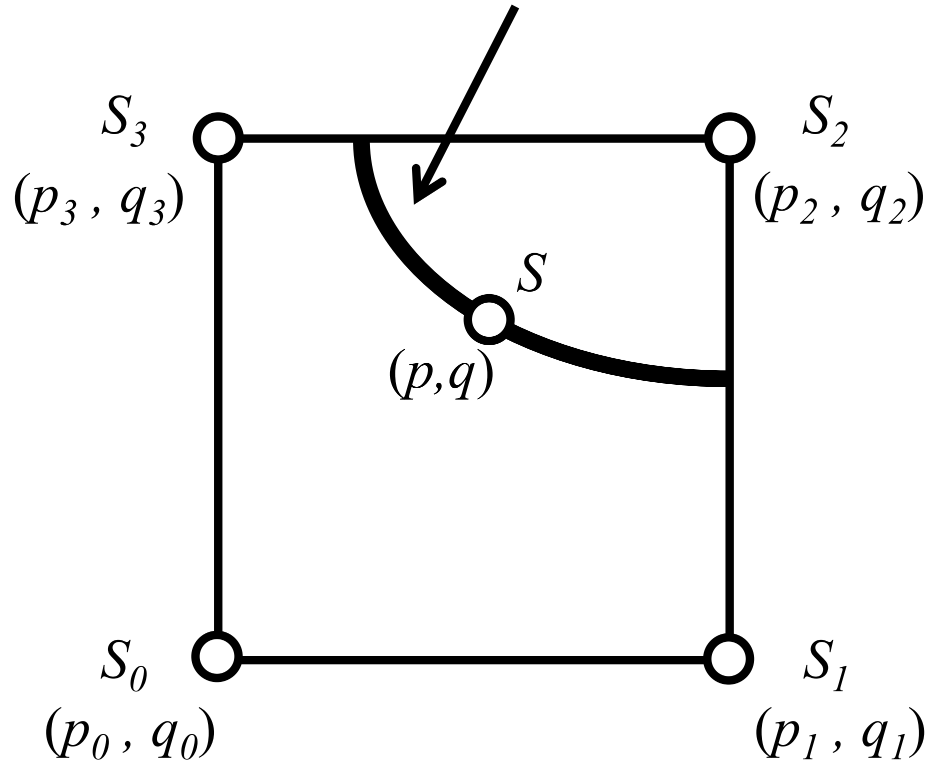
$$S(p,q) = \sum_{k=0}^{3} N_k(p,q) S_k$$

$$= \sum_{k=0}^{3} \frac{1}{4}(1 + p_k p)(1 + q_k q) S_k$$

Isolines on a rectangle

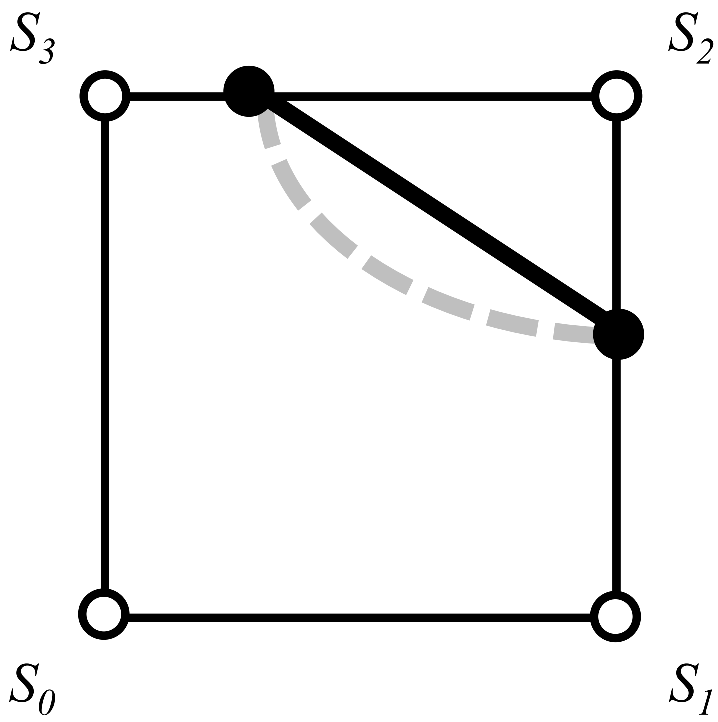
- Isoline for S

$$S = \sum_{k=0}^{3} \frac{S_k}{4} + \left(\sum_{k=0}^{3} \frac{p_k S_k}{4} \right) p + \left(\sum_{k=0}^{3} \frac{q_k S_k}{4} \right) q + \left(\sum_{k=0}^{3} \frac{p_k q_k S_k}{4} \right) pq = C$$



