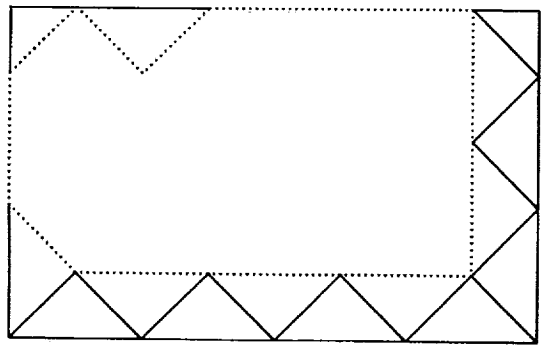
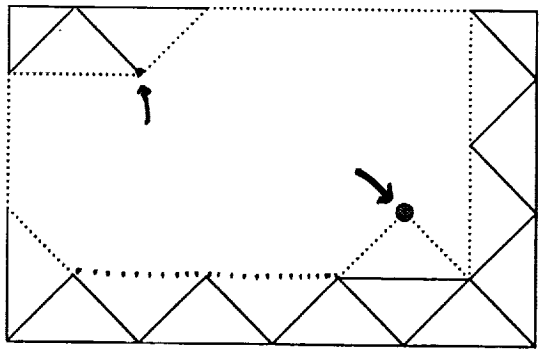
AN ADVANCING-FRONT DELAUNAY-TRIANGULATION ALGORITHM DESIGNED FOR ROBUSTNESS

D. J. MAVRIPLIS
I.C.A.S.E.- NASA LANGLEY RESEARCH CENTER

UNSTRUCTURED MESH GENERATION

- Advancing Front Method
- Delaunay Triangulation Techniques
- Combinations of Both
  - Merriam
  - Rebay, Muller and Roe
- Others (Computational Geometry)
  - Edulsbrunner, Bern, Eppstein



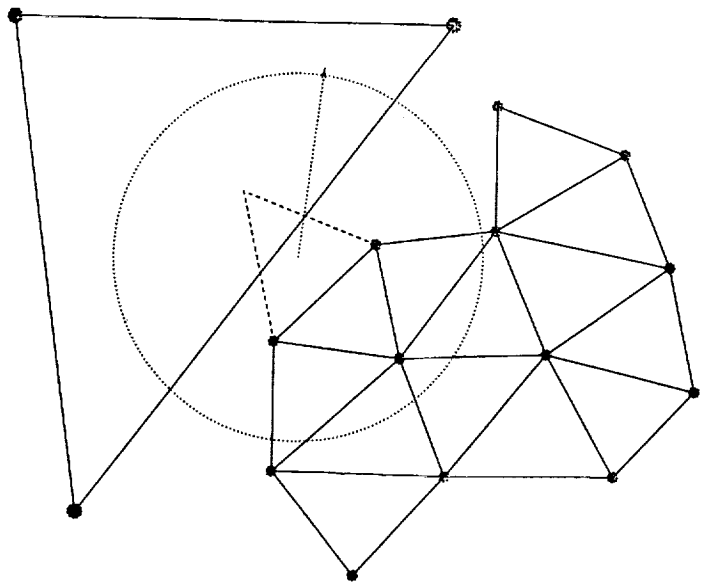


ADVANCING FRONT

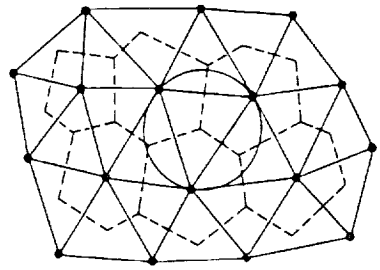
- Always Pick Smallest Front Edge
  - Front edges form heap-list
  - Dynamic data structure (insert-delete)
- Join Edge to New Point or Existing Front Point
  - Intersection checking
- Requires Location of "Close" Front Points
  - Quadtree Data Structures
  - Dynamic (insert-delete)

FAILURE OF ADVANCING FRONT

- Merging Two Fronts of Dissimilar Length Scales
- Usually Result of Rapid Variation in Field Function f(x,y)

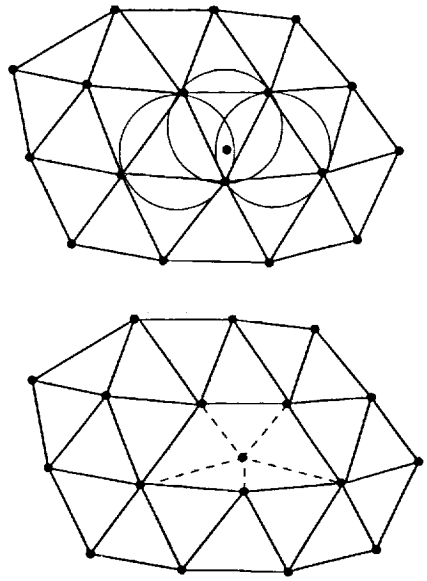


DELAUNAY TRIANGULATION



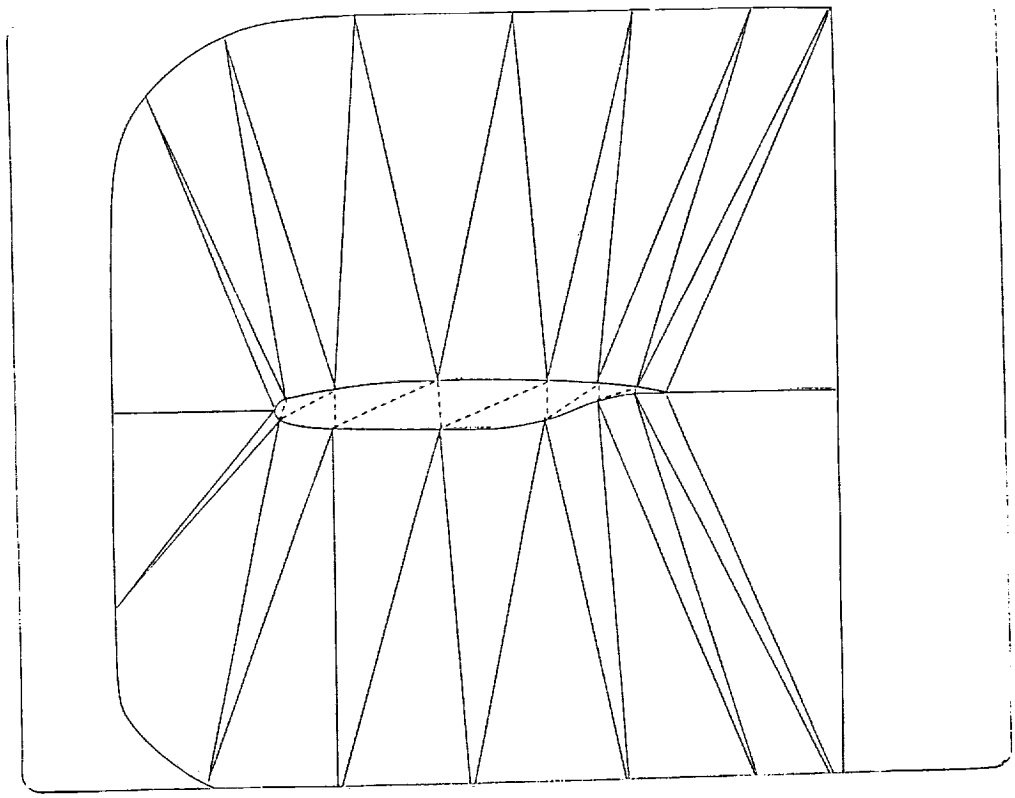
- Decouples Grid Points from Triangulation Procedure
- Produces Most Equiangular Triangles
- Purely Local Construction

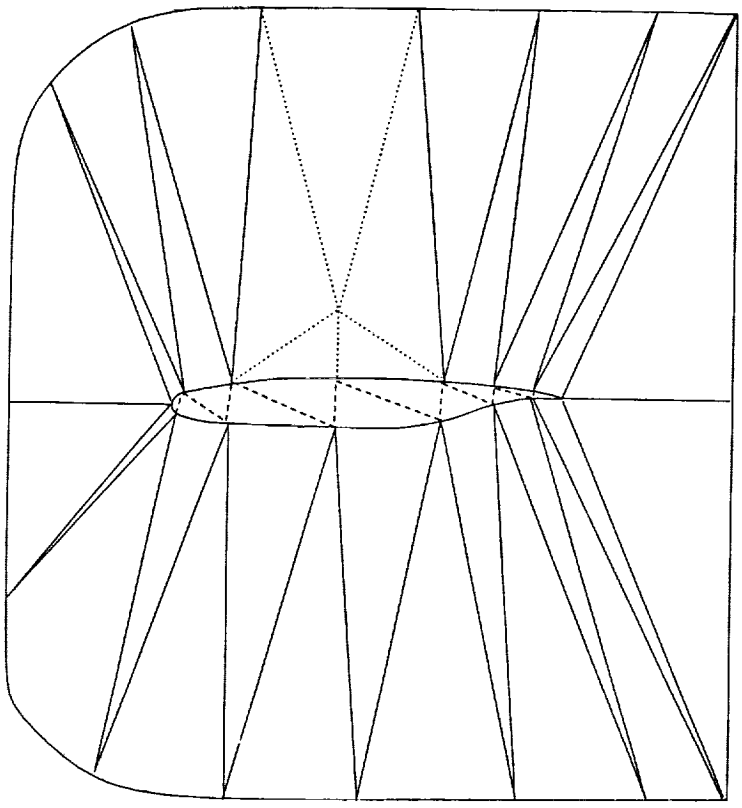
BOWYER'S ALGORITHM FOR DELAUNAY TRIANGULATION

