King Abdullah University of Science and Technology

KAUST

CS 247 – Scientific Visualization
Lecture 10: Scalar Field Visualization, Pt. 4

Markus Hadwiger, KAUST

Reading Assignment #6 (until Mar 8)

Read (required):

- Real-Time Volume Graphics, Chapter 2
  (*GPU Programming*)
- Reminder: Real-Time Volume Graphics, Chapter 5.4

Read (optional):

- Paper:
  Gregory M. Nielson and Bernd Hamann,
  The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes,
  Visualization 1991
  https://dl.acm.org/doi/abs/10.5555/949607.949621

Quiz #1: Mar 8

Organization

- First 30 min of lecture
- No material (book, notes, ...) allowed

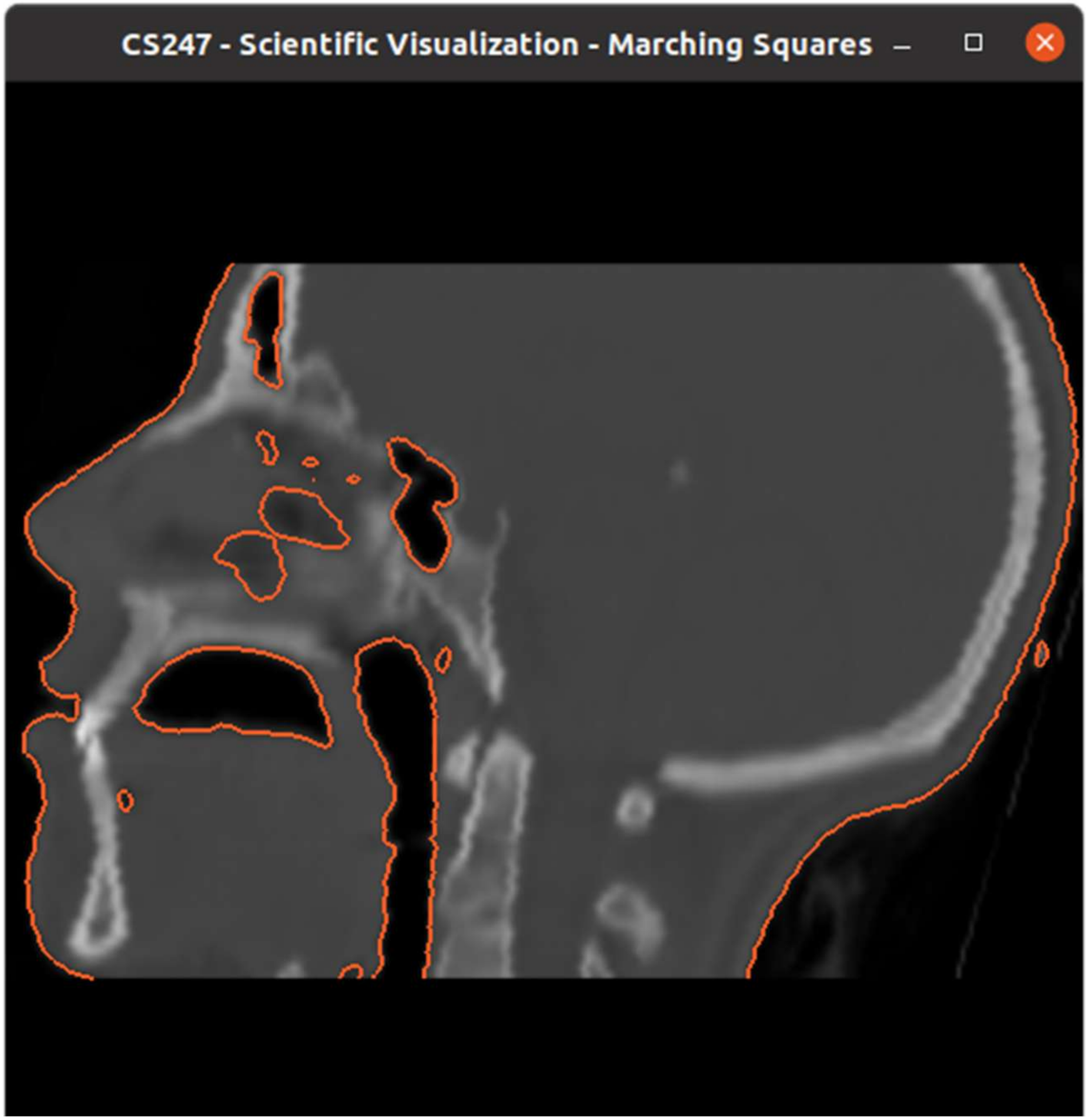
Content of questions

- Lectures (both actual lectures and slides)
- Reading assignments (except optional ones)
- Programming assignments (algorithms, methods)
- Solve short practical examples

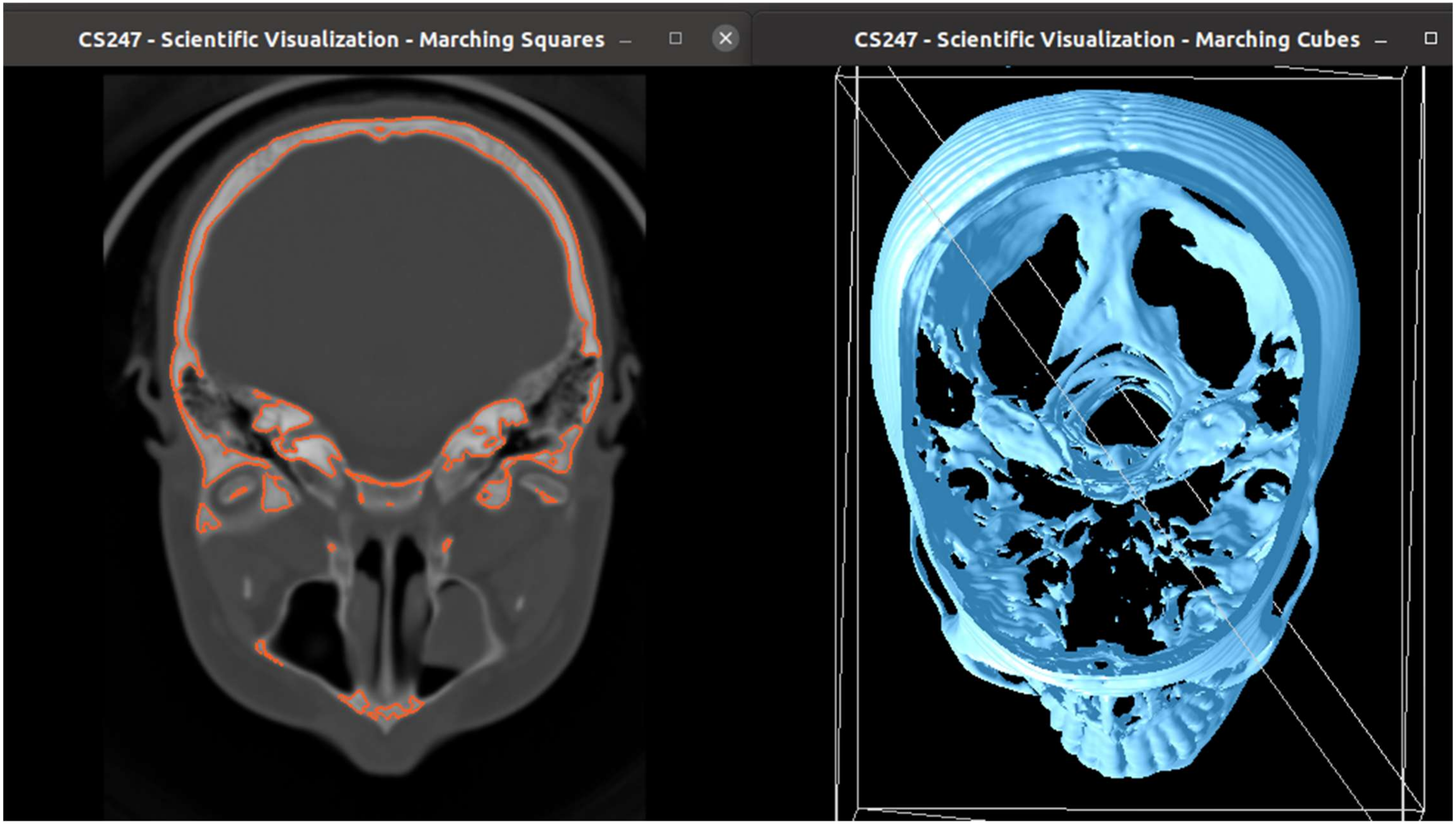
Programming Assignments Schedule (tentative)

Assignment 0:	Lab sign-up: join discord, setup github account + get repo	until	**Feb 1**
	Basic OpenGL example		
Assignment 1:	Volume slice viewer	until	**Feb 15**
Assignment 2:	Iso-contours (marching squares)	until	**Mar 1**
Assignment 3:	Iso-surface rendering (marching cubes)	until	**Mar 15**
Assignment 4:	Volume ray-casting, part 1	until	**Apr 12**
	Volume ray-casting, part 2	until	**Apr 19**
Assignment 5:	Flow vis, part 1 (hedgehog plots, streamlines, pathlines)	until	**May 3**
Assignment 6:	Flow vis, part 2 (LIC with color coding)	until	**May 13**

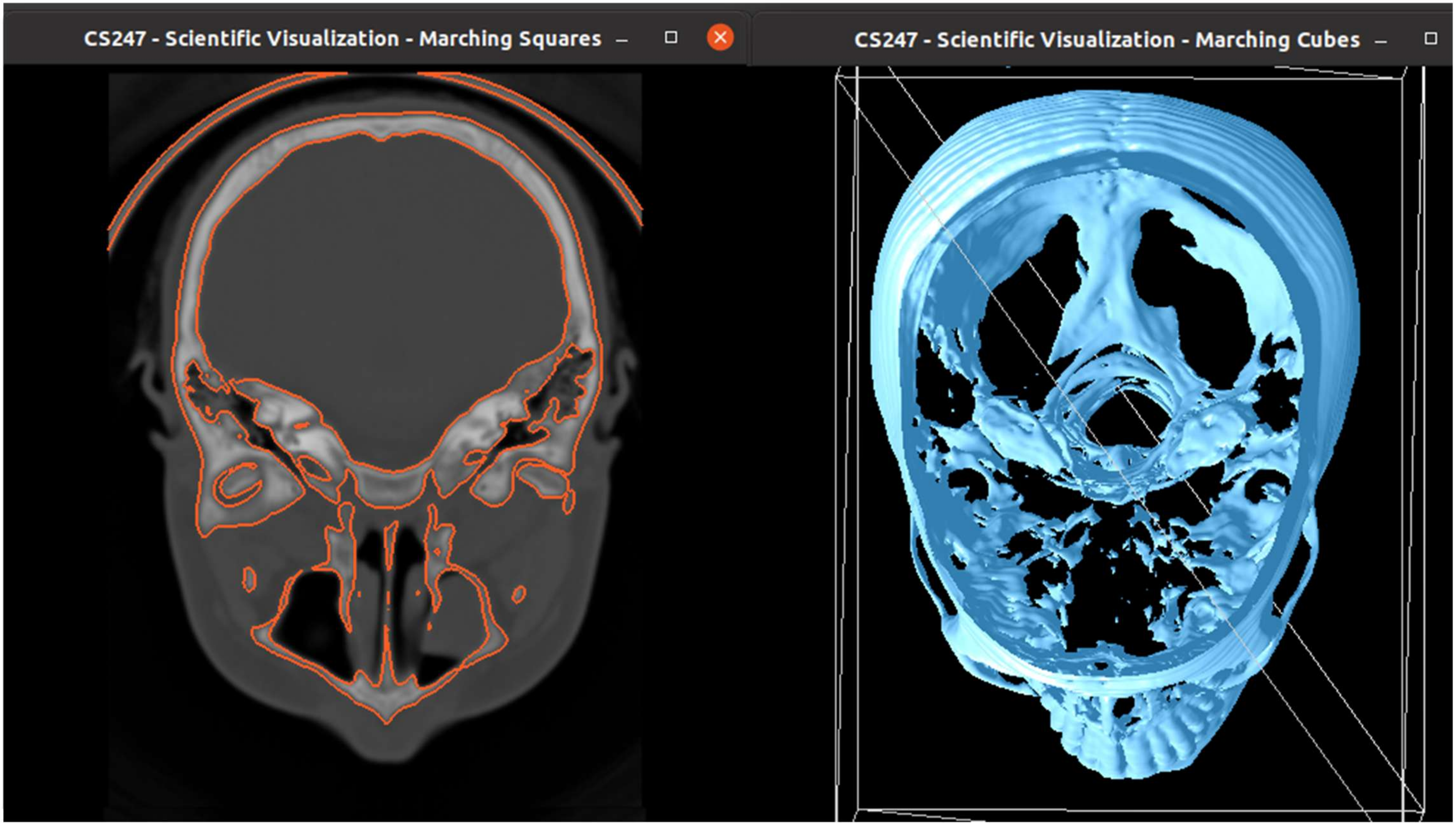
Programming Assignment 2



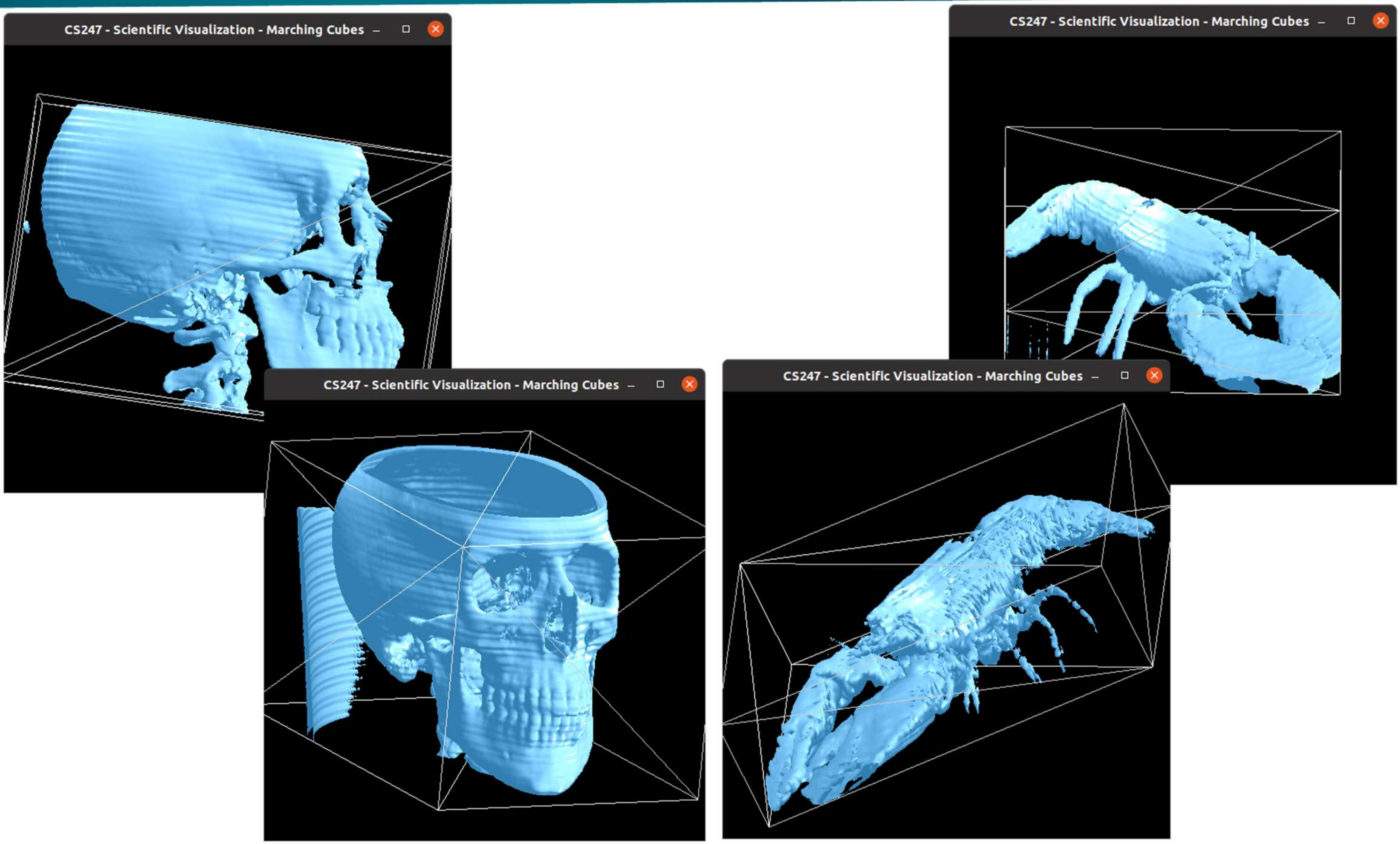
Programming Assignment 2 + 3



Programming Assignment 2 + 3



Programming Assignment 3



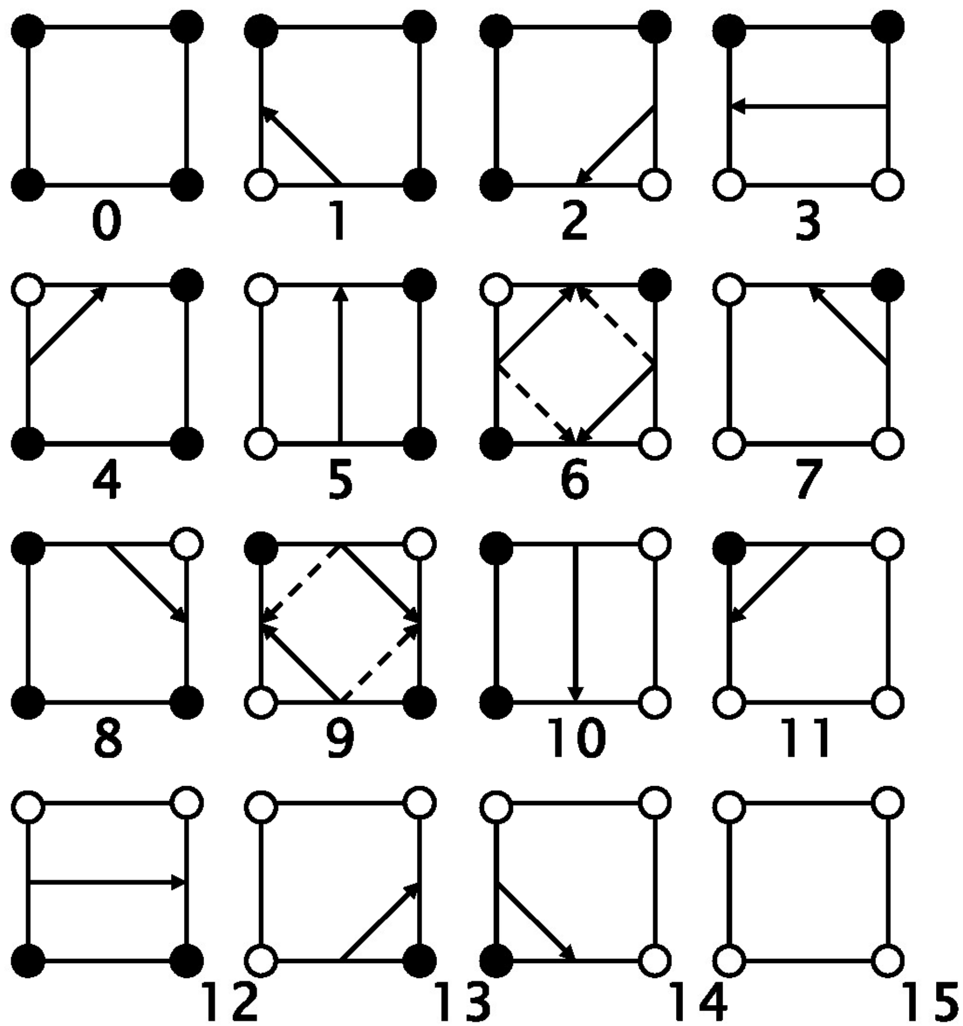
Basic contouring algorithms:

- cell-by-cell algorithms: simple structure, but generate disconnected segments, require post-processing
- contour propagation methods: more complicated, but generate connected contours

"Marching squares" algorithm (systematic cell-by-cell):

- process nodes in ccw order, denoted here as x_0, x_1, x_2, x_3
- compute at each node $\mathbf{x}_i$ the reduced field $\tilde{f}(x_i) = f(x_i) - (c - \varepsilon)$ (which is forced to be nonzero)
- take its sign as the i^{th} bit of a 4-bit integer
- use this as an index for lookup table containing the connectivity information:

Contours in a quadrangle cell



- $\tilde{f}(x_i) < 0$ (filled circle)
- $\tilde{f}(x_i) > 0$ (open circle)

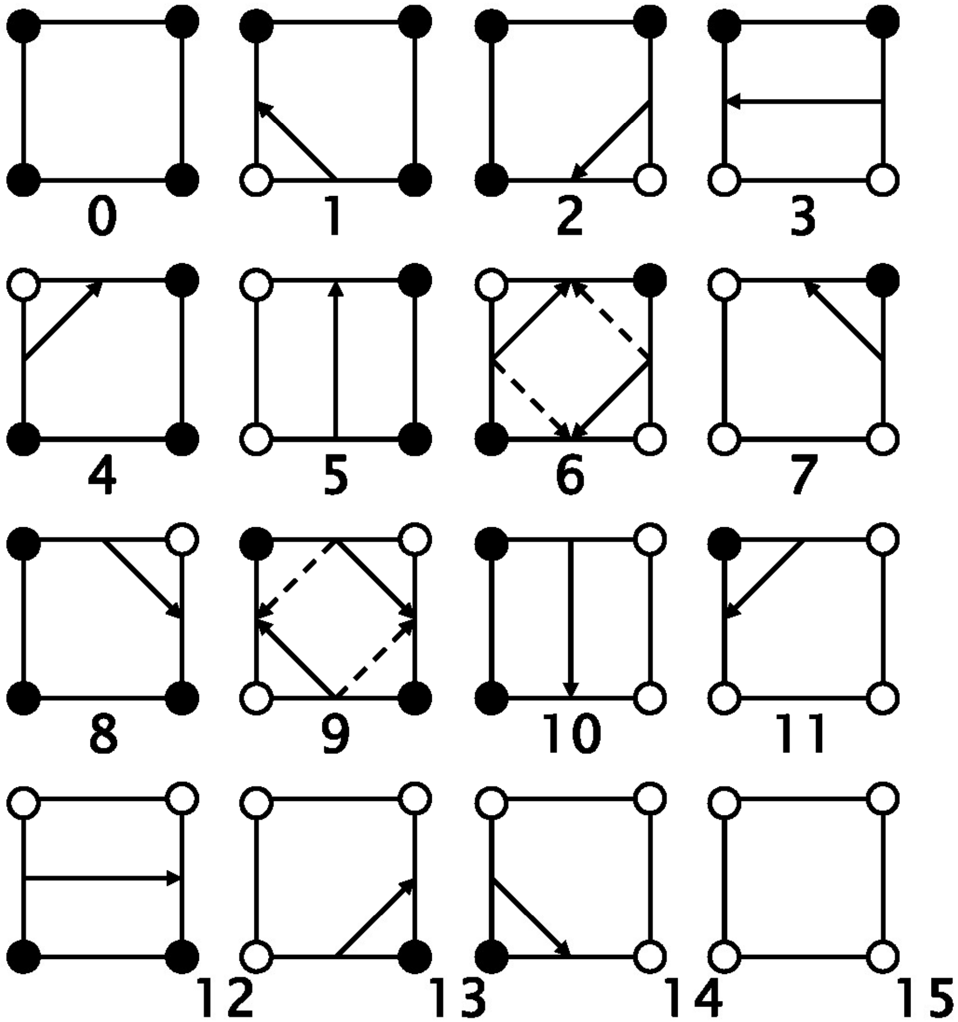
Alternating signs exist in cases 6 and 9.

Choose the solid or dashed line?

Both are possible for topological consistency.

This allows to have a fixed table of 16 cases.

Contours in a quadrangle cell



- ● $f(x_i) < c$
- ○ $f(x_i) \geq c$

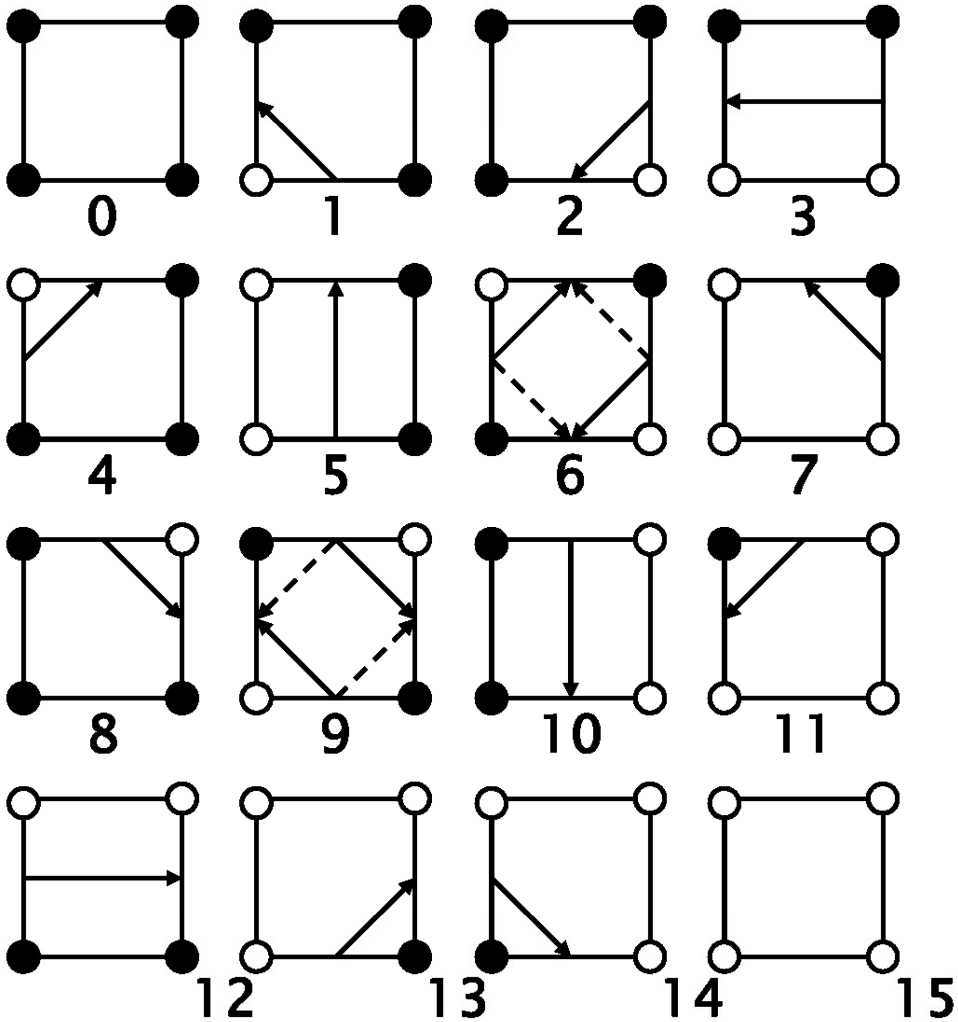
Alternating signs exist in cases 6 and 9.

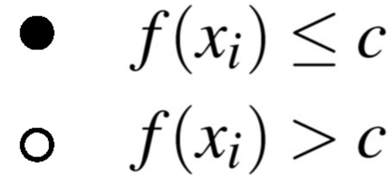
Choose the solid or dashed line?

Both are possible for topological consistency.

This allows to have a fixed table of 16 cases.

Contours in a quadrangle cell





- Alternating signs exist in cases 6 and 9.
- Choose the solid or dashed line?
- Both are possible for topological consistency.
- This allows to have a fixed table of 16 cases.

Orientability (1-manifold embedded in 2D)

Orientability of 1-manifold:

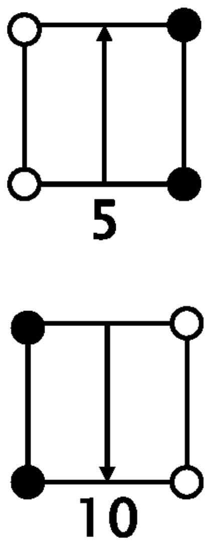
Possible to assign consistent left/right orientation

not orientable

Moebius strip
(only one side!)

Iso-contours

- Consistent side for scalar values…
  - greater than iso-value (e.g, *left* side)
  - less than iso-value (e.g., *right* side)
- Use consistent ordering of vertices
  (e.g., larger vertex index is "tip" of arrow;
  if (0,1) points "up", "left" is left, …)



● $\tilde{f}(x_i) < 0$

○ $\tilde{f}(x_i) > 0$

Orientability (2-manifold embedded in 3D)

Orientability of 2-manifold:

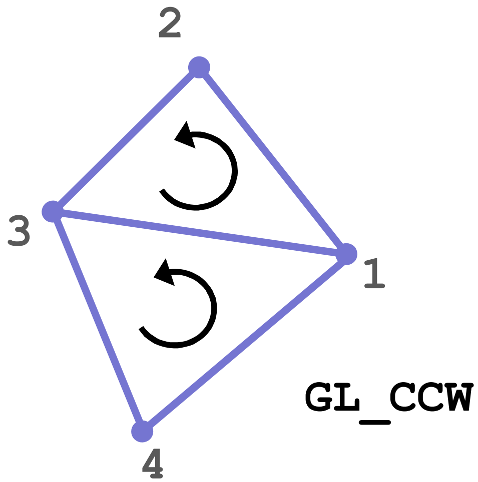
Possible to assign consistent normal vector orientation

not orientable

Moebius strip
(only one side!)

Triangle meshes

- Edges
  - Consistent ordering of vertices: CCW (counter-clockwise) or CW (clockwise) (e.g., (3,1,2) on one side of edge, (1,3,4) on the other side)
- Triangles
  - Consistent front side vs. back side
  - Normal vector; or ordering of vertices (CCW/CW)
  - See also: “right-hand rule”



Topological consistency

To avoid degeneracies, use symbolic perturbations:

If level c is found as a node value, set the level to $c-\varepsilon$ where ε is a symbolic infinitesimal.

Then:

- contours intersect edges at some (possibly infinitesimal) distance from end points
- flat regions can be visualized by pair of contours at $c-\varepsilon$ and $c+\varepsilon$
- contours are topologically consistent, meaning:

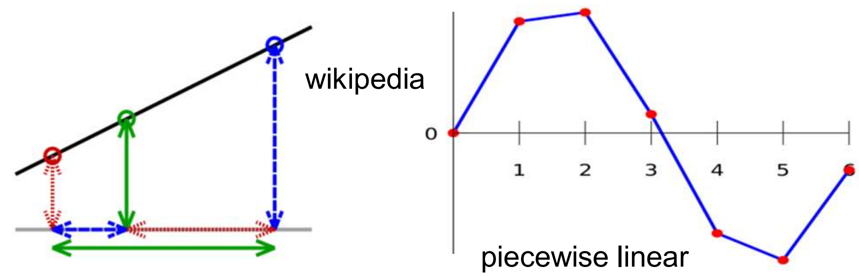
Contours are closed, orientable, nonintersecting lines.

(except where the boundary is hit)

Linear Interpolation / Convex Combinations

Linear interpolation in 1D:

$$f(\alpha) = (1-\alpha)v_1 + \alpha v_2$$



Line embedded in 2D (linear interpolation of vertex coordinates/attributes):

$$f(\alpha_1, \alpha_2) = \alpha_1 v_1 + \alpha_2 v_2 \qquad\qquad f(\alpha) = v_1 + \alpha(v_2 - v_1)$$

$$\alpha_1 + \alpha_2 = 1 \qquad\qquad \alpha = \alpha_2$$

Line segment: $\alpha_1, \alpha_2 \geq 0$ (→ convex combination)

Compare to line parameterization
with parameter t:

$$v(t) = v_1 + t(v_2 - v_1)$$

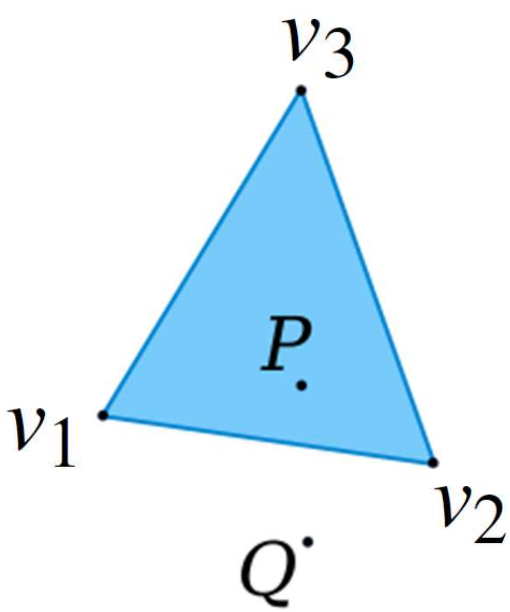
Linear Interpolation / Convex Combinations

Linear combination (n-dim. space):

$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = \sum_{i=1}^{n} \alpha_i v_i$$

Affine combination: Restrict to $(n-1)$-dim. subspace:

$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = \sum_{i=1}^{n} \alpha_i = 1$$



Convex combination: $\alpha_i \geq 0$

(restrict to simplex in subspace)

Linear Interpolation / Convex Combinations

The weights α_i are the n normalized ***barycentric*** coordinates

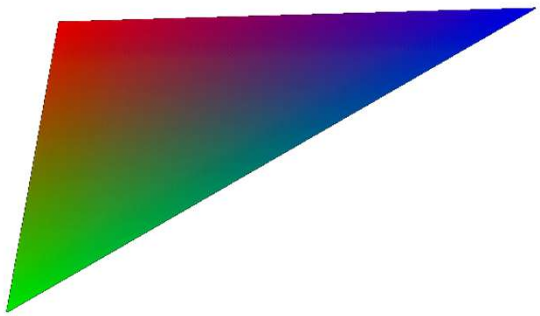
→ linear attribute interpolation in simplex

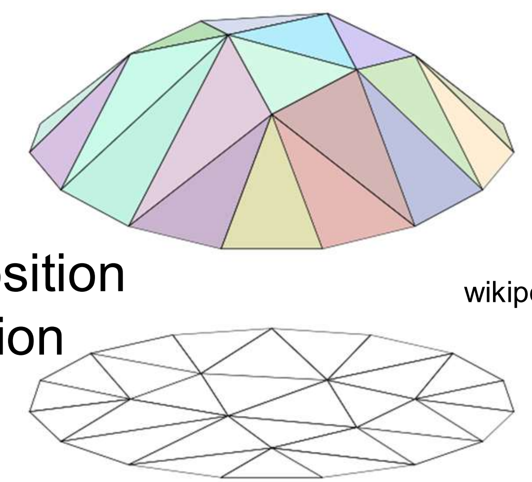
$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = \sum_{i=1}^{n} \alpha_i v_i$$

$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = \sum_{i=1}^{n} \alpha_i = 1$$

$$\alpha_i \geq 0$$

attribute interpolation





spatial position interpolation

wikipedia

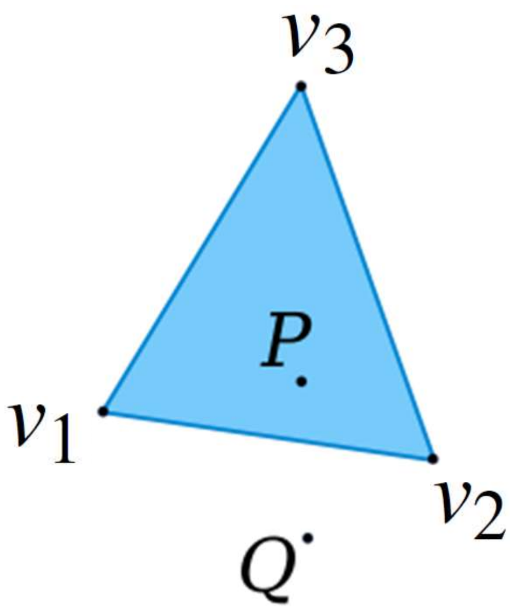
Linear Interpolation / Convex Combinations

$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = \sum_{i=1}^{n} \alpha_i v_i$$

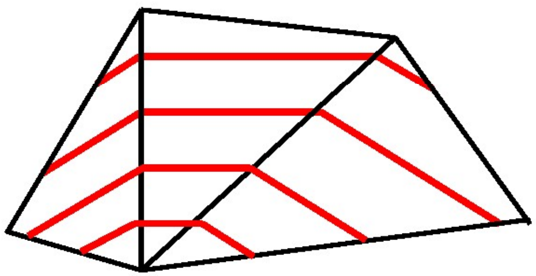
$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = \sum_{i=1}^{n} \alpha_i = 1$$

Can re-parameterize to get $(n-1)$ ***affine*** coordinates:

$$\alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 =$$
$$\tilde{\alpha}_1 (v_2 - v_1) + \tilde{\alpha}_2 (v_3 - v_1) + v_1$$
$$\tilde{\alpha}_1 = \alpha_2$$
$$\tilde{\alpha}_2 = \alpha_3$$



Contours in triangle/tetrahedral cells



Linear interpolation of cells implies piece-wise linear contours.

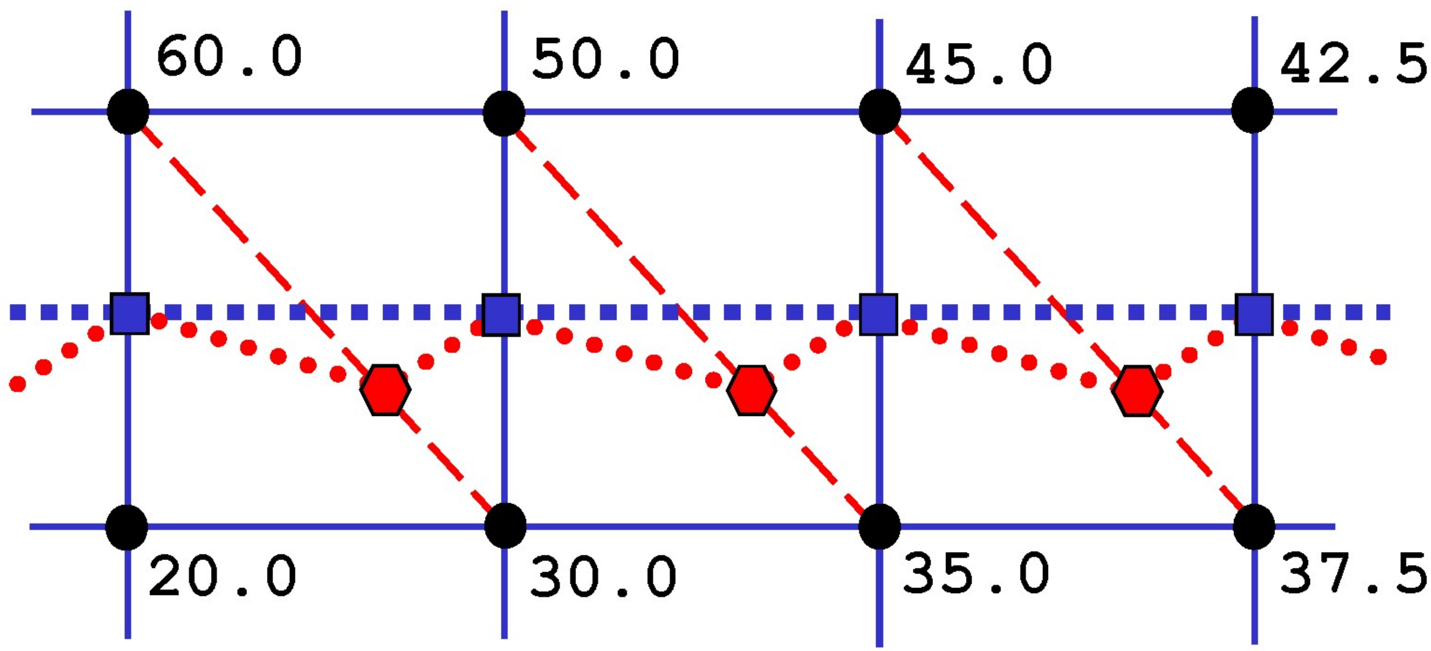
Contours are unambiguous, making "marching triangles" even simpler than "marching squares".

Question: Why not split quadrangles into two triangles (and hexahedra into five or six tetrahedra) and use marching triangles (tetrahedra)?

Answer: This can introduce periodic artifacts!

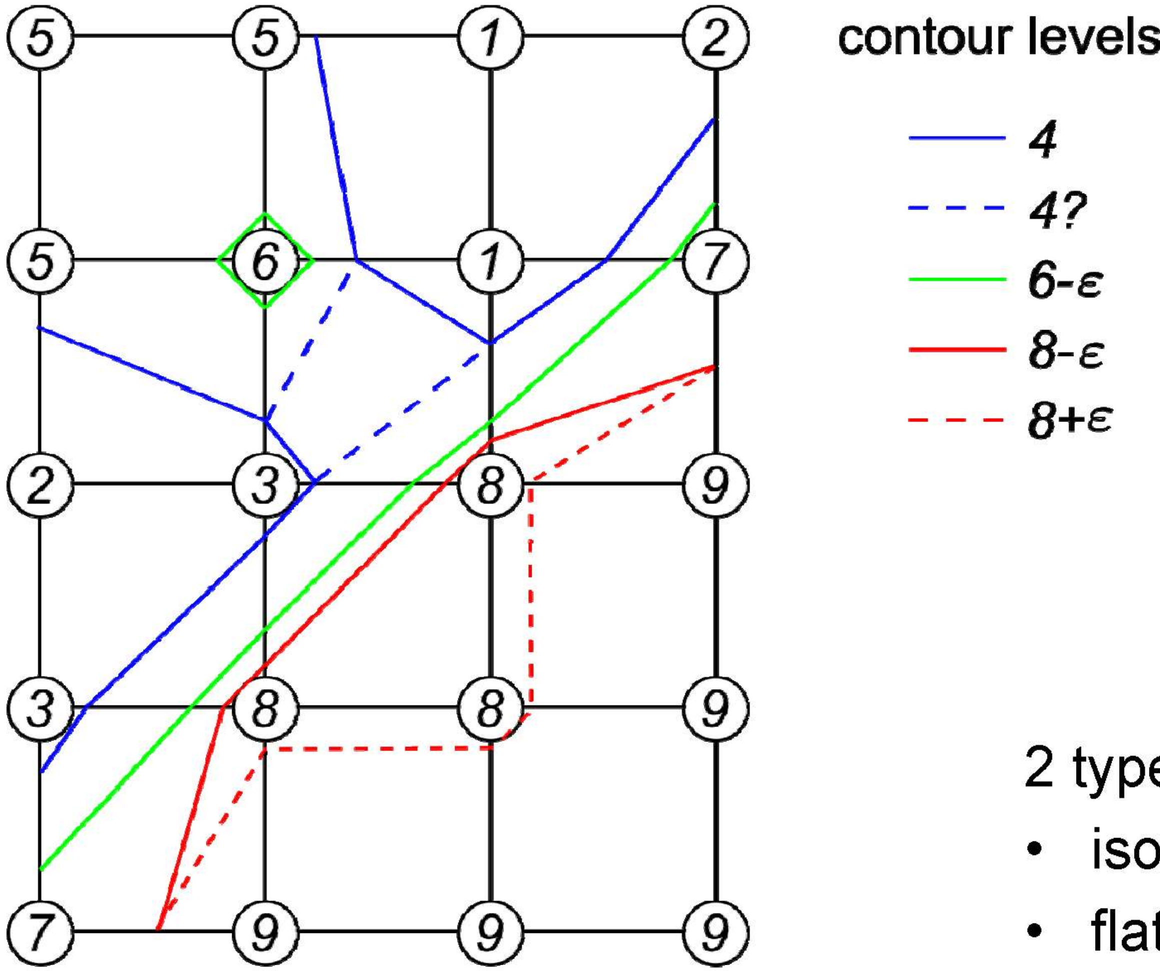
Contours in triangle/tetrahedral cells

Illustrative example: Find contour at level c=40.0 !



original quad grid, yielding vertices ■ and contour ▪▪▪▪▪
triangulated grid, yielding vertices ⬢ and contour ••••••

Example



2 types of degeneracies:

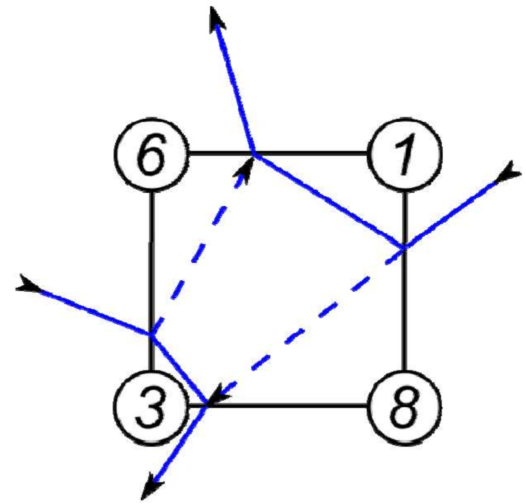
- isolated points (c=6)
- flat regions (c=8)

Ambiguities of contours

What is the correct contour of c=4?

Two possibilities, both are orientable:

- connect high values ———————
- connect low values - - - - - - - - - - -



Answer: correctness depends on interior values of $f(x)$.

But: different interpolation schemes are possible.

Better question: What is the correct contour with respect to bilinear interpolation?

Bi-Linear Interpolation

Consider area between 2x2 adjacent samples (e.g., pixel centers):

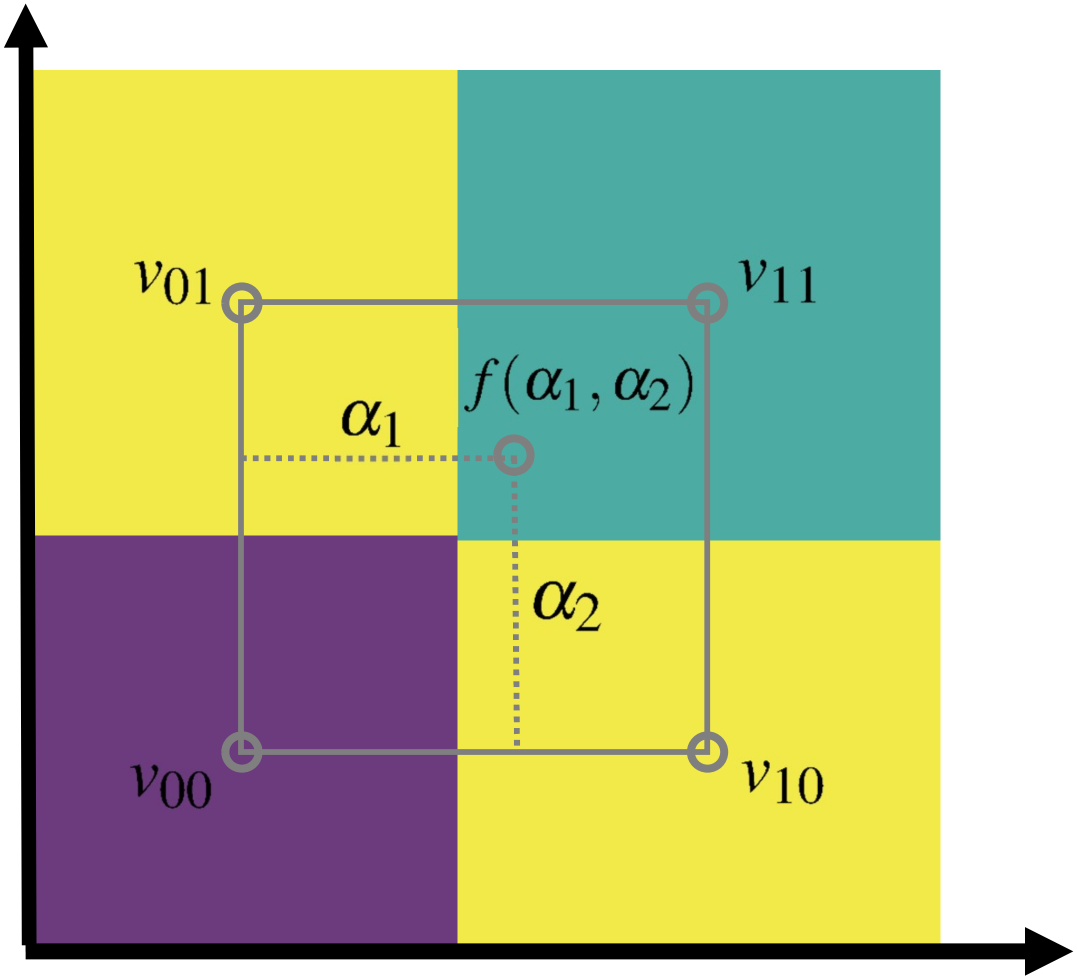
Given any (fractional) position

$$\alpha_1 := x_1 - \lfloor x_1 \rfloor \qquad \alpha_1 \in [0.0, 1.0)$$
$$\alpha_2 := x_2 - \lfloor x_2 \rfloor \qquad \alpha_2 \in [0.0, 1.0)$$

and 2x2 sample values

$$\begin{bmatrix} v_{01} & v_{11} \\ v_{00} & v_{10} \end{bmatrix}$$

Compute: $f(\alpha_1, \alpha_2)$



Bi-Linear Interpolation

Consider area between 2x2 adjacent samples (e.g., pixel centers):

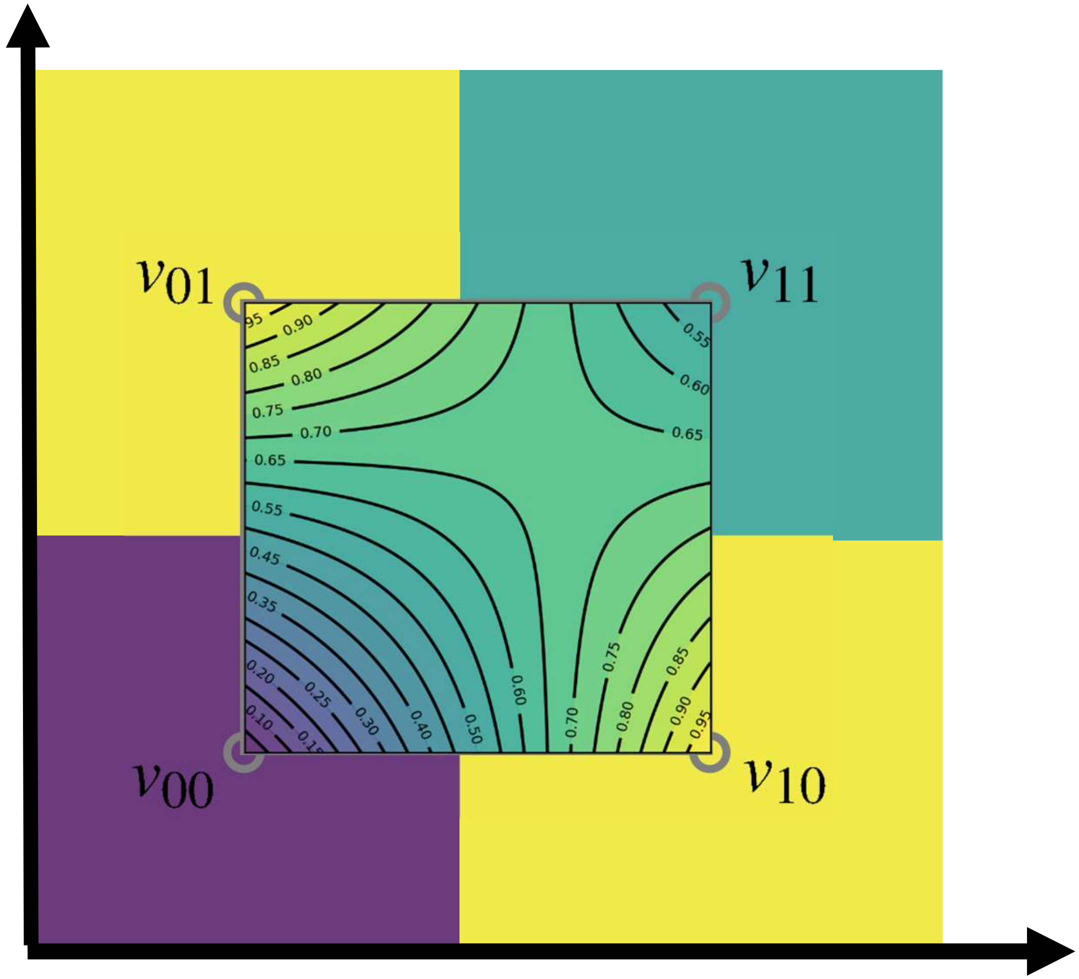
Given any (fractional) position

$$\alpha_1 := x_1 - \lfloor x_1 \rfloor \qquad \alpha_1 \in [0.0, 1.0)$$
$$\alpha_2 := x_2 - \lfloor x_2 \rfloor \qquad \alpha_2 \in [0.0, 1.0)$$

and 2x2 sample values

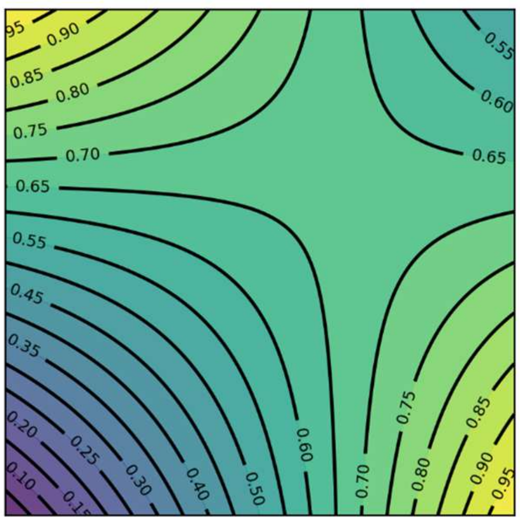
$$\begin{bmatrix} v_{01} & v_{11} \\ v_{00} & v_{10} \end{bmatrix}$$

Compute: $f(\alpha_1, \alpha_2)$

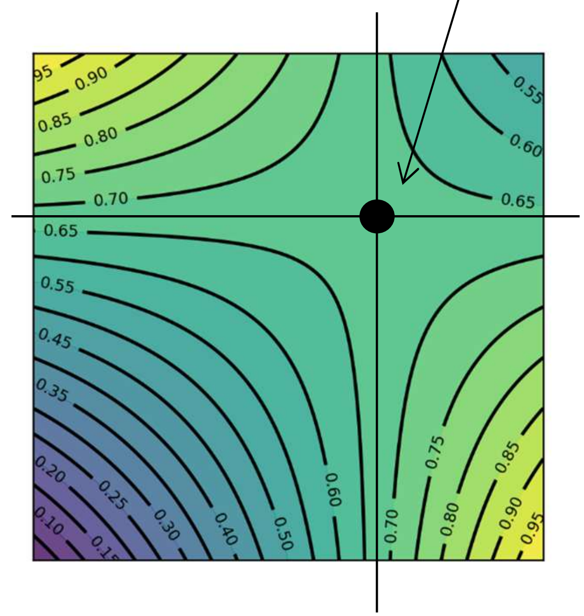


Bi-Linear Interpolation: Critical Points

Critical points are where the gradient vanishes (i.e., is the zero vector)



critical point
(saddle point)



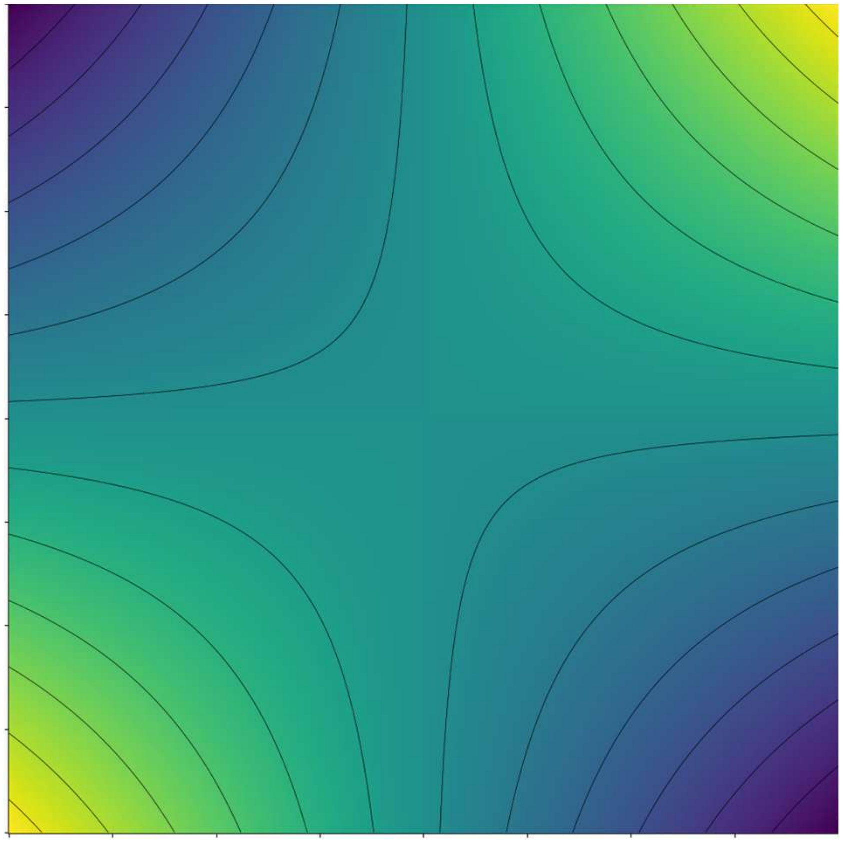
here, the critical
value is 2/3=0.666…

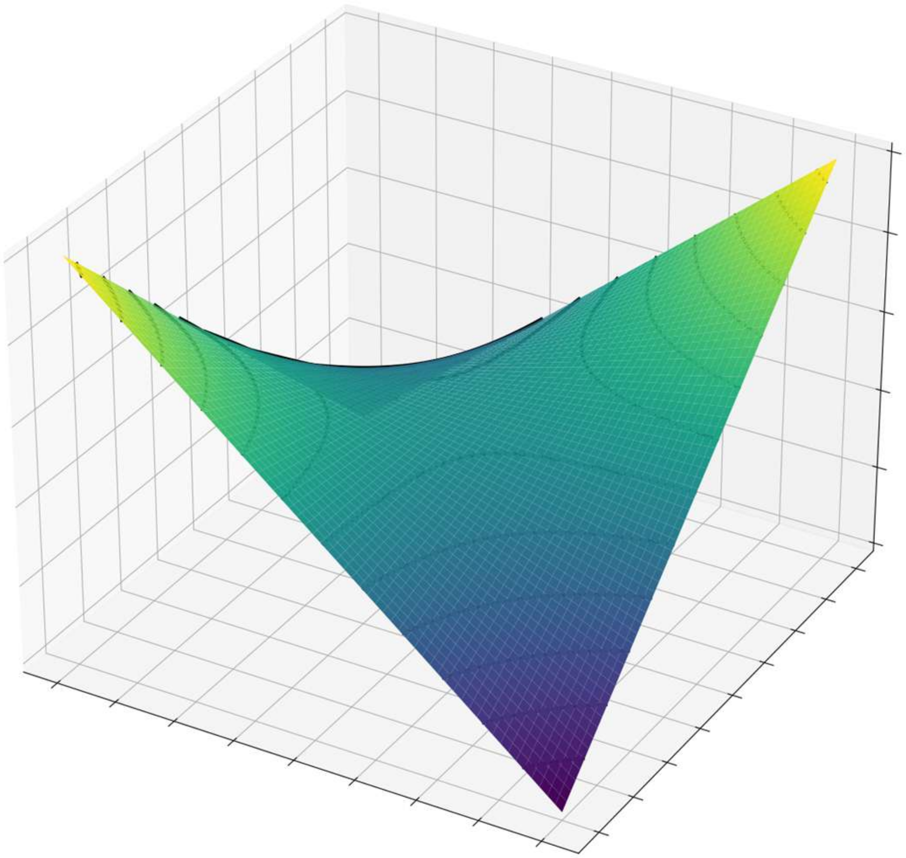
“Asymptotic decider”: resolve ambiguous configurations (6 and 9) by comparing specific iso-value with critical value (scalar value at critical point)

Bi-Linear Interpolation

Consider area between 2x2 adjacent samples (e.g., pixel centers)

Example #1: 1 at bottom-left and top-right, 0 at top-left and bottom-right

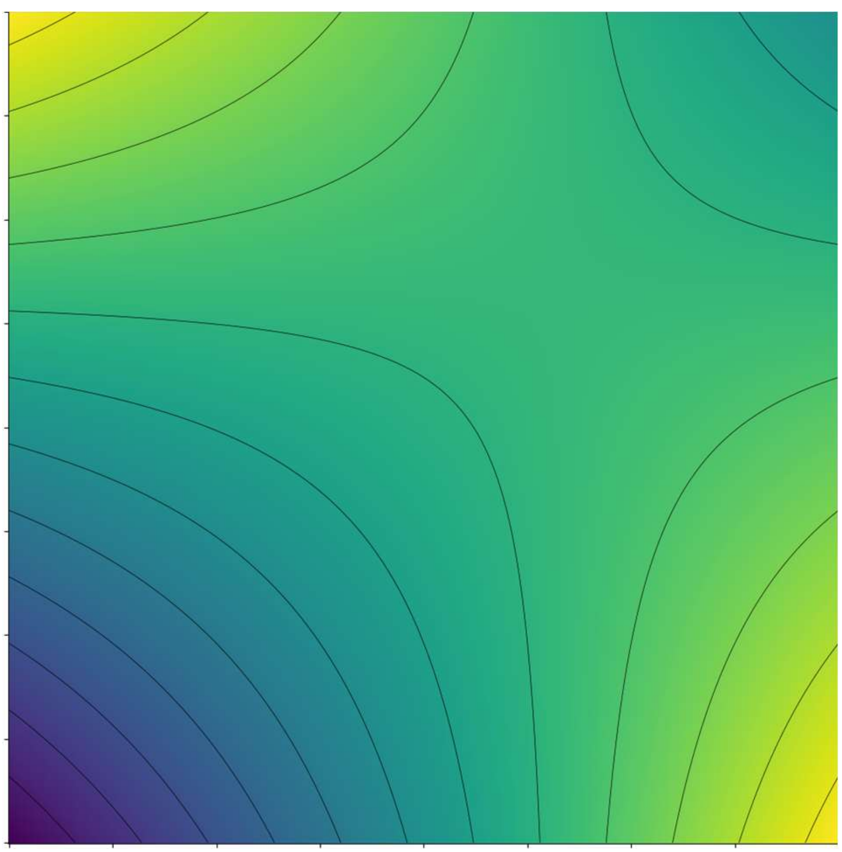


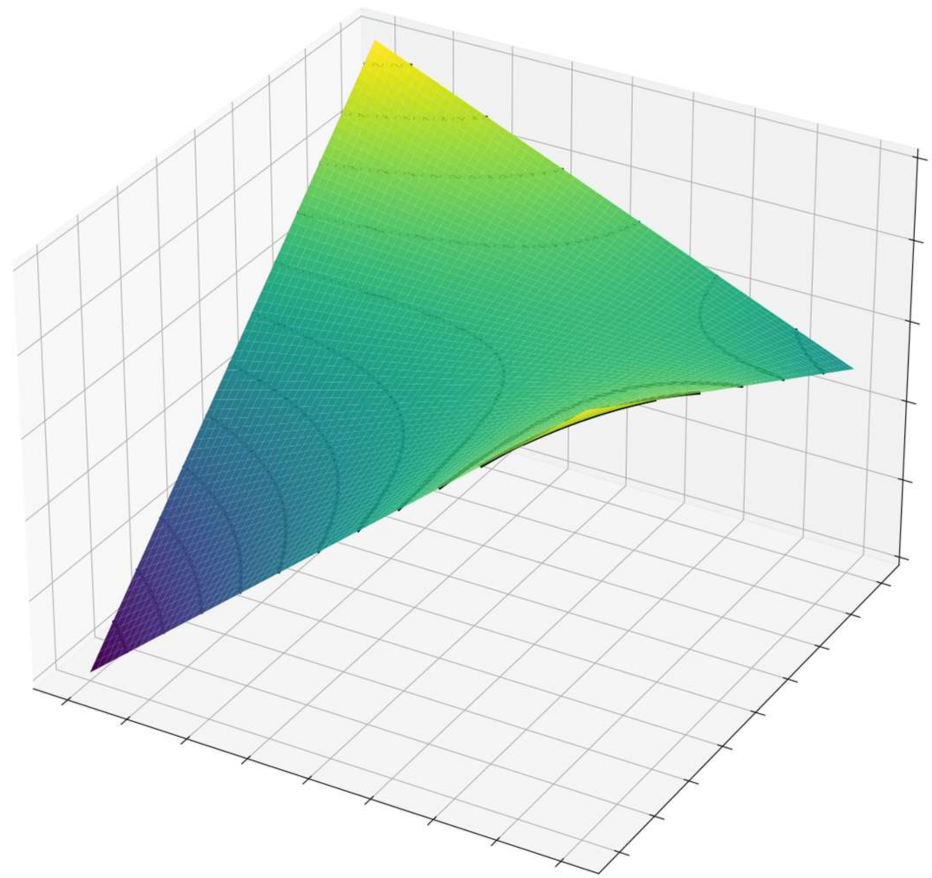


Bi-Linear Interpolation

Consider area between 2x2 adjacent samples (e.g., pixel centers)

Example #2: 1 at top-left and bottom-right, 0 at bottom-left, 0.5 at top-right





Bi-Linear Interpolation

Interpolate function at (fractional) position (α_1, α_2) :

$$f(\alpha_1, \alpha_2) = \begin{bmatrix} \alpha_2 & (1-\alpha_2) \end{bmatrix} \begin{bmatrix} v_{01} & v_{11} \\ v_{00} & v_{10} \end{bmatrix} \begin{bmatrix} (1-\alpha_1) \\ \alpha_1 \end{bmatrix}$$

$$= (1-\alpha_1)(1-\alpha_2)v_{00} + \alpha_1(1-\alpha_2)v_{10} + (1-\alpha_1)\alpha_2 v_{01} + \alpha_1\alpha_2 v_{11}$$

$$= v_{00} + \alpha_1(v_{10} - v_{00}) + \alpha_2(v_{01} - v_{00}) + \alpha_1\alpha_2(v_{00} + v_{11} - v_{10} - v_{01})$$

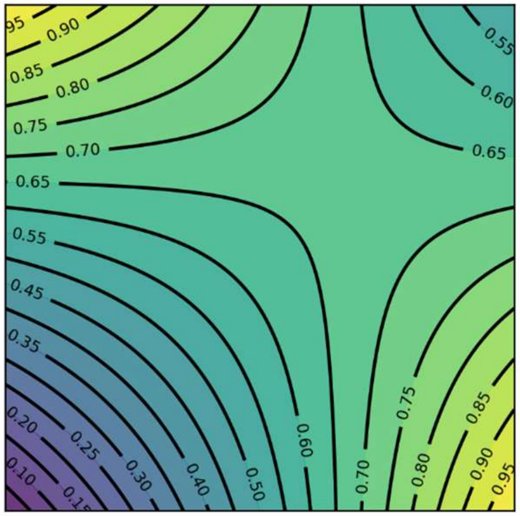
Bi-Linear Interpolation: Contours

Find one specific iso-contour (can of course do this for any/all isovalues):

$$f(\alpha_1, \alpha_2) = c$$

Find all (α_1, α_2) where:

$$v_{00} + \alpha_1 (v_{10} - v_{00}) + \alpha_2 (v_{01} - v_{00}) + \alpha_1 \alpha_2 (v_{00} + v_{11} - v_{10} - v_{01}) = c$$



Bi-Linear Interpolation: Critical Points

Compute gradient (critical points are where gradient is zero vector):

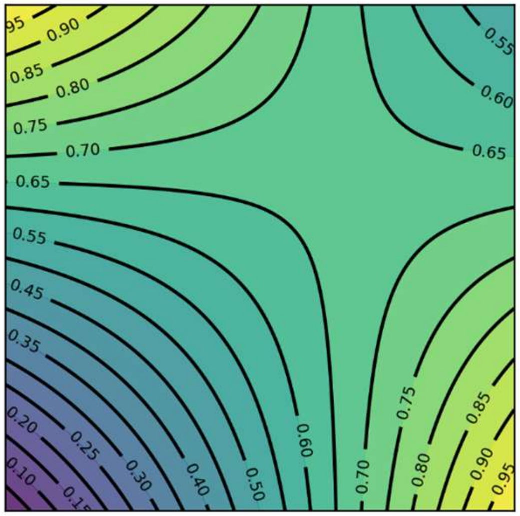
$$\frac{\partial f(\alpha_1,\alpha_2)}{\partial \alpha_1} = (v_{10} - v_{00}) + \alpha_2 (v_{00} + v_{11} - v_{10} - v_{01})$$

$$\frac{\partial f(\alpha_1,\alpha_2)}{\partial \alpha_2} = (v_{01} - v_{00}) + \alpha_1 (v_{00} + v_{11} - v_{10} - v_{01})$$

Where are lines of constant value / critical points?

$$\frac{\partial f(\alpha_1,\alpha_2)}{\partial \alpha_1} = 0: \qquad \alpha_2 = \frac{v_{00} - v_{10}}{v_{00} + v_{11} - v_{10} - v_{01}}$$

$$\frac{\partial f(\alpha_1,\alpha_2)}{\partial \alpha_2} = 0: \qquad \alpha_1 = \frac{v_{00} - v_{01}}{v_{00} + v_{11} - v_{10} - v_{01}}$$



Bi-Linear Interpolation: Critical Points

Compute gradient (critical points are where gradient is zero vector):

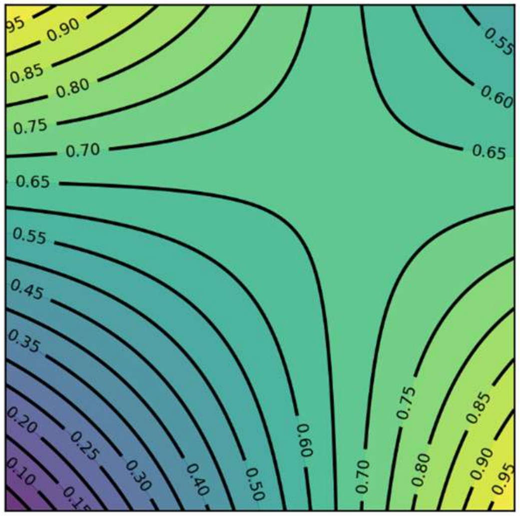
$$\frac{\partial f(\alpha_1, \alpha_2)}{\partial \alpha_1} = (v_{10} - v_{00}) + \alpha_2 (v_{00} + v_{11} - v_{10} - v_{01})$$

$$\frac{\partial f(\alpha_1, \alpha_2)}{\partial \alpha_2} = (v_{01} - v_{00}) + \alpha_1 (v_{00} + v_{11} - v_{10} - v_{01})$$

Where are lines of constant value / critical points?

$$\frac{\partial f(\alpha_1, \alpha_2)}{\partial \alpha_1} = 0: \qquad \alpha_2 = \frac{v_{00} - v_{10}}{v_{00} + v_{11} - v_{10} - v_{01}}$$

$$\frac{\partial f(\alpha_1, \alpha_2)}{\partial \alpha_2} = 0: \qquad \alpha_1 = \frac{v_{00} - v_{01}}{v_{00} + v_{11} - v_{10} - v_{01}}$$



if denominator is zero, bi-linear interpolation has degenerated to linear interpolation (or const)! (also means: no isolated critical points!)