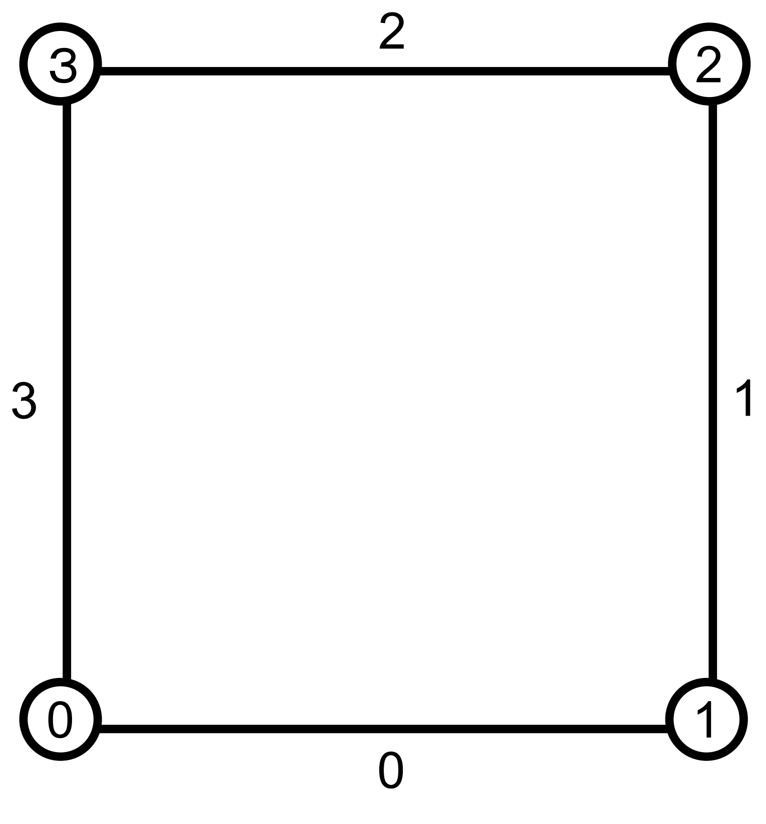
Marching Squares

- Efficient extraction of isolines by using the tables same as Marching triangles.



Marching Squares

- Vertex and edge IDs



Vertex ID table

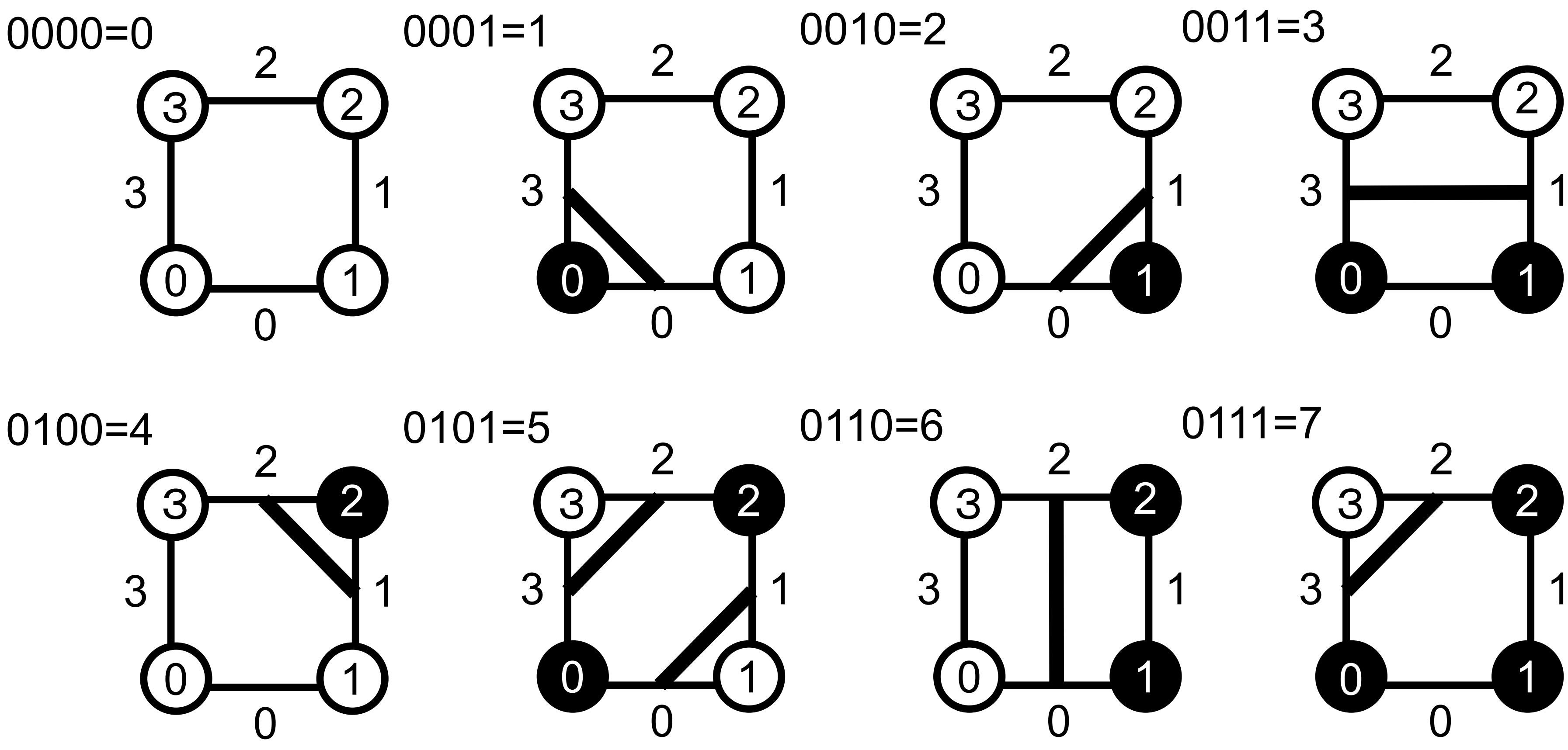
Edge ID	Vert ID1	Vert ID2
0	0	1
1	1	2
2	2	3
3	3	0