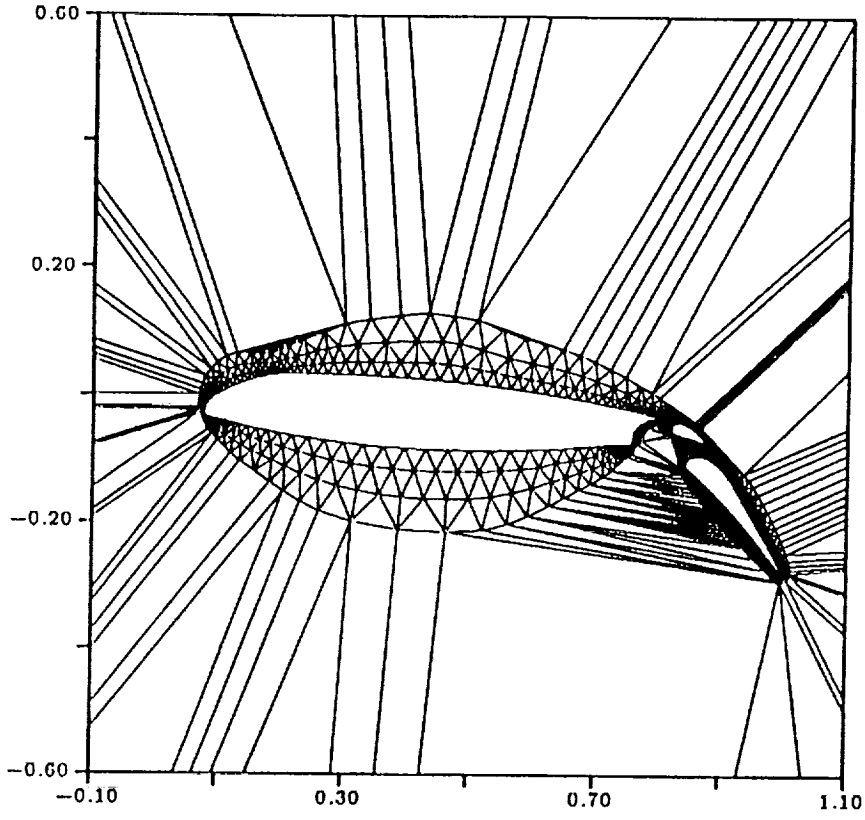


DELAUNAY TRIANGULATIONS

- Fundamental Data Structure in Computational Geometry
- Essentially a Reconnection Strategy
- Rigorous CG Construct
- Must be Modified for Non-Convex Domains
- Heuristic Point Placement Strategies
- Very Simple and Efficient Algorithms

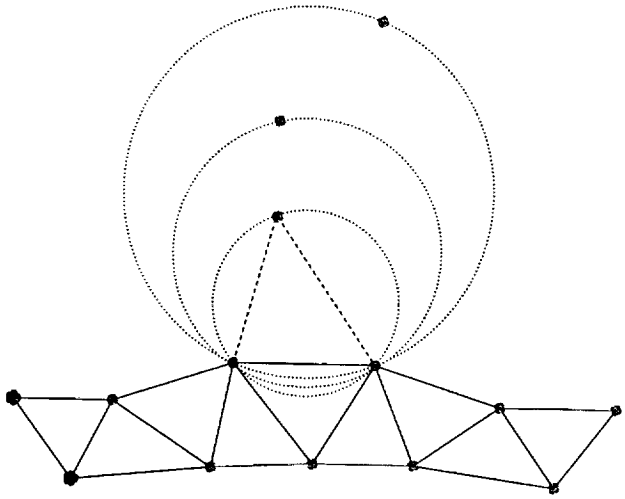





0.60
0.20
−0.20
−0.60
−0.10
0.30
0.70
1.10

TANAMURA-MERRIAM ALGORITHM

- Delaunay Triangulation of a Given Set of Points
- Advancing Front Generating Each Triangle Sequentially
- Never Modify Already Generated Triangles



Y A G G

Yet Another Grid Generator

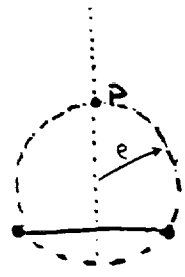
- 2-D Non-Stretched Grid Generation Fairly Easy
- Existing Methods Still Unsatisfactory
  - Advancing Front
    - Efficiency
    - Robustness
      (Counter Examples for Merging Fronts)
  - Delaunay Triangulation
    - Boundary Integrity
    - Round-off Error Failures
- Objectives:
  - High Quality Mesh
  - Efficient Strategy
  - Theoretically Guaranteed Robustness
  - Extendible to 3-D and Stretched Meshes

ADVANCING-FRONT DELAUNAY-TRIANGULATION

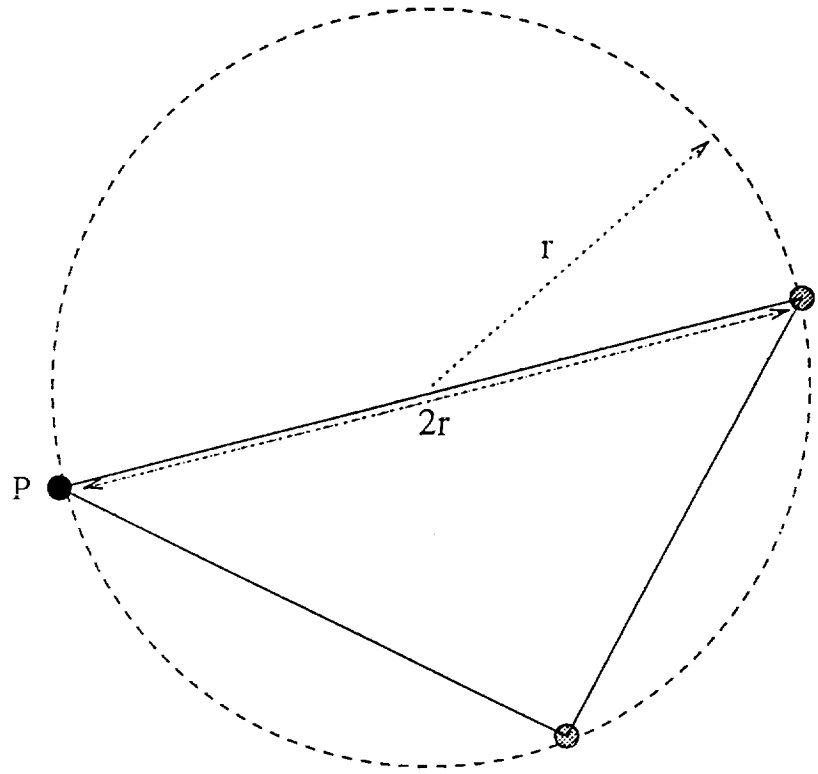
- Advancing Front Point Placement
- Delaunay Triangulation Reconnection
- Combines Advantages of Both Methods
  - — Boundary Integrity Guaranteed
  - — Rigorous CG Construction (Constrained Delaunay Triangulation)
  - — Local Operations Only

ADVANCING-FRONT DELAUNAY-TRIANGULATION

- Define Field Function for Circumcircles $\rho = f(x,y)$
- Choose Front Edge (Heap List)
- Place New Point (determined by $\rho = f(x,y)$)

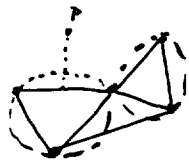


- Construct All Triangles with New Point such that $\rho_{new} < \rho$
  - — Join New Point to All Point Pairs of Grid and Retain only Valid Triangles
  - — Only Test Subset of Grid Points Less than 2ρ away from New Point

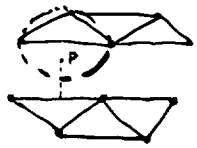


3 POSSIBILITIES FOR NEW POINT

- New Point Does Not Lie in Any Existing Circumcircles
  - — All Existing Triangles Remain Valid
  - — New Triangles Formed with Front Points Only



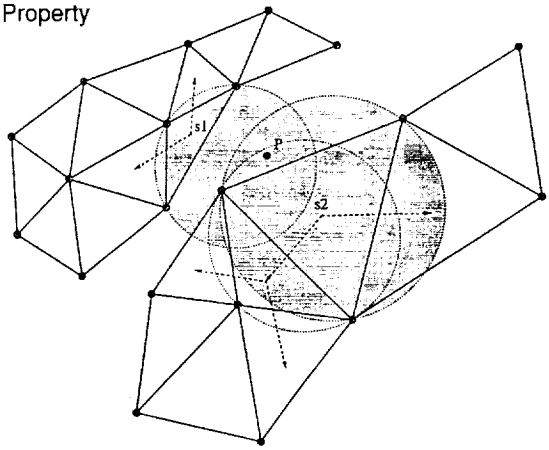
- New Point Lies in Existing Circumcircle(s)
  - — These Triangles Must Be Deleted Before Generation of New Elements
  - — Requires Search for Intersected Circumcircles



- New Point Not Needed
  - — Valid Triangle by Joining Current Edge to a Front Point
  - — Due to Variation in $\rho = f(x,y)$
  - — Determined by Tanamura-Merriam Algorithm

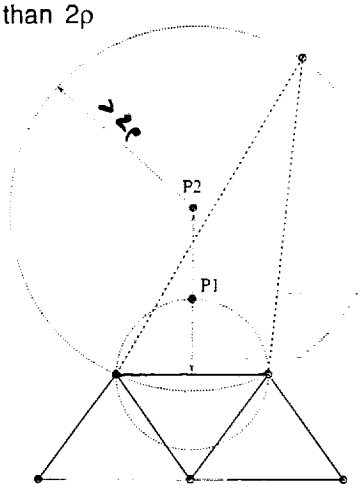
INTERSECTED CIRCUMCIRCLE SEARCH

- Grid Not Fully Connected (Neighbor Search Not Valid)
- Search All Front Triangles for Intersections
- Search Through Neighbors from Each Intersected Front Triangle
- Correctness Guaranteed by Delaunay Visibility Property



NEW POINT PLACEMENT

- Positioned Along Median to Yield Triangle of Radius $\rho = f(x,y)$
- Lower Limit P1 (Smallest Circumcircle)
- Upper Limit P2 (Equidistant from Other Points)
  - Only Relevent if There Exists a Point Closer than 4ρ which yields a Delaunay Triangle Smaller than 2ρ



AF-DT ALGORITHM

1.) Construct Original Front (Boundary Edges)

2.) Choose Edge of Front (Heap List)

3.) Determine Max Circumradius as $\rho = f(x,y)$

4.) Locate All Front Points Less Than 4ρ from Edge

5.) Use TM Algorithm to Determine The Triangle Formed Between Edge and "Close" Points

- If Triangle Exists and is Acceptable Go To 9
- If Triangle is Too Large: Create New Point, Limit Position by Center
- If Triangle Does Not Exist: Create New Point

AF-DT ALGORITHM

6.) Determine All Front Triangle Circles Intersected by New Point

7.) Determine All Interior Intersected Triangles (Neighbor Search)

8.) Remove All Intersected Triangles and Update Front

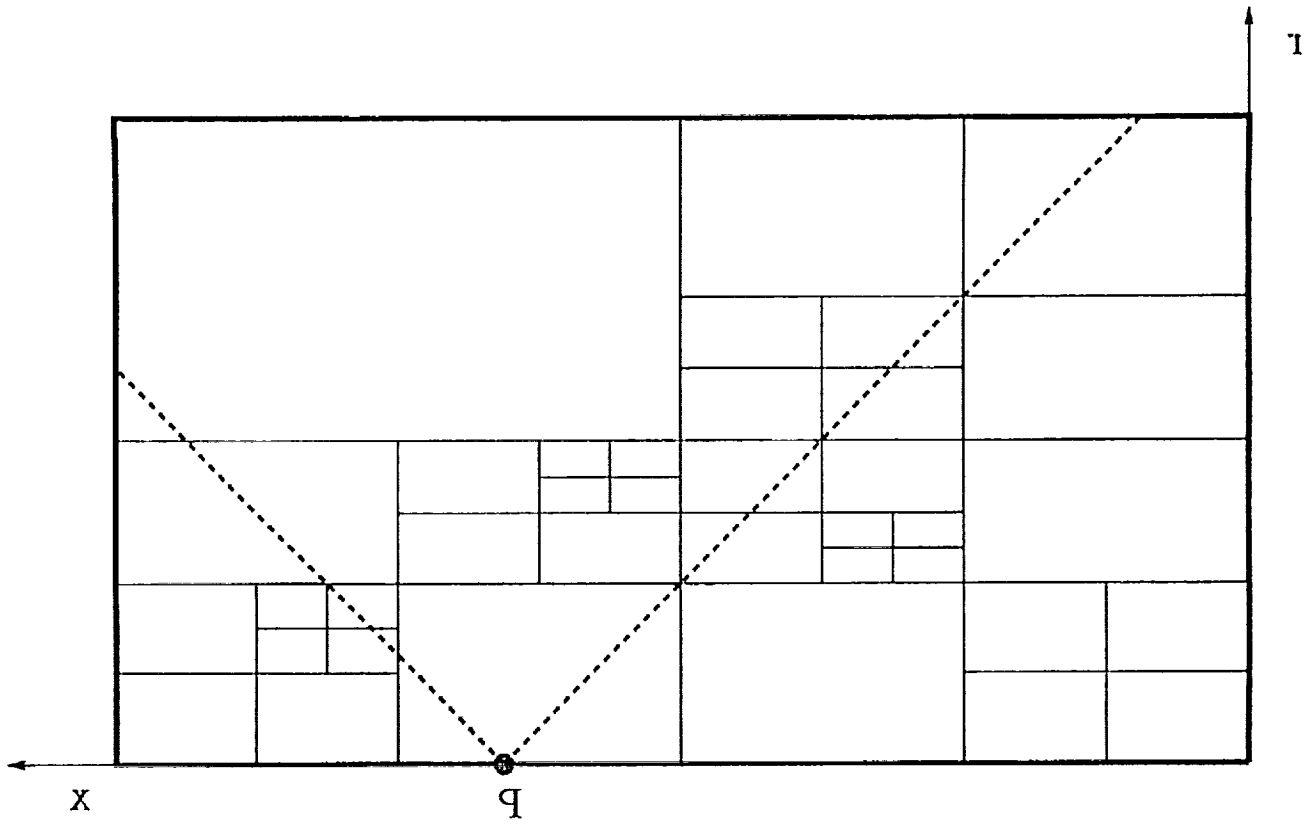
9.) Form All Acceptable Triangles With New Point and "Close" Points (which do not intersect boundary edges)