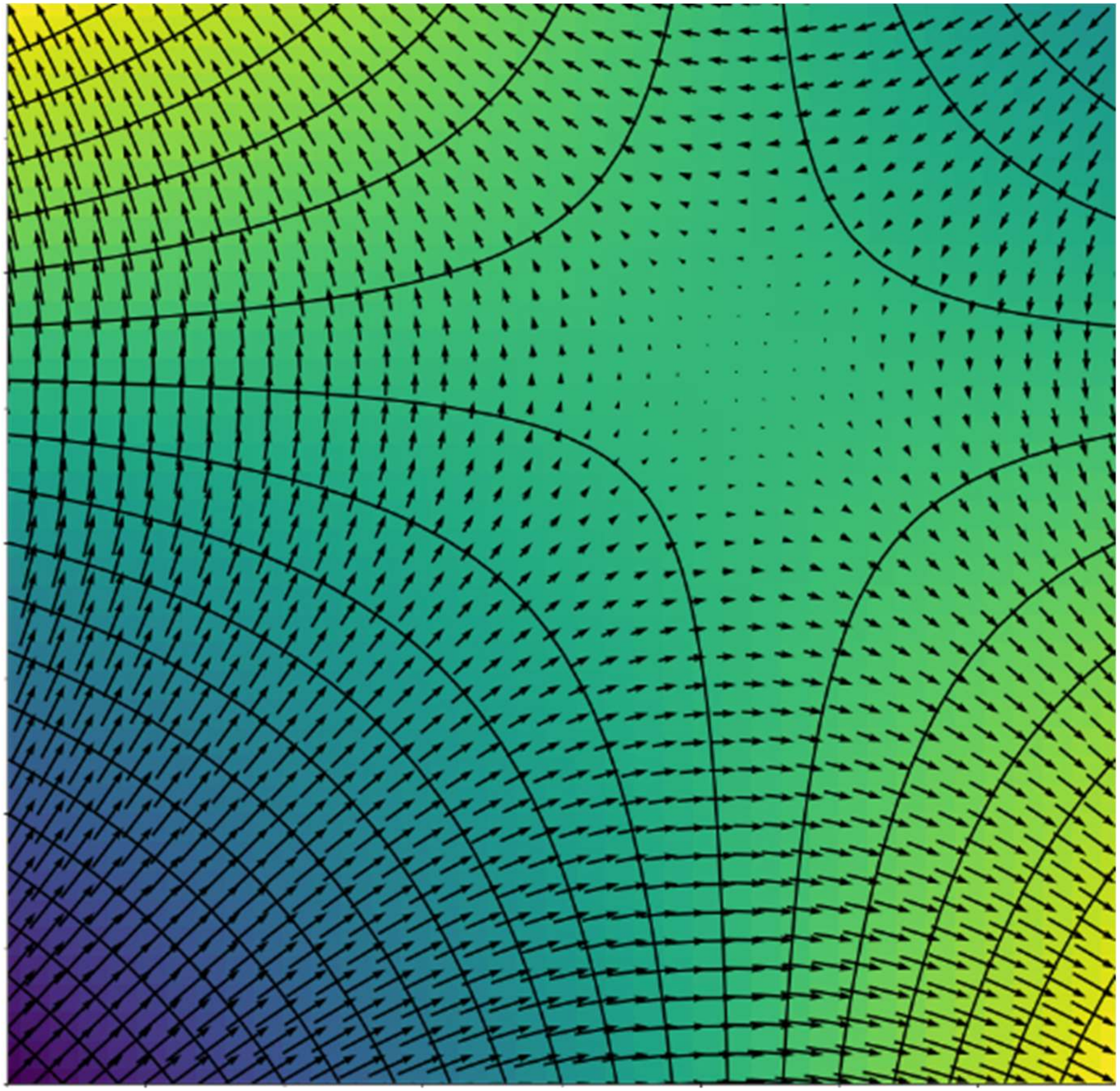
Bi-Linear Interpolation: Critical Points

Compute gradient

Note that isolines are farther apart where gradient is smaller

Note the horizontal and vertical lines where gradient becomes vertical/horizontal

Note the critical point



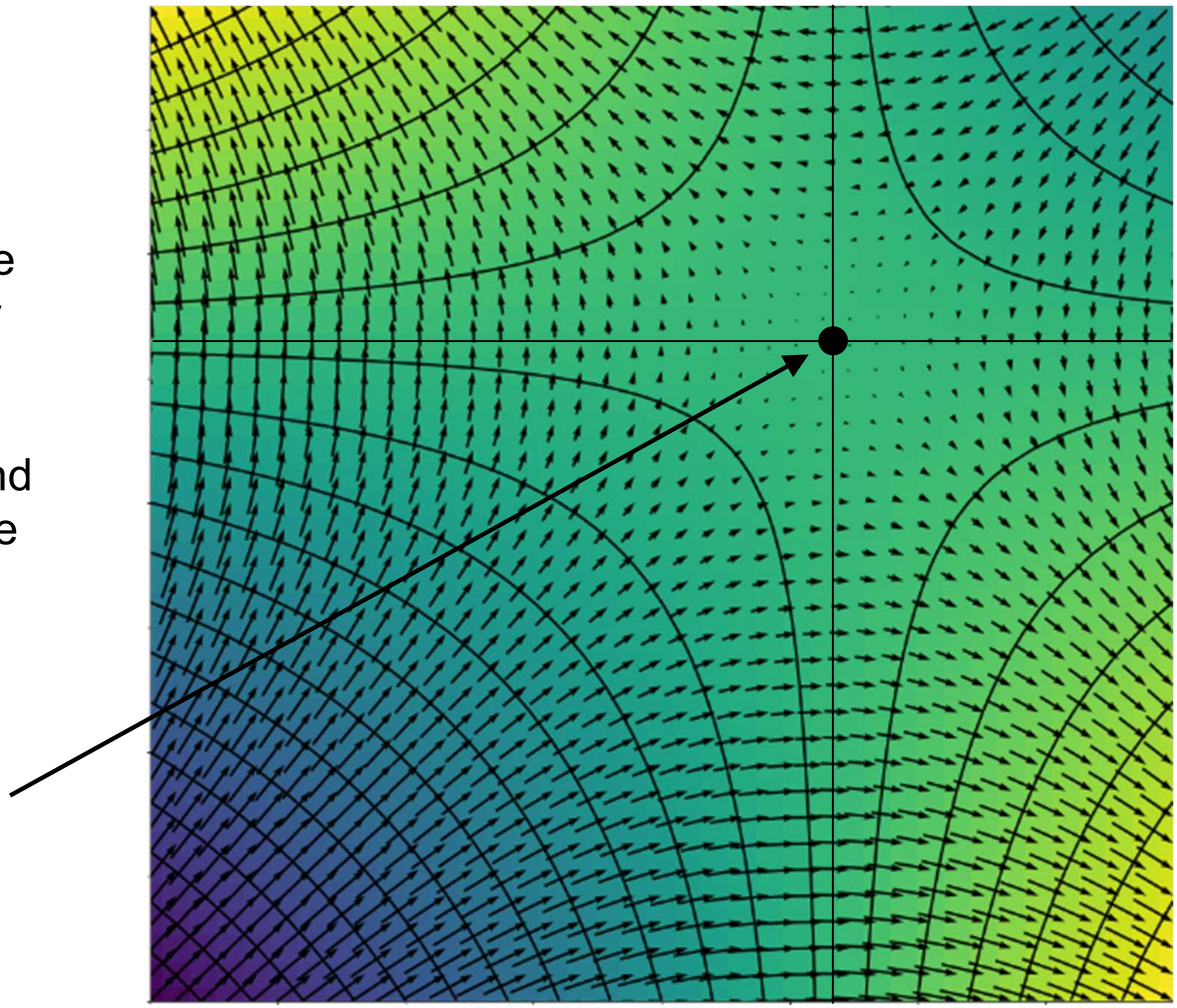
Bi-Linear Interpolation: Critical Points

Compute gradient

Note that isolines are
farther apart where
gradient is smaller

Note the horizontal and
vertical lines where
gradient becomes
vertical/horizontal

Note the critical point



Bi-Linear Interpolation: Critical Points

Examine Hessian matrix at critical point (non-degenerate critical p.?, ...)

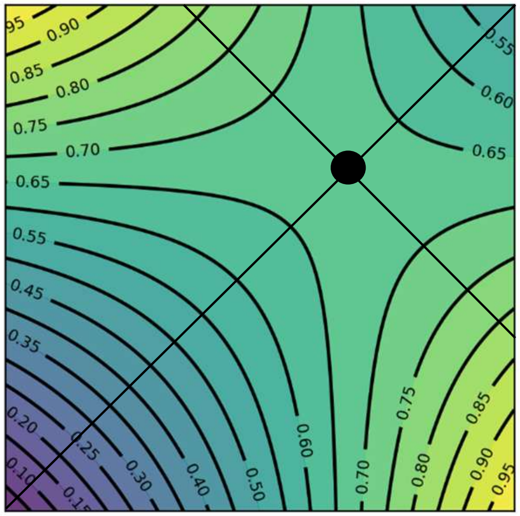
$$\begin{bmatrix} \frac{\partial^2 f}{\partial \alpha_1^2} & \frac{\partial^2 f}{\partial \alpha_1 \partial \alpha_2} \\ \frac{\partial^2 f}{\partial \alpha_2 \partial \alpha_1} & \frac{\partial^2 f}{\partial \alpha_2^2} \end{bmatrix} = \begin{bmatrix} 0 & a \\ a & 0 \end{bmatrix}$$

$$a = v_{00} + v_{11} - v_{10} - v_{01}$$

Eigenvalues and eigenvectors (Hessian is symmetric: always real)

$$\lambda_1 = -a \text{ and } \lambda_2 = a$$

$$v_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, v_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$



(here also: principal curvature magnitudes and directions
of this function's graph == surface embedded in 3D)

Bi-Linear Interpolation: Critical Points

Examine Hessian matrix at critical point (non-degenerate critical p.?, ...)

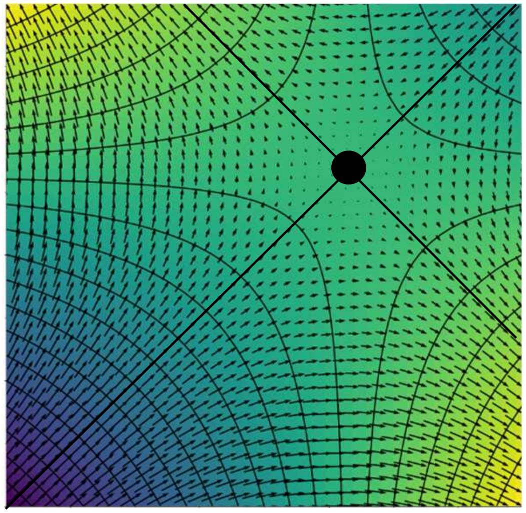
$$\begin{bmatrix} \frac{\partial^2 f}{\partial \alpha_1^2} & \frac{\partial^2 f}{\partial \alpha_1 \partial \alpha_2} \\ \frac{\partial^2 f}{\partial \alpha_2 \partial \alpha_1} & \frac{\partial^2 f}{\partial \alpha_2^2} \end{bmatrix} = \begin{bmatrix} 0 & a \\ a & 0 \end{bmatrix}$$

$$a = v_{00} + v_{11} - v_{10} - v_{01}$$

Eigenvalues and eigenvectors (Hessian is symmetric: always real)

$$\lambda_1 = -a \text{ and } \lambda_2 = a$$

$$v_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, v_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$



(here also: principal curvature magnitudes and directions of this function's graph == surface embedded in 3D)

Bi-Linear Interpolation: Critical Points

Examine Hessian matrix at critical point (non-degenerate critical p.?, ...)

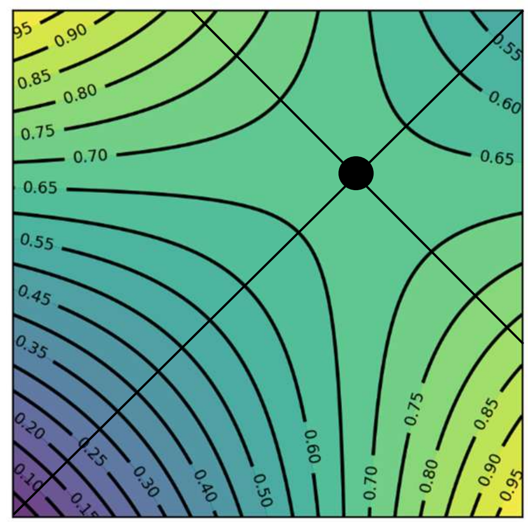
$$\begin{bmatrix} \frac{\partial^2 f}{\partial \alpha_1^2} & \frac{\partial^2 f}{\partial \alpha_1 \partial \alpha_2} \\ \frac{\partial^2 f}{\partial \alpha_2 \partial \alpha_1} & \frac{\partial^2 f}{\partial \alpha_2^2} \end{bmatrix} = \begin{bmatrix} 0 & a \\ a & 0 \end{bmatrix}$$

$$a = v_{00} + v_{11} - v_{10} - v_{01}$$

Eigenvalues and eigenvectors (Hessian is symmetric: always real)

$$\lambda_1 = -a \text{ and } \lambda_2 = a$$

$$v_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, v_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$



degenerate means determinant = 0 (at least one eigenvalue = 0);
bi-linear is simple: a = 0 means degenerated to
linear anyway: no critical point at all! (except constant function)
(but with more than one cell: can have max or min at vertices)

Interlude: Implicit Function Theorem

When can I write an implicit function in $\mathbb{R}^{n+m}$ such that it is the graph of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ *at least locally?*

That is: is this implicitly described function an *n*-manifold embedded in $\mathbb{R}^{n+m}$? (with local coordinates in $\mathbb{R}^n$)

$$G(f) := \{(x, f(x)) | x \in \mathbb{R}^n\} \subset \mathbb{R}^n \times \mathbb{R}^m \simeq \mathbb{R}^{n+m}$$

Theorem: if *m* x *m* Jacobian matrix is invertible

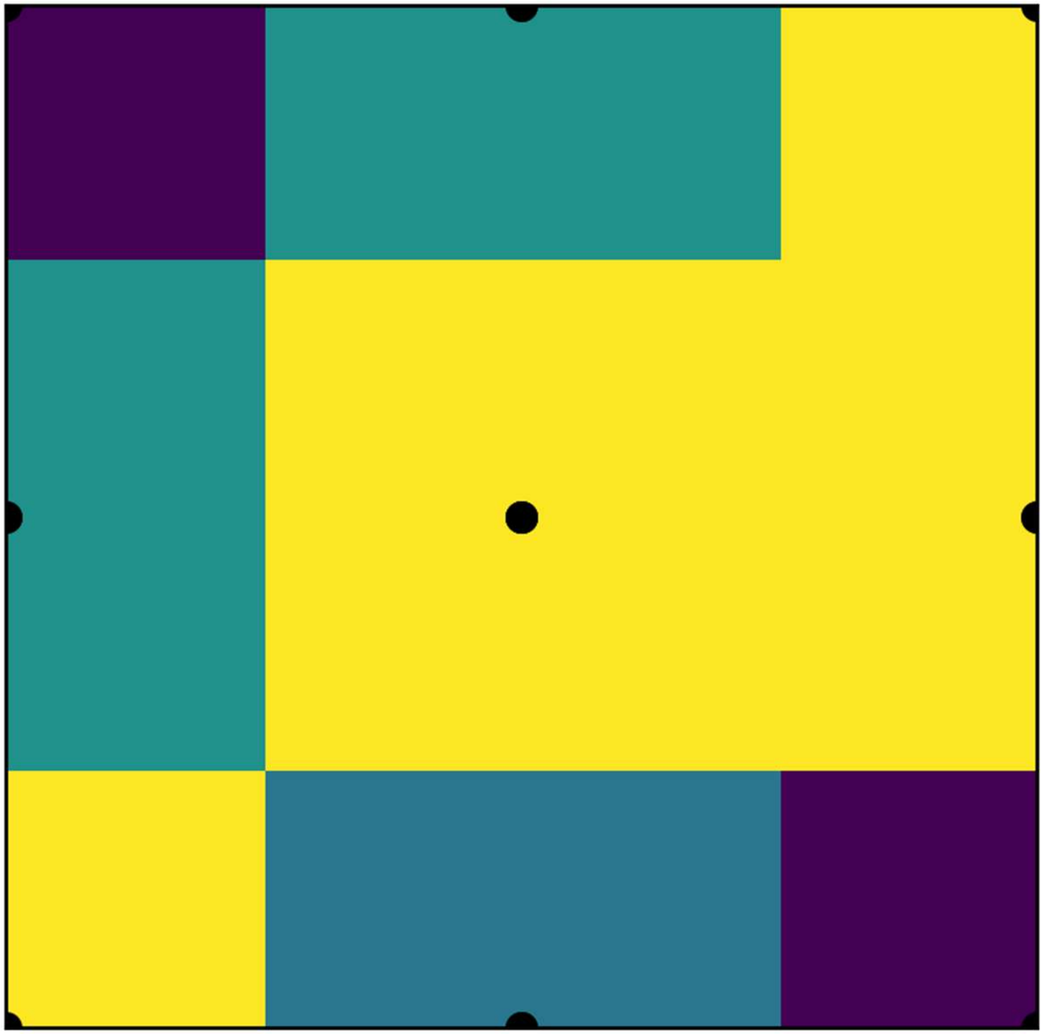
(easier for scalar field: check if gradient of f is non-zero)