Marching Squares

- Intersection patterns
  – $2^4 = 16$ patterns

: $S > S_i$

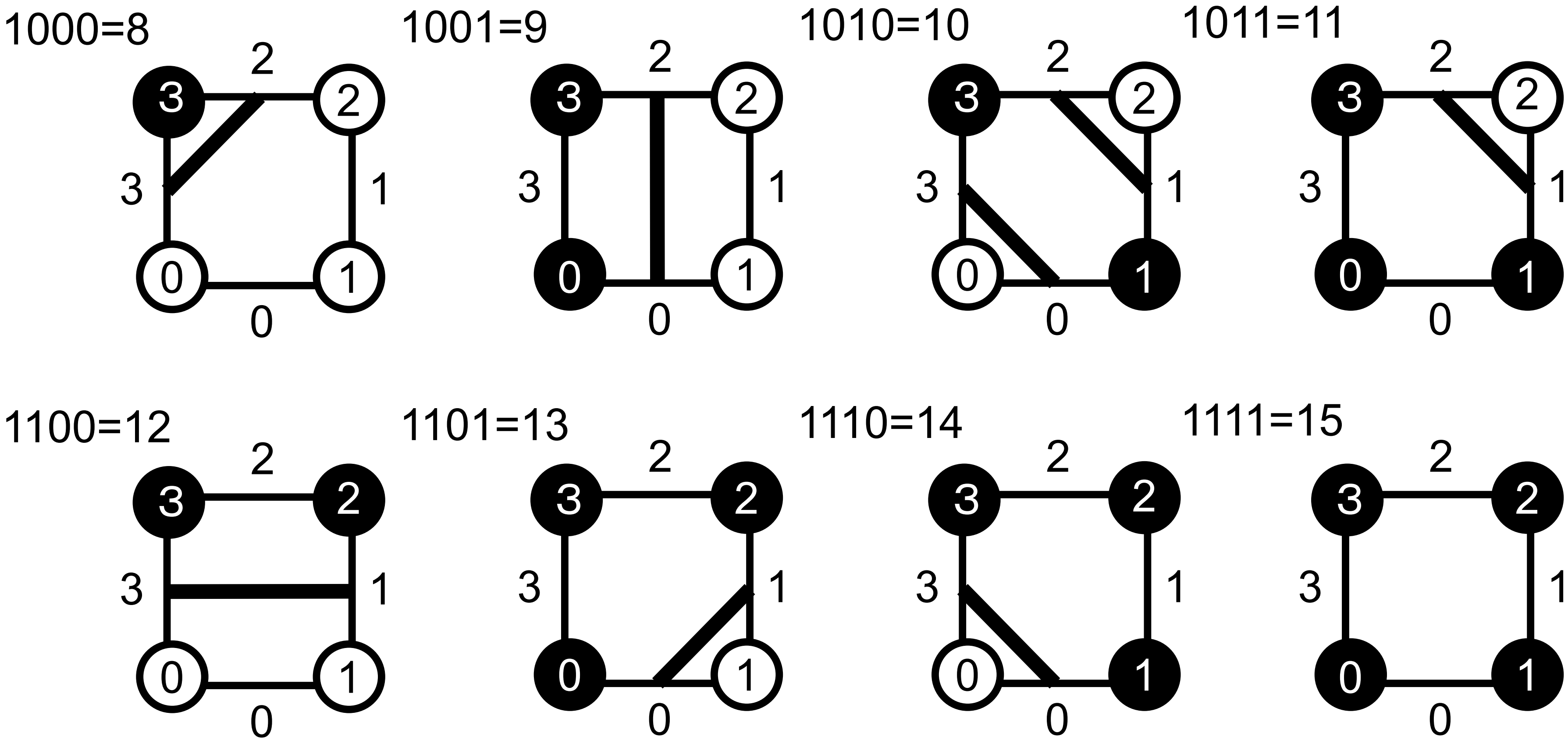
: $S \leqq S_i$



Marching Squares

- Intersection patterns
  - 2^4 = 16 patterns

○ : $S > S_i$
● : $S \leqq S_i$



Marching Squares

- Index calculation
  - logical sum of the scalar value and the threshold.

```
var index = 0;                            // 0 = 0000
if ( S0 > S ) { index |= 1; } // 1 = 0001
if ( S1 > S ) { index |= 2; } // 2 = 0010
if ( S2 > S ) { index |= 4; } // 3 = 0100
if ( S3 > S ) { index |= 8; } // 4 = 1000
```

Marching Squares

- Intersection pattern table

Edge ID table

Index	Edge1 ID1	Edge1 ID2	Edge2 ID1	Edge2 ID2
0000 = 0	-1	-1	-1	-1
0001 = 1	0	3	-1	-1
0010 = 2	0	1	-1	-1
0011 = 3	3	1	-1	-1
0100 = 4	1	2	-1	-1
0101 = 5	1	0	3	2
0110 = 6	2	0	-1	-1
0111 = 7	3	2	-1	-1

