A FINITE POINT METHOD FOR COMPRESSIBLE FLOW

Rainald Löhner, Carlos Sacco, Eugenio Oñate and Sergio Idelsohn

School of Computational Science and Informatics
M.S. 4C7, George Mason University
Fairfax, VA 22030-4444, USA
http://www.science.gmu.edu/∼rlohner

International Center for Numerical Methods in Engineering
Universidad Politécnica de Catalunya, Edificio C1
Campus Norte, Gran Capitan s/n
08034 Barcelona, Spain

OUTLINE

- