Algorithms for Contour Maps and Isosurfaces

Alexander Pasko

pasko@acm.org

http://hm.softalliance.net/

Alexander Pasko, Evgenii Maltsev

Contents

Contour Map

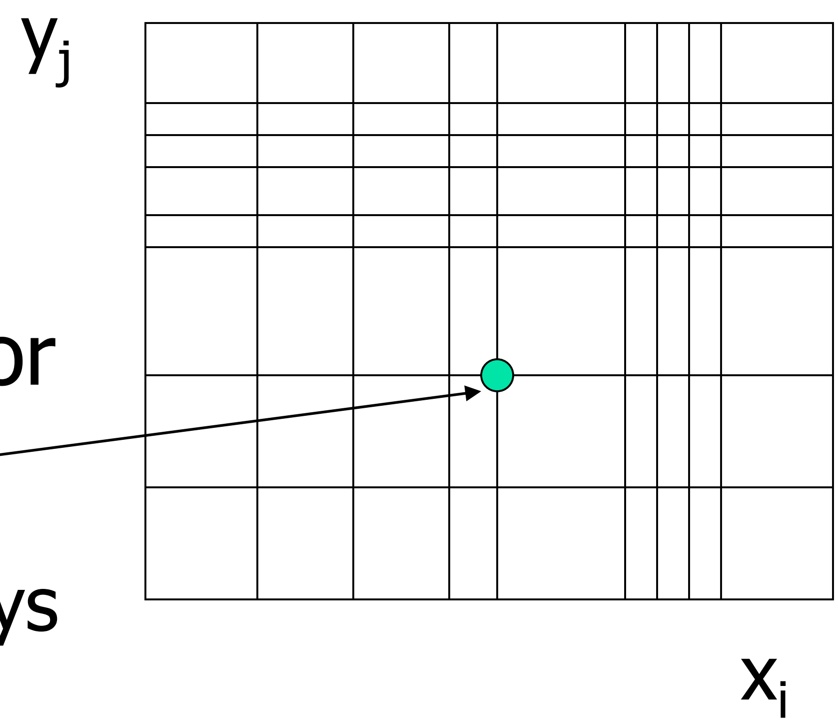
Data:

1) Function $z = f(x,y)$ or

2D array $F_{ij}=f(x_i,y_j)$

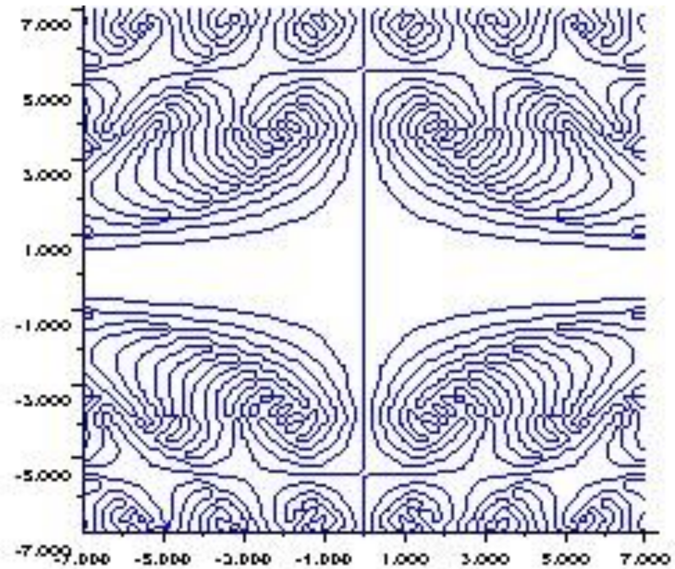
- + two linear scalar arrays x_i and y_j
- x_i, y_j can be given by default

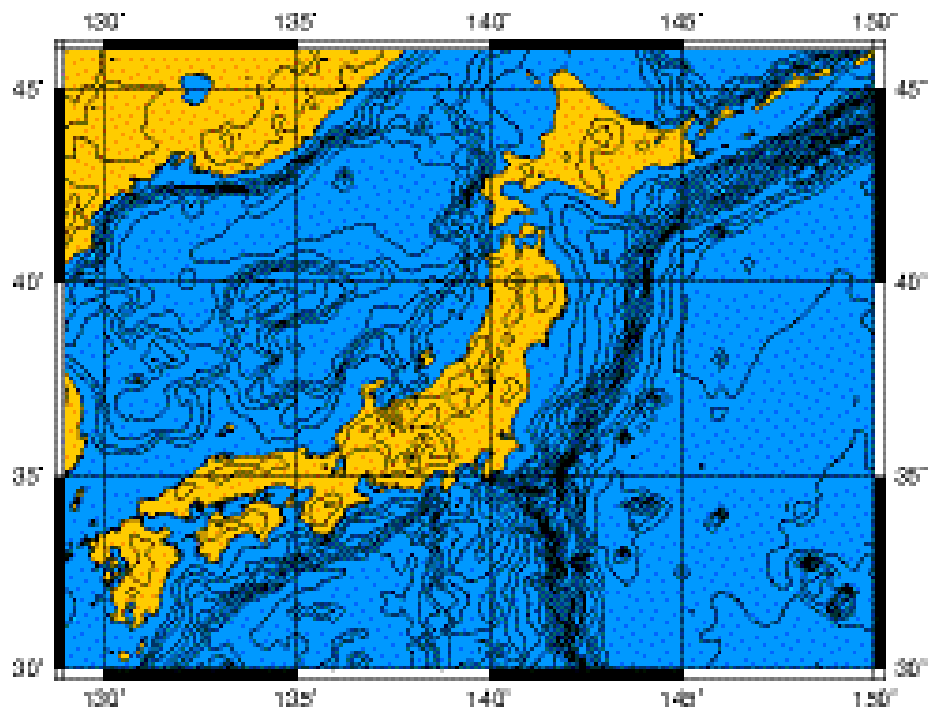
2) Levels c_k



Contour Map

Contour is
an "implicit" curve
or several curves
$f(x,y)=c_k$





Other terms:
iso-contours, isolines,
topographic map

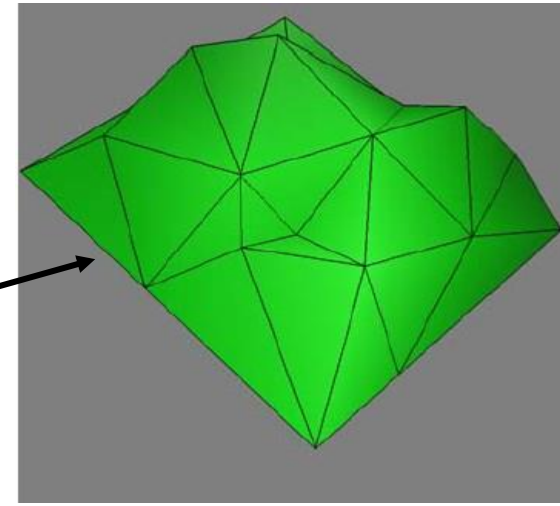
Contour is defined with:

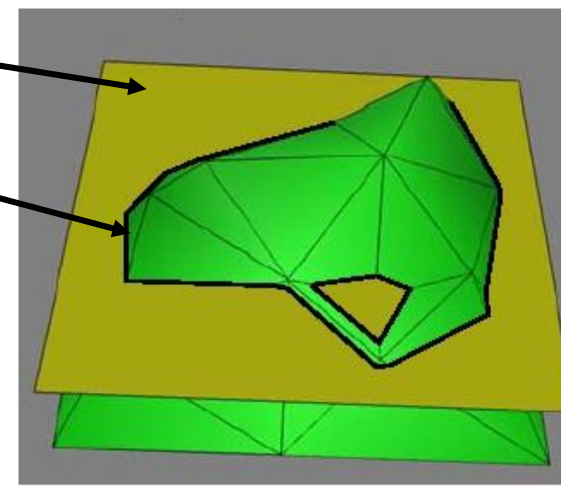
1) Surface z=f(x,y)

2) Plane z=c

3) Intersection between the surface and the plane

4) Projection of the curve onto xy-plane.





Images by P. Agarwal et al., Duke University

Contour: Simple Example

Data:

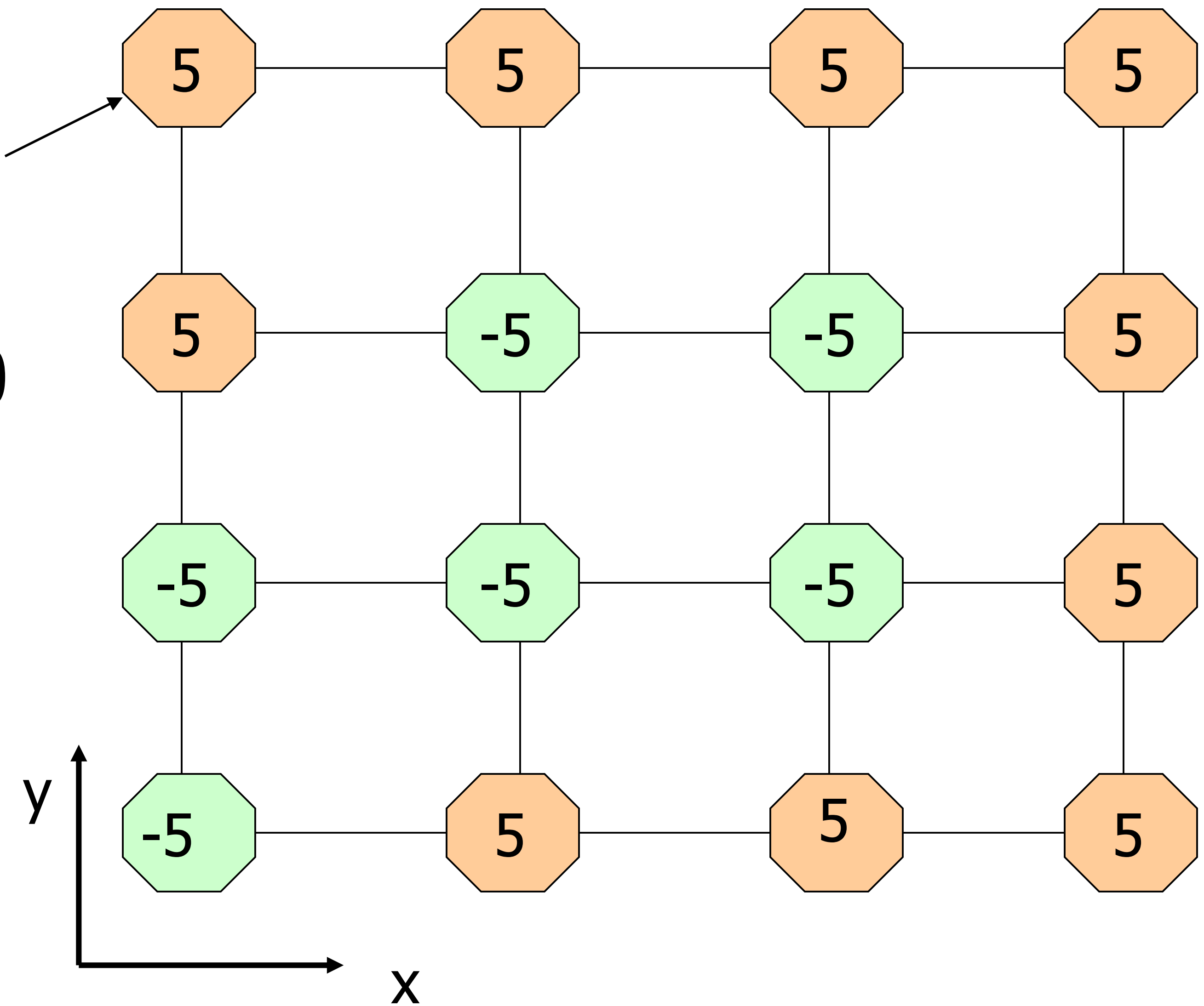
$z_{ij}=f(x_i,y_j)$

Grid 4x4

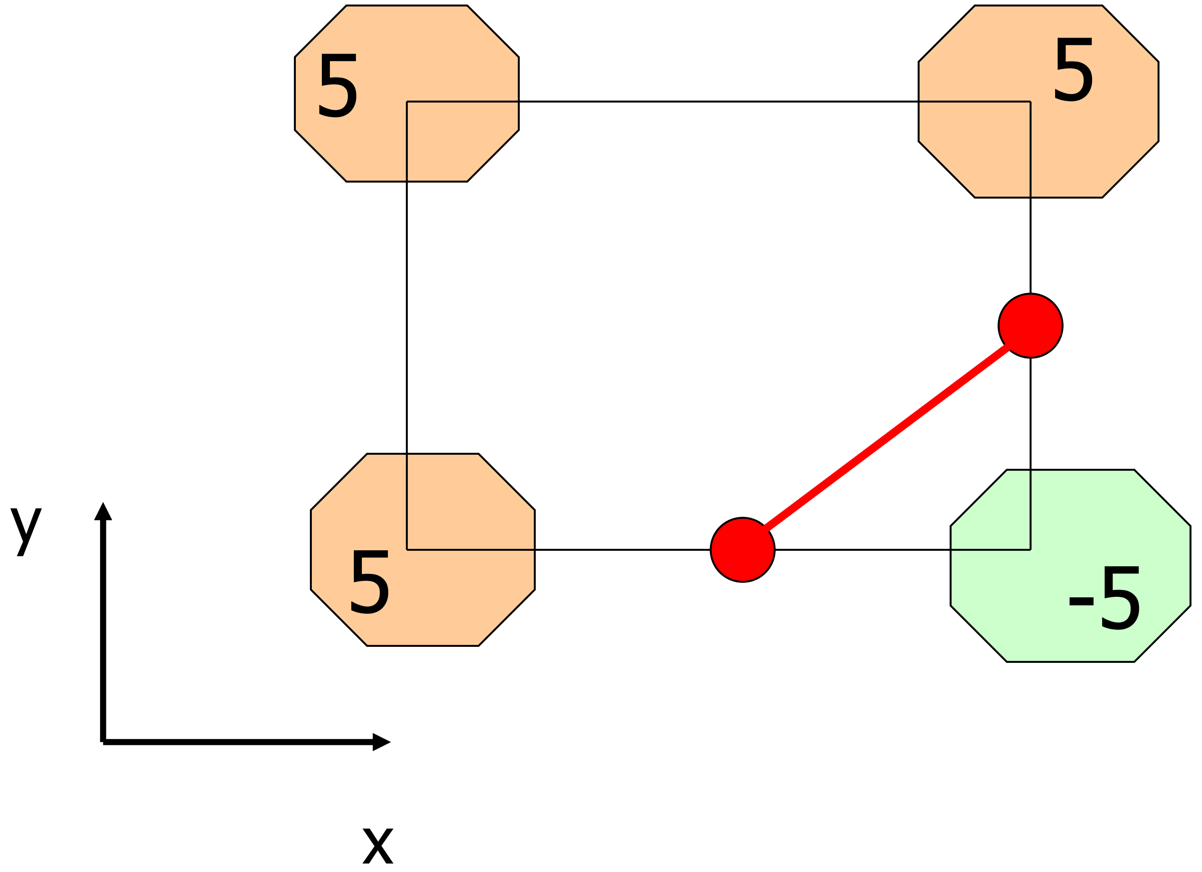
Level z=0

Contour

f(x,y)=0



One Cell



Contour

f(x,y)=0

Steps of Contour Generation

1. Select a cell with
   an intersection point
2. If no cells to process – End
3. Process a cell:
   construct segments of the contour
4. Select next cell
5. Repeat Step 2

Exhaustive Enumaration

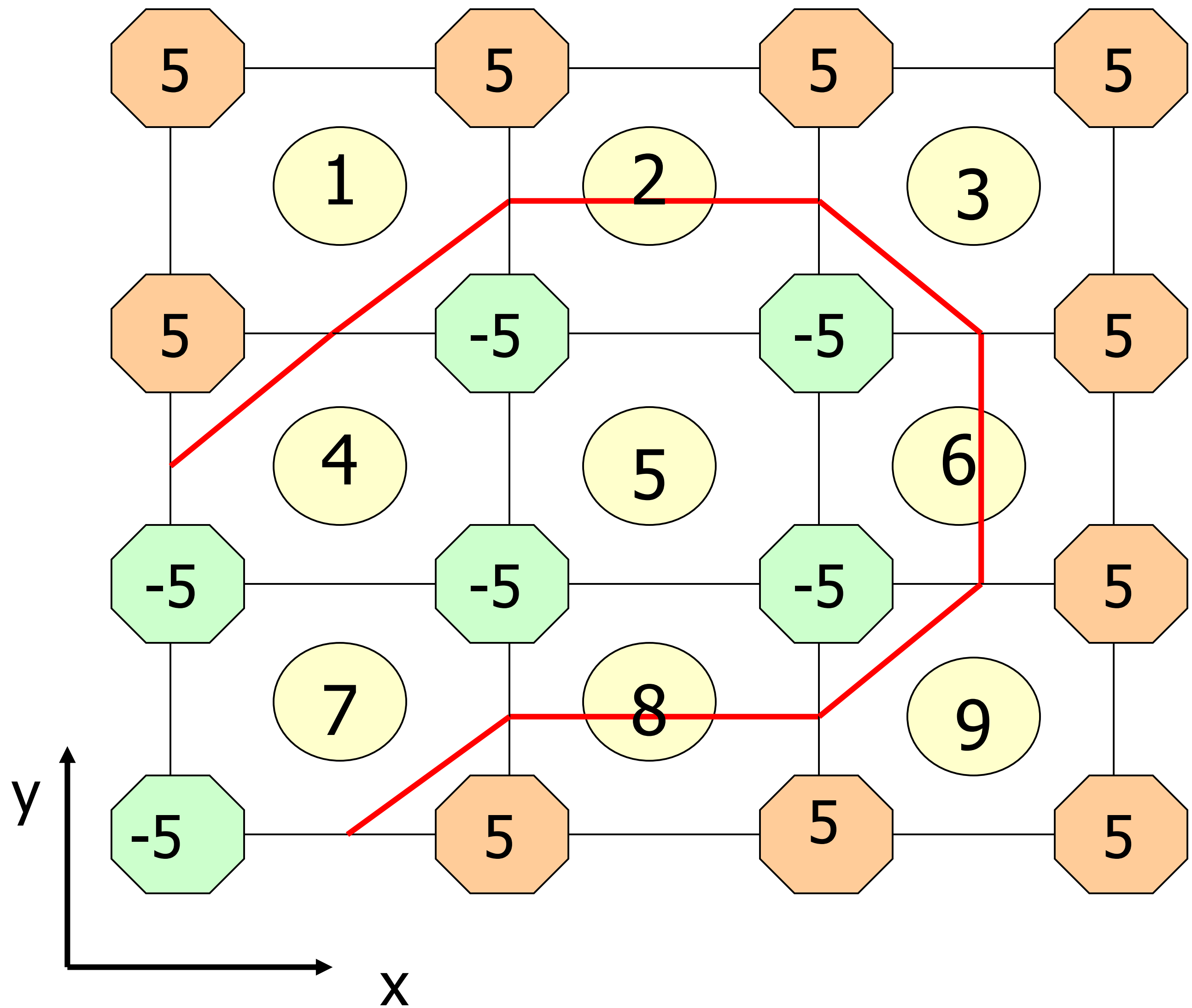
Check all MxN cells as:

```
for (i=1,M){
    for (j=1,N){
        select Cell_ij
    }
}
```

Exhaustive enumeration example

Contour

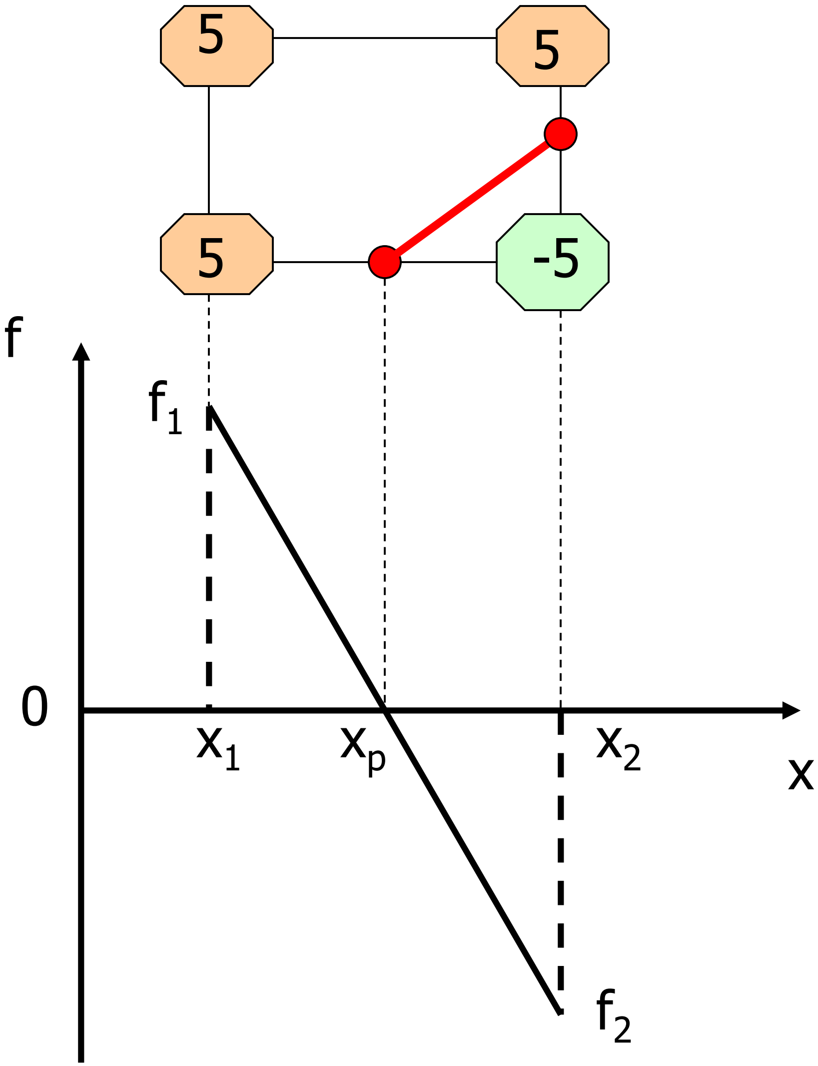
f(x,y)=0



Cell processing

1) Find all edge-surface intersection points – vertices of contour lines
2) Connect vertices into segments
3) Add segments to the contour or render segments

Edge intersection



Linear interpolation

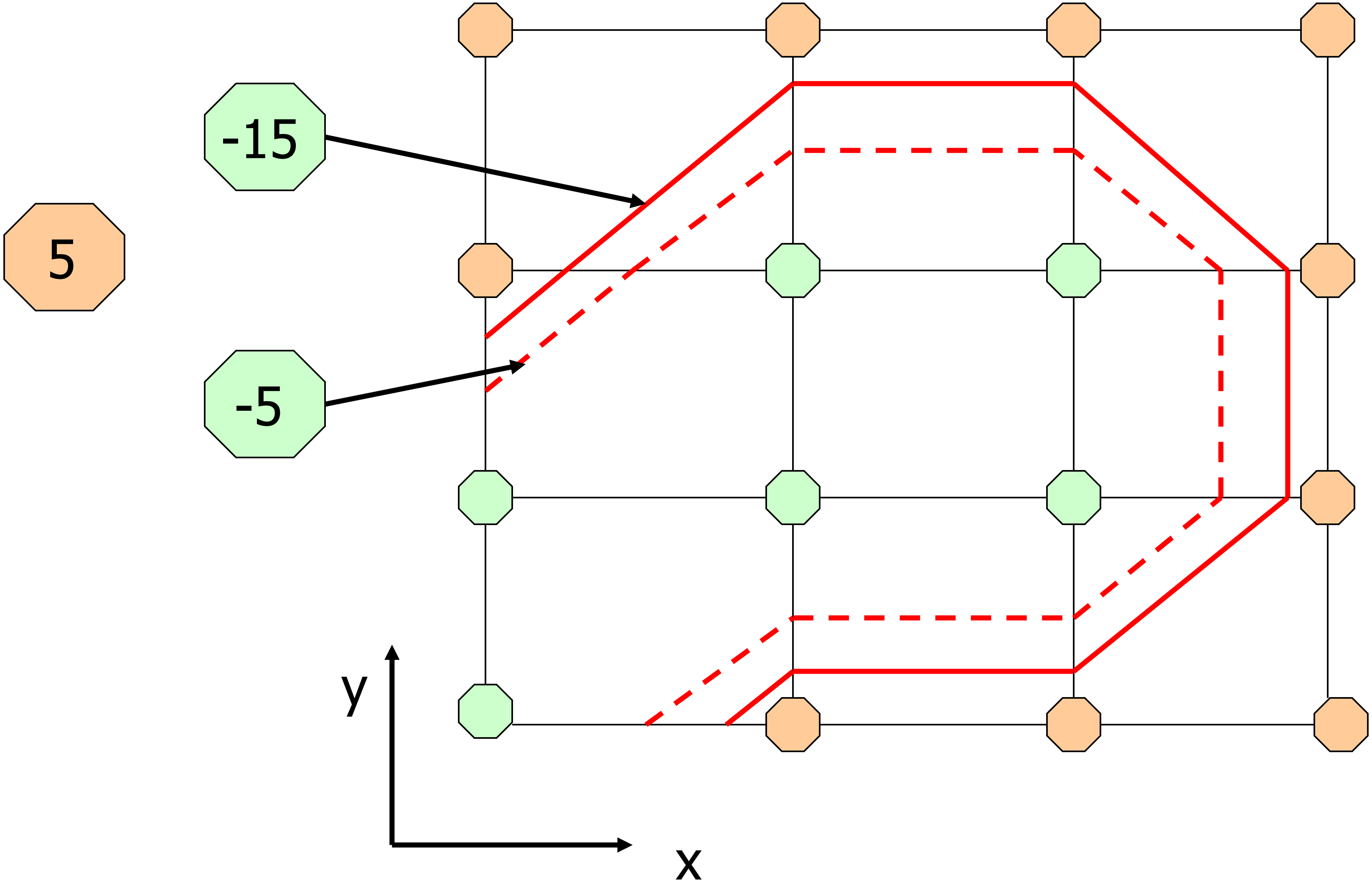
$$f = f_1(1-t) + f_2 t$$

$$t = \frac{x - x_1}{x_2 - x_1}$$

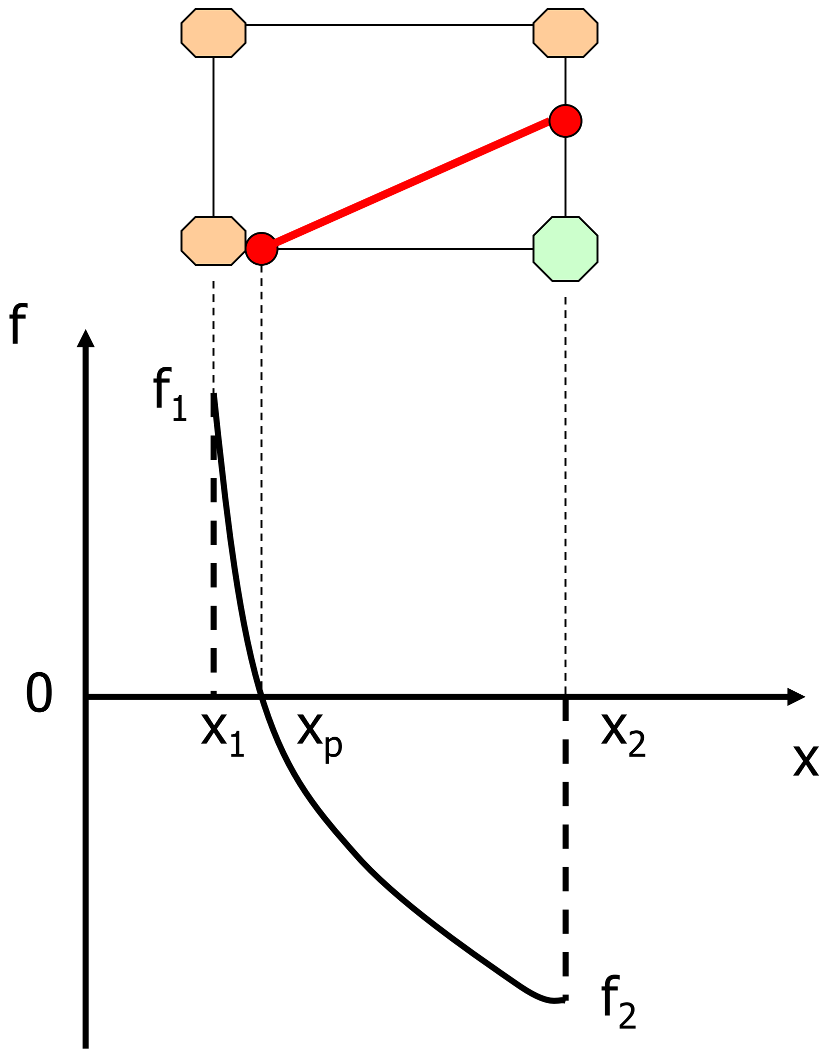
For $f=0$

$$x_p = x_1 + \frac{f_1(x_2 - x_1)}{f_1 - f_2}$$

Change of function values



Edge intersection



Search on the edge

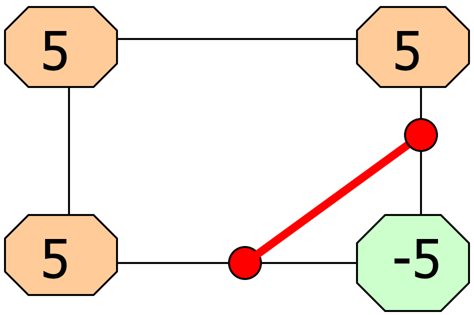
1) Continuous f(x,y)

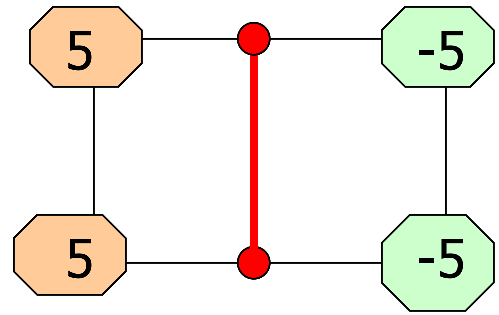
2) Analytical solution for polynomial f

3) Numerical solution:
   - bisections
   - Newton search
   - …

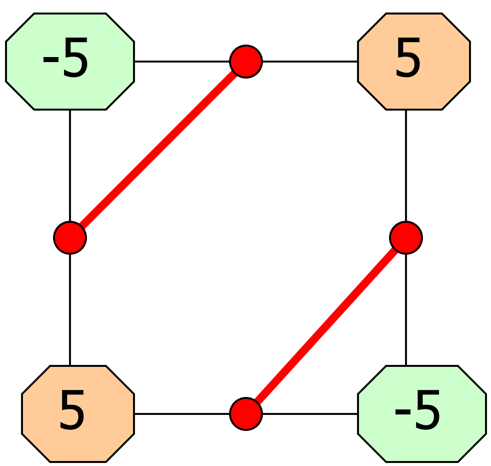
Typical Cases in the Cell

Two intersection points:





Four intersection points:



The case of one intersection point is reduced to 2 points by f+df in a vertex

All Cases in the Cell

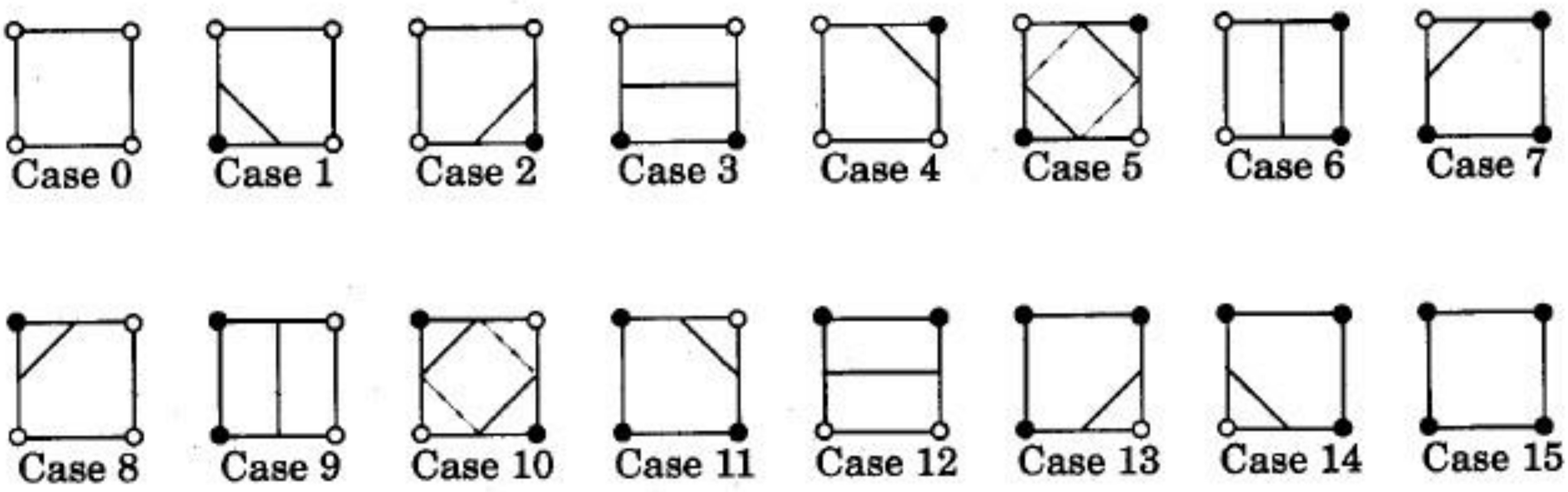
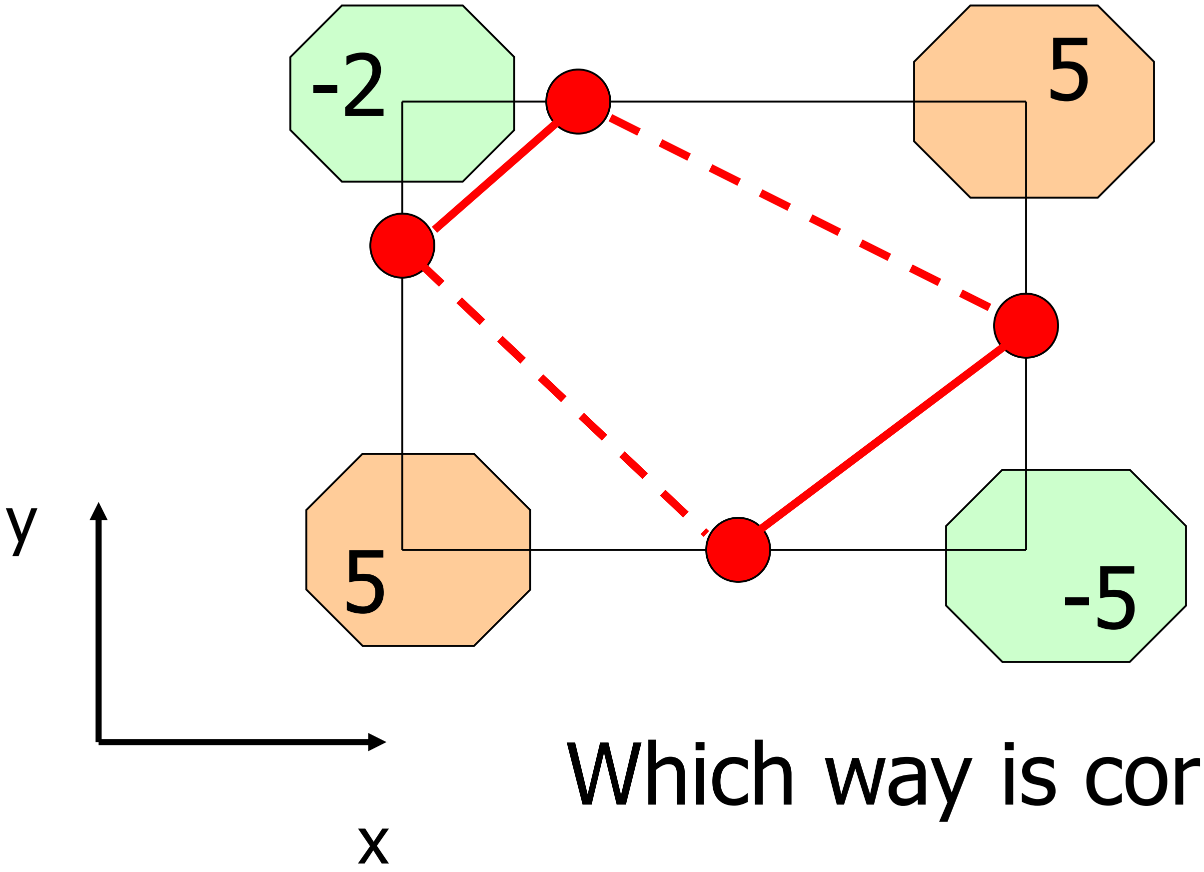


Image by P. Rheingans,

Topological Ambiguity

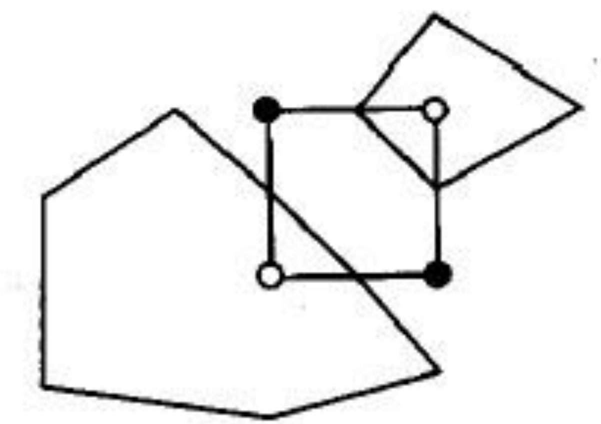


Contour

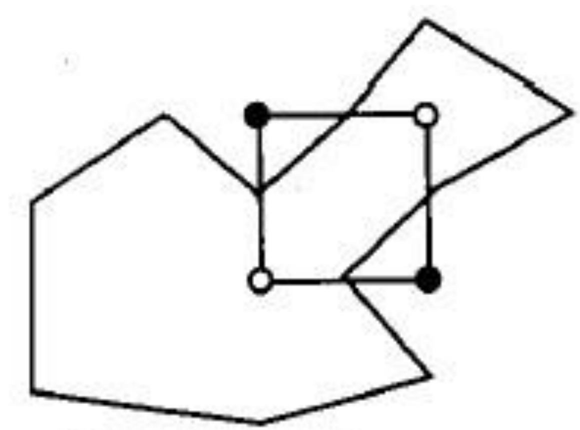
f(x,y)=0

Which way is correct?

Two possible contours with one ambiguous cell:



a) break contour



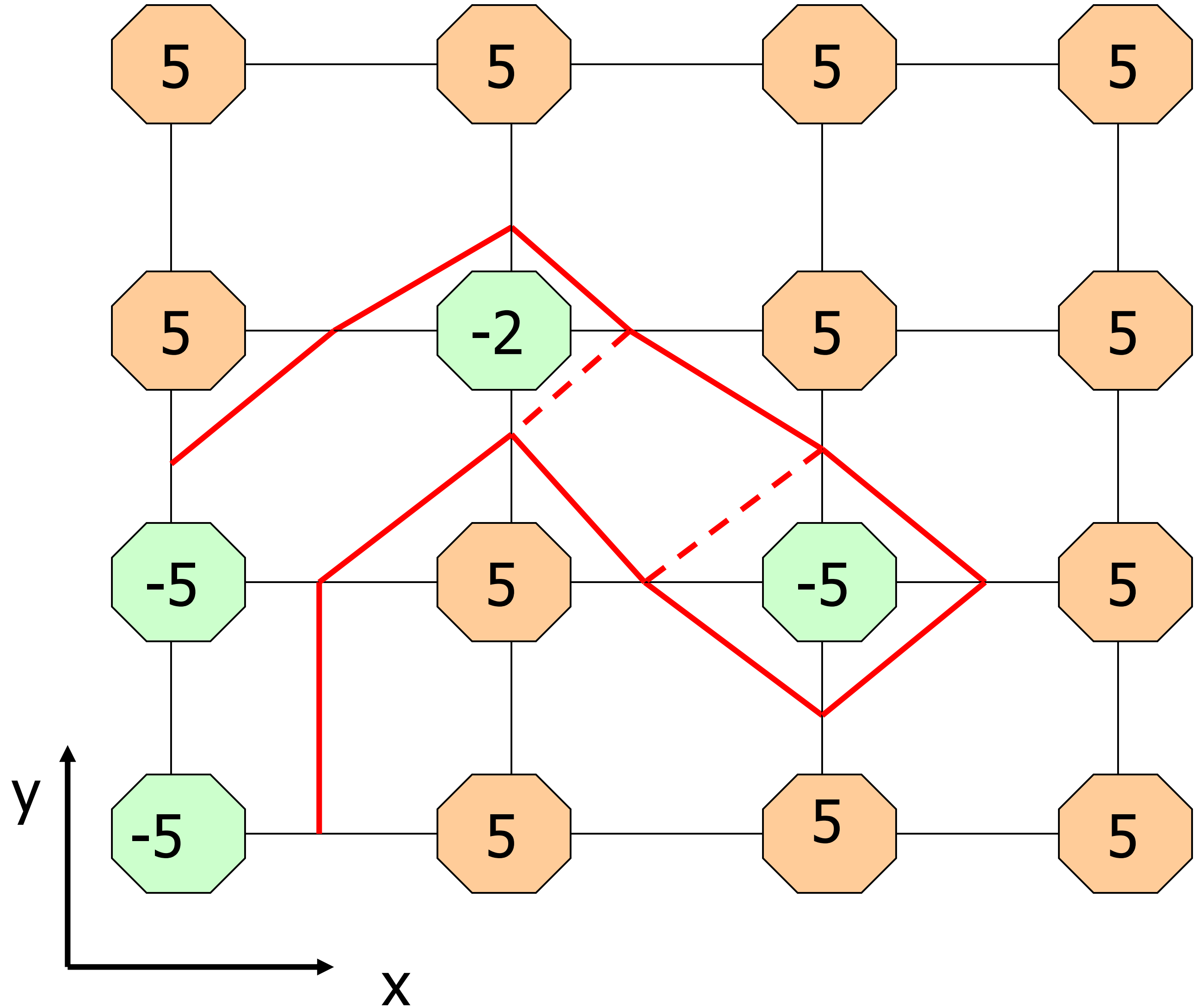
b) join contour

Image by P. Rheingans,

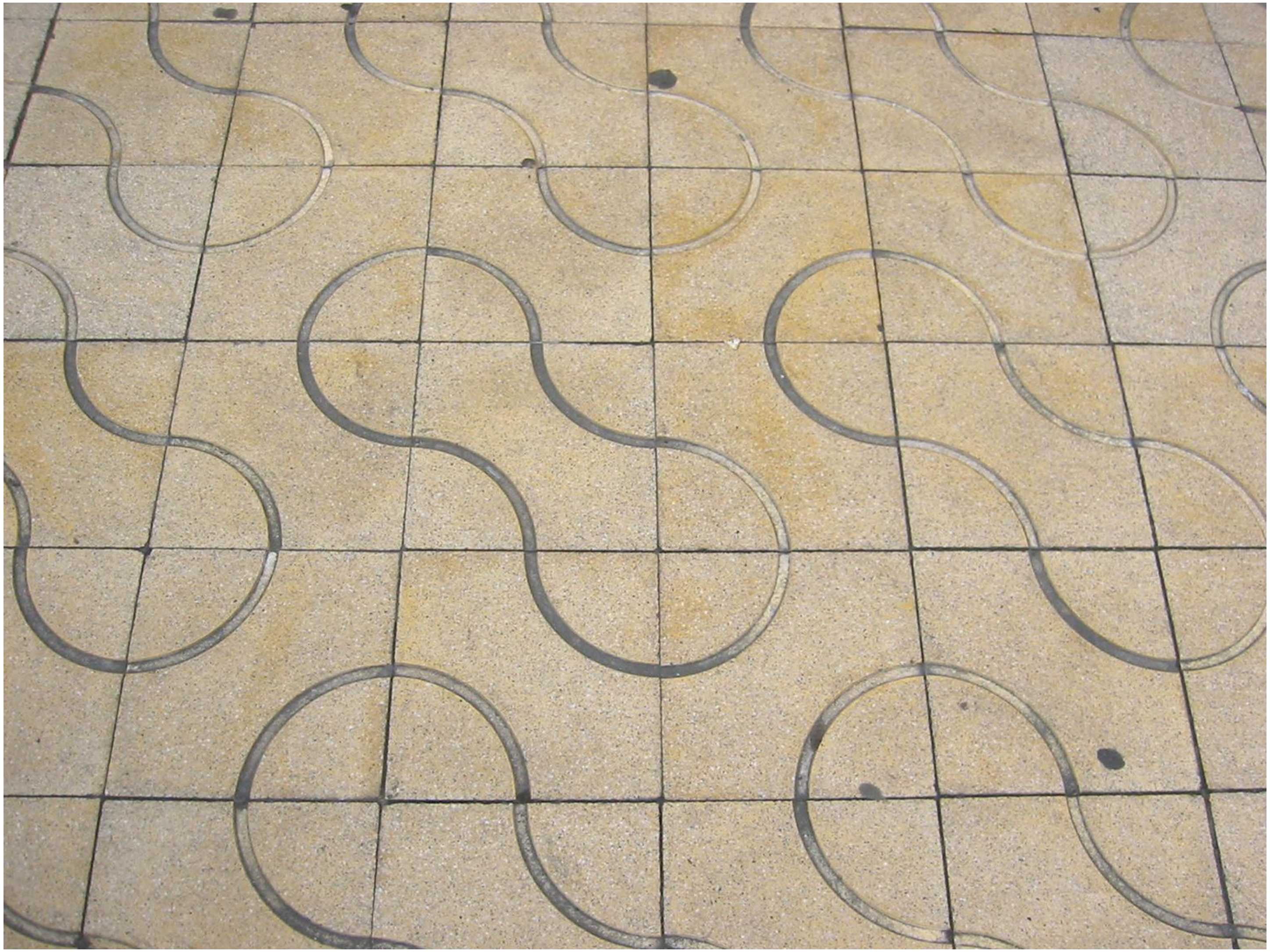
Topological ambiguity example

Contour

f(x,y)=0



Ueno: topological ambiguity example

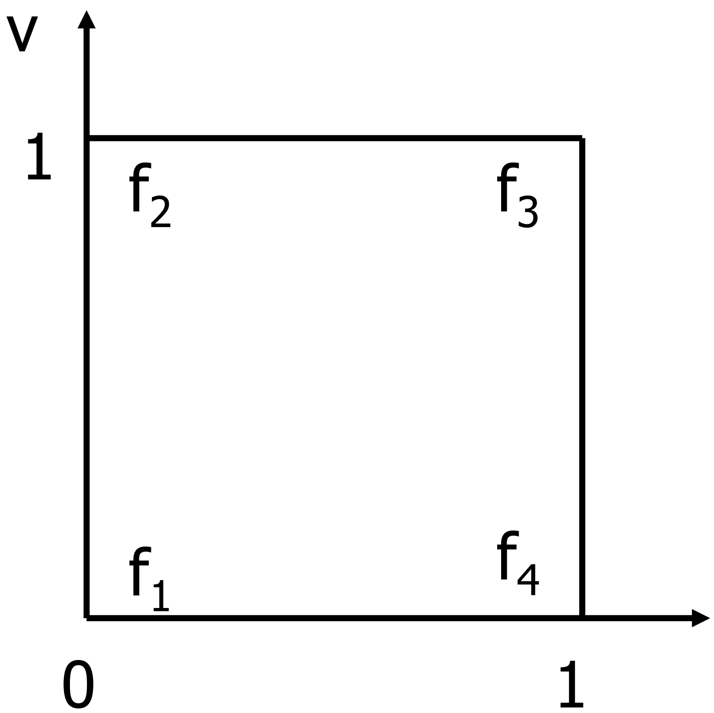


The worst case of a contour map

Resolving the Ambiguity with Hyperbolic Arcs

1) Bilinear interpolation inside the cell
2) Contour as a hyperbola
3) Calculate center of hyperbola
4) Use center of hyperbola to resolve the topological ambiguity

Bilinear interpolation inside the cell

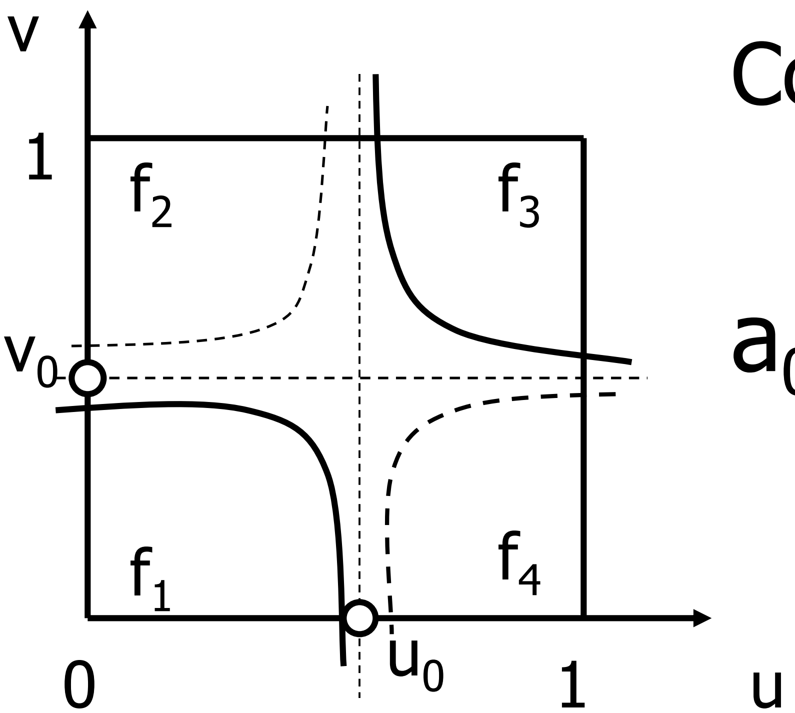


$$f = [f_1(1-u) + f_4u](1-v) + [f_2(1-u) + f_3u]v$$

$$\Downarrow$$

$$f = a_0uv + a_1v + a_2u + a_3$$

Contour as a hyperbola



Contour f=0

⇩

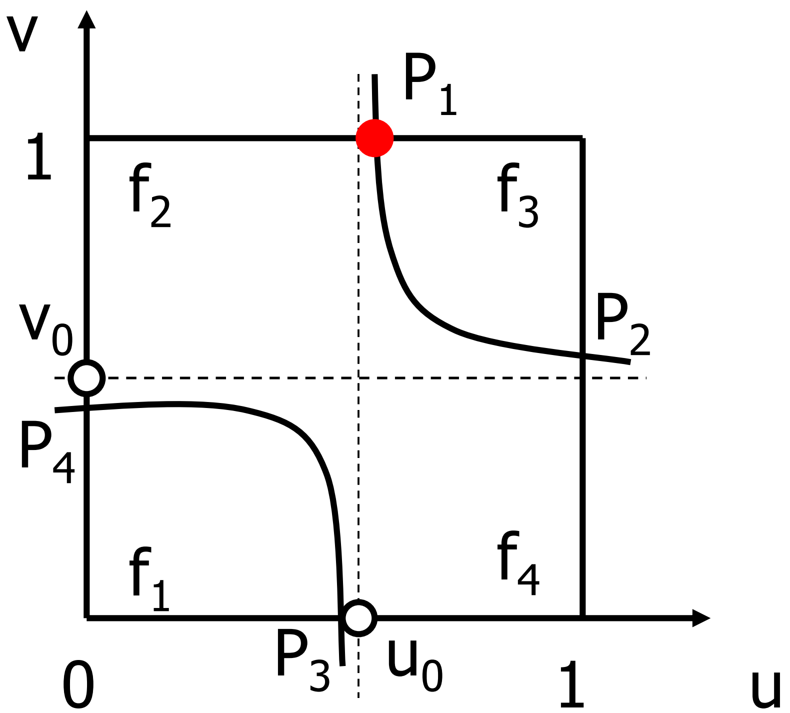
$$a_0uv + a_1v + a_2u + a_3 = 0$$

$$v = \frac{-a_3 - a_2u}{a_1 + a_0u}$$

Center of hyperbola:

$u_0 = -a_1/a_0 \quad v_0 = -a_2/a_0$

Resolving ambiguities



Four intersection points:

$P_1(u_1,1)$, $P_2(1, v_2)$
$P_3(u_3,0)$, $P_4(0, v_4)$

Hyperbolic arcs selection:

if($u_1 > u_0$) $P_1 P_2$ and $P_3 P_4$

if($u_1 < u_0$) $P_1 P_4$ and $P_2 P_3$

Isosurface

Data:

1) Function $\xi = f(x,y,z)$ or

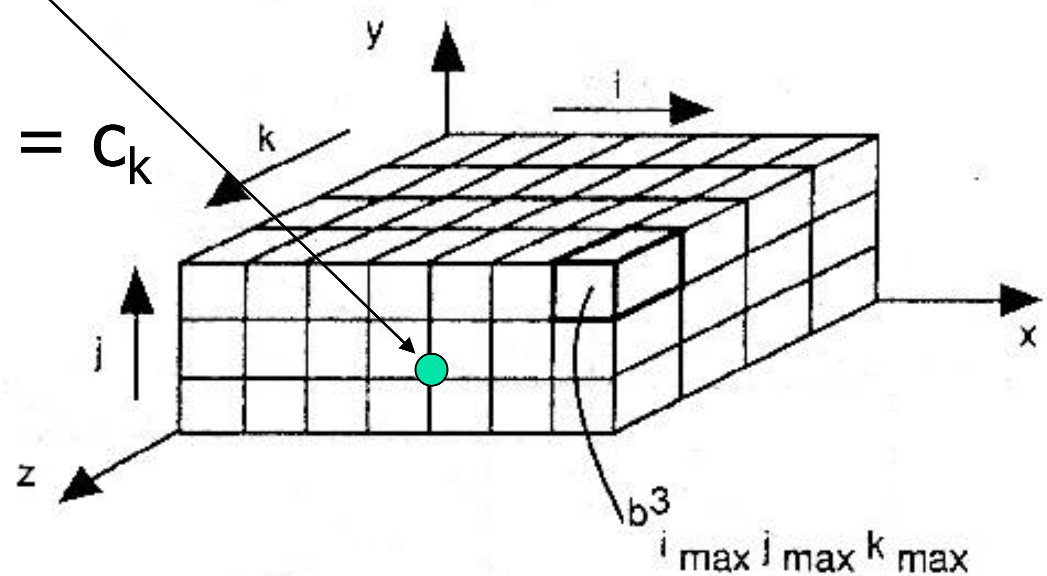
3D array $\xi_{ijk}=f(x_i,y_j,z_k)$

2) Levels c_k

Shape model:

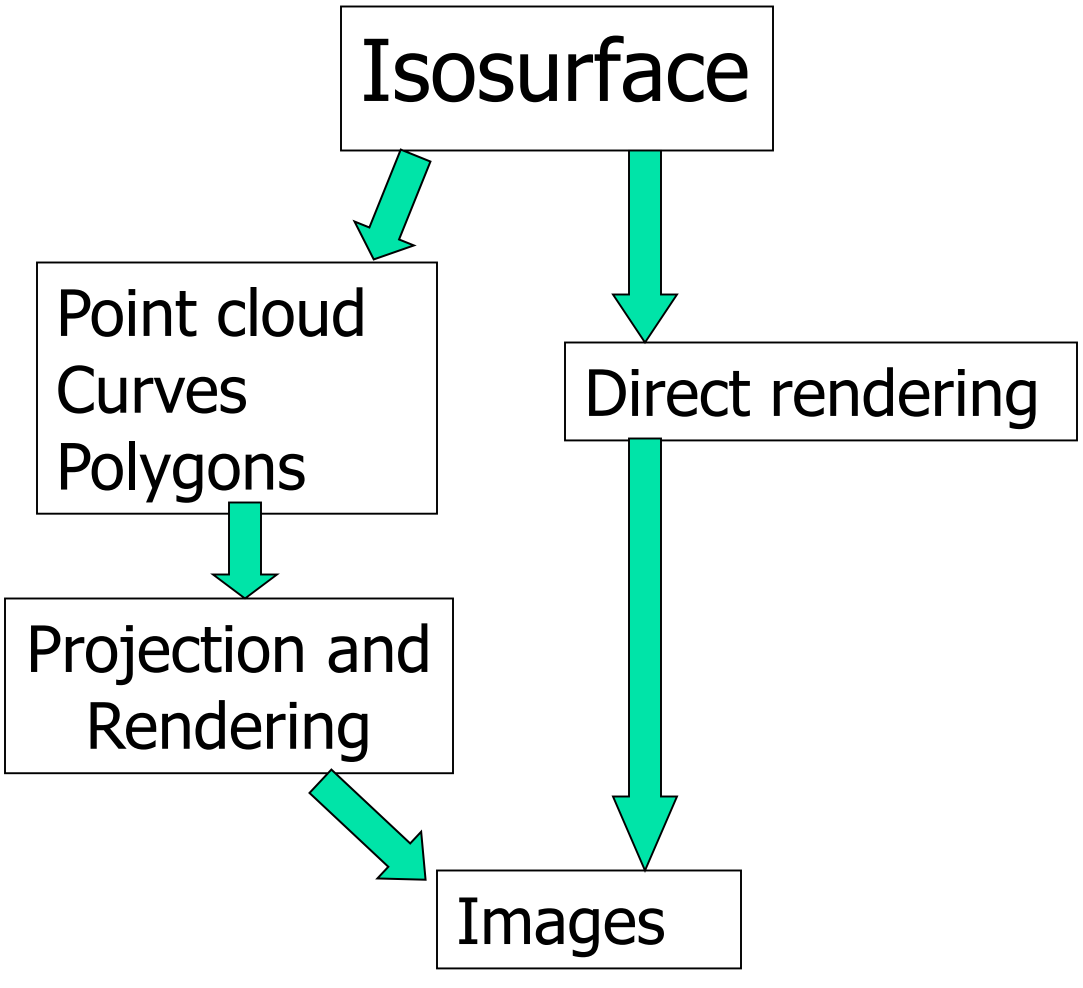
isosurfaces $f(x,y,z) = c_k$

3D grid & data



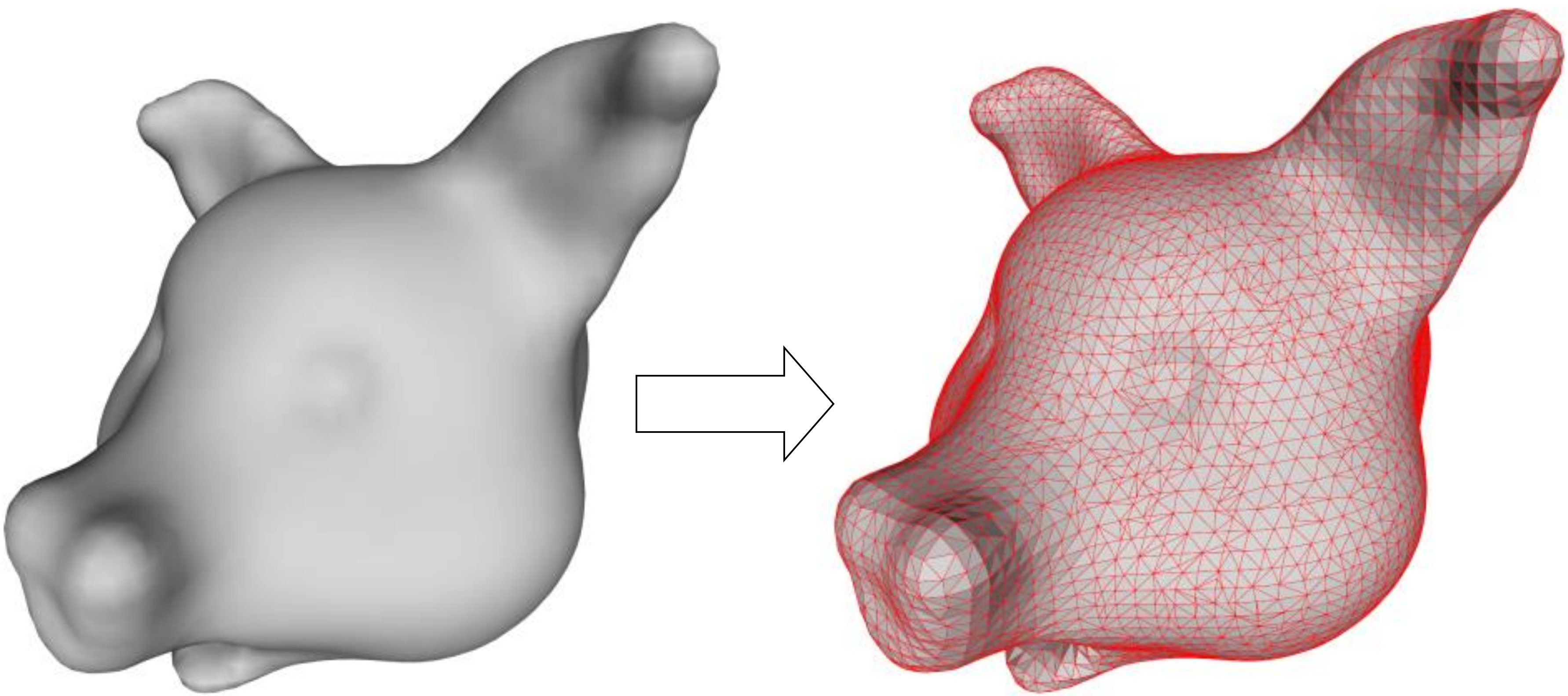
Other terms:
implicit surface,
3D contour,
equipotential surface

Isosurface
Transformations and Rendering



Isosurface Polygonization

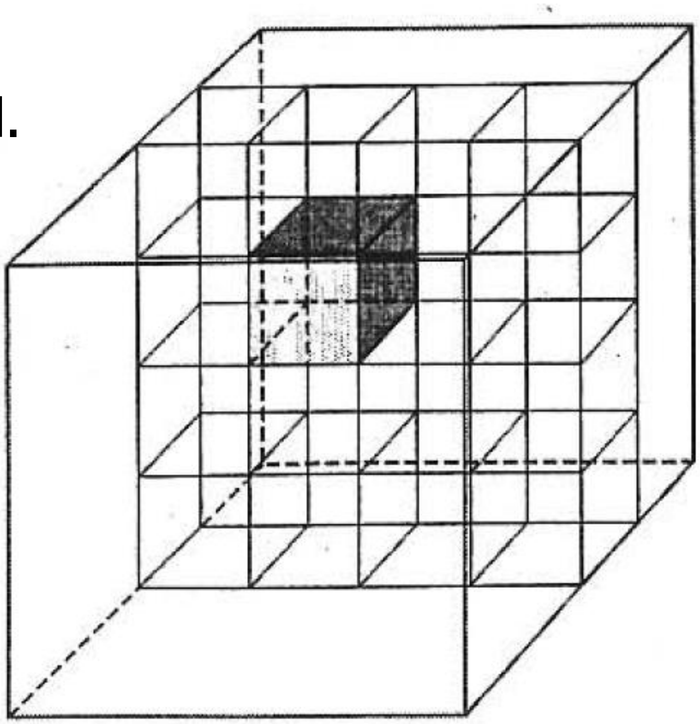
Polygonization is the generation of a polygonal approximation of an isosurface.



Images by J. Bærenten

Exhaustive Enumeration

- Discrete data (voxel data from CT or MRI) is usually a set of points with scalar values in the nodes of a *regular* (number of neighbors is constant) and *uniform* (constant step size) grid.
- Exhaustive enumeration
  - examines **every** cell, determining which cells intersect the surface;
  - is very fast, because data values are known;
  - surface/edge intersections are usually computed by linear interpolation.
  - for N^3 cells and N>1000, memory management becomes a problem.
- Example: "*Marching Cubes*" [Lorensen and Cline 1987] processes a rectangular grid one plane at a time. Each cubic cell is polygonized according to a 256-entry table of ready polygon configurations.



Cell Polygonization

- *Cell polygonization* generates a set of polygons for the surface patch inside a single transversal cell. Steps:

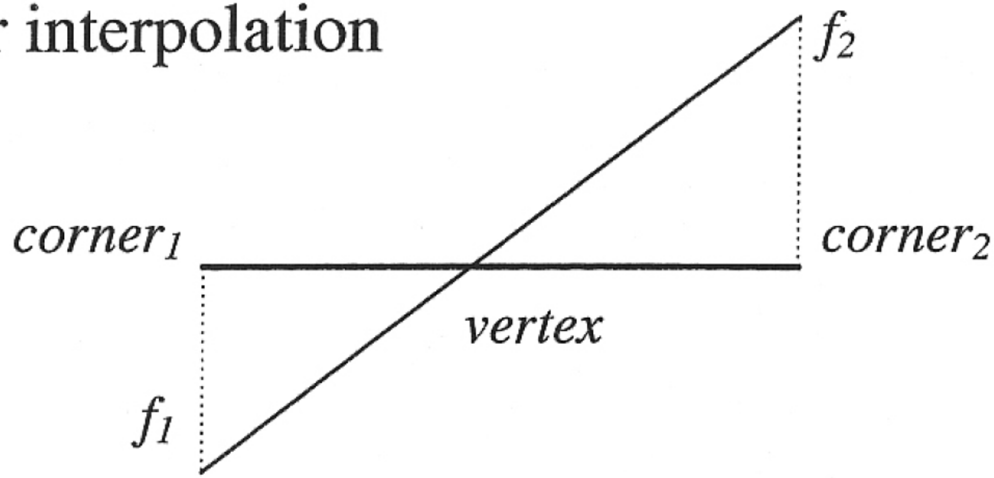
  1) Detect a cell edge which intersects the surface (different function signs in the endpoints). Such edge is assumed to contain a single intersection.

  2) Compute an intersection point (surface vertex).

  3) Connect surface vertices to form polygons.