※ ”-1”: No intersection point on the edge

Marching Squares

- Intersection pattern table

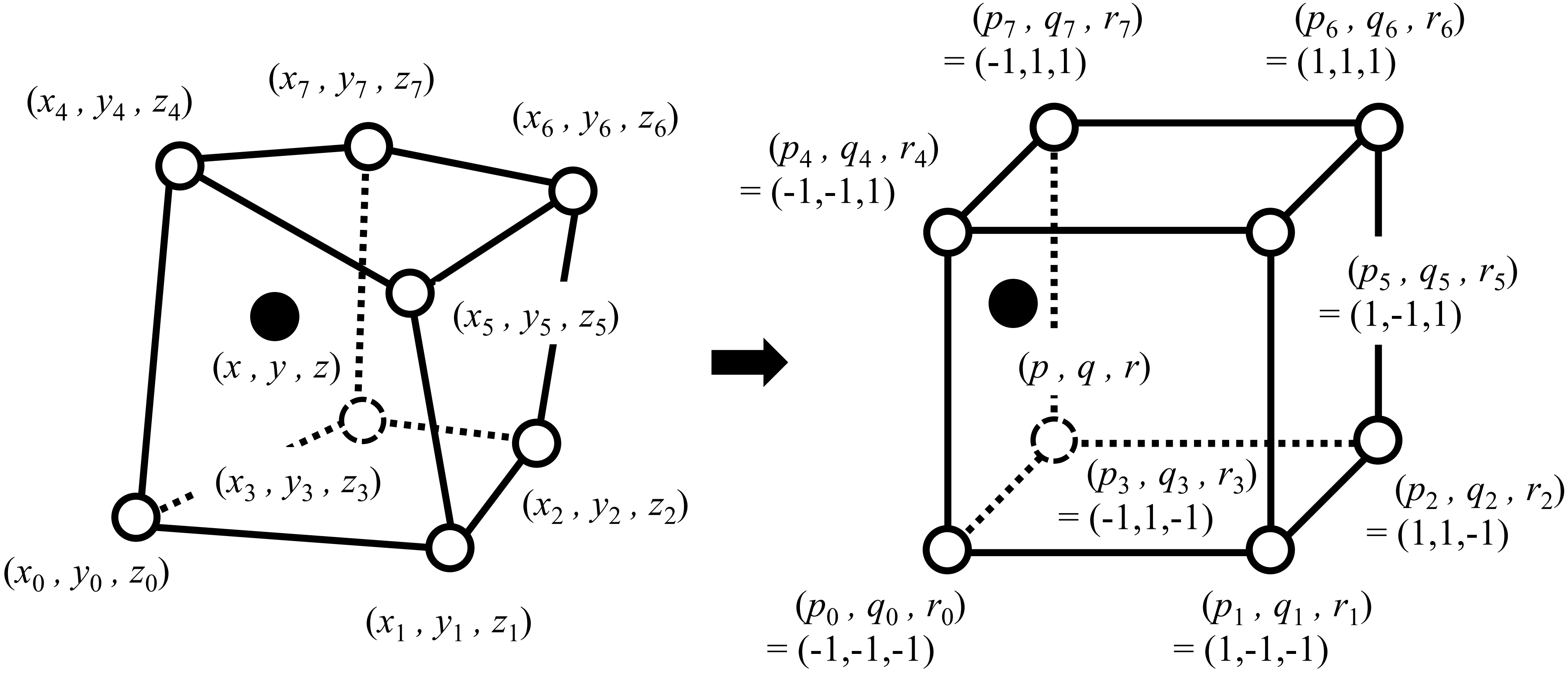
Edge ID table

Index	Edge1 ID1	Edge1 ID2	Edge2 ID1	Edge2 ID2
1000 = 8	2	3	-1	-1
1001 = 9	0	2	-1	-1
1010 = 10	1	2	0	3
1011 = 11	1	2	-1	-1
1100 = 12	1	3	-1	-1
1101 = 13	0	1	-1	-1
1110 = 14	0	3	-1	-1
1111 = 15	-1	-1	-1	-1

※ ”-1”: No intersection point on the edge

Interpolation: Hexahedron

- The point (p, q, r) in local coordinates for a point (x, y, z) within a hexahedron in global coordinates



