Delaunay Triangulation

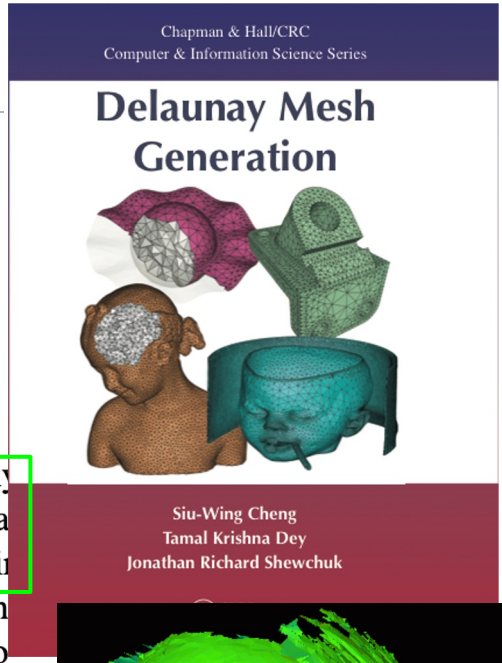
09.10.2023

Ihor Makar

Agenda

1. Definition and properties of Delaunay Triangulation
2. Relation to 3D convex hull;
3. Overview of popular algorithms to construct Delaunay Triangulation (Incremental edge flip, delaunay refinement)
4. What is good linear finite element ?
5. Overview and demo of popular commercial packages for 2D ("Triangle" by Richard Shevchuk, "Delos")

Delaunay Mesh Generation (Chapman & Hall/CRC Computer and Information Science Series)



by S.W. Cheng, T. K. Dey, J. R. Shewchuk, 2012

https://people.eecs.berkeley.edu/~jrs/papers/meshbook/chapter1.pdf

1.4 A personal history of working in mesh generation

When we came to study mesh generation in the 1990s, we were drawn by the unusually strong way it combines theory and practice, complexity and elegance, and combinatoria and numerical computing. There is a strong tradition of practical meshing algorithms i scientific computing and computer graphics, yet their difficulty and fragility bring up fun damental theoretical questions in approximation theory, surface sampling, topology, algo rithm design, numerical computing, and the structure of Delaunay triangulations and their weighted and constrained relatives. Mesh generation demands an understanding of both combinatorial and numerical algorithms, because meshing is geometric and most meshes are used by numerical applications. Lastly, meshes and their applications are as attractive to the eye as their mathematics are to the soul.

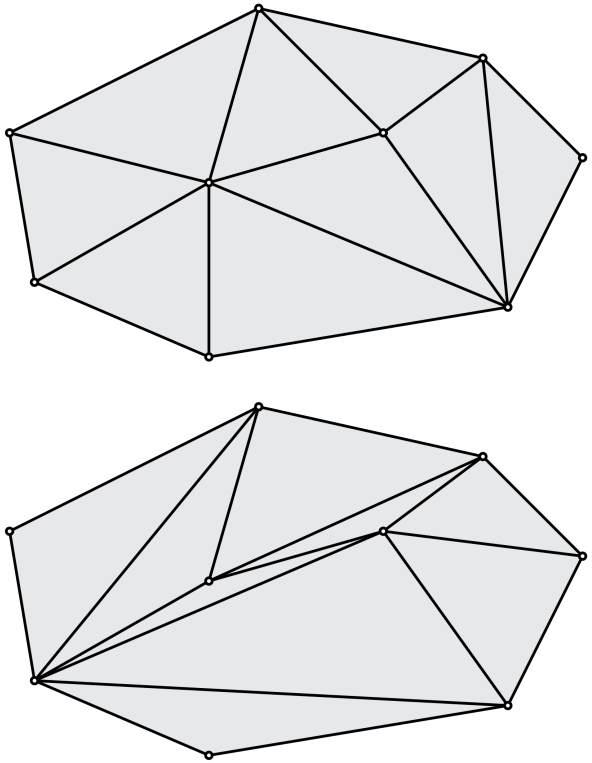
Galvanized by the publication of Ruppert's algorithm, Jonathan generalized it to three dimensions in 1997. The tetrahedral Delaunay refinement algorithm described in Chapter 8 accepts nonconvex domains with internal boundaries and offers guaranteed good grading.

https://people.eecs.berkeley.edu/~jrs/

Triangulation

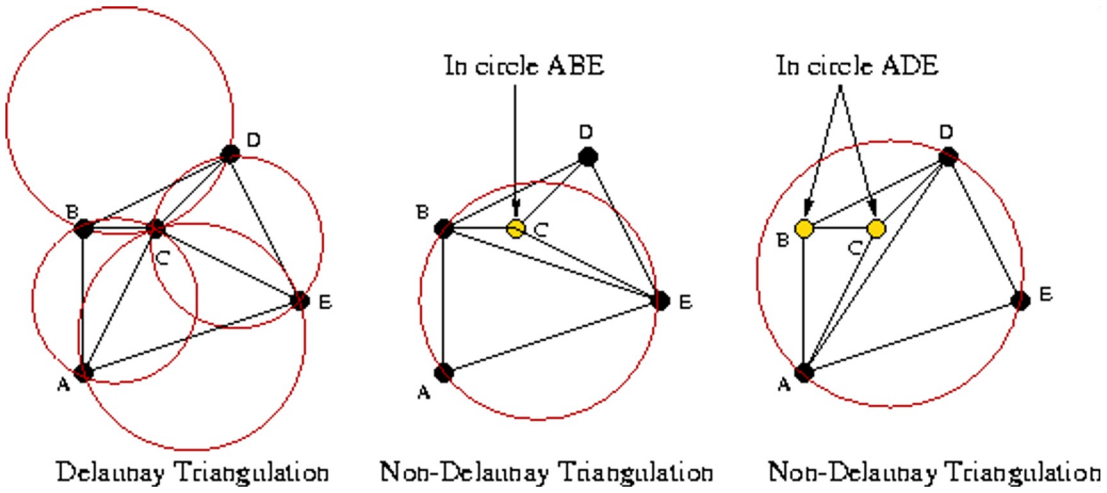
In the plane (when P is a set of points in), **triangulations** are made up of triangles, together with their edges and vertices.

Two different triangulations of the same set of 9 points in the plane:



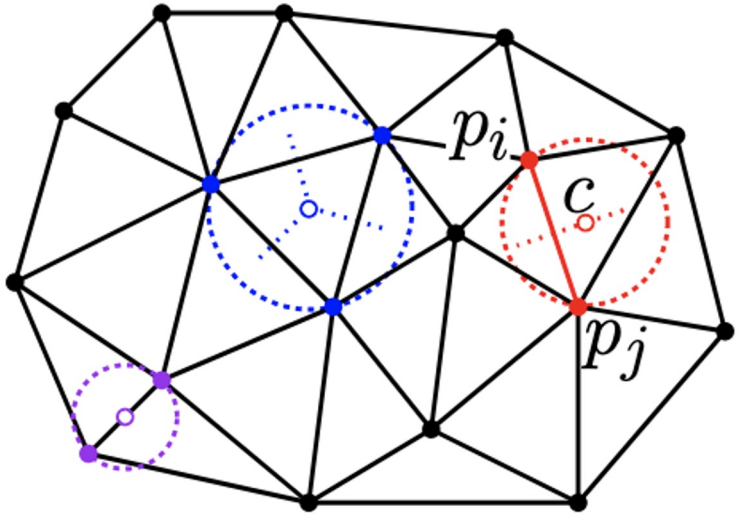
Delaunay triangulation

In mathematics and computational geometry, a **Delaunay triangulation** for a given set P of discrete points is a triangulation DT(P) such that no point in P is inside the circumcircle of any triangle in DT(P).



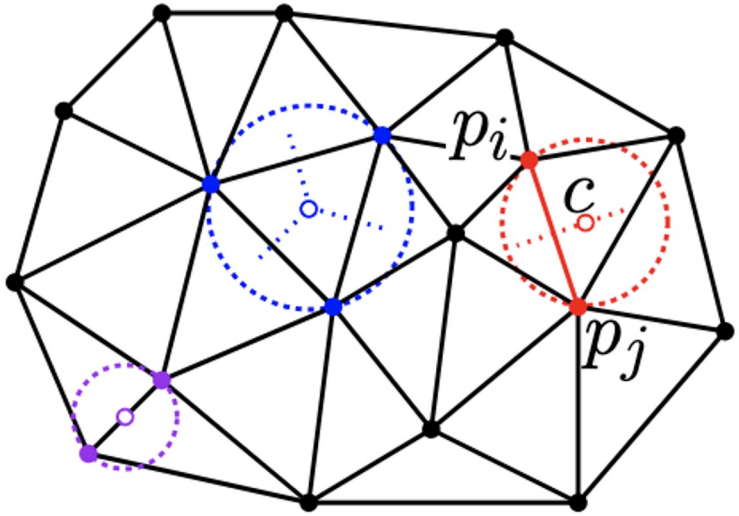
Circumcircle property

The circumcircle of any triangle in the Delaunay triangulation is "empty," that is, the interior of the associated circular disk contains no sites of P (see blue line).



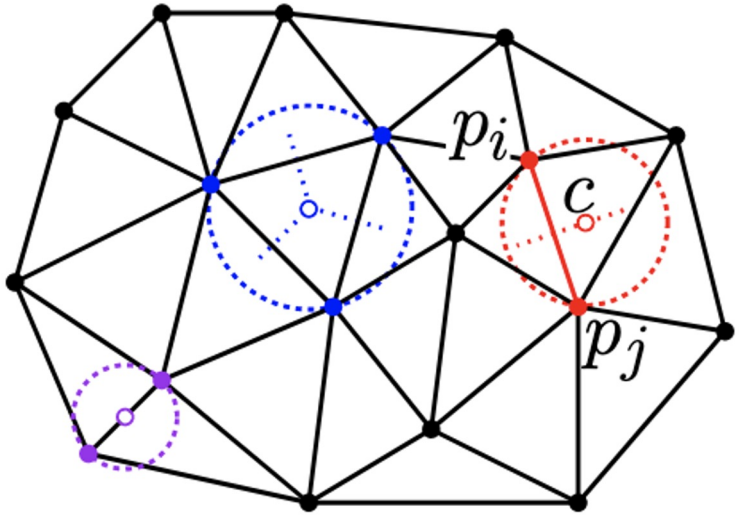
Empty circle property

Two sites pi and pj are connected by an edge in the Delaunay triangulation, if and only if there is an empty circle passing through pi and pj. (see red line)



Closest pair property

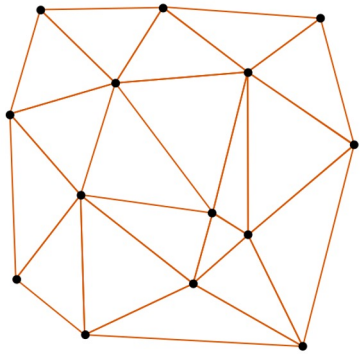
The closest pair of sites in P are neighbors in the Delaunay triangulation (see the purple circle)



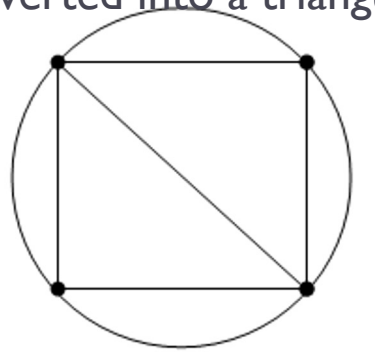
https://en.wikipedia.org/wiki/Gabriel_graph

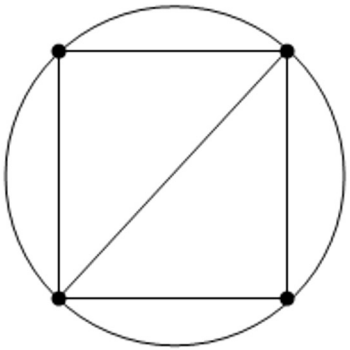
Is Delaunay triangulation unique ?

- A Delaunay triangulation is unique iff the circumcircle of every triangle contains exactly three points o e vertices of the triangle.



- For instance, the Delaunay diagram of the four vertices of a square is a square, and can be converted into a triangulation in two different ways.





Relation to the minimum spanning tree

Given a graph, the minimum spanning tree (MST) is a set of (n − 1) edges that connect the points (into a free tree) such that the total weight of edges is minimized.

Theorem: The minimum spanning tree of a set P of point sites (in any dimension) is a subgraph of the Delaunay triangulation

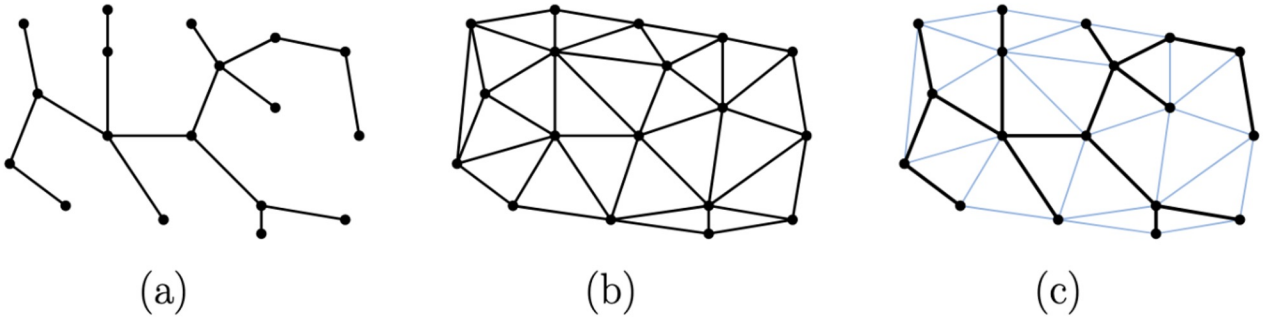


Fig. 2: (a) A point set and its EMST, (b) the Delaunay triangulation, and (c) the overlay of the two.