Surface vertex computation

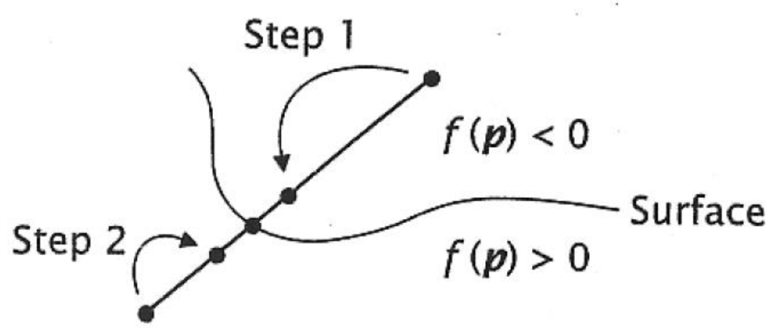
- Linear interpolation



vertex = α *corner1* + (1 - α) *corner2*

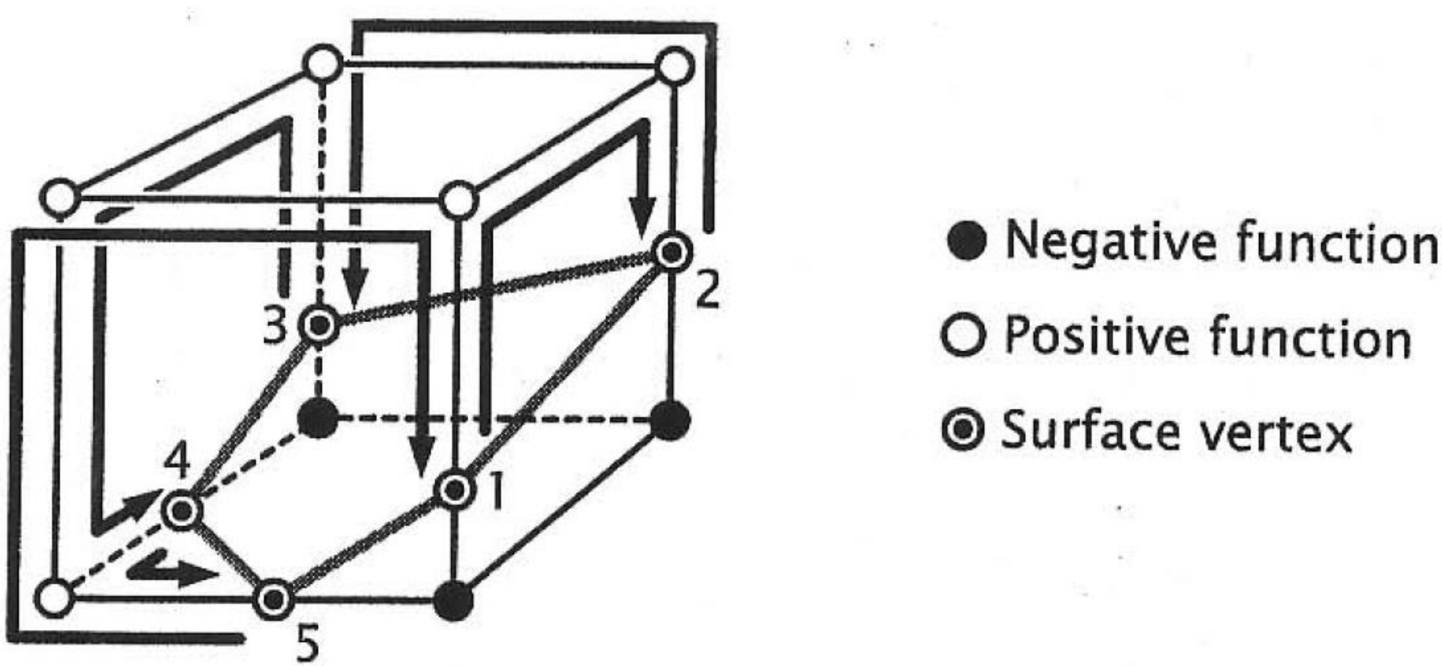
α = *f2* / (*f2* - *f1*)

- Binary section (binary subdivision)



Surface vertices connection

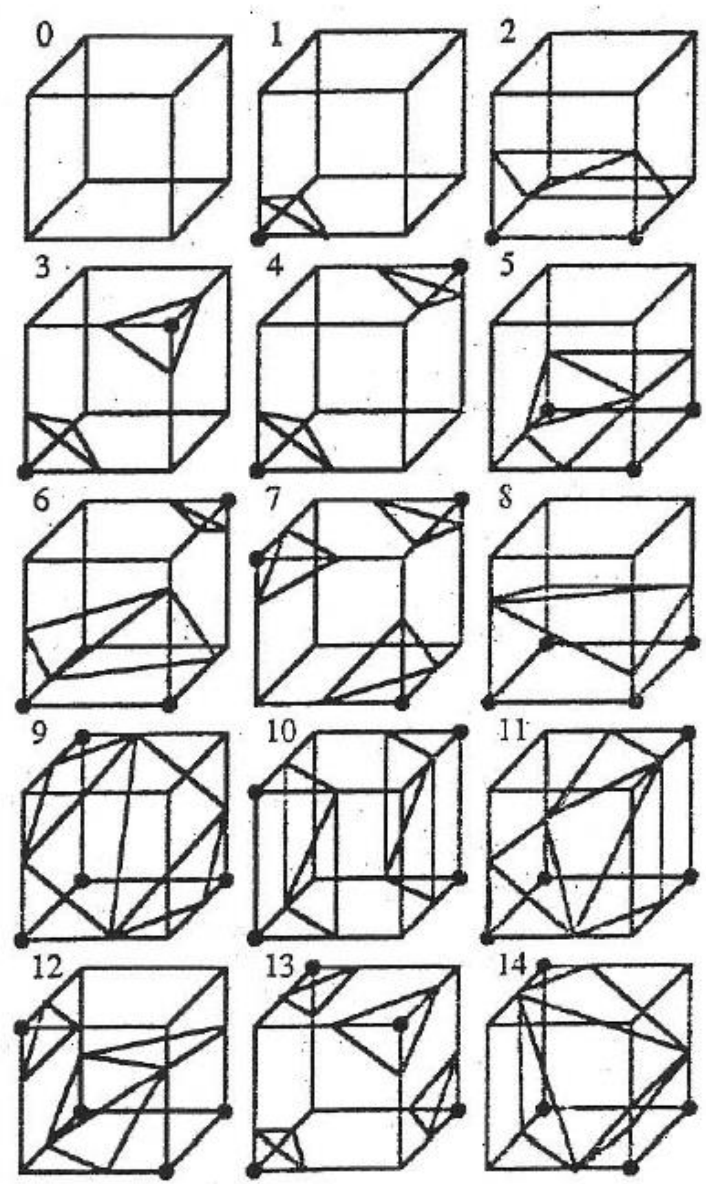
Cubic cell
Algorithm starts with a transversal edge, looks for the next transversal edge in the face and stops when the polygon is complete.



Surface vertices connection

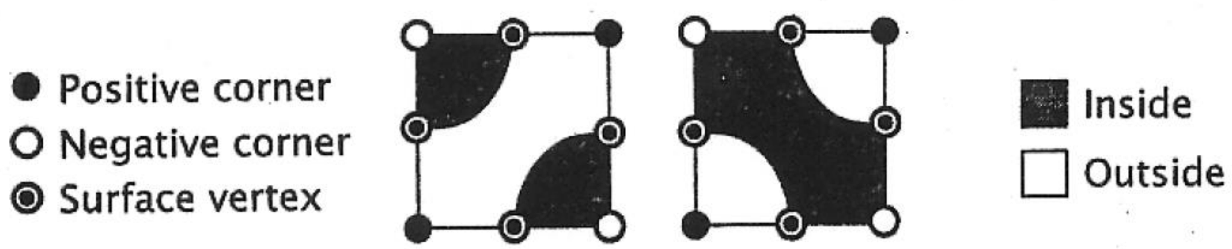
Table for cubic cells ("Marching Cubes")

The configuration of the set of polygons for a cubic cell depends on the number of cell corners with positive function values. For 8 corners, there are $2^8 = 256$ possible configurations. Only 15 basic configurations have to be stored. Others are equivalent to them due to symmetry and rotations.

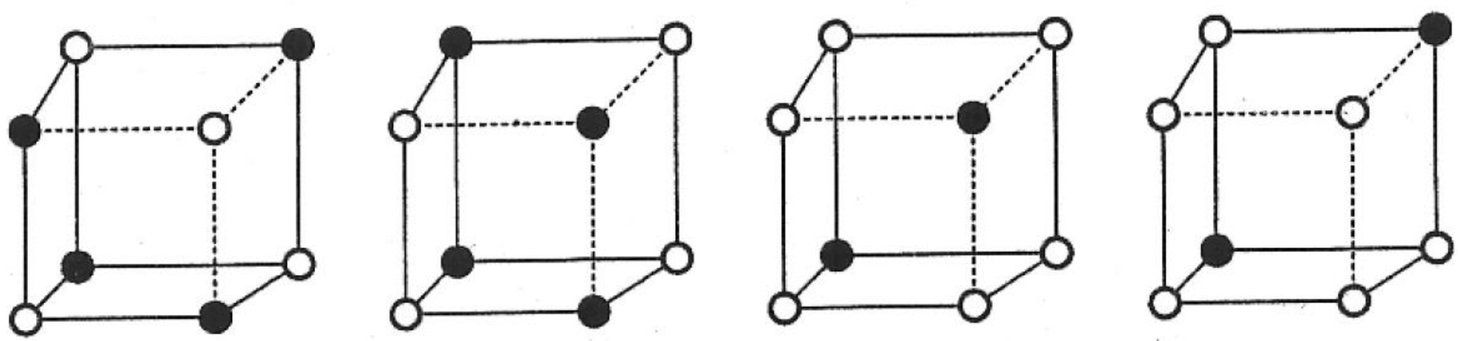


Topological Ambiguities

- Ambiguity occurs for certain configurations at the cell level.
- Alternate surface vertex connection for a cell face:



Ambiguous corner configurations for a cube



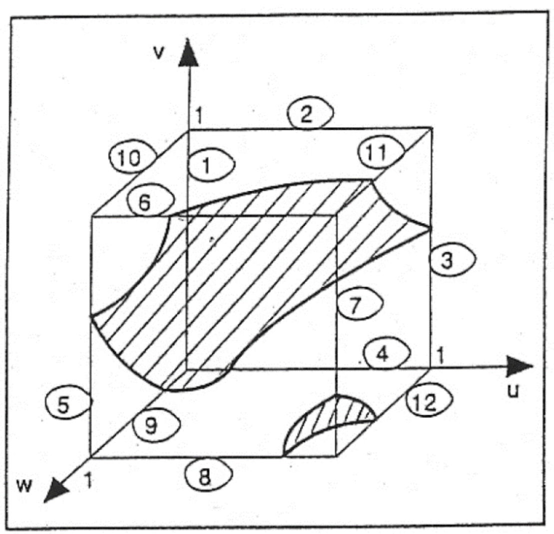
Polygonization with Hyperbolic Arcs

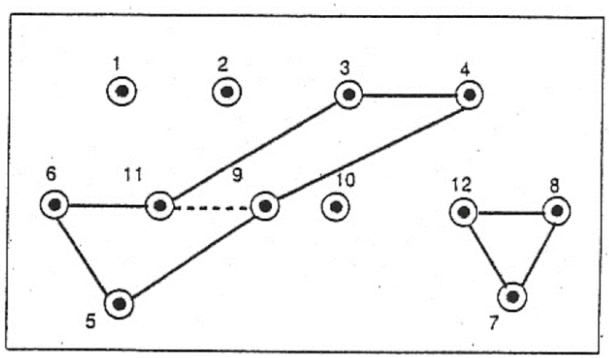
- Class: continuous data and discrete data (with trilinear interpolation).
- Spatial partitioning: exhaustive enumeration with the given number of cells for each axis.
- Surface vertex computation: linear interpolation or binary search.
- Surface vertices connection: algorithm of a connectivity graph construction and tracing.
- Ambiguity: trilinear interpolation in the cell and local bilinear interpolation on the cell face.

http://hyperfun.org/wiki/doku.php?id=frep:isopol

Connectivity graph construction and tracing

1) Process 6 cell faces independently
2) Resolve topological ambiguities on each cell
3) Construct a graph with 12 nodes representing edges of the cell
4) Nodes in the graph are connected if there is a hyperbolic arc connecting them on some face
5) Find all cycles in the connectivity graph – they correspond to the polygons