Interpolation: Hexahedron

- The scalar value S for the point (p, q) can be calculated using interpolation functions N_0, N_1, N_2, …, N_7

$$S(p,q) = \sum_{k=0}^{7} N_k(p,q) S_k$$

- Interpolation Functions

$$N_k(p,q) = \frac{1}{8}(1 + p_k p)(1 + q_k q)(1 + r_k r), k = 0,1,2,\ldots,7$$

Interpolation: Hexahedron

- Data interpolation

$$S(p,q,r) = \sum_{k=0}^{7} N_k(p,q,r)S_k$$

$$= \sum_{k=0}^{7} \frac{1}{8}(1 + p_k p)(1 + q_k q)(1 + r_k r)S_k$$

Interpolation: Hexahedron

- Transformation from (p,q,r) to (x,y,z)

$$x(p,q,r) = \sum_{k=0}^{7} N_k(p,q,r)x_k = \frac{1}{8}(1+p_k p)(1+q_k q)(1+r_k r)x_k$$

$$y(p,q,r) = \sum_{k=0}^{7} N_k(p,q,r)y_k = \frac{1}{8}(1+p_k p)(1+q_k q)(1+r_k r)y_k$$

$$z(p,q,r) = \sum_{k=0}^{7} N_k(p,q,r)z_k = \frac{1}{8}(1+p_k p)(1+q_k q)(1+r_k r)z_k$$

Interpolation: Hexahedron

- Transformation from (x,y,z) to (p,q,r)
  - Newton-Raphson method
    - Jacobian matrix J

$$\mathbf{J} = \begin{pmatrix} \frac{\partial x}{\partial p} & \frac{\partial x}{\partial q} & \frac{\partial x}{\partial r} \\ \frac{\partial y}{\partial p} & \frac{\partial y}{\partial q} & \frac{\partial y}{\partial r} \\ \frac{\partial z}{\partial q} & \frac{\partial z}{\partial q} & \frac{\partial z}{\partial r} \end{pmatrix} = \sum_{i=0}^{3} \begin{pmatrix} \frac{\partial N_i(p,q,r)}{\partial p} x_i & \frac{\partial N_i(p,q,r)}{\partial q} x_i & \frac{\partial N_i(p,q,r)}{\partial r} x_i \\ \frac{\partial N_i(p,q,r)}{\partial p} y_i & \frac{\partial N_i(p,q,r)}{\partial q} y_i & \frac{\partial N_i(p,q,r)}{\partial r} y_i \\ \frac{\partial N_i(p,q,r)}{\partial p} z_i & \frac{\partial N_i(p,q,r)}{\partial q} z_i & \frac{\partial N_i(p,q.r)}{\partial r} z_i \end{pmatrix}$$

Interpolation: Hexahedron

- Transformation from (x,y,z) to (p,q,r)
  - In case of a cube (regular hexahedron)