Spanner Properties

A natural observation about Delaunay triangulations is that its edges would seem to form a reasonable **transportation road network between the points.**

This is closely related to the theory of **geometric spanners**, that is, geometric graphs whose shortest paths are not significantly longer than the straight-line distance.

Given any parameter $t \geq 1$, we say that G is a **t-spanner** if for any two points $p, q \in P$, the shortest path length between p and q in G is at most a factor t longer than the Euclidean distance between these points, that is

$$\delta_G(p, q) \leq t\|pq\|$$

Spanner Properties

Theorem: Given a set of points P in the plane, the Delaunay triangulation of P is a t-spanner for $t = 4\pi\sqrt{3}/9 \approx 2.418$.

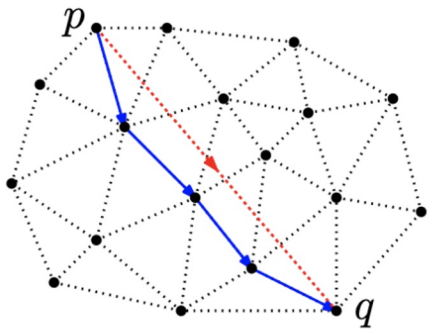


Fig. 4: Spanner property of the Delaunay Triangulation.

It had been conjectured for many years that the Delaunay triangulation is a $(\pi/2)$-spanner ($\pi/2 \approx 1.5708$). This was disproved in 2009, and the lower bound now stands at roughly 1.5846. Closing the gap between the upper and lower bound is an important open problem.

Relation to 3D Convex Hull

Theorem

Let $P = \{p_1, \ldots, p_n\}$ with $p_i = (a_i, b_i, 0)$. Let $p_i^* = (a_i, b_i, a_i^2 + b_i^2)$ be the vertical projection of each point p_i onto the paraboloid $z = x^2 + y^2$. Then $Del(P)$ is the orthogonal projection onto the plane $z = 0$ of the lower convex hull of P^*.

p_i^*, p_j^*, p_k^* form a (triangular) face of the lower convex hull of P^*

$\Updownarrow$

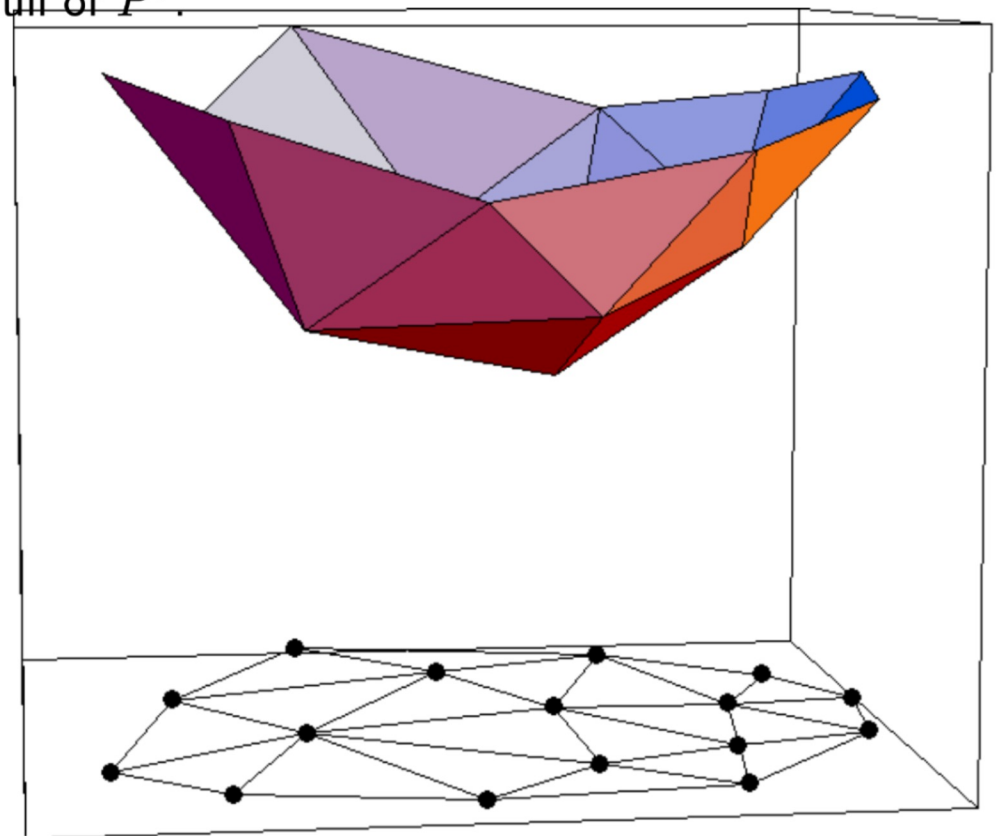
The plane through p_i^*, p_j^*, p_k^* leaves all the remaining points of P^* above it

$\Updownarrow$

The circle through p_i, p_j, p_k leaves all the remaining points of P in its exterior

$\Updownarrow$

p_i, p_j, p_k form a triangle of $Del(P)$



```
main()
{
        int [NMAX],y[NMAX],z[NMAX];  /* input points xy,z = x^2 + y^2 */
        int n;                       /* number of input points */
        int i, j, k, m;              /* indices of four points */
        int xn, yn, zn;              /* outward normal to (i,j,k) */
        int flag;                    /* t if m above of (i,j,k) */

        /* Input points and compute z = x^2 + y^2. */
        scanf("%d", &n);
        for ( i=0; i<n; i++ ) {
                scanf("%d %d", &x[i], &y[i]);
                z[i]=x[i] * x[i]+y[i] * y[i];
        }

        /* For each triple (i,j,k) */
        for ( i=0; i < n-2; i++ )
        for ( j=i+1; j < n; j++ )
        for ( k=i+1; k < n; k++ )
        if ( j != k ) {
           /* Compute normal to triangle (i,j,k). */
           xn=(y[j]-y[i])*(z[k]-z[i])-(y[k]-y[i])*(z[j]-z[i]);
           yn=(x[k]-x[i])*(z[j]-z[i])-(x[j]-x[i])*(z[k]-z[i]);
           zn=(x[j]-x[i])*(y[k]-y[i])-(x[k]-x[i])*(y[j]-y[i]);

           /* Only examine faces on bottom of paraboloid: zn < 0. */
           if ( flag=(zn < 0) )
                   /* For each other point m */
                   for (m=0; m<n; m++)
                           /* Check if m above (i,j,k). */
                           flag=flag &&
                                   ((x[m]-x[i])*xn +
                                    (y[m]-y[i])*yn +
                                    (z[m]-z[i])*zn <= 0);
        if (flag)
                printf("%d\t%d\t%d\n", i, j, k);
    }
}
```

Code 5.1 $O(n^4)$ Delaunay triangulation algorithm.

output its two endpoints (either their coordinates, or an index into your Voronoi vertex list). For each unbounded Voronoi [illegible] endpoint and a vector

J. O'Rourke
Computational Geometry in C,
Cambridge University Press, 1994

page 202

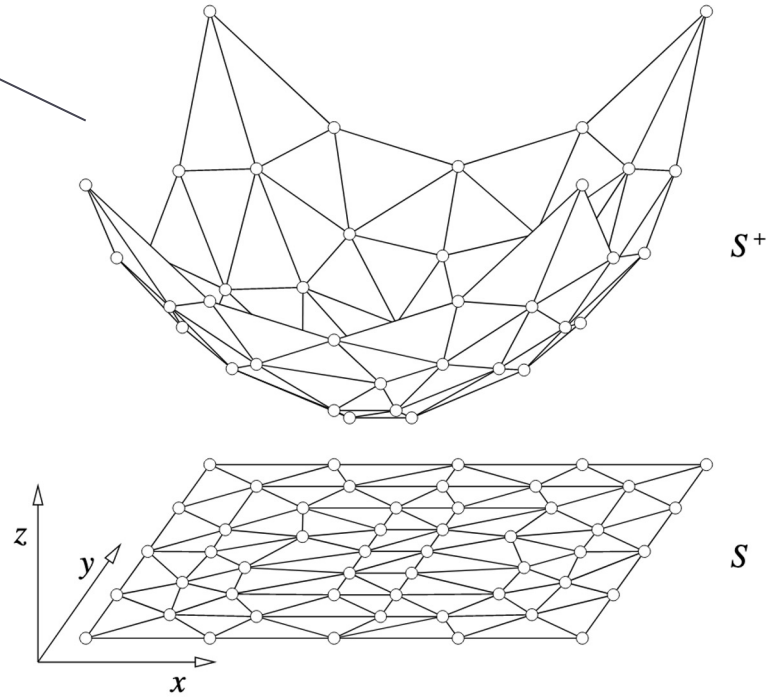


Figure 2.5: The parabolic lifting map.

Relation to 3D Convex Hull

This relationship between Delaunay triangulations and convex hulls has two consequences. First, it makes many properties of the Delaunay triangulation intuitive. For example, from the fact that every finite point set has a polyhedral convex hull, it follows that every finite point set has a Delaunay triangulation. Second, it brings to mesh generation the power of a huge literature on polytope theory and algorithms. For example, every convex hull algorithm is a Delaunay triangulation algorithm!

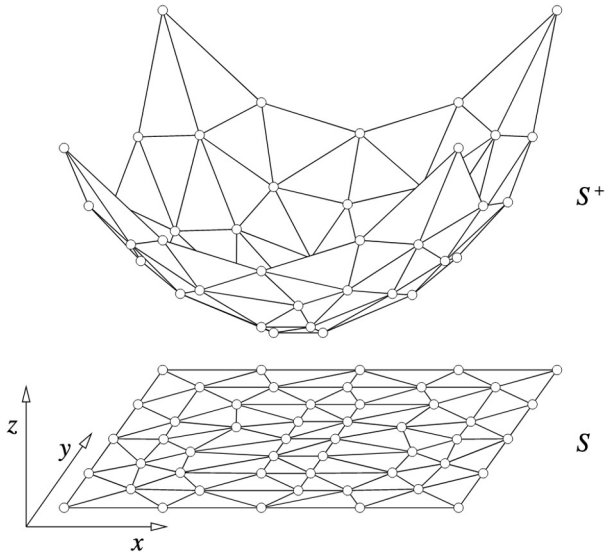


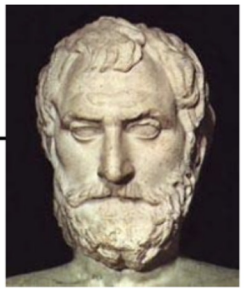
Figure 2.5: The parabolic lifting map.

Maximizing Angles and Edge Flipping

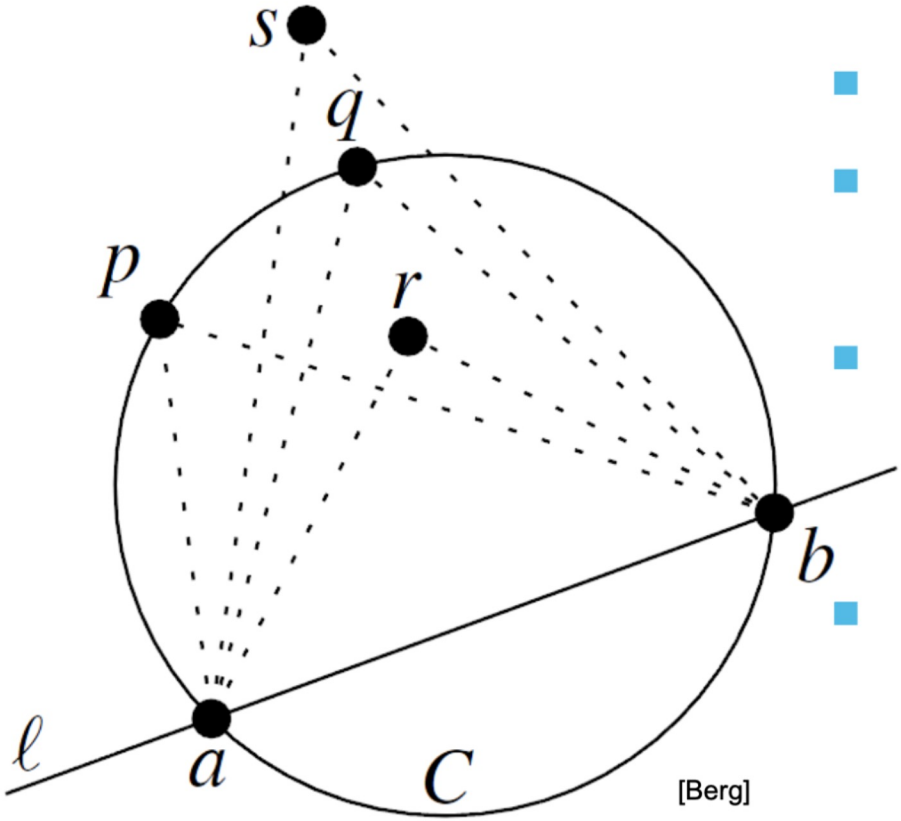
- Let *T* be a triangulation with *m* triangles (and 3*m angles*)
- Angle-vector

  = non-decreasing ordered sequence $(\alpha_1, \alpha_2, \ldots, \alpha_{3m})$
  inner angles of triangles, $\alpha_i \leq \alpha_j$, for $i < j$
- In the plane, Delaunay triangulation has the lexicographically largest angle sequence
  - It maximizes the minimal angle (the first angle in angle-vector)
  - It maximizes the second minimal angle, …
  - It maximizes all angles
  - It is an angle sequence optimal triangulation

Thales's theorem (624-546 BC)



Respective Central Angle Theorem

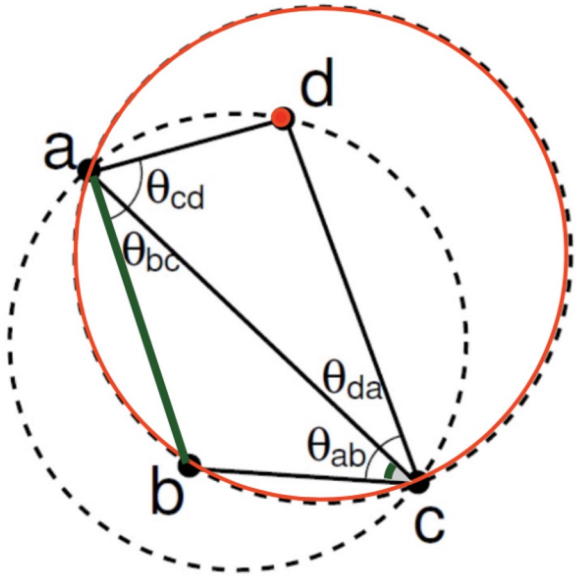


- Let C = circle,
- l =line intersecting C in points a, b
- p, q, r, s = points on the same side of l
  p,q on C , r is *in*, s is out
- Then for the angles holds:
  $\sphericalangle arb > \sphericalangle apb = \sphericalangle aqb > \sphericalangle asb$

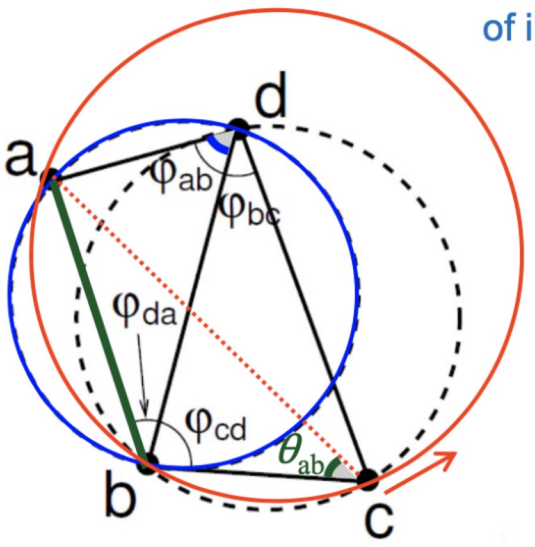
http://www.mathopenref.com/arccentralangletheorem.html

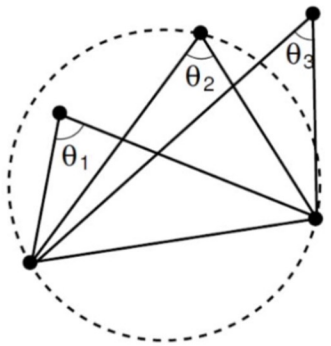
The minimum angle increases after the edge flip

of illegal edge ac > bd



flip(ac)





$\theta_1 > \theta_2 > \theta_3$

[Mount]

$|bd| < |ac|$ $\varphi_{ab} > \theta_{ab}$ $\varphi_{bc} > \theta_{bc}$ $\varphi_{cd} > \theta_{cd}$ $\varphi_{da} > \theta_{da}$

=> After limited number of edge flips

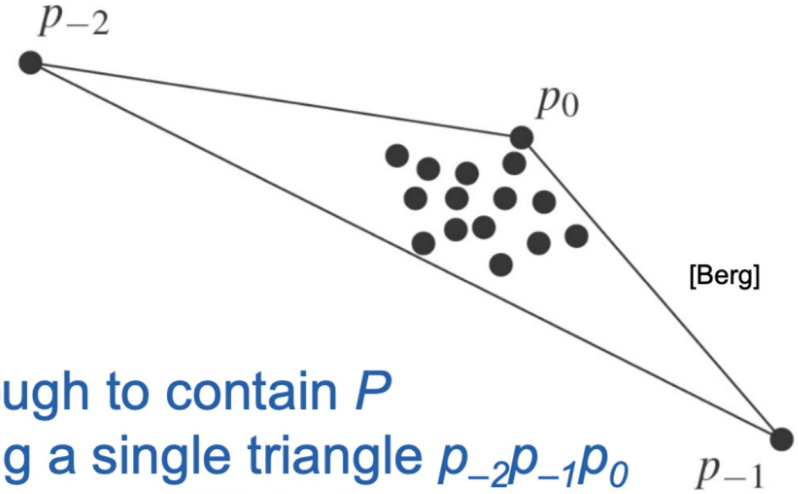
- Terminate with lexicographically maximum triangulation
- It satisfies the empty circle condition => Delauney T.

Incremental algorithm principle

1. Create a large triangle containing all points (to avoid problems with unbounded cells)
   - must be larger than the largest circle through 3 points
   - will be discarded at the end
2. Insert the points in random order
   - Find triangle with inserted point *p*
   - Add edges to its vertices (these new edges are correct)
   - Check correctness of the old edges (triangles) "around *p*" and legalize (flip) potentially illegal edges
3. Discard the large triangle and incident edges

Incremental algorithm in detail

DelaunayTriangulation(*P*)
Input: Set *P* of *n* points in the plane
Output: A Delaunay triangulation *T* of *P*

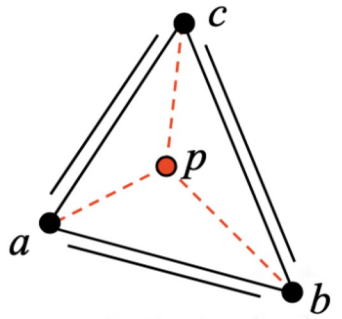


1. Let p_{-2}, p_{-1}, p_0 form a triangle large enough to contain *P*
2. Initialize *T* as the triangulation consisting a single triangle $p_{-2}p_{-1}p_0$
3. Compute random permutation $p_1, p_2, \ldots, p_n$ of $P \setminus \{p_0\}$
4. **for** $r = 1$ **to** n **do**
5. T = Insert(p_r, T)
6. Discard p_{-1}, p_{-2} with all incident edges from *T*
7. **return** *T*

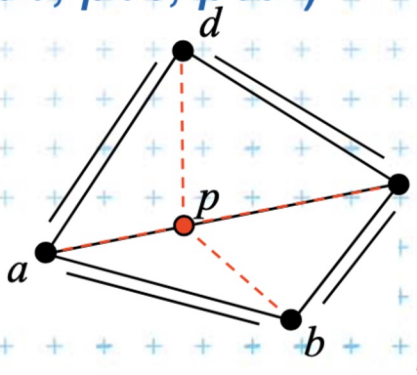
Incremental algorithm – insertion of a point

Insert(*p*, *T*)
Input: Point *p* being inserted into triangulation *T*
Output: Correct Delaunay triangulation after insertion of *p*

1. Find a triangle $abc \in T$ containing *p*
2. **if** *p* lies in the interior of *abc* **then**
3. Insert edges *pa, pb, pc* into triangulation *T*
   (splitting *abc* into 3 triangles *pab, pbc, pca*)
4. LegalizeEdge(*p, ab, T*)
5. LegalizeEdge(*p, bc, T*)
6. LegalizeEdge(*p, ca, T*)
7. **else** // *p* lies on the edge of *abc*, say *ab*, point *d* is right from edge *ab*
8. Remove *ab* and insert edges *pa, pb, pc, pd* into triangulation *T*
   (splitting *abc* and *abd* into 4 triangles *pad, pdb, pbc, pca*)
9. LegalizeEdge(*p, ab, T*)
10. LegalizeEdge(*p, bc, T*)
11. LegalizeEdge(*p, cd, T*)
12. LegalizeEdge(*p, da, T*)
13. **return** *T*



[Berg]



[Berg]

Incremental algorithm – edge legalization

LegalizeEdge(*p, ab, T*)

Input: Edge *ab* being checked after insertion of point *p* to triangulation *T*

Output: Delaunay triangulation of $p \cup T$

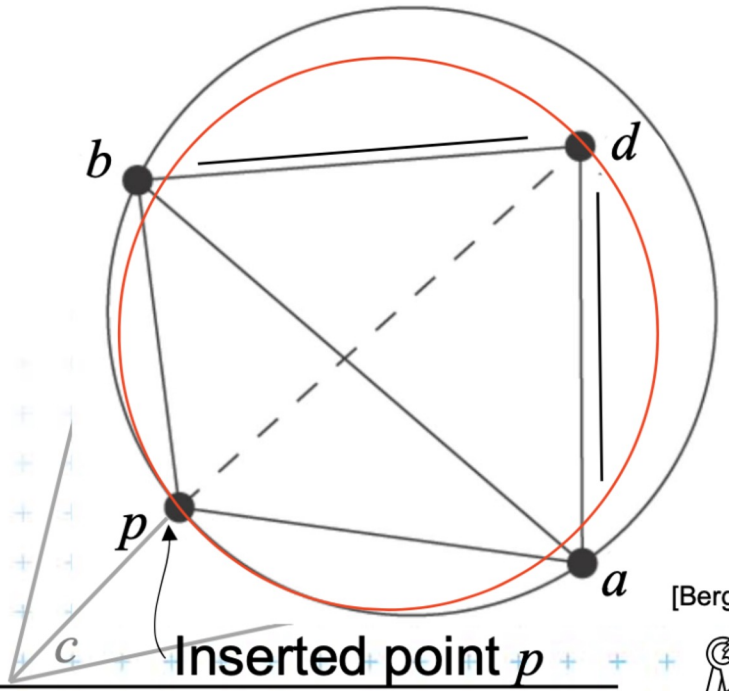
1. **if**(*ab* is edge on the exterior face) **return**
2. let *d* be the vertex to the right of edge *ab*
3. if(inCircle(*p, a, b, d*)) // *d* is in the circle around *pab* => *d* is illegal
4. Flip edge *ab* for *pd*
5. LegalizeEdge(*p, ad, T*)
6. LegalizeEdge(*p, db, T*)

Insertion of *p* may make edges *ab, bc & ca* illegal (circle around *pab* will contain point *d*)

After edge flip, the edge *pd* will be legal (the circumcircles of the resulting triangles *pdb,* and *pad* will bee empty)

We must check and possibly flip edges *ad, db*

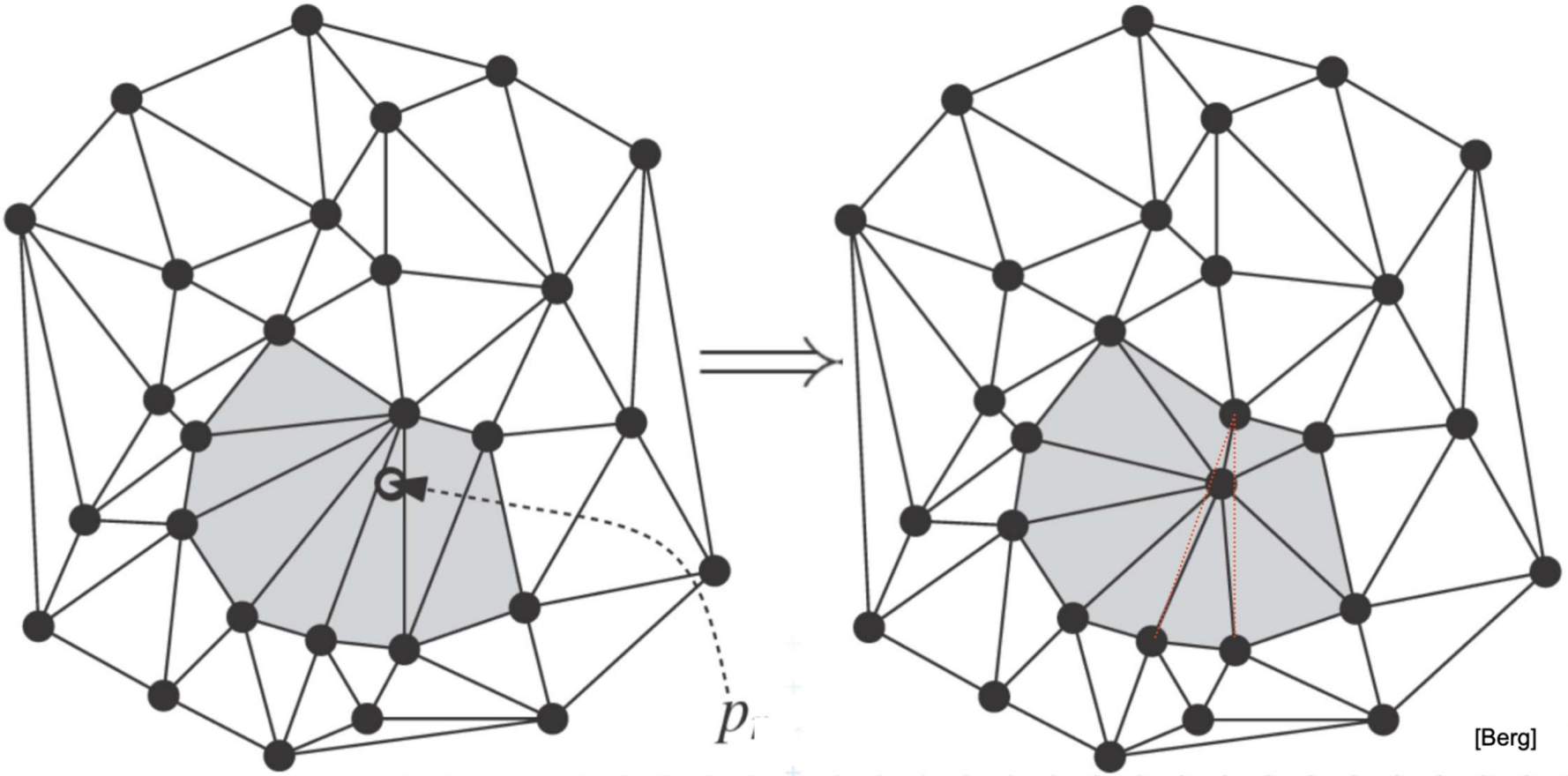
(We must check and possibly flip edges *bc & ca* - lines 5,6 in Insert(p, T))



[Berg]

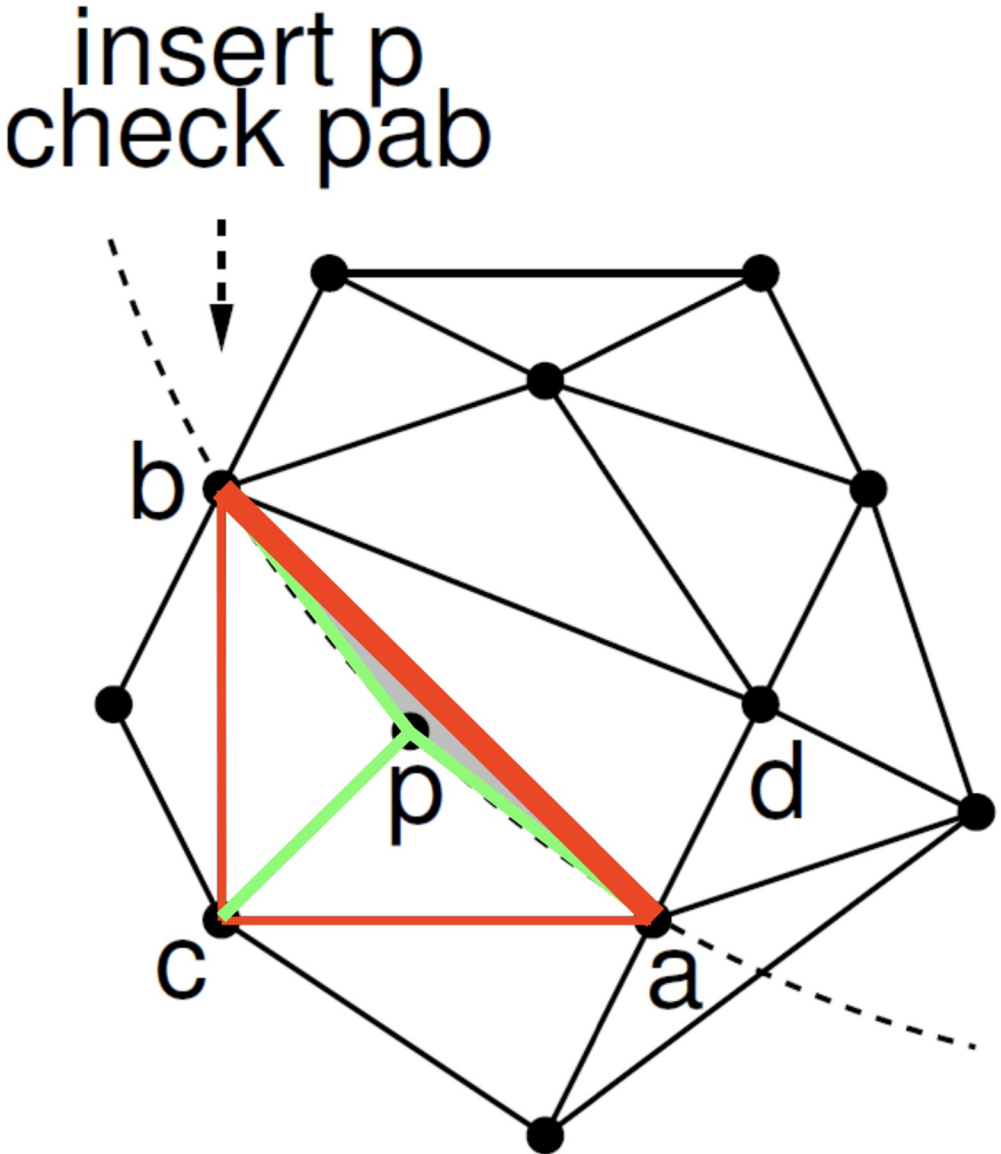
DCGI

DT- point insert and mesh legalization



Every new edge created due to insertion of *p* will be incident to *p*

Delaunay triangulation – other point insert

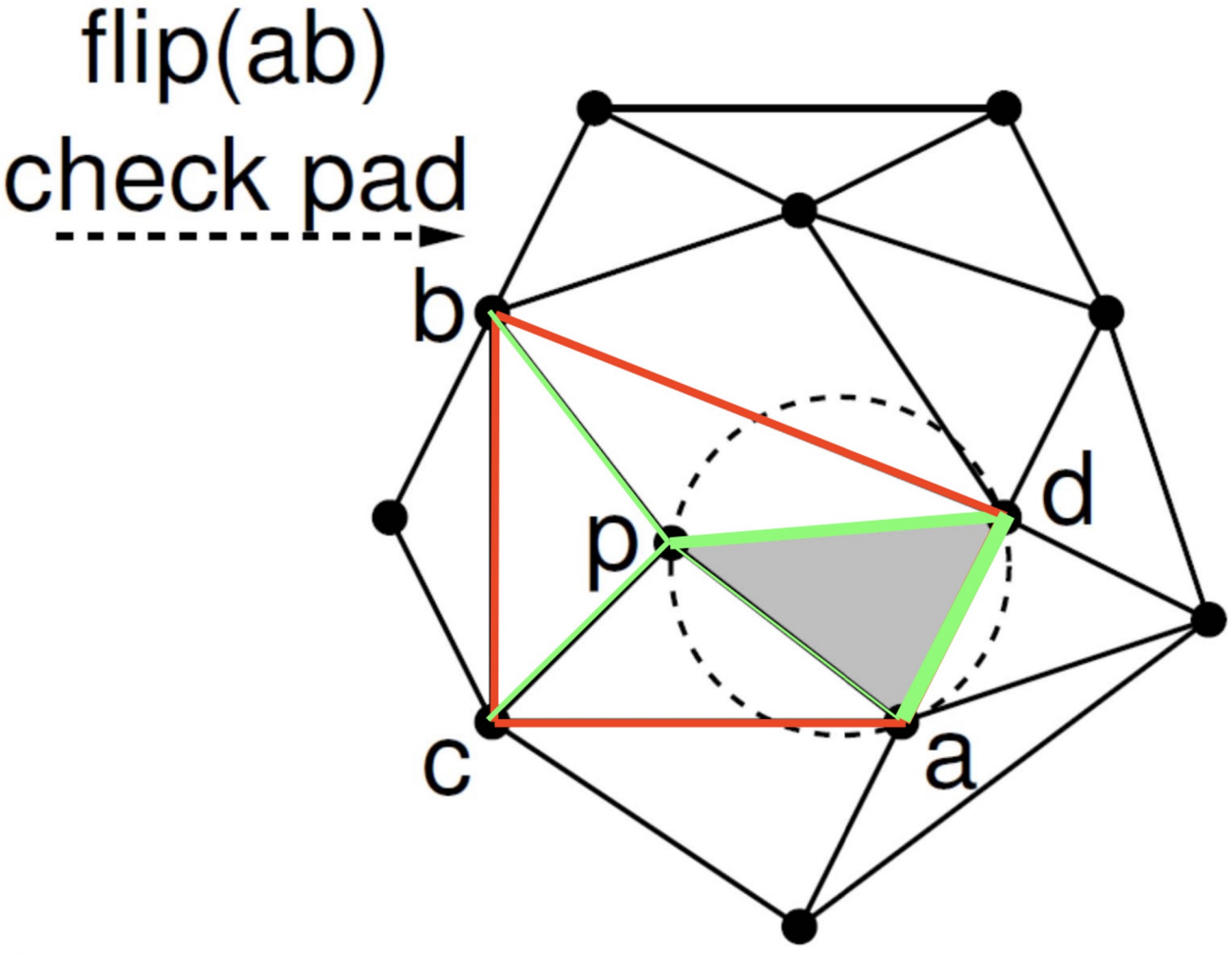


Legalize now

Legalize later

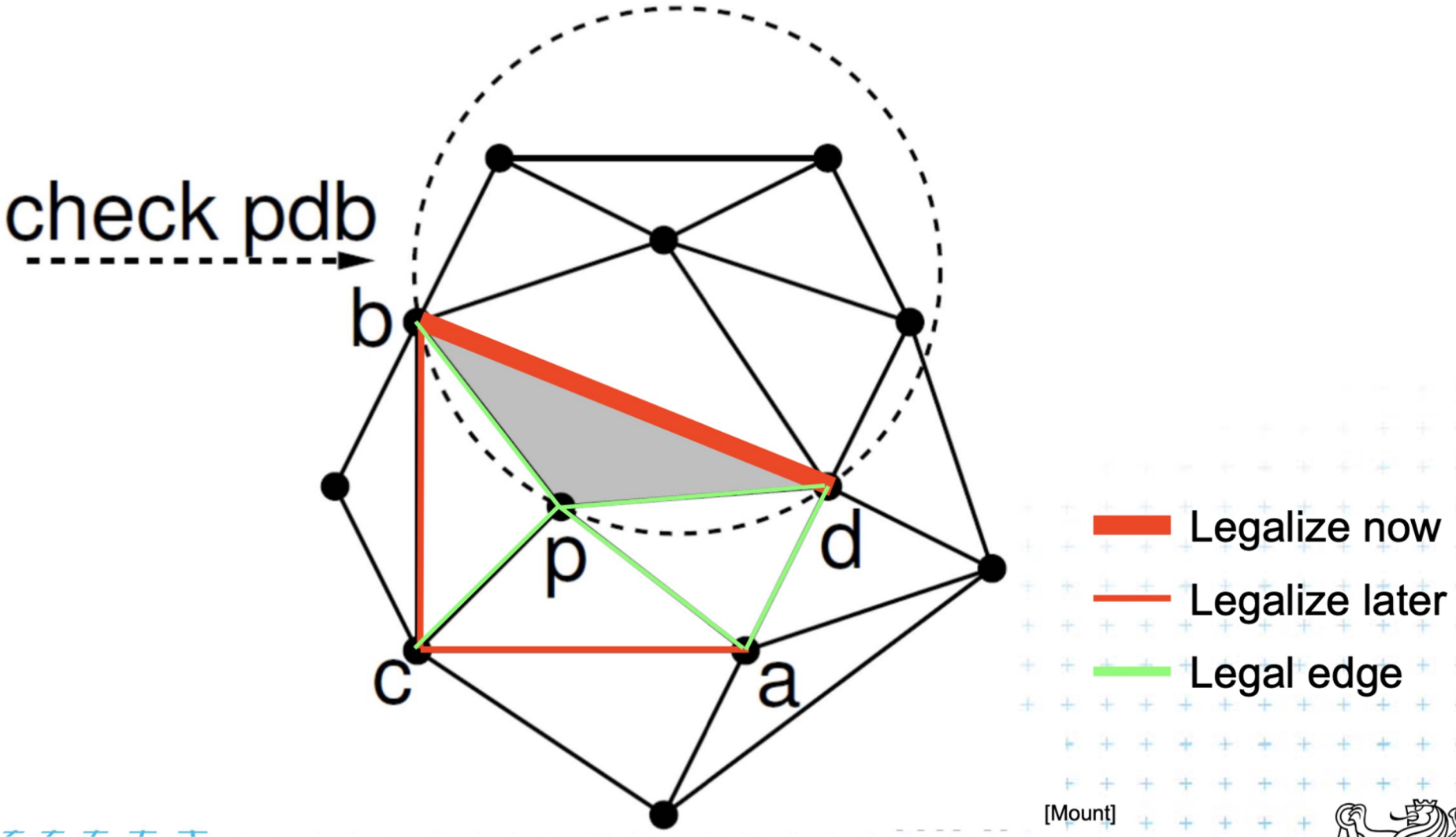
Legal edge

Delaunay triangulation – other point insert



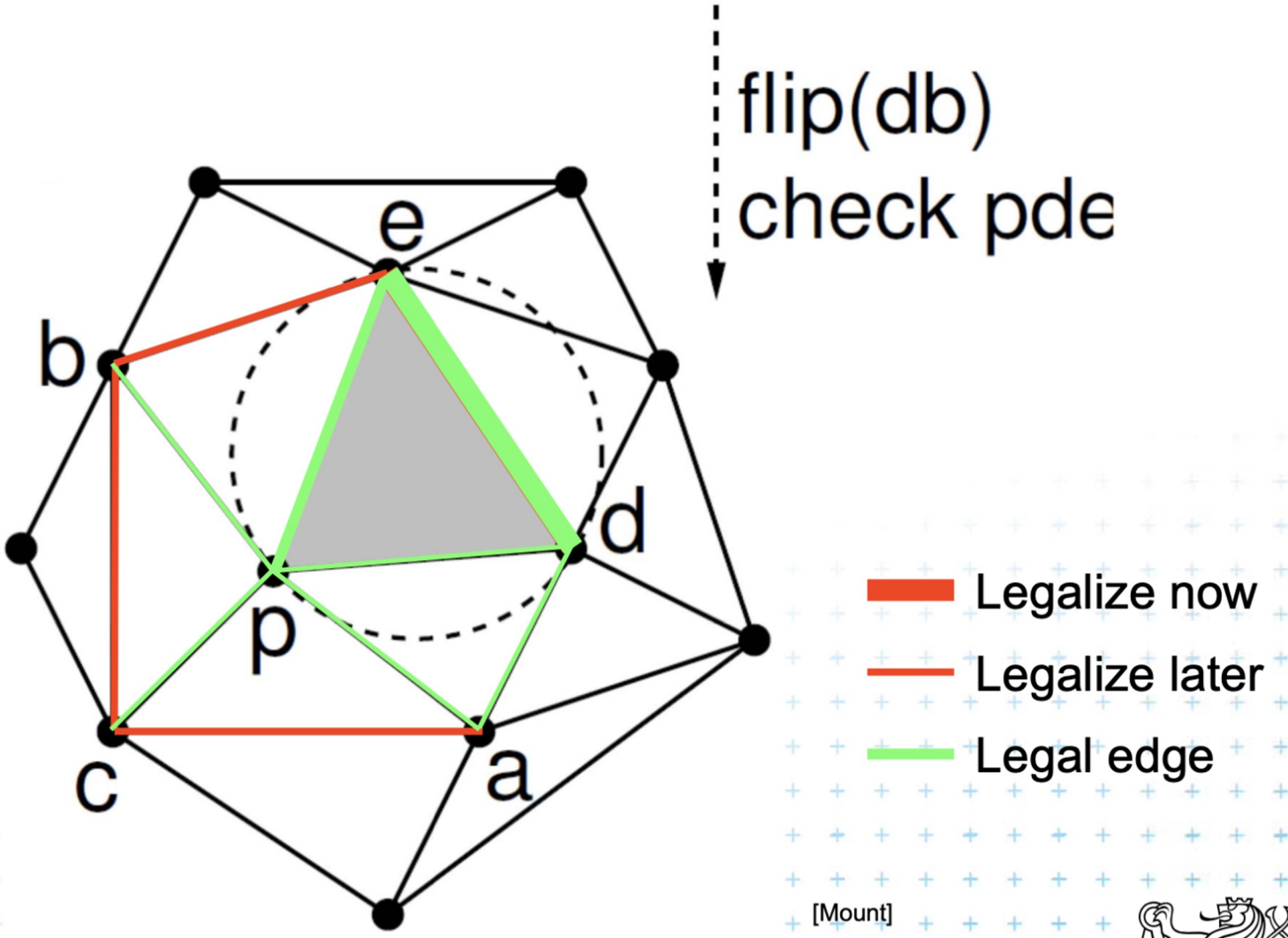
[Mount]

Delaunay triangulation – other point insert



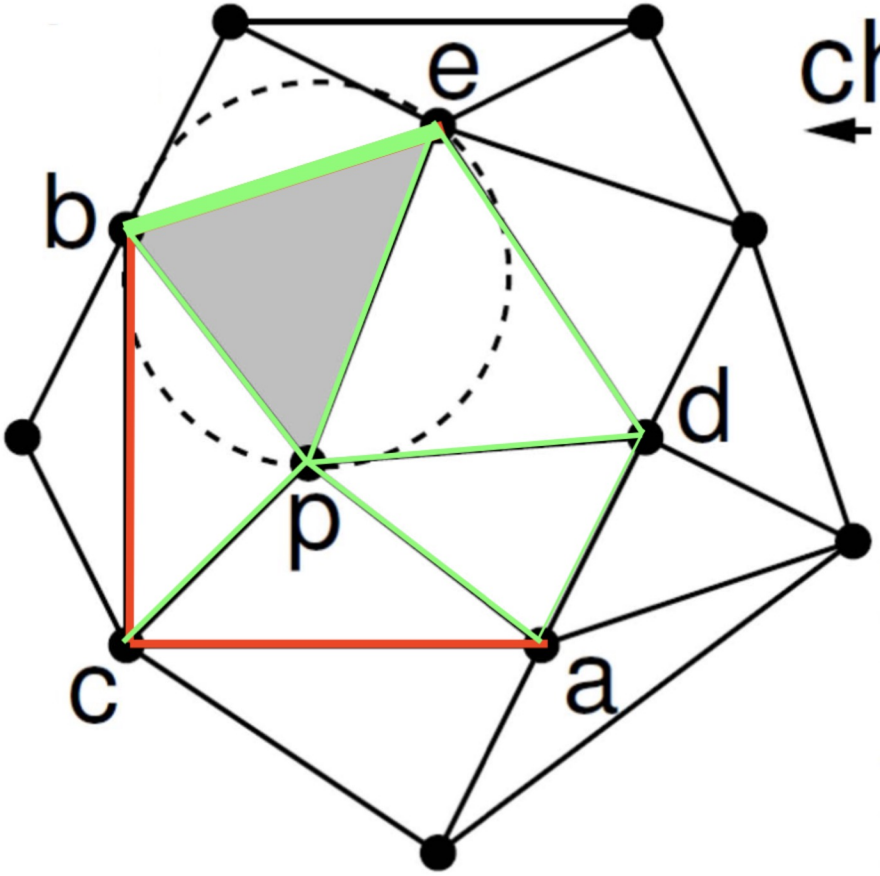
[Mount]

Delaunay triangulation – other point insert



[Mount]

Delaunay triangulation – other point insert



check peb

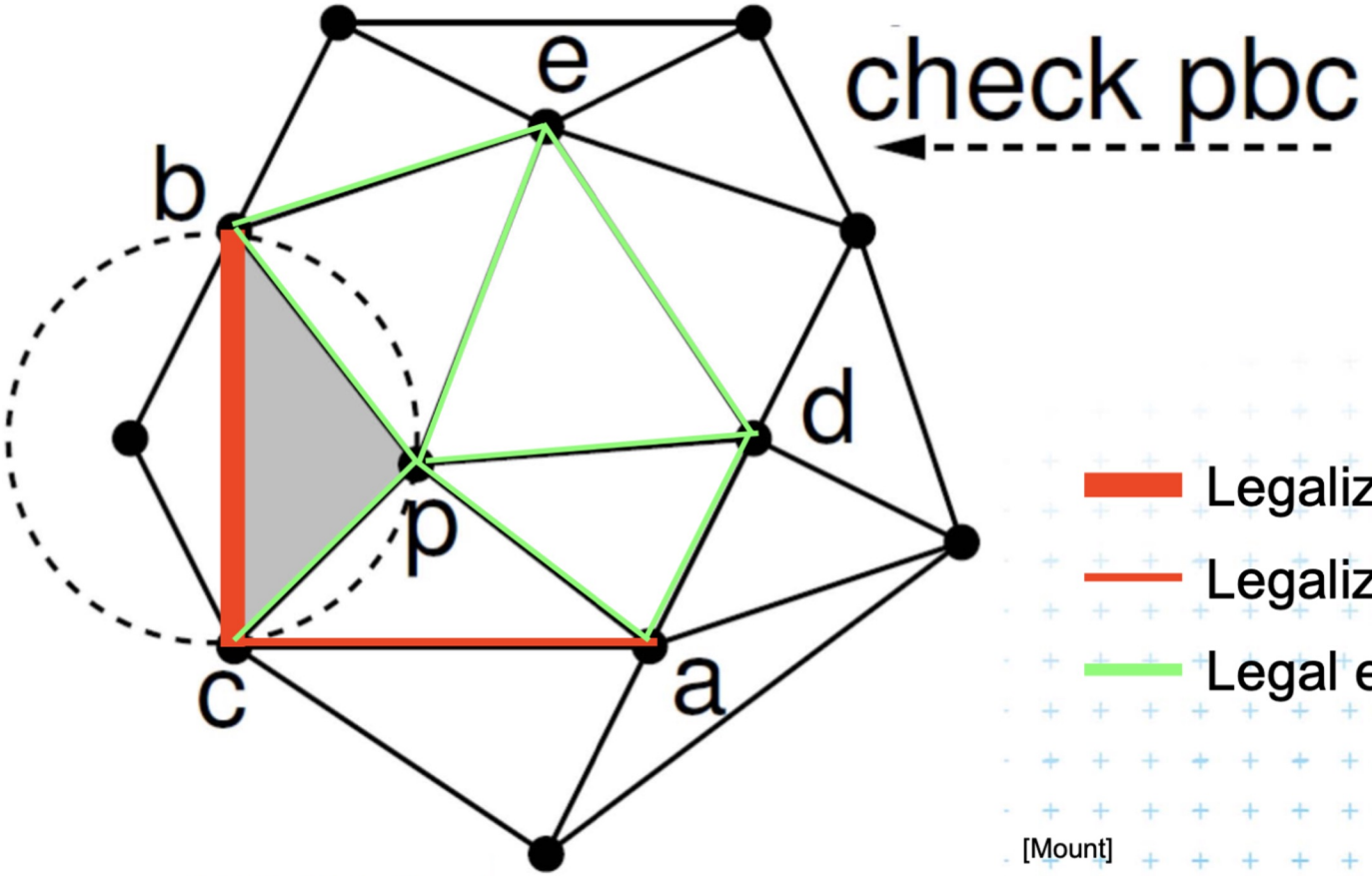
Legalize now

Legalize later

Legal edge

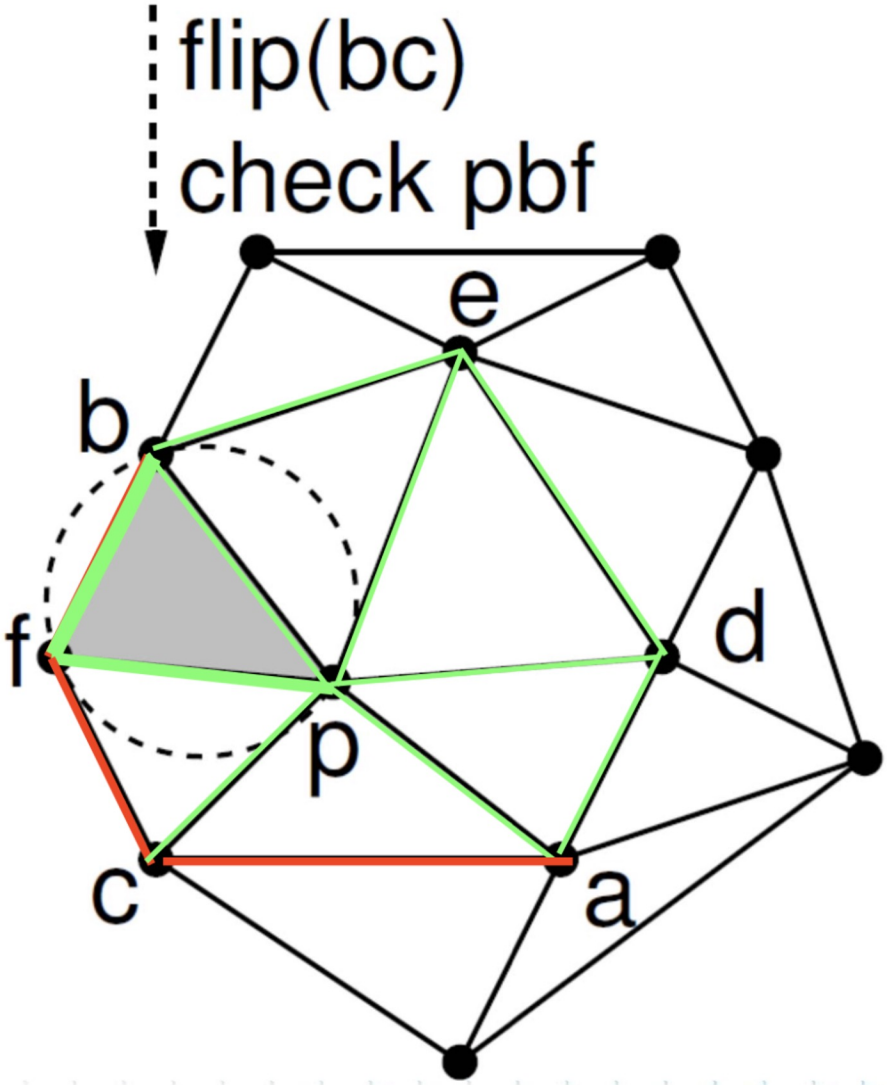
[Mount]

Delaunay triangulation – other point insert



[Mount]

Delaunay triangulation – other point insert



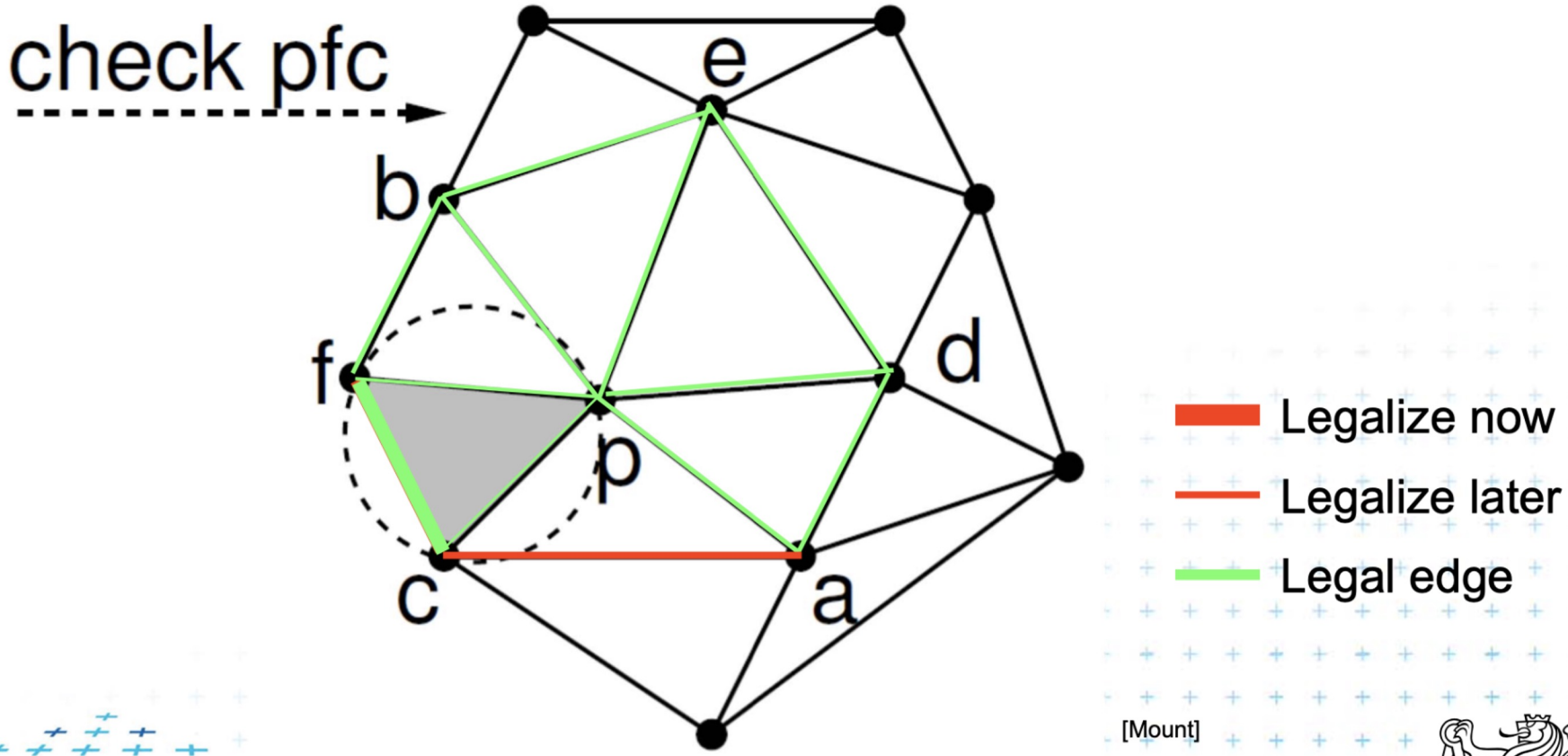
Legalize now

Legalize later

Legal edge

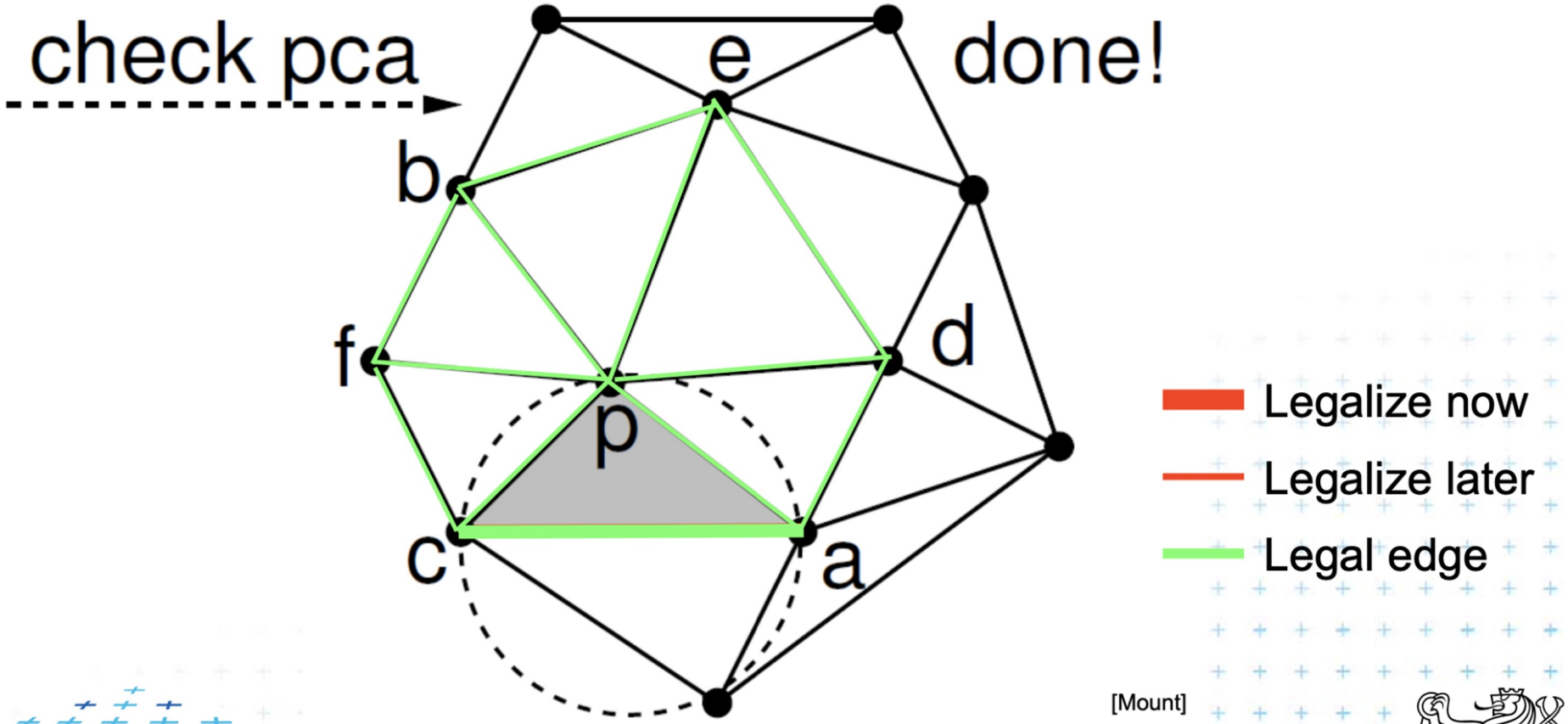
[Mount]

Delaunay triangulation – other point insert



[Mount]

Delaunay triangulation – other point insert



[Mount]

Correctness of the algorithm

- Every new edge (created due to insertion of *p*)
  - is incident to *p*
  - must be legal
    => no need to test them
- Edge can only become illegal if one of its incident triangle changes
  - Algorithm tests any edge that may become illegal

    => the algorithm is correct
- Every edge flip makes the angle-vector larger

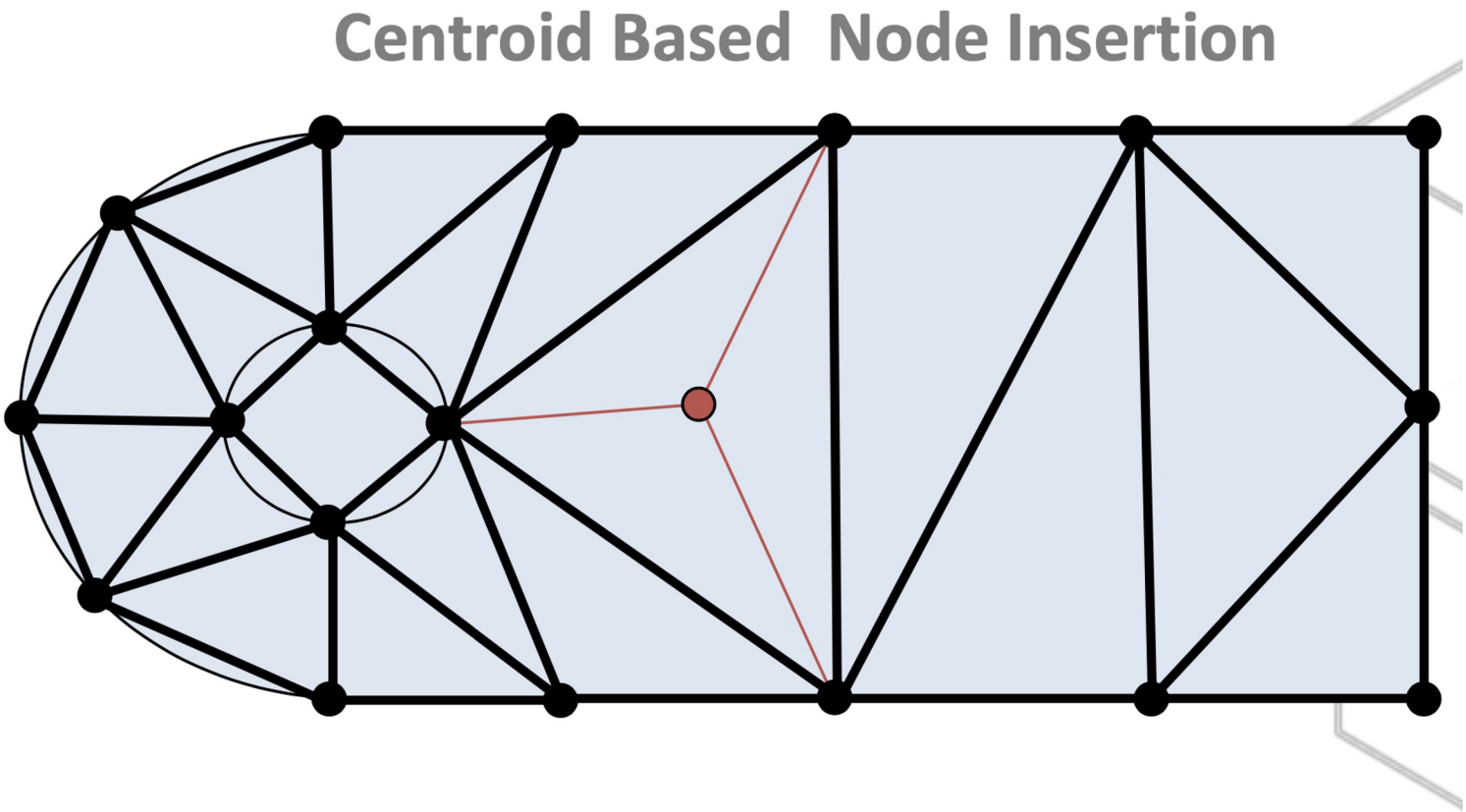
=> algorithm can never get into infinite loop

How insert new points ?

For example, for planar graph we build the initial triangilation. Then:

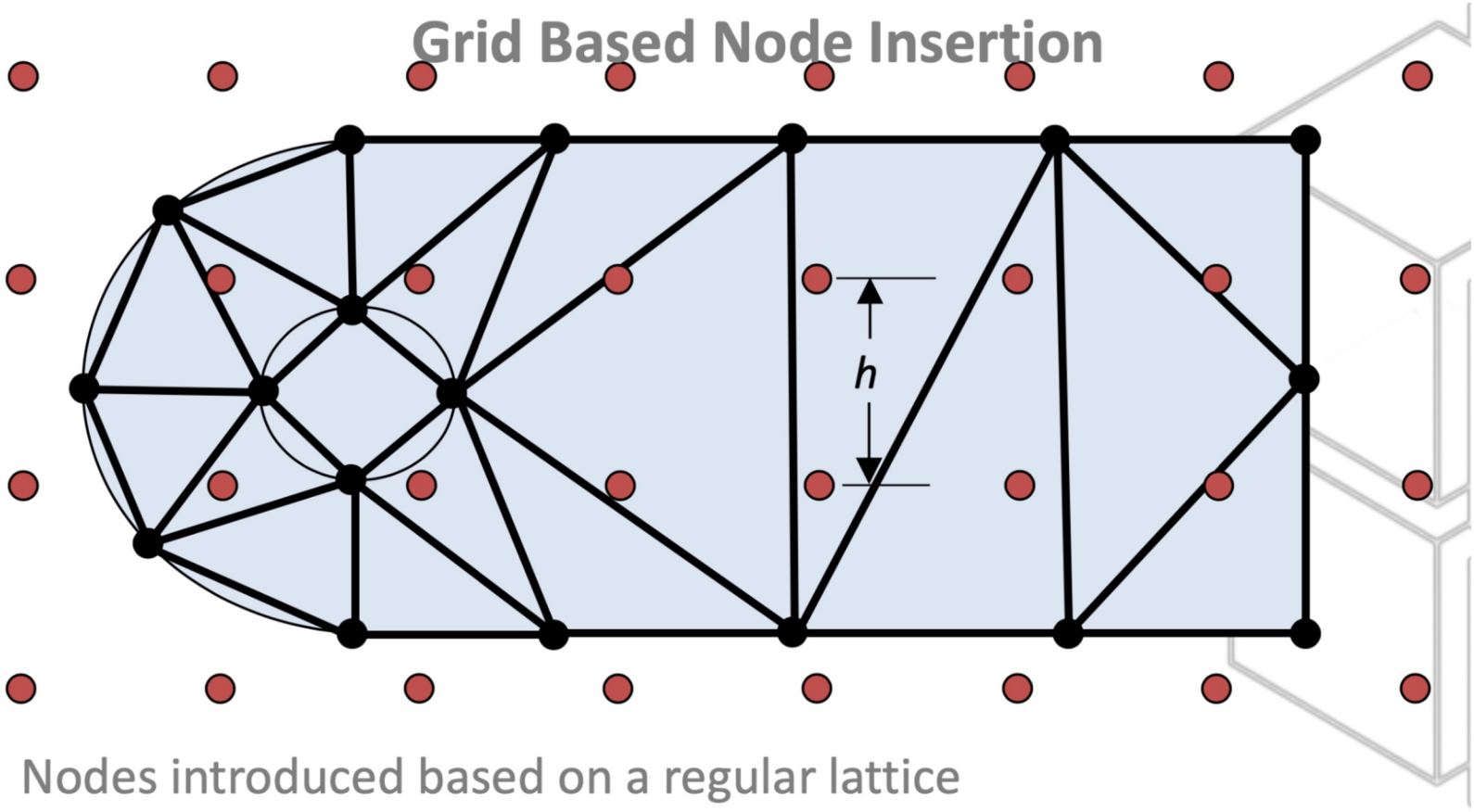
1. Centroid based insertion - insert new point in center of each triangle with area less than **A;**
2. Grid based insertion;
3. Circumcenter (“Guaranteed Quality”) Node Insertion.