See **`https://en.wikipedia.org/wiki/Implicit_function_theorem`**

General result: *constant rank theorem*

Bi-Linear Interpolation: Comparisons

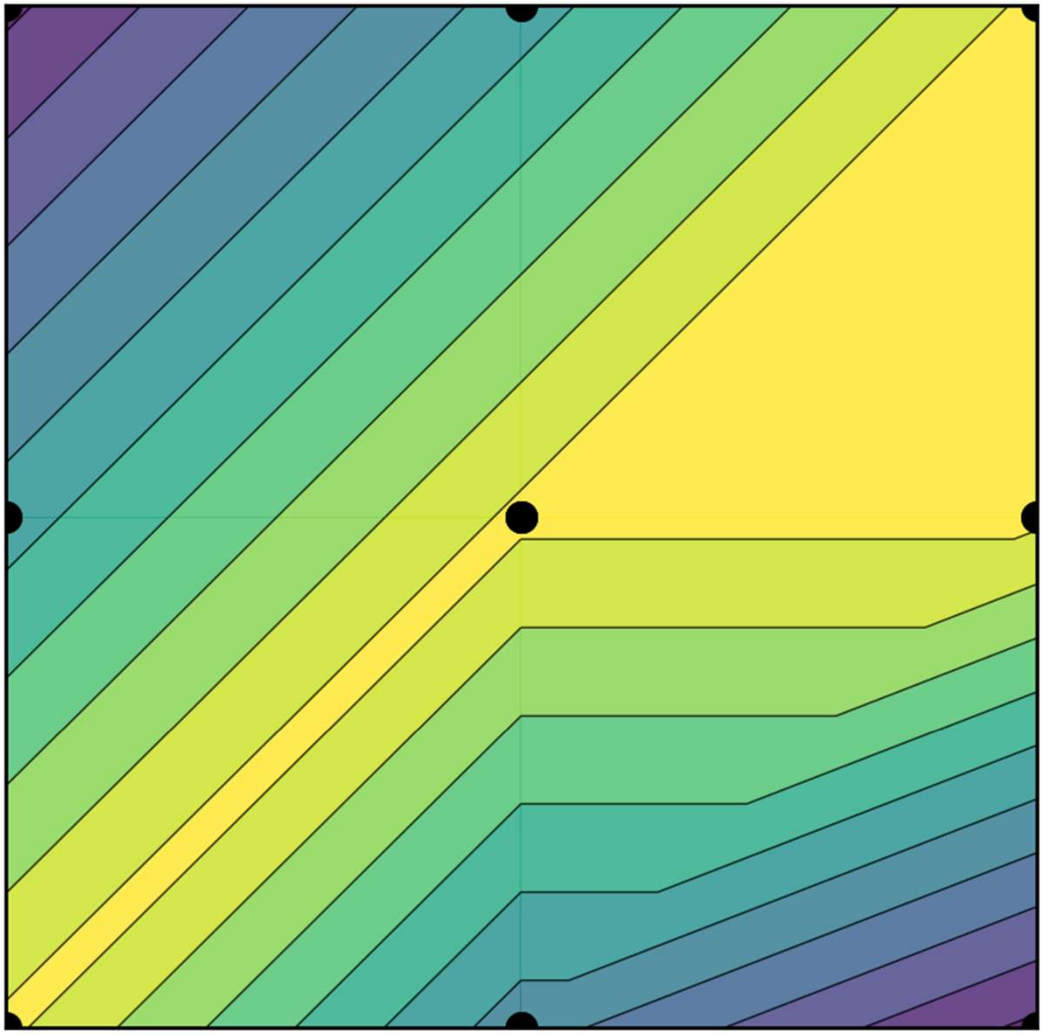
nearest-neighbor



Bi-Linear Interpolation: Comparisons

piecewise linear

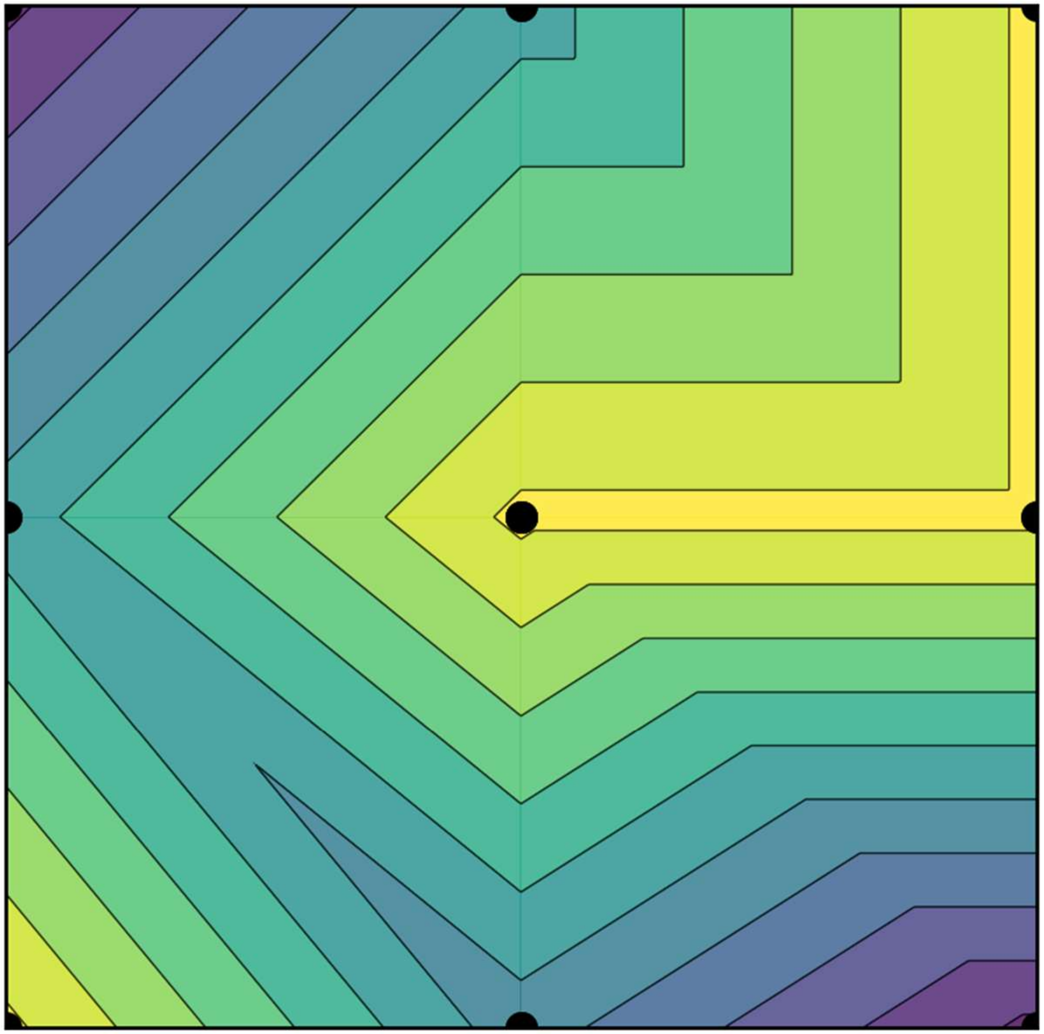
(2 triangles per quad;
diagonal:
bottom-left,
top-right)



Bi-Linear Interpolation: Comparisons

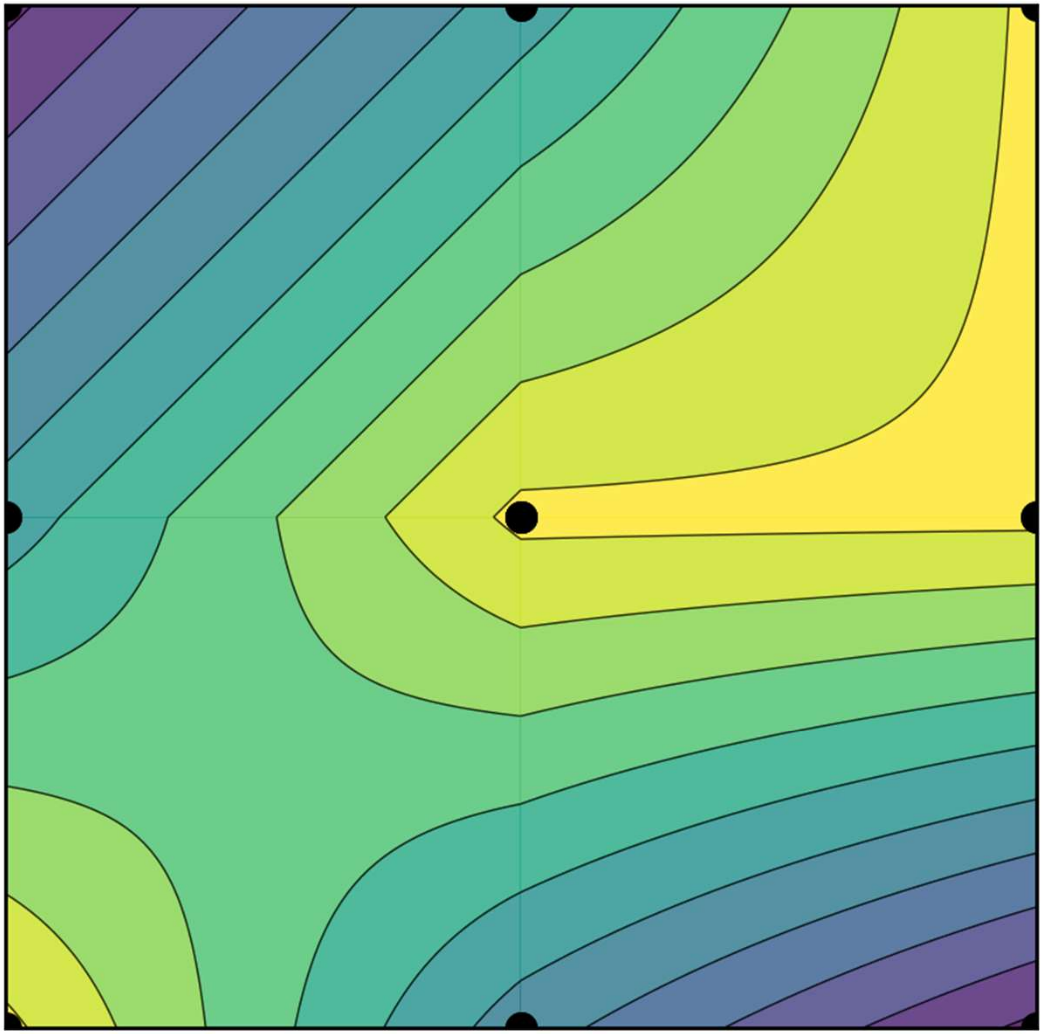
piecewise linear

(2 triangles per quad;
diagonal:
top-left,
bottom-right)



Bi-Linear Interpolation: Comparisons

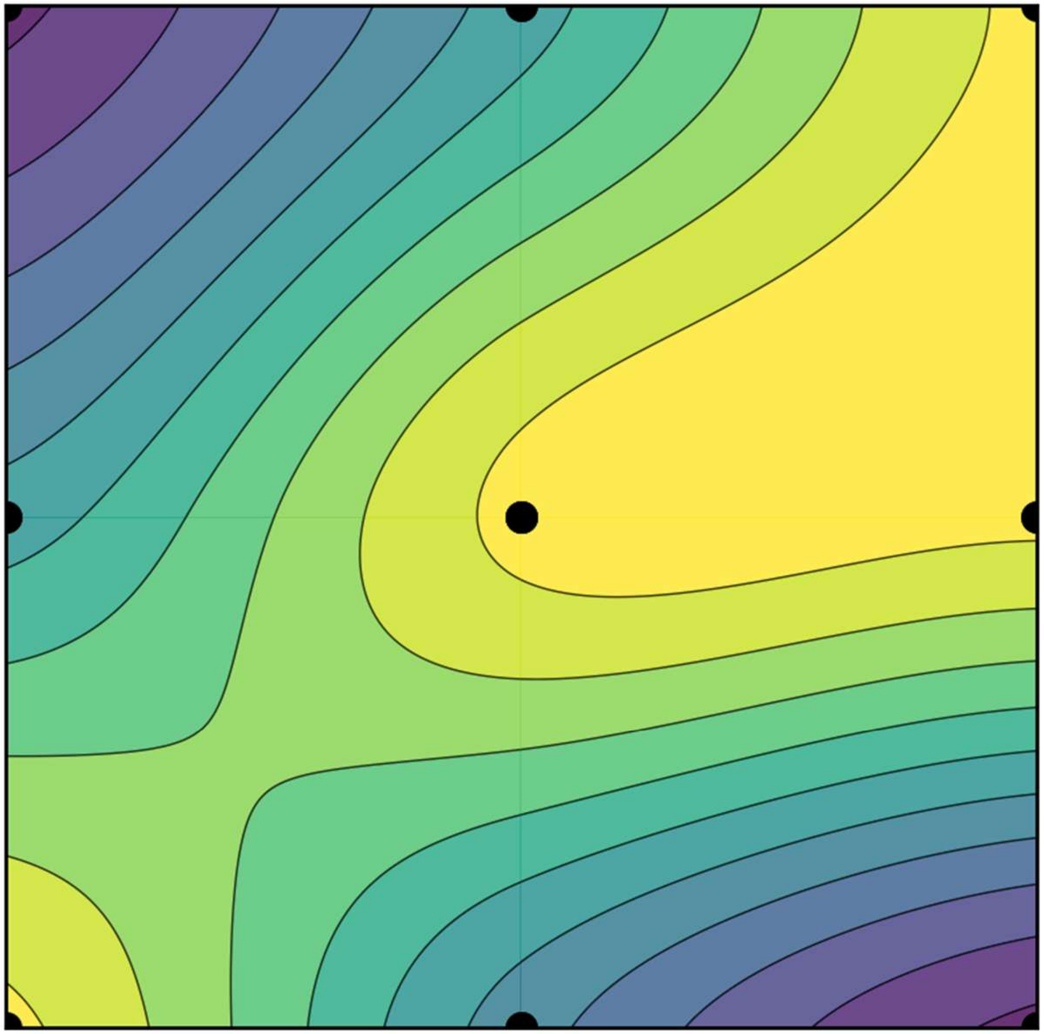
bi-linear



Bi-Linear Interpolation: Comparisons

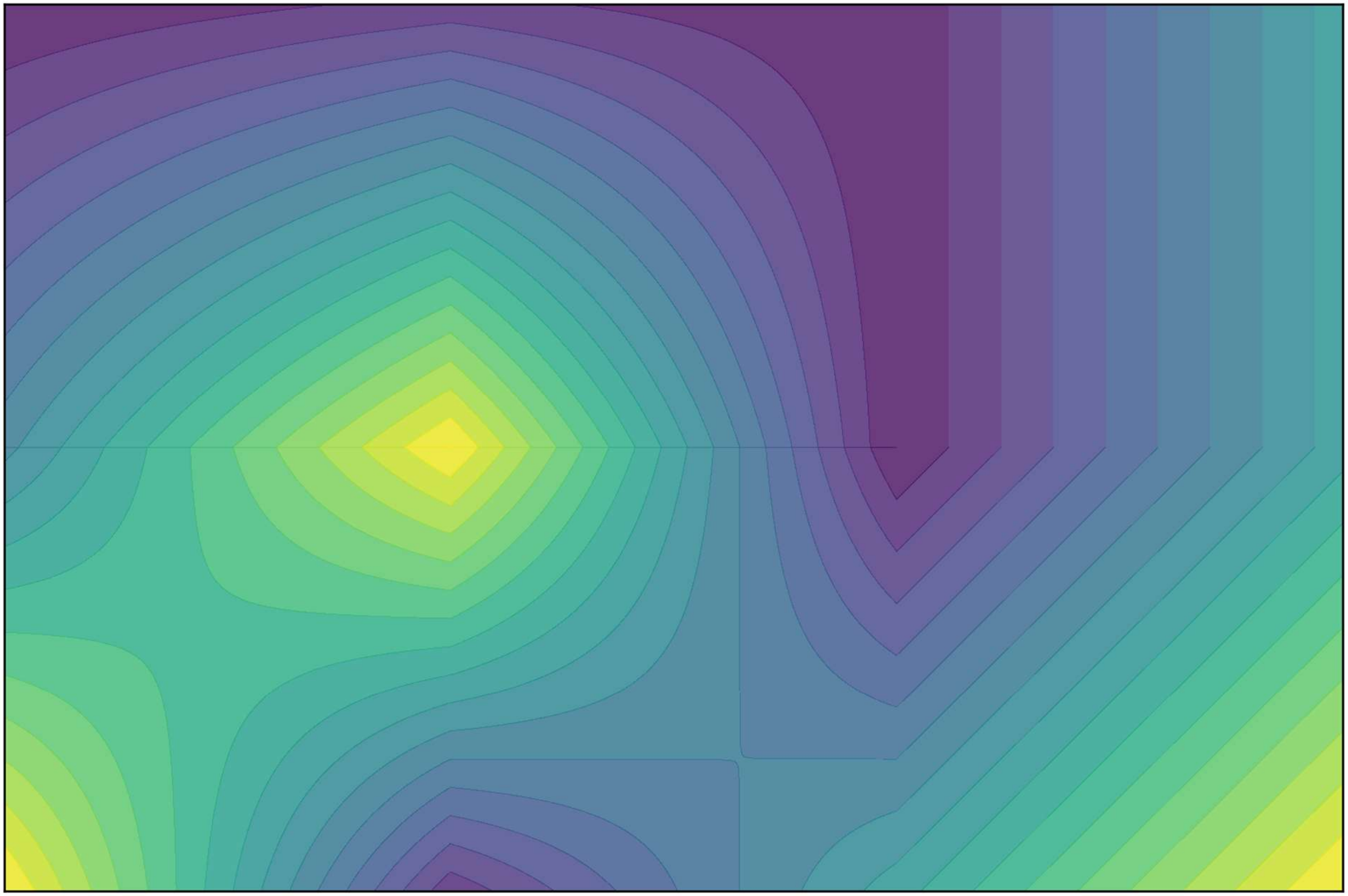
bi-cubic

(Catmull-Rom spline)