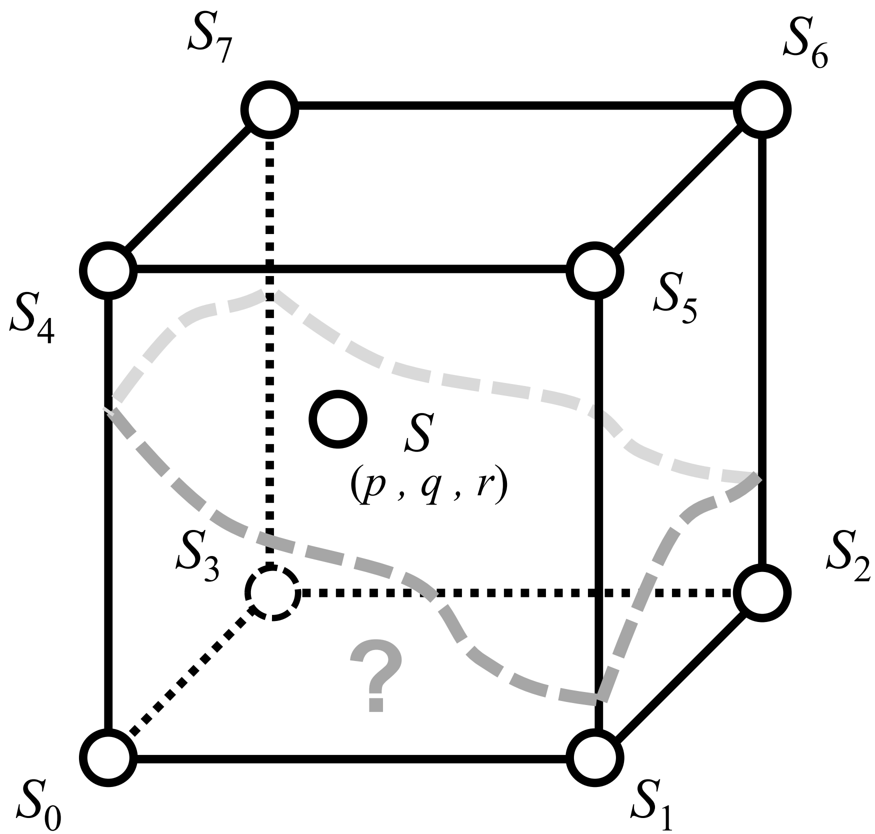
$$p = 2\frac{x - x_0}{x_1 - x_0} - 1$$

$$q = 2\frac{y - y_0}{y_1 - y_0} - 1$$

$$r = 2\frac{z - z_0}{z_1 - z_0} - 1$$

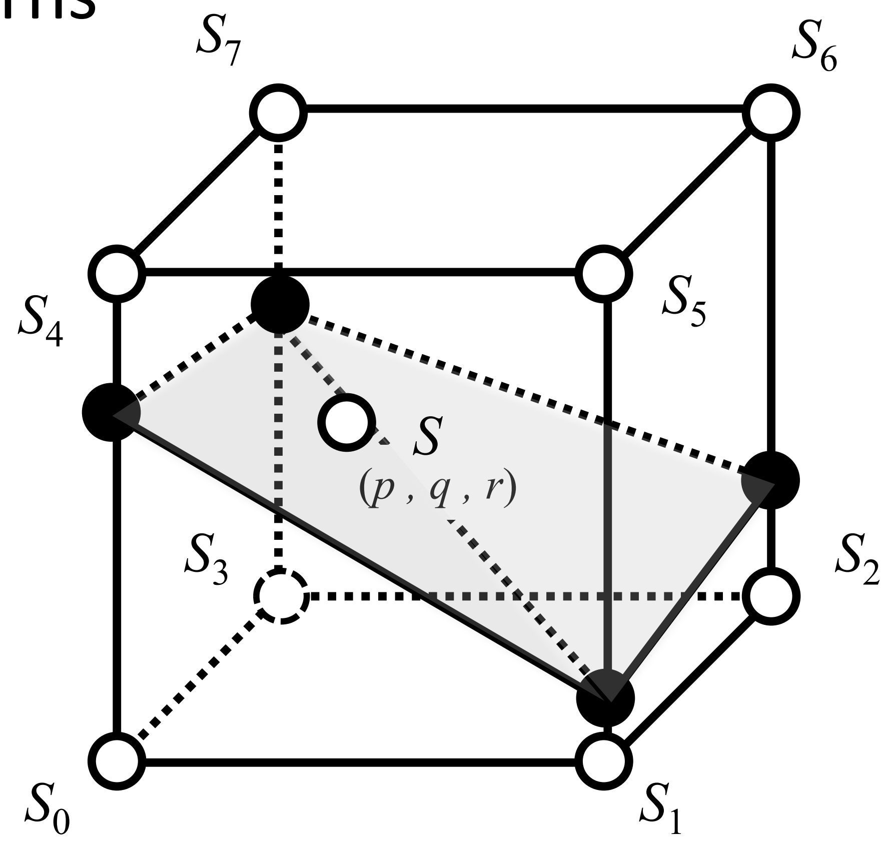
Isosurfaces within a cube

- A set of points have a scalar value S



Marching Cubes

- Intersection patterns
  - ? patterns



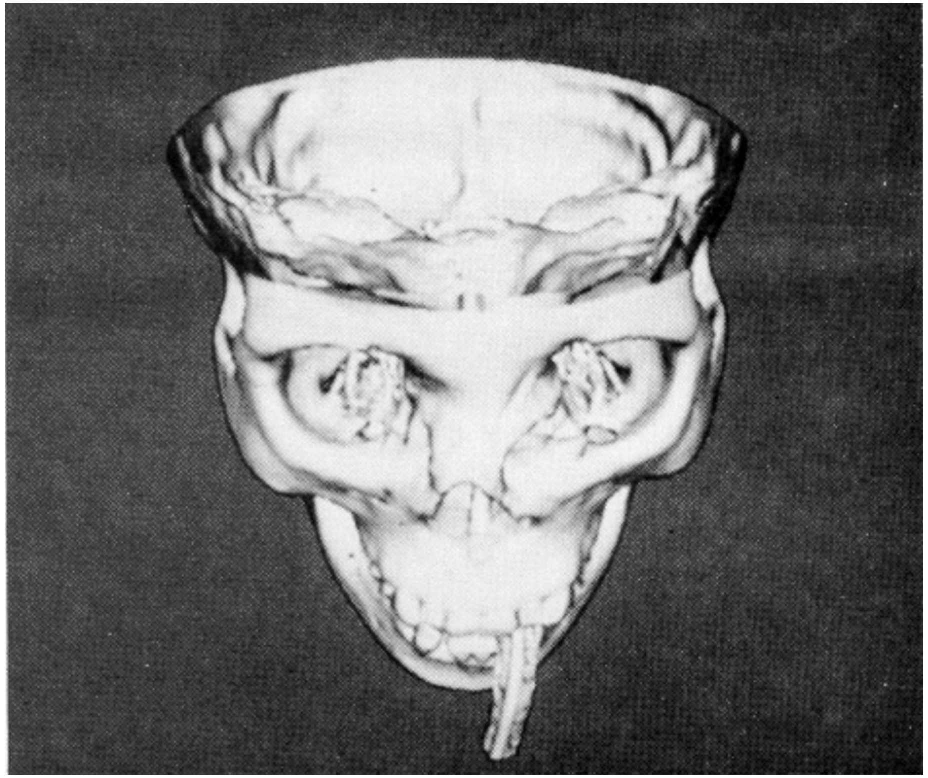
Marching Cubes

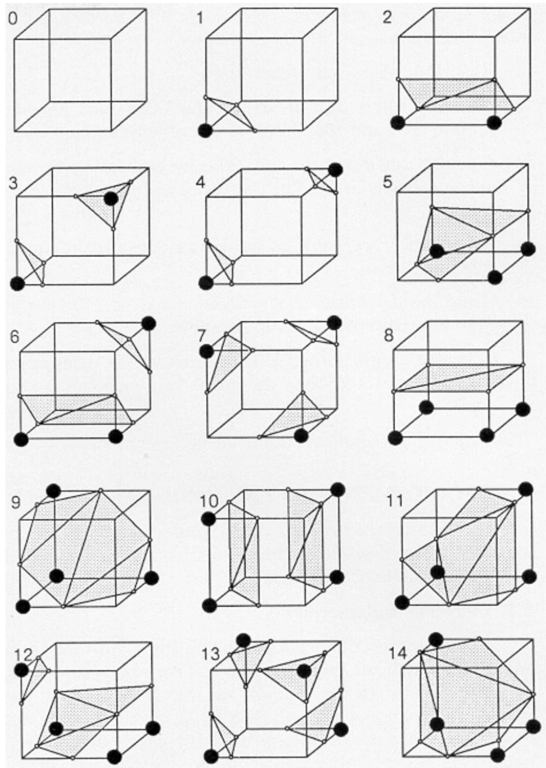
- Index calculation
  - logical sum of the scalar value and the threshold.

```
var index = 0;                           //   0 = 0000,0000
if ( S0 > S ) { index |=   1; } //   1 = 0000,0001
if ( S1 > S ) { index |=   2; } //   2 = 0000,0010
if ( S2 > S ) { index |=   4; } //   3 = 0000,0100
if ( S3 > S ) { index |=   8; } //   4 = 0000,1000

...

if ( S6 > S ) { index |=  64; } //  64 = 0100,0000
if ( S7 > S ) { index |= 128; } // 128 = 1000,0000
```

Marching Cubes

- W.Lorensen and H.Cline, “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”, Computer Graphics, 21 (4): 163-169, July 1987.





Polling

- Take the poll
  - Student ID Number
  - Name