Centroid based insertion



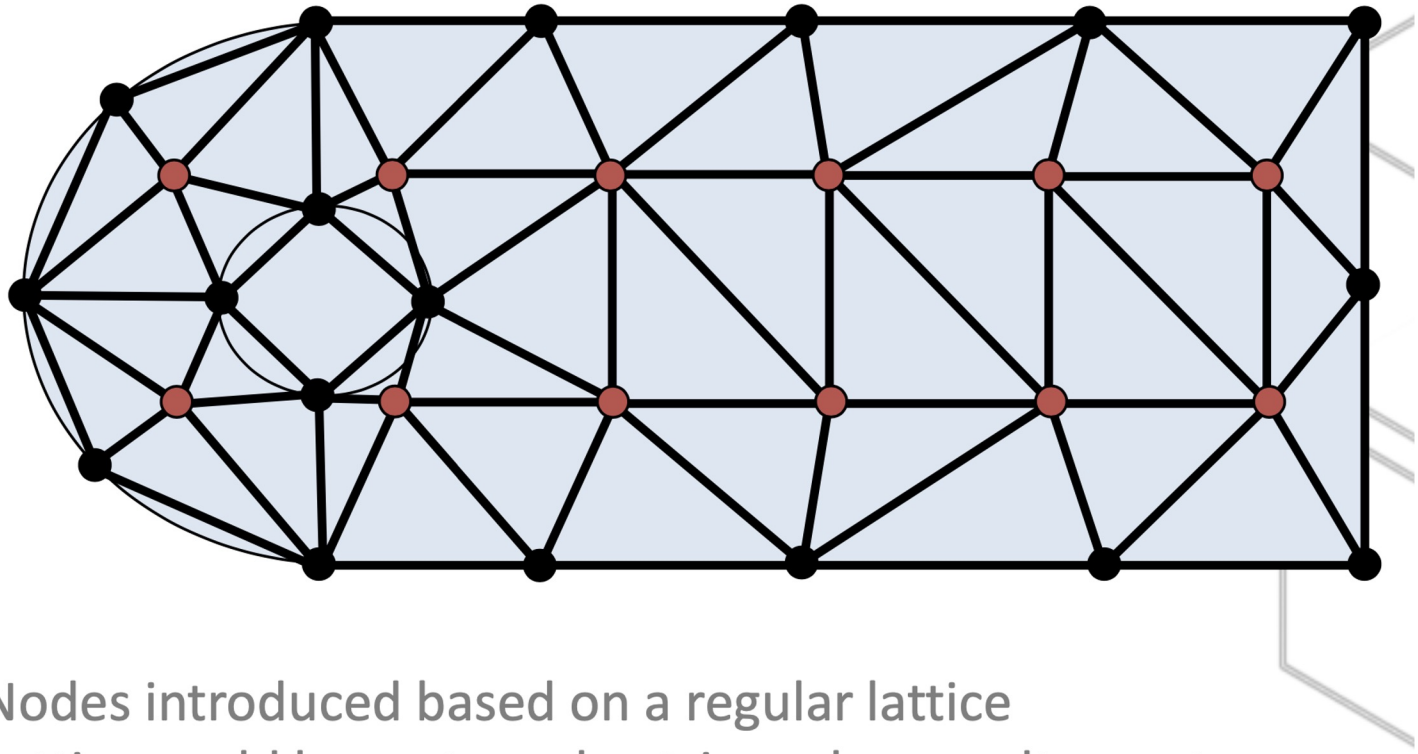
Nodes introduced at triangle centroids
Continues until edge length, $l \approx h$

Grid nodes insertion



Nodes introduced based on a regular lattice
Lattice could be rectangular, triangular, quadtree, etc…
Outside nodes ignored

Grid nodes insertion



Nodes introduced based on a regular lattice

Lattice could be rectangular, triangular, quadtree, etc…

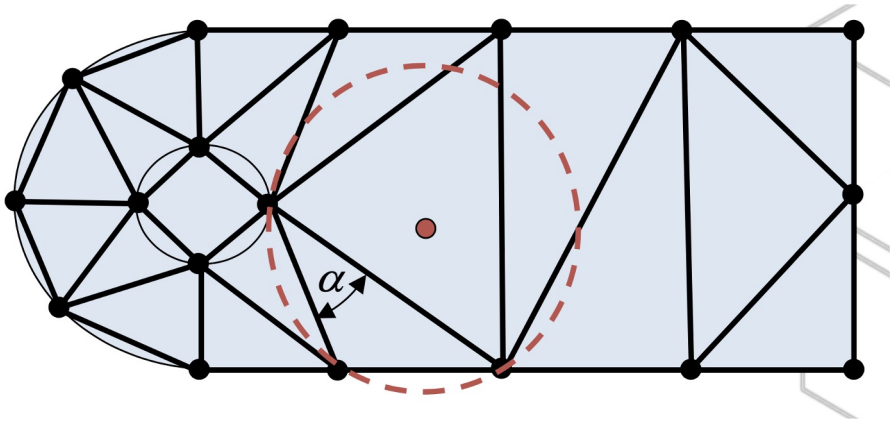
Outside nodes ignored

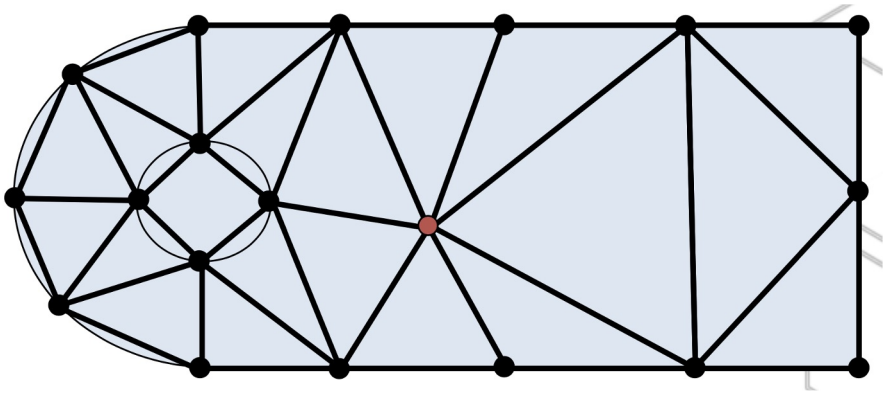
How can we achieve that min(ange) > A?

Circumcenter ("Guaranteed Quality") Node Insertion.

A strategy: Nodes introduced at triangle circumcenters.

Order of insertion based on minimum angle of any triangle.
Continues until minimum angle > predefined minimum.





Popular 2D mesh generators

Nice demo tool for insertion algorithm from **Washington University**:

https://students.engineering.wustl.edu/comp_geo_algorithms/Voronoi_Diagram/VoronoiDiagram.html

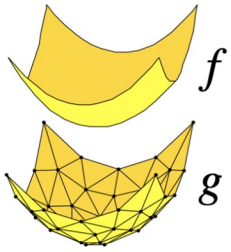
- Jonathan **Richard Shewchuk**'s two-dimensional quality mesh generator : http://www.cs.cmu.edu/~quake/triangle.html
- https://github.com/drufat/triangle
  Python wrapper around Jonathan Richard Shewchuk's "**Triangle**"
- **Delos** (executable, fortran/c++) automatic 2D mesh generator for "almost flat surfaces" - mesh generator by Oliver Stab. https://people.minesparis.psl.eu/olivier.stab/delos/frames_gb.html
- **CGAL C++ library**. CGAL is a software project that provides easy access to efficient and reliable geometric algorithms in the form of a C++ library. https://www.cgal.org/
- **Delaunau2D** (python) - https://github.com/BrunoLevy/geogram/wiki/Delaunay2D
- **Delaunator** (javascript + ports to other languages) https://github.com/mapbox/delaunator

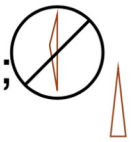
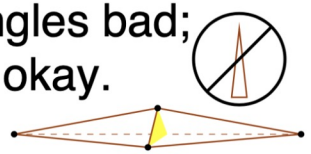
What is good linear finite element ?

1. Richard Shewchuk: https://people.eecs.berkeley.edu/~jrs/papers/elemtalk.pdf

Three Criteria for Linear Elements

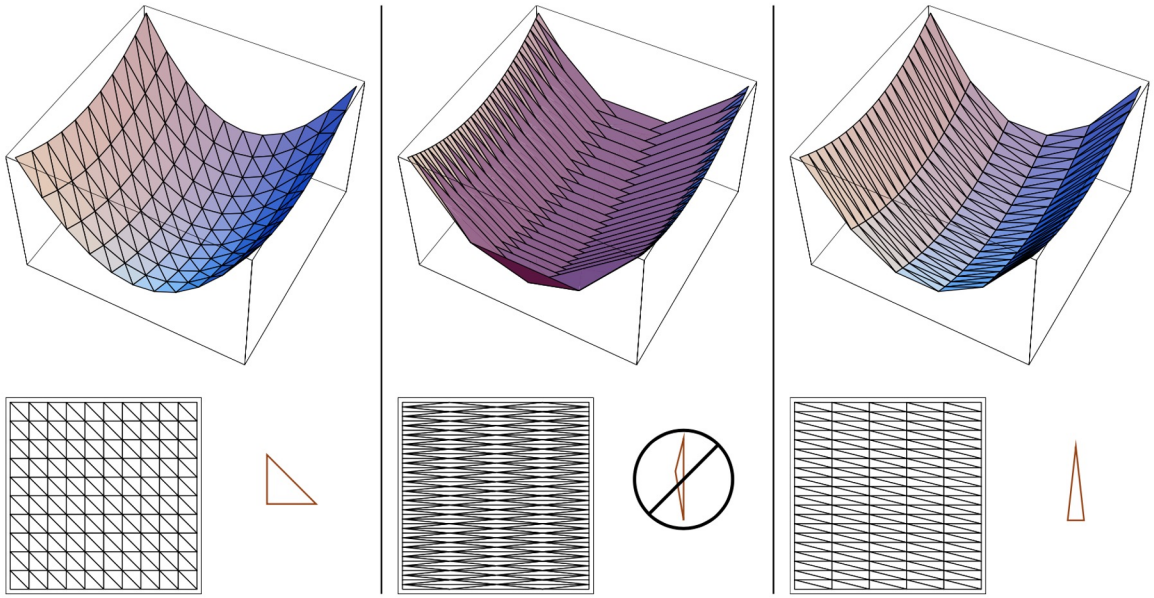
Let f be a function.
Let g be a piecewise linear interpolant of f over some triangulation.



Criterion	
Interpolation error $\|f-g\|_\infty$	Size very important. Shape only marginally important.
Gradient interpolation error $\|\nabla f-\nabla g\|_\infty$	Size important. Large angles bad; small okay. 
Element stiffness matrix maximum eigenvalue $\lambda_{\max}$	Small angles bad; large okay. 

What is good linear finite element ?

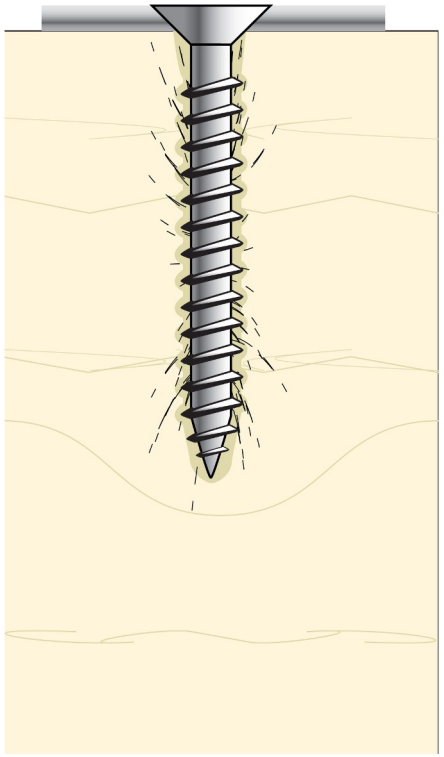
1. Richard Shewchuk: https://people.eecs.berkeley.edu/~jrs/papers/elemtalk.pdf
   The Importance of Approximating Gradients Accurately



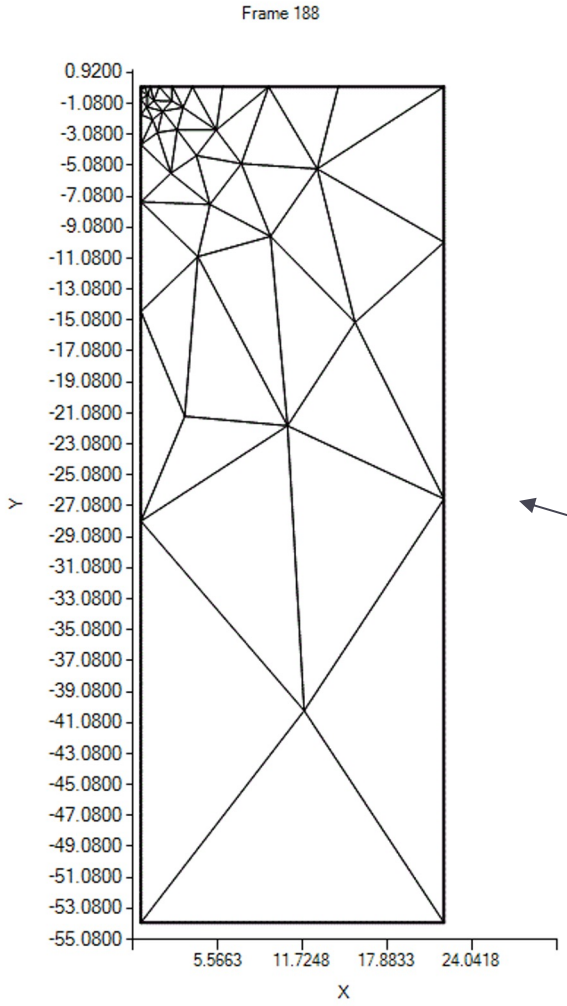
- $\| \nabla f - \nabla g \|_\infty$ affects discretization error in FEM.
- In mechanics, ∇f is the strains.