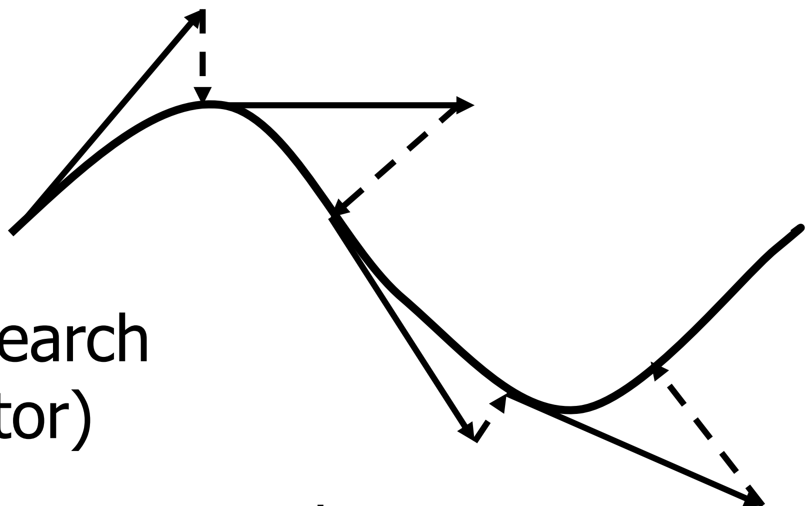


Other Methods for Contouring

- Predictor-corrector continuation
- Subdivision
- Shrinkwrap

Predictor-Corrector Continuation

1) Select an initial contour point
2) Calculate the tangent line
3) Make step along the tangent direction (predictor)
4) Correct the selected point by search in the normal direction (corrector)
5) Connect the previous and the new points by a segment

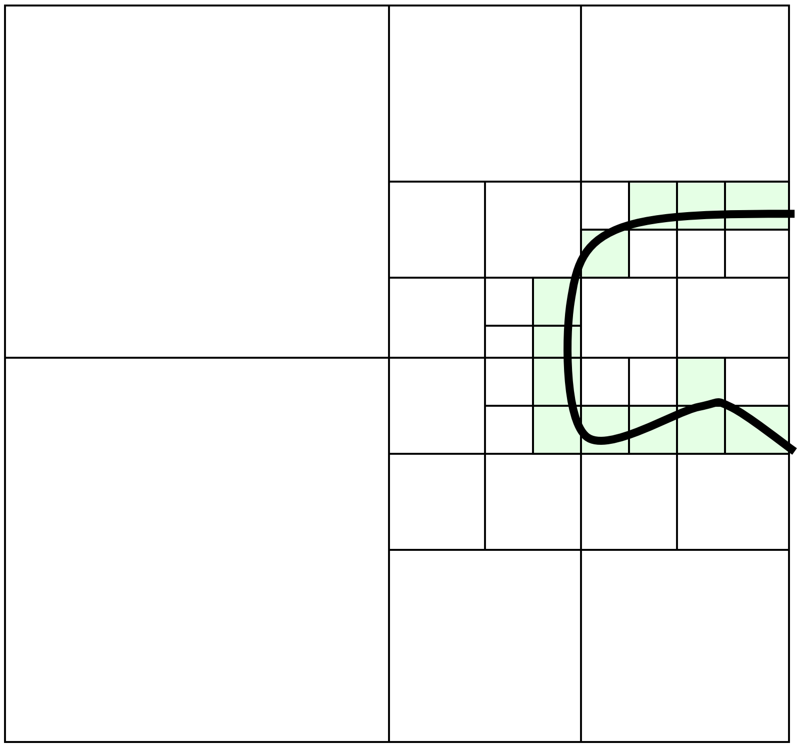


Problems:

high curvature areas, multiple components

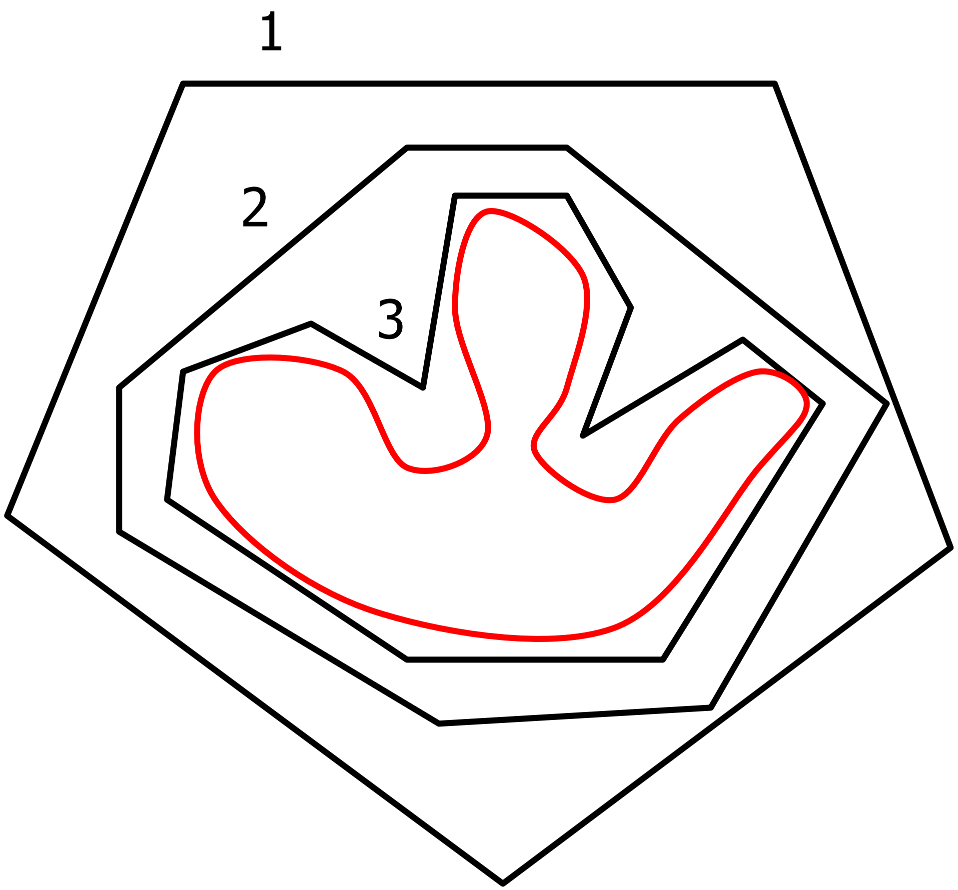
Subdivision Method

1) Define a bounding box for the contour – original cell
2) Subdivide the cell in four subcells
3) Check all subcells for cell-contour intersection
4) Repeat step 2 for all non-empty subcells
5) Result: collection of cells enclosing the contour



Shrinkwrap algorithm

1) Define an external polygon
2) Move its vertices
   to the contour
3) Subdivide its edges
4) Repeat steps 2 and 3
   until the given precision
   is reached



Problems:
high curvature areas, multiple components

HyperFun Polygonizer

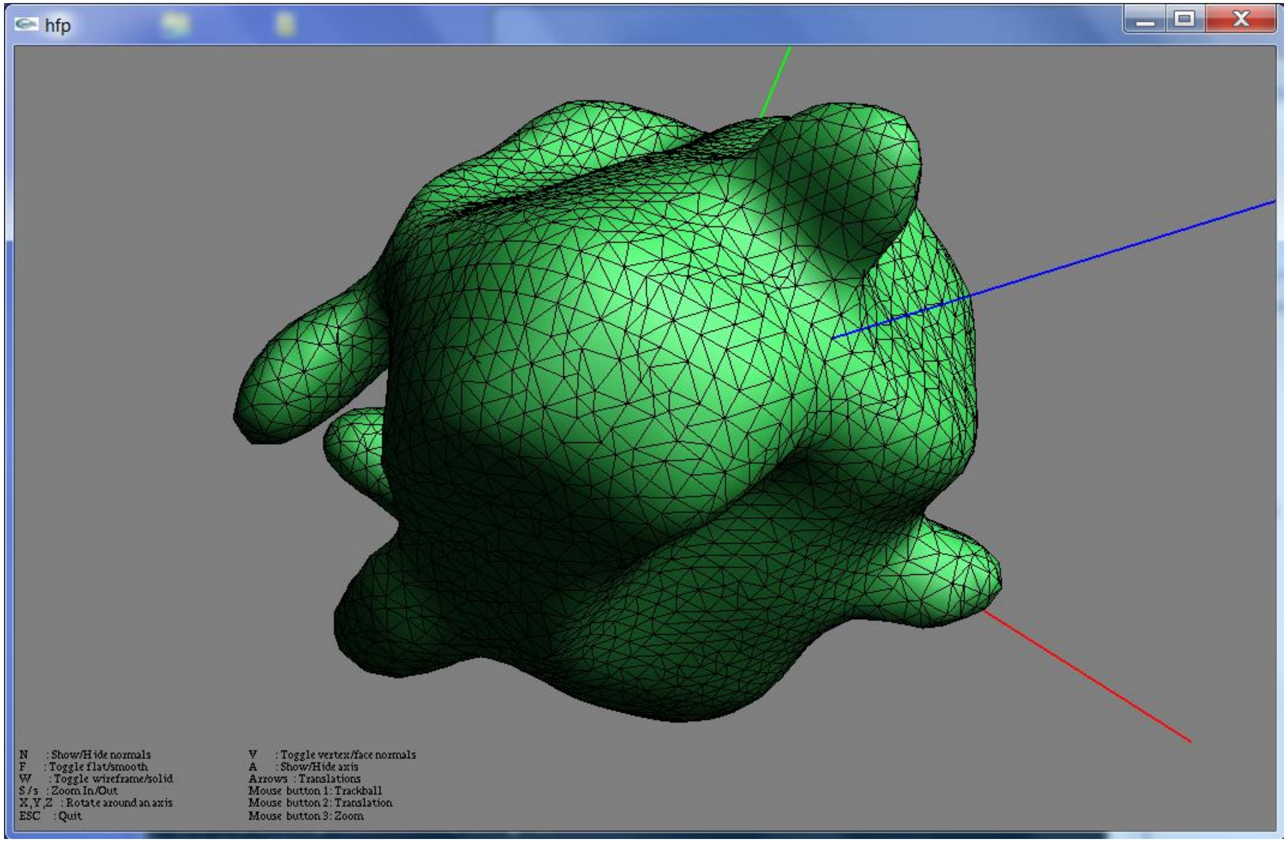
Input: function *F(x,y,z)* definition in HyperFun language and isovalue *C*

Algorithm: polygonization of the isosurface *F(x,y,z) = C* using hyperbolic arcs

Output:
- triangular mesh (polygonized isosurface) rendered with OpenGL
- export to files in VRML and STL formats

HyperFun Polygonizer

```
function(x[3], a[1])
{
sphere = 5^2 - (x[1]^2 + x[2]^2 + x[3]^2);
function = sphere + hfNoiseG(x, 1.8, 0.7, 1.4);
}
```



References

- Introduction to Implicit Surfaces, J. Bloomenthal et al. (Eds.), Morgan Kaufmann, 1997.
- W. Lorensen, H. Cline, Marching Cubes: A high resolution 3D surface construction algorithm, *Computer Graphics*, Vol. 21, Nr. 4, July 1987.
- Pasko A., Pilyugin V., Pokrovskiy V. Geometric modeling in the analysis of trivariate functions, Computers and Graphics, vol.12, Nos.3/4, 1988, pp.457-465.