10.) Add Triangles to Mesh, Update Front

11.) If Front Queue Empty: Stop
Else: Go to 2

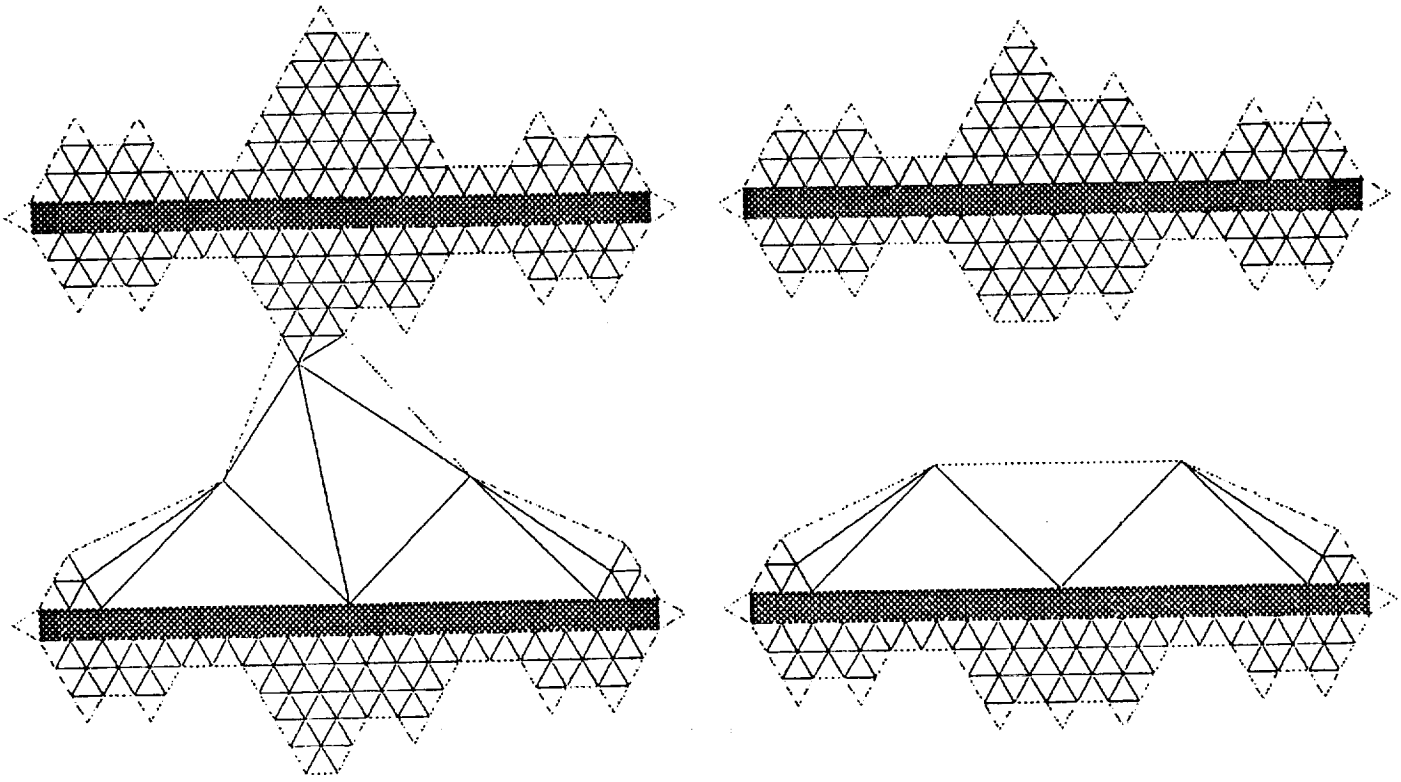
REQUIRED SEARCHES

- All Searches Based on Front, $O(\sqrt{N})$:
  - Dynamic
  - $O(N\log N)$
- Heap List for Choosing Front Edge
- Quadtree for Locating "Close" Points
- Octree for Intersected Front Triangle Circles
  - Point (x,y,r) in 3D
  - Generates Additional Length Scale

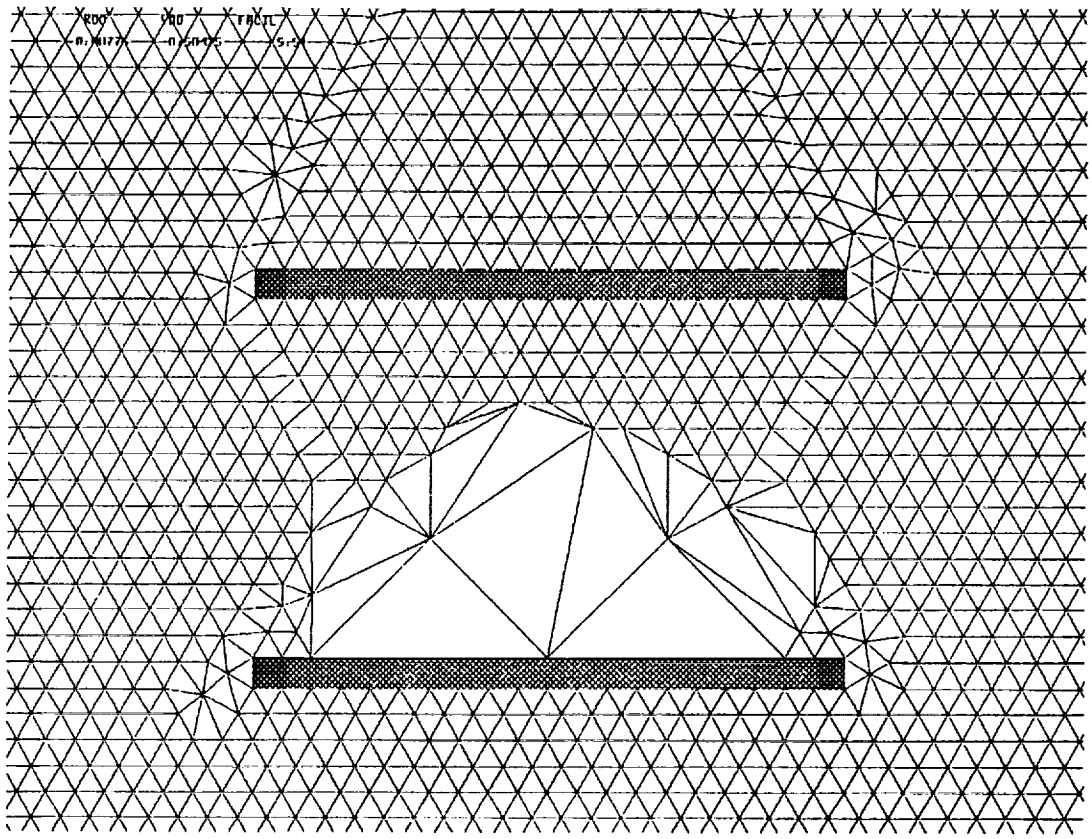


INTERSECTED CIRCUMCIRCLES

- Radius of Intersected Circles Provides Additional Length Scale
- Corresponds to f(x,y) on that (opposing) Front
- Useful in Regions of Rapid Variations in f(x,y)
  - if f(x,y) = constant circumcircles never intersected
- Additional Length Scale is Missing in Traditional Advancing Front Method