Вкручення шурупа Wuerth

Задача: оптимізація геометрії шурупа і дослідження обертового моменту



Mesh over the frames. Example



Full-length screw penetration

Step file:
170 04 0004-SK-DG-VG_schnitt1_4.STEP

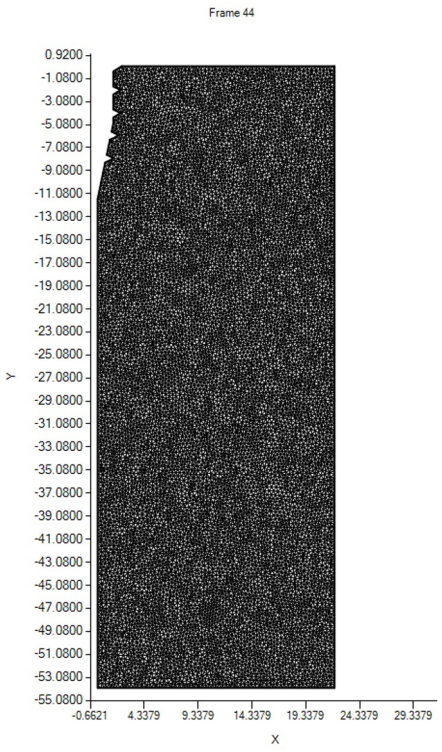
Mesh parameters:
Element size = 0.3
Growth factor = 1.5

Note the graded non-uniform mesh, automatically adjusted

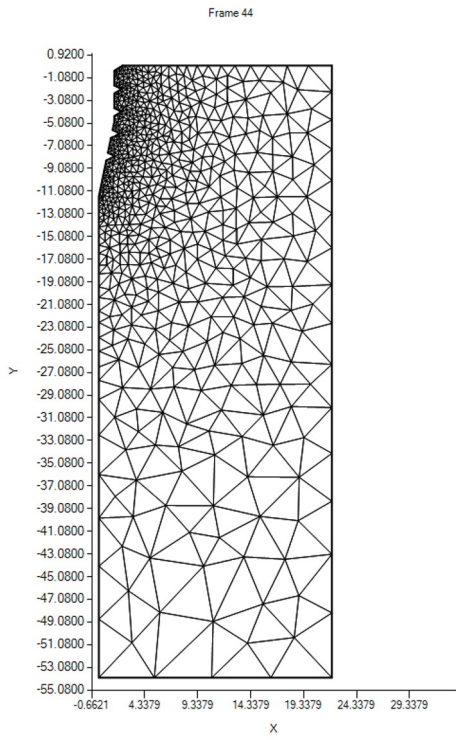
Influence of the mesh parameters

Mesh size = 0.3

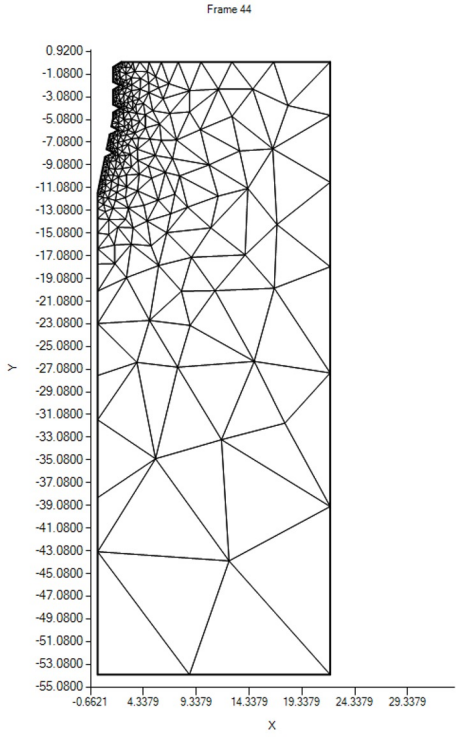
Growth Factor = 1



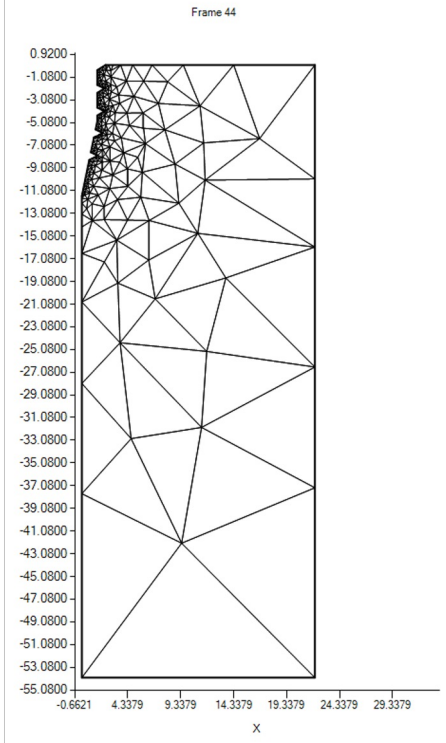
Growth Factor = 1.1



Growth Factor = 1.3

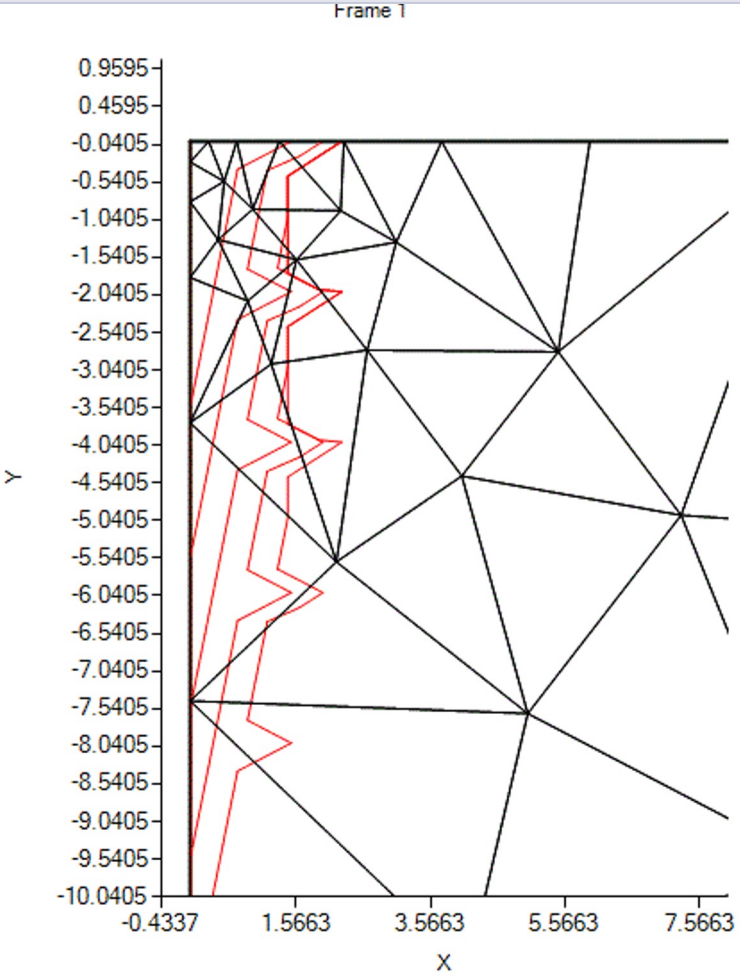


Growth Factor = 1.5



Automatic limitation on the element size

Maximum X-displacement between two consequent frames should be less then element size multiplied by defined coeficcient



Start slide show *(Shift+F5)* to see the animation
Red lines = key frames

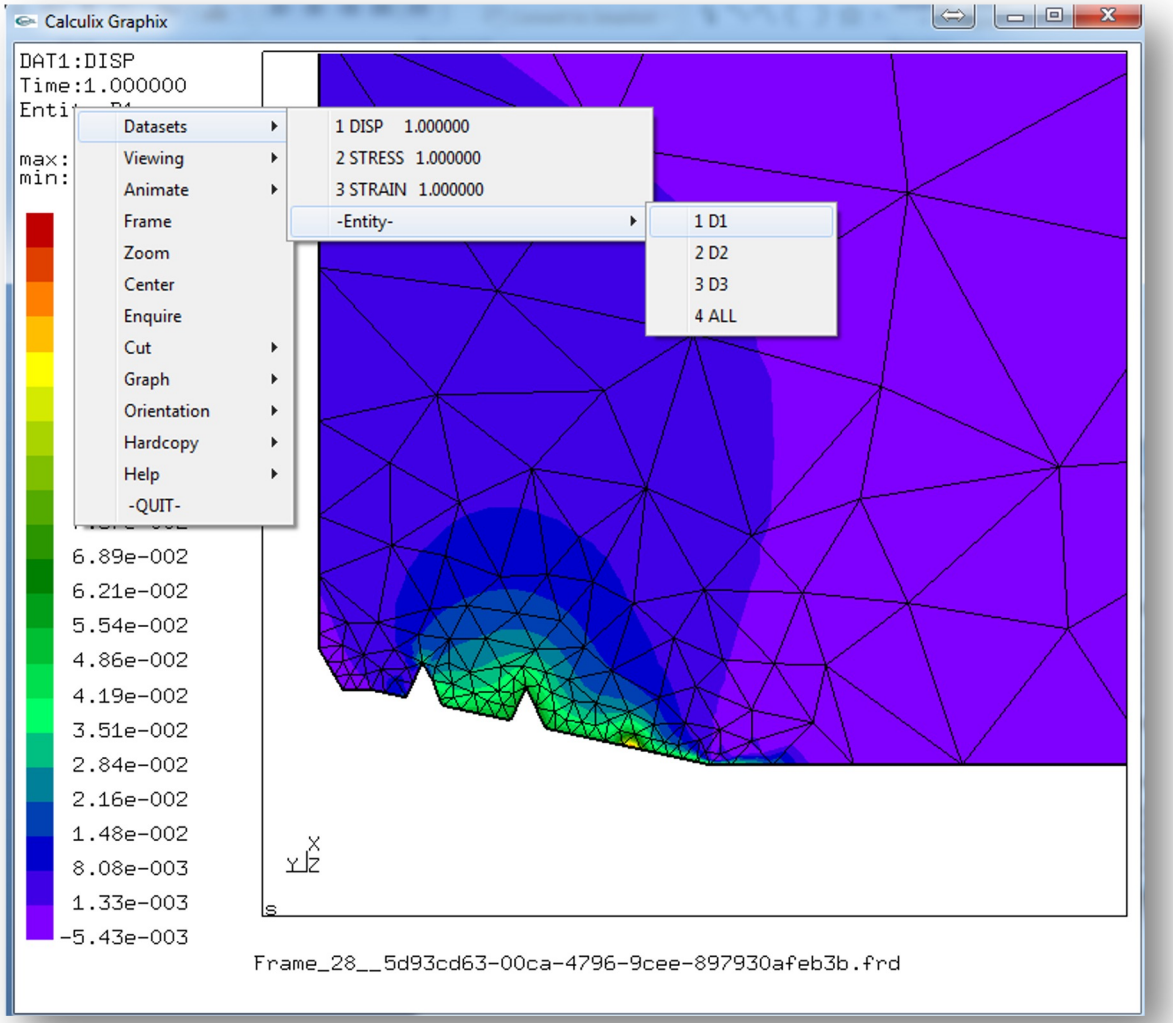
Post-processing tools

1. Left mouse click on the *panel area* to activate the menu

2. Select the *dataset* to show from the *Datasets* menu:
select DISP, select D1

3. Select *Viewing -> Toggle element edges* to show the mesh

4. On the charting area use:
Left mouse button to *Rotate*;
Middle mouse button to *Zoom in/out*;
Right mouse button to *Pan*.



Useful links

1. Books of Joseph O'Rourke: https://www.science.smith.edu/~jorourke/
2. https://people.eecs.berkeley.edu/~jrs/jrspapers.html
3. Computational Geometry Algorithms and Applications, Mark Berg and others.
4. Petr Felkel Triangulations (lecture notes, some slides used in this presentation), 2017
5. Siu-Wing Cheng, Tamal Krishna Dey, and Jonathan Richard Shewchuk, *Delaunay Mesh Generation*, xii+375 pages. CRC Press, Boca Raton, Florida, December 2012
6. https://people.eecs.berkeley.edu/~jrs/papers/elemtalk.pdf