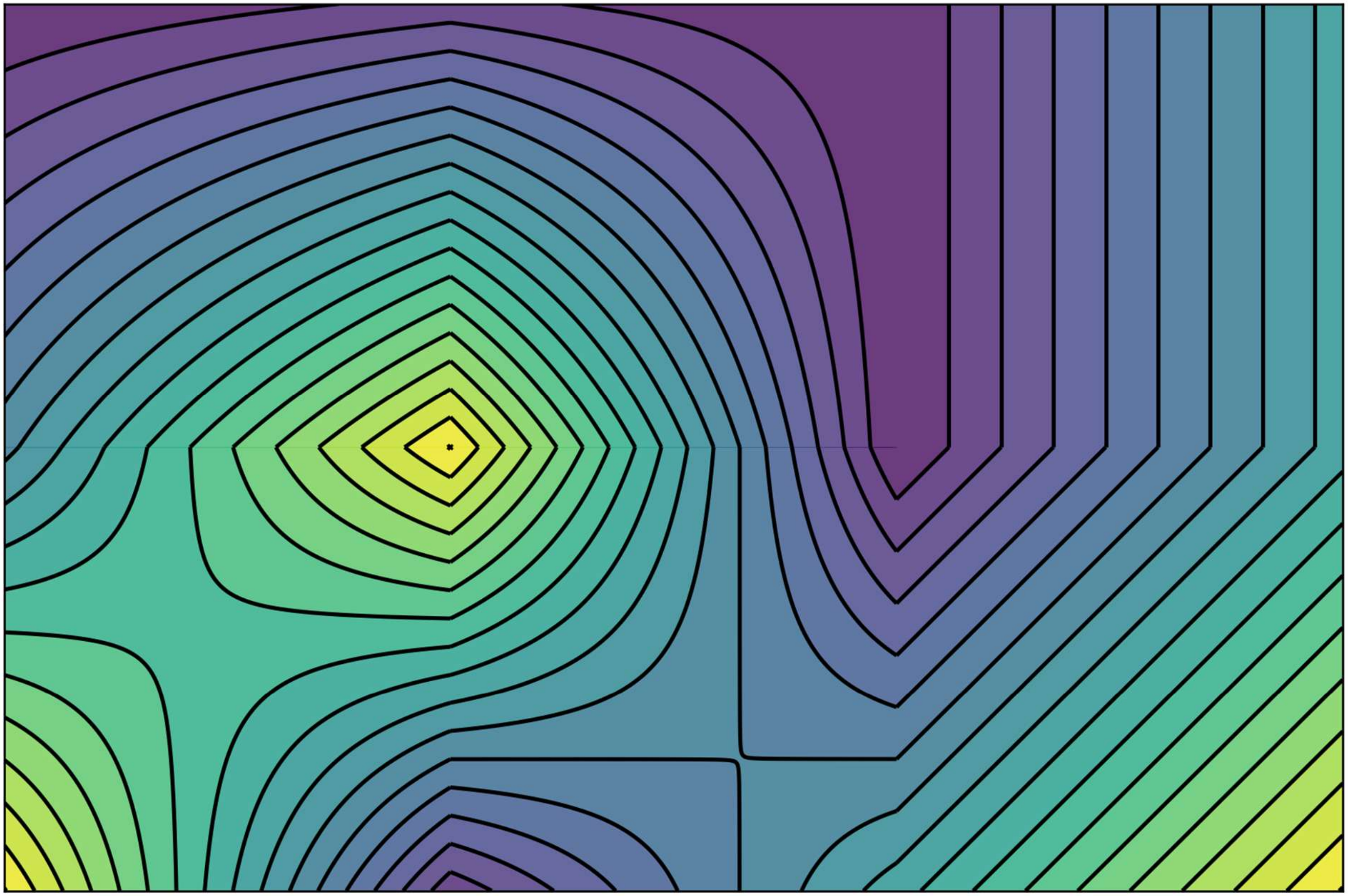


More examples

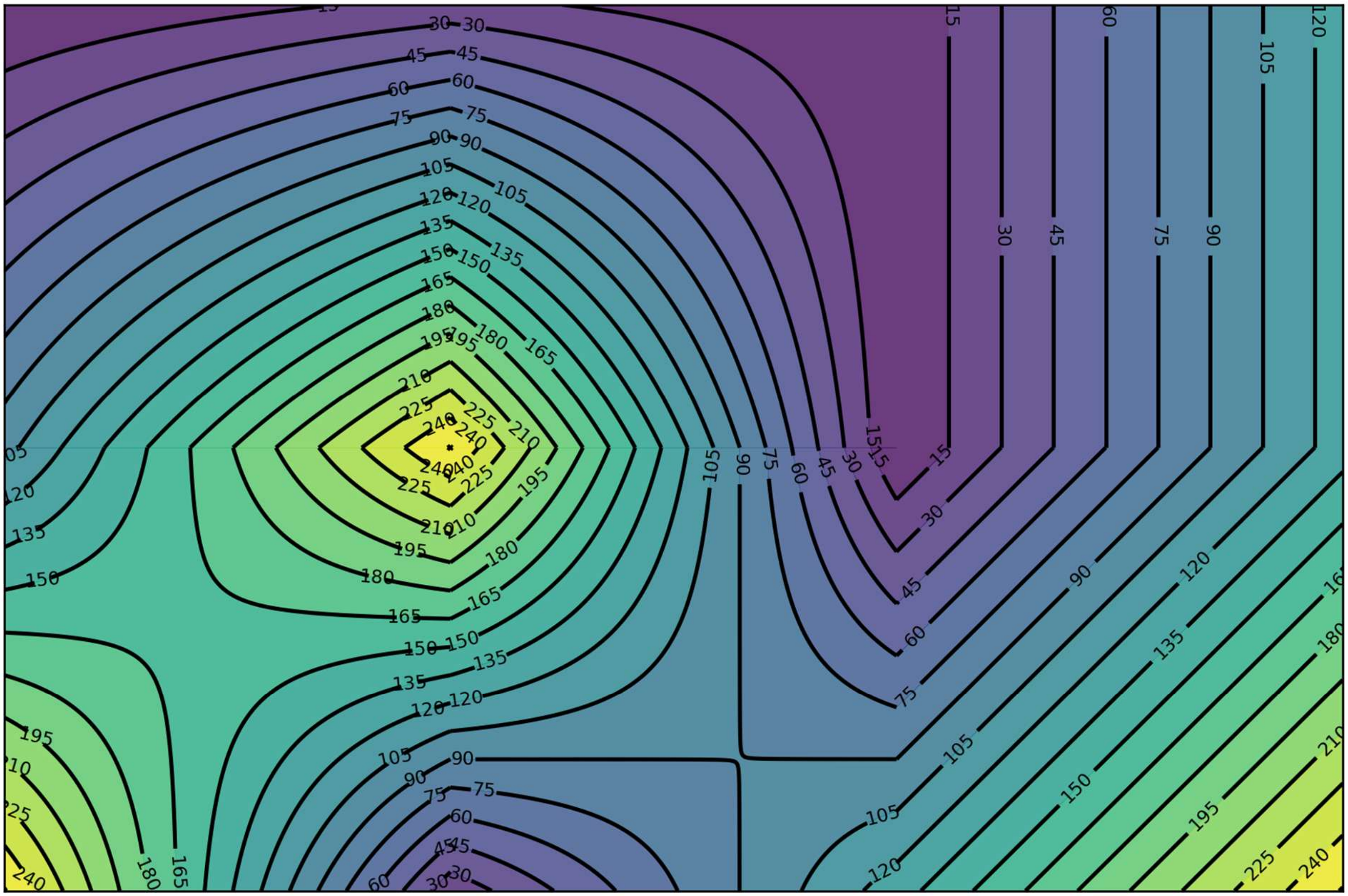
Piecewise Bi-Linear (Example: 3x2 Cells)



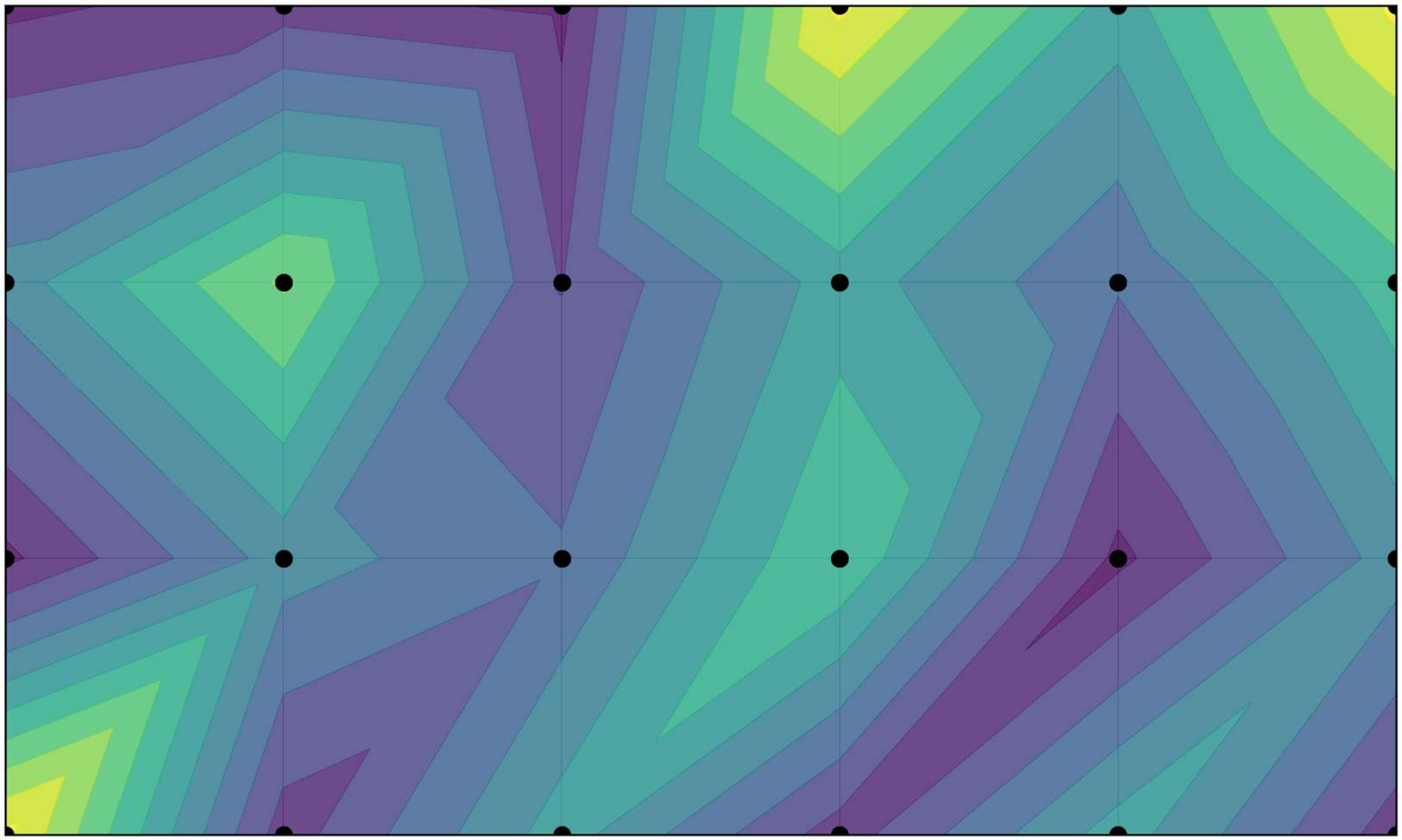
Piecewise Bi-Linear (Example: 3x2 Cells)



Piecewise Bi-Linear (Example: 3x2 Cells)

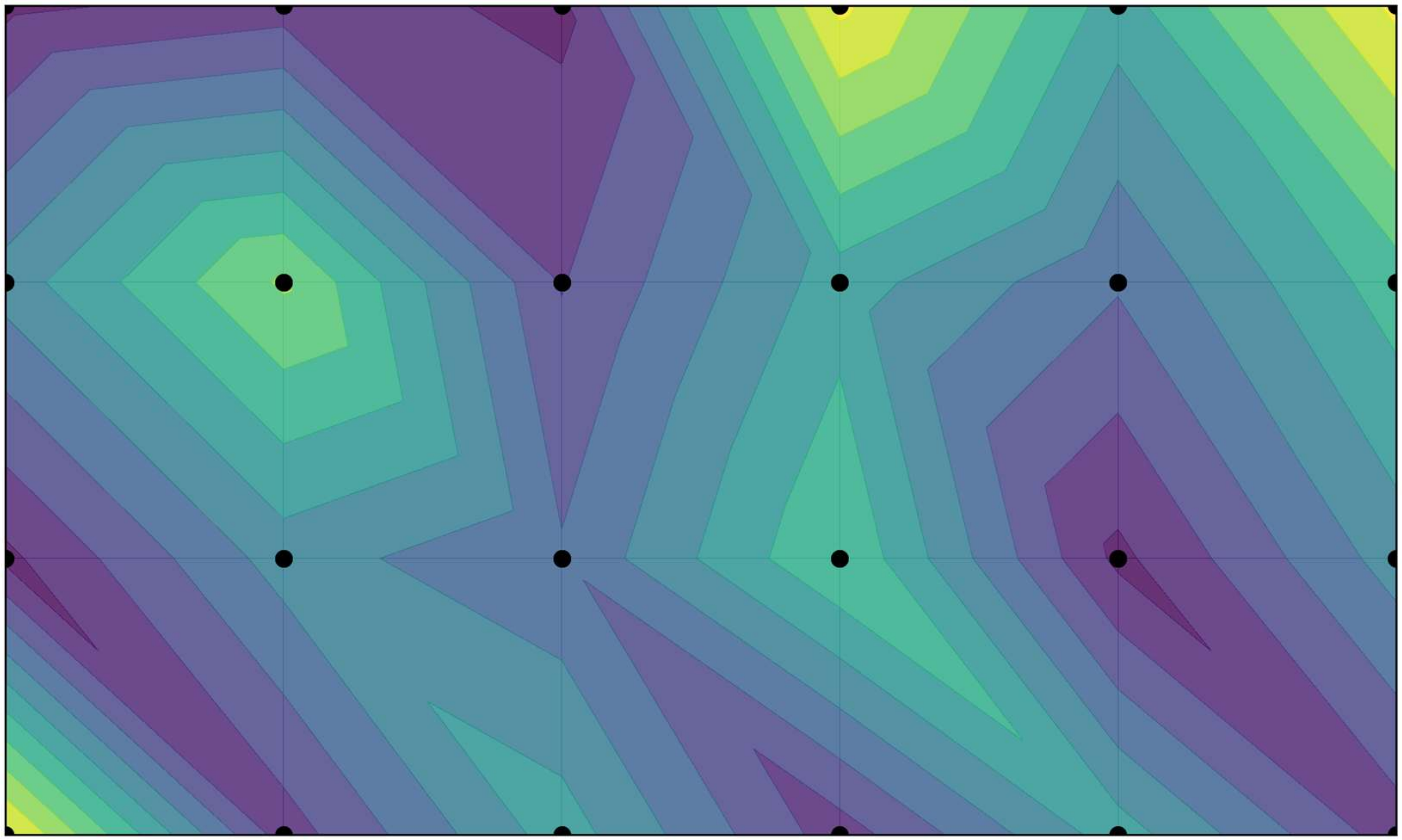


Bi-Linear Interpolation: Comparisons



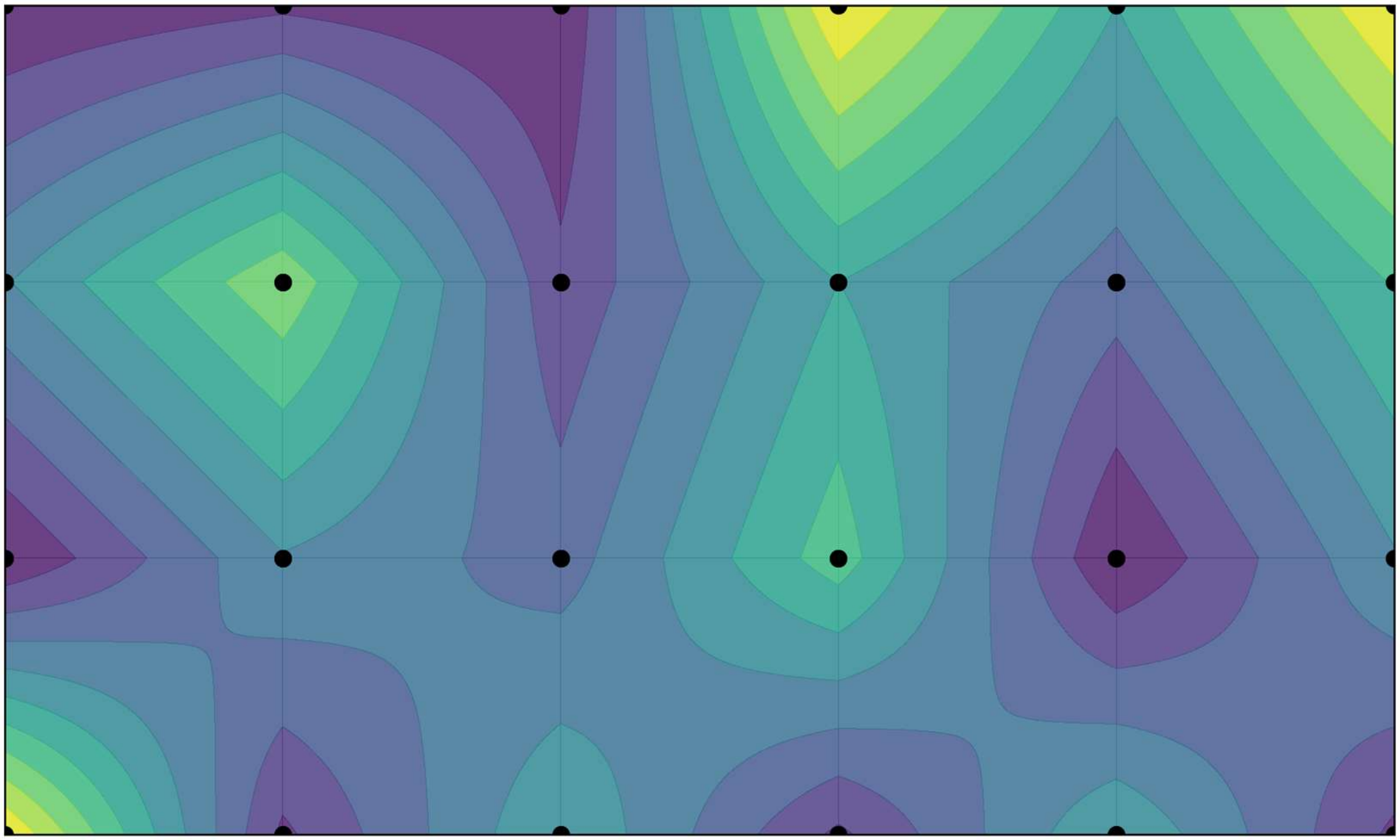
linear (diagonal 1)

Bi-Linear Interpolation: Comparisons



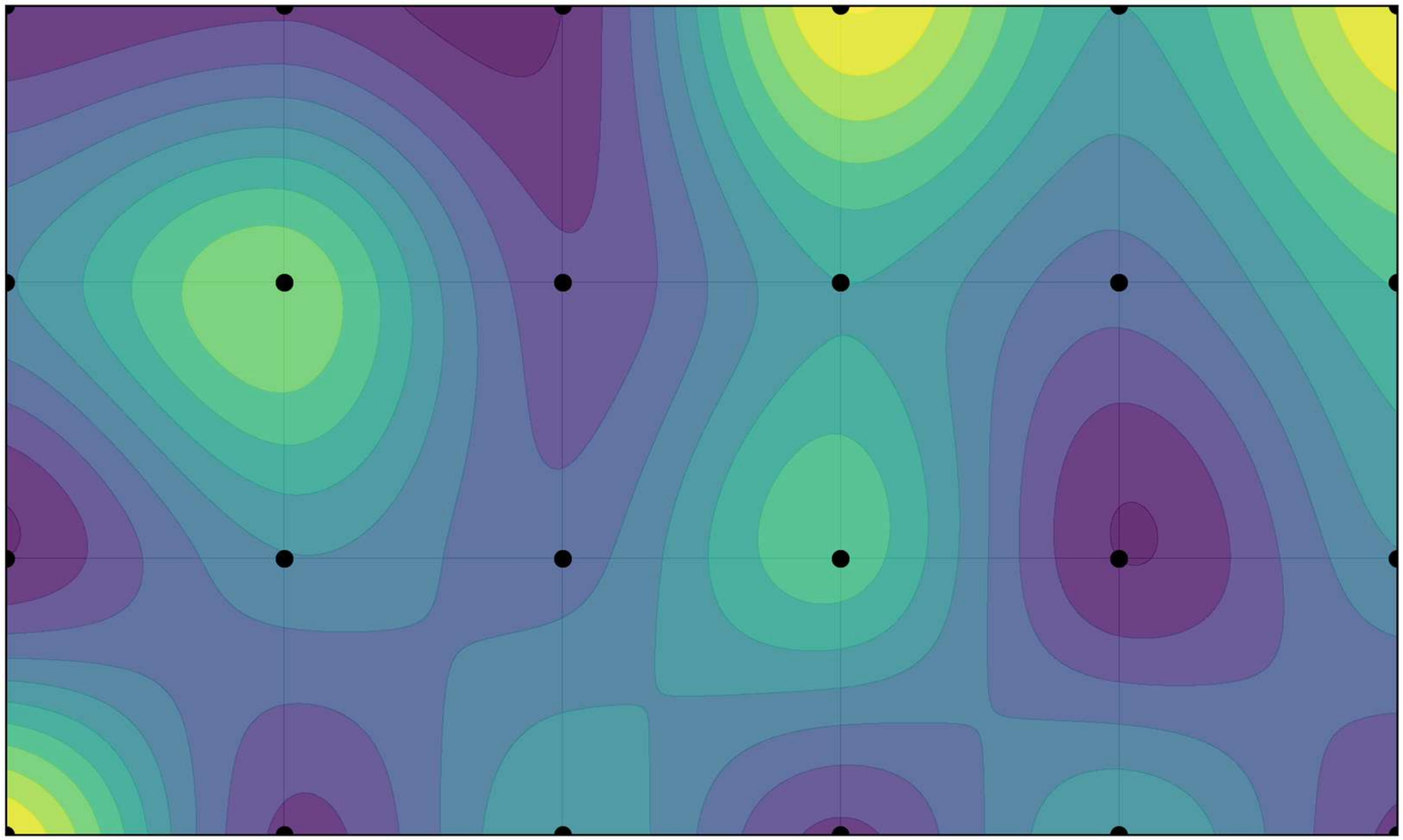
linear (diagonal 2)

Bi-Linear Interpolation: Comparisons



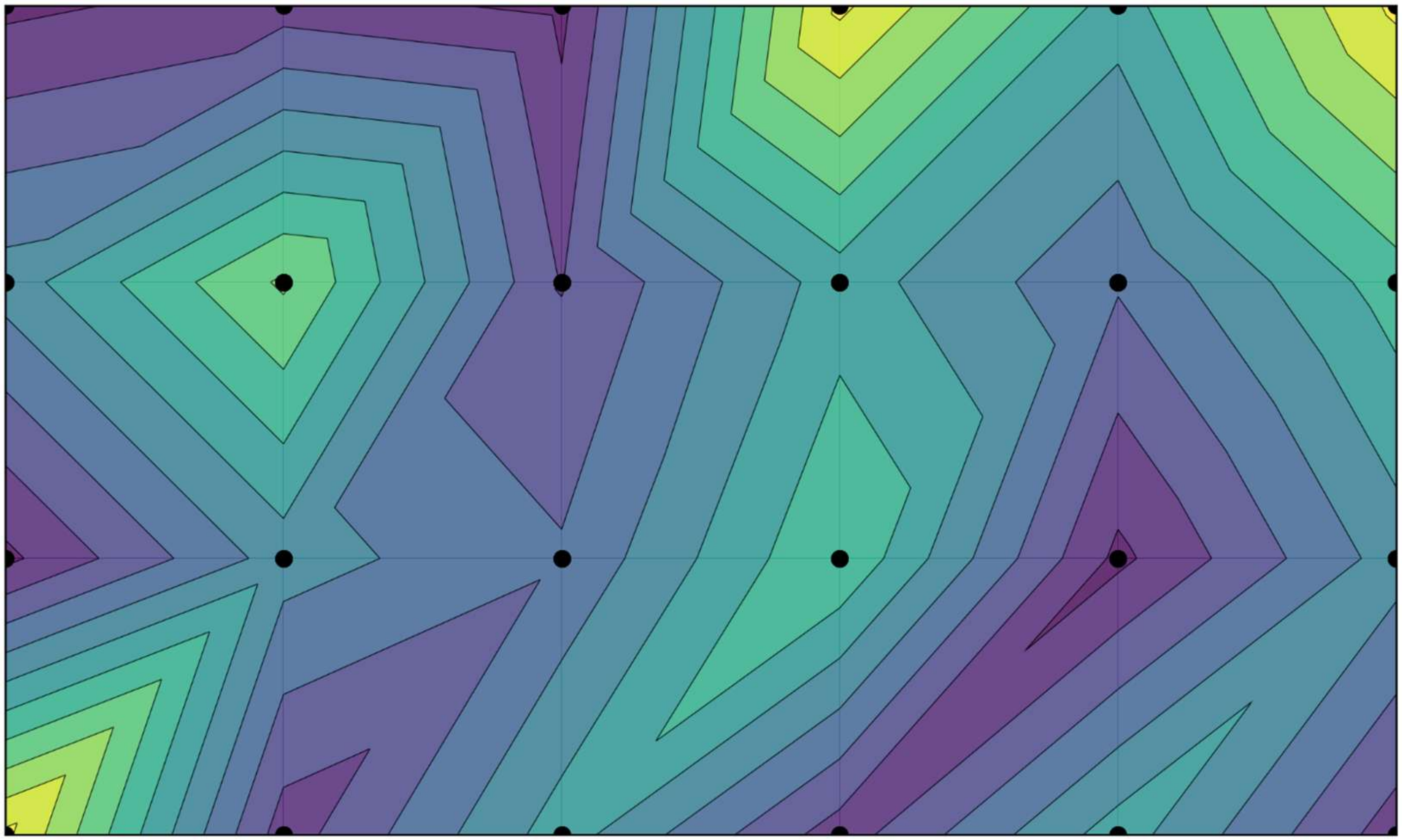
bi-linear (in 3D: tri-linear)

Bi-Linear Interpolation: Comparisons



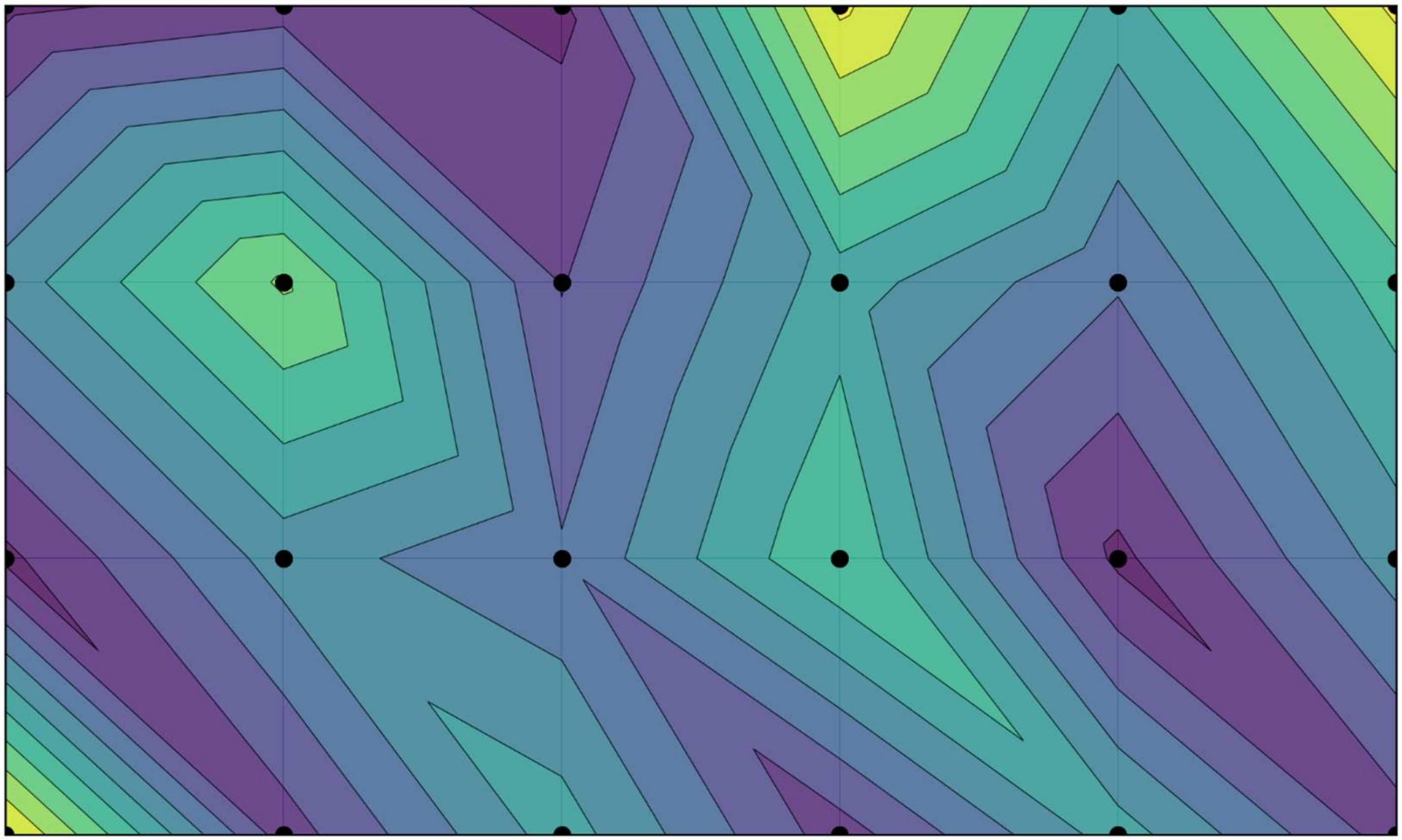
bi-cubic (in 3D: tri-cubic)

Bi-Linear Interpolation: Comparisons



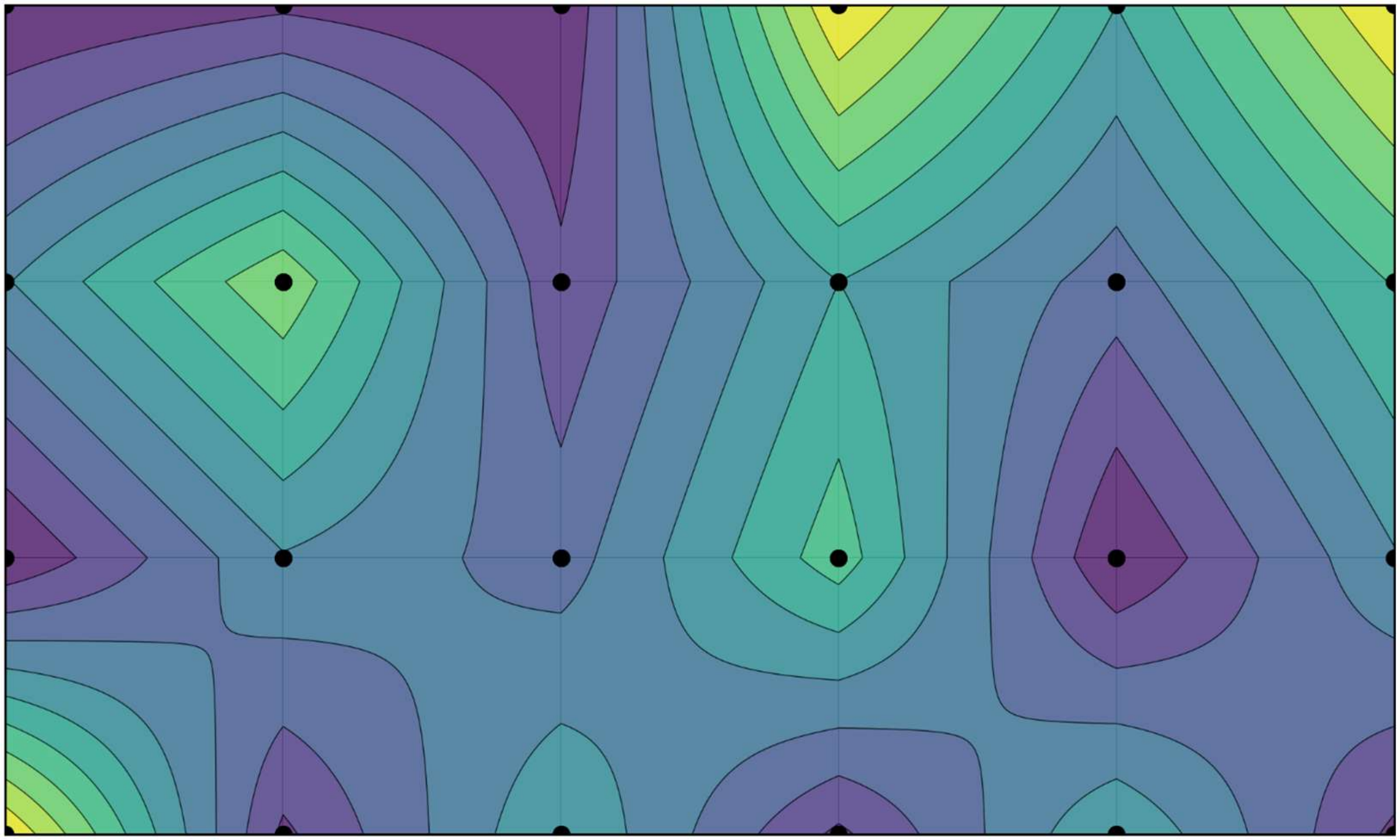
linear (diagonal 1)

Bi-Linear Interpolation: Comparisons



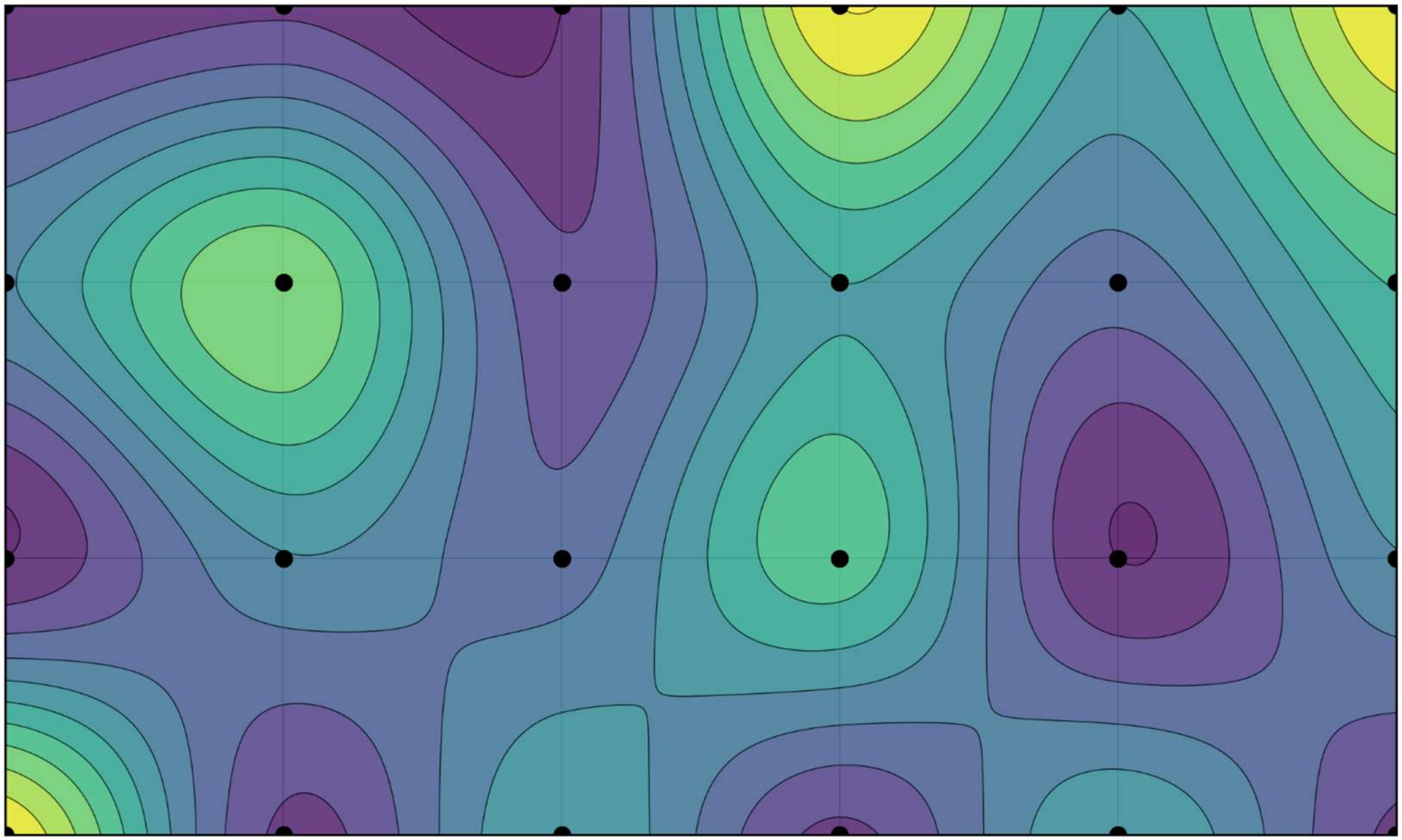
linear (diagonal 2)

Bi-Linear Interpolation: Comparisons



bi-linear (in 3D: tri-linear)

Bi-Linear Interpolation: Comparisons



bi-cubic (in 3D: tri-cubic)

Thank you.

Thanks for material

- Helwig Hauser
- Eduard Gröller
- Daniel Weiskopf
- Torsten Möller
- Ronny Peikert
- Philipp Muigg
- Christof Rezk-Salama