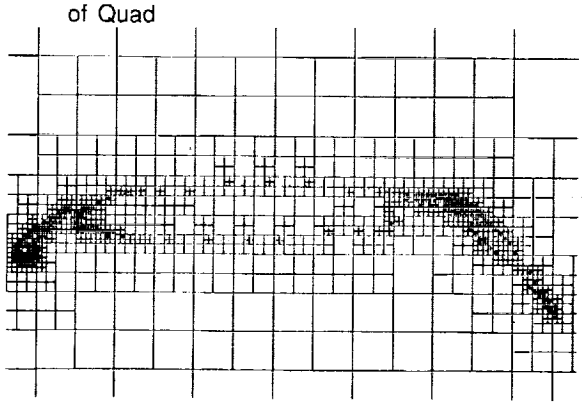


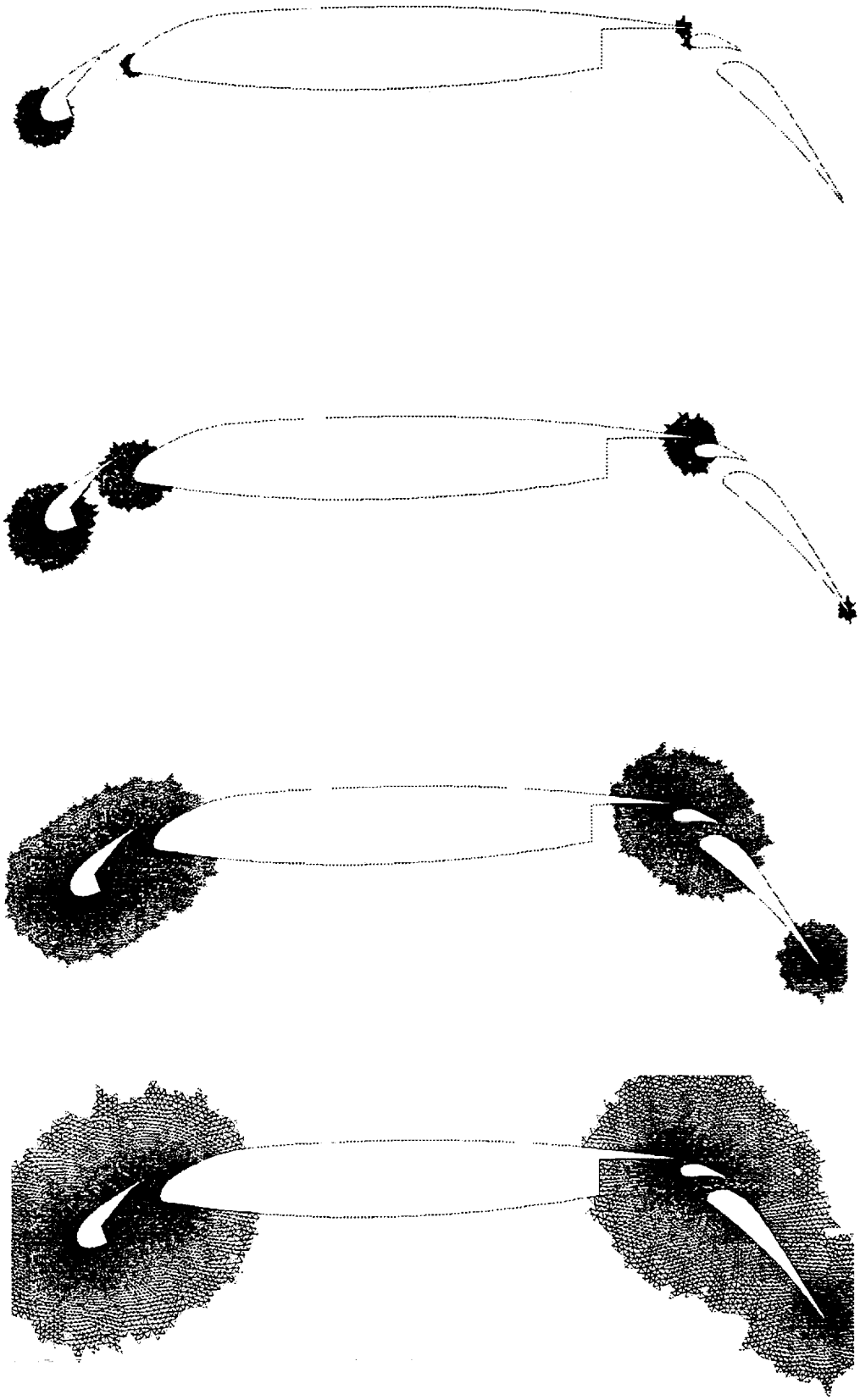
C-3

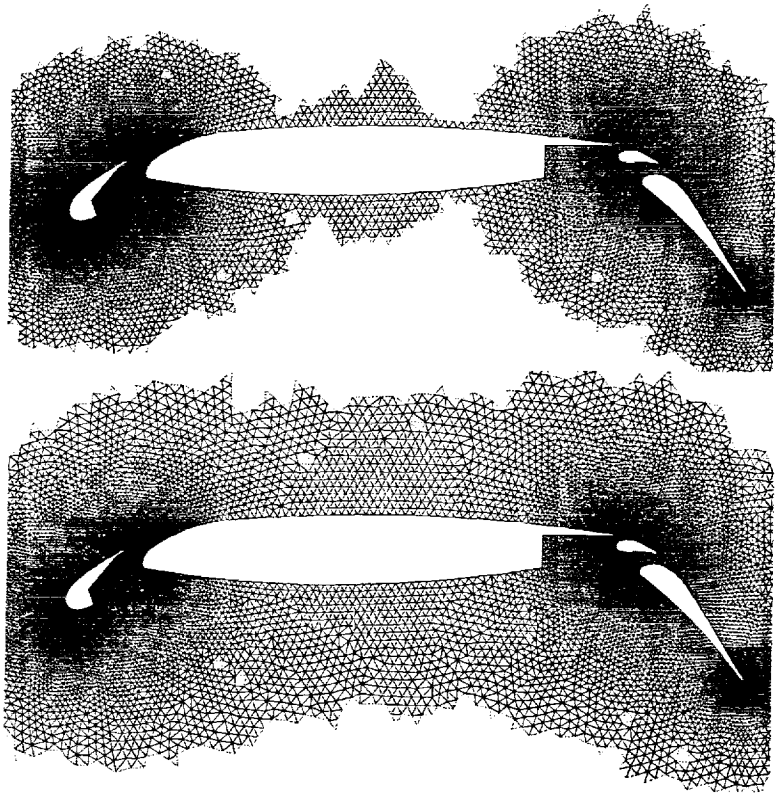


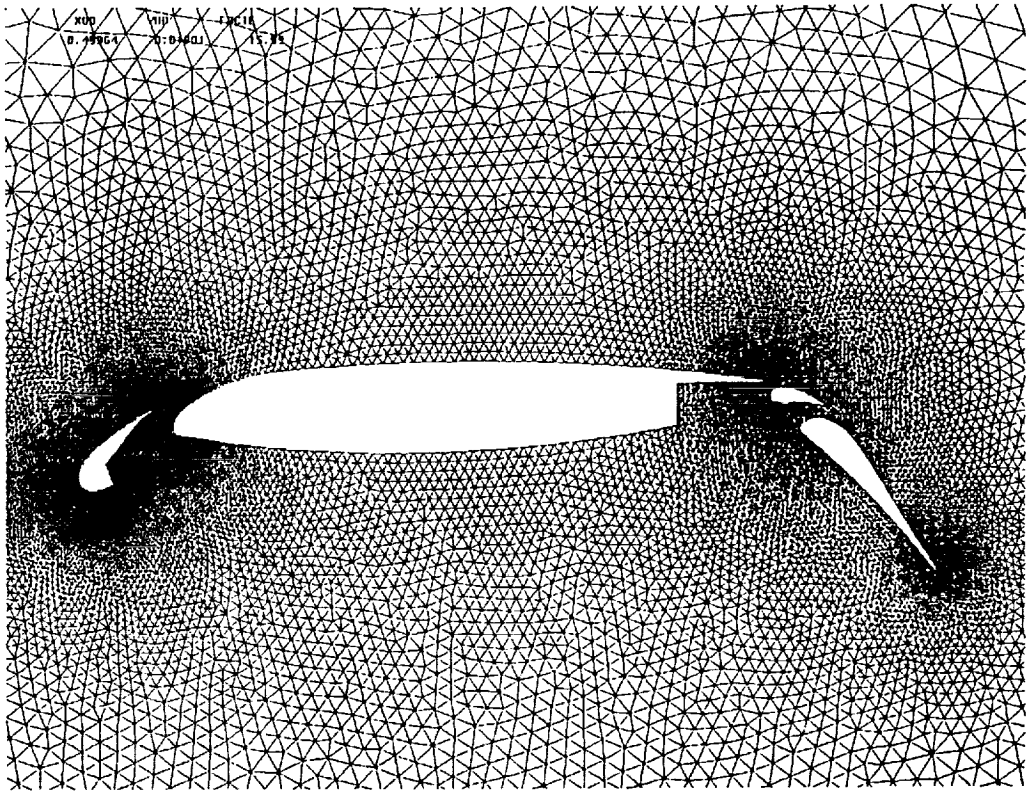
FIELD FUNCTION: F(X,Y)

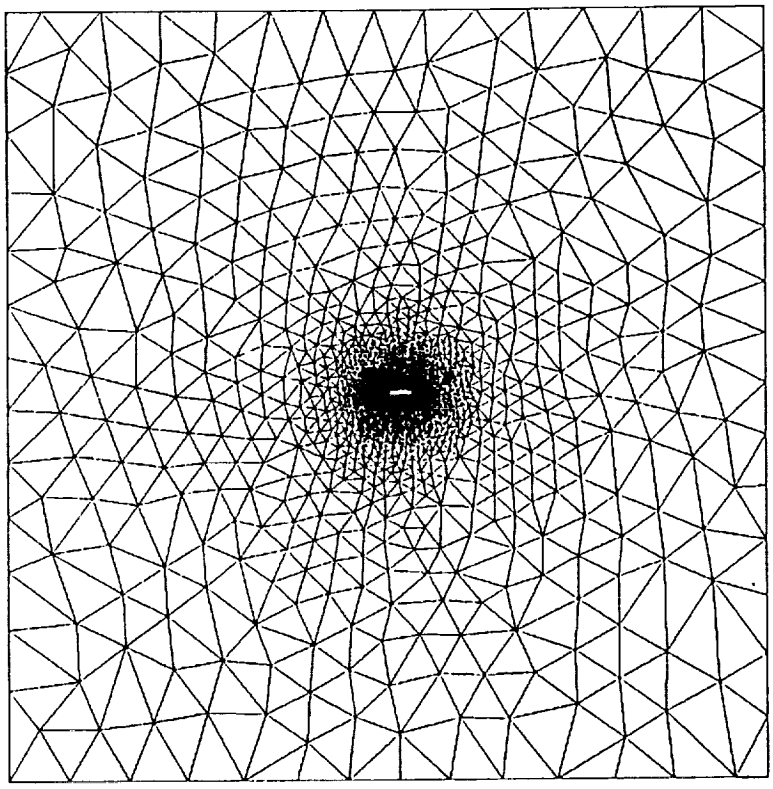
- Create Point Sources in Field and Solve Poisson Equation on Background Grid (Pirzadeh)
- Supporting Grid Taken as Initial Quadtree of Boundary Points
- To Determine f(x,y):
  - Traverse Quadtree to Locate Quad Containing (x,y)
  - $\rho(x,y)$ = Bilinear Interpolation of 4 Corners of Quad











AF-DT ALGORITHM

- Boundary Integrity Guaranteed (Initial Condition)
- Robustness:
  - All Local Operations
    (Never Create Unacceptable Triangles)
  - Validity Guaranteed by Constrained DT
    (Two Length Scales Required)
- Efficiency:
  - Generates Grid 1 Point at a Time
    (vs 1 Triangle at a Time)
  - Complexity: $O(N\log N)$
  - Storage: $O(\sqrt{N})$
- Counterpoint: Increased Coding Complexity

CONCLUSIONS

- Generates 500 Triangles/secd on SGI 4D35 Workstation
- 35% - 40% of Time Spent in Front Circle Test
- Extensions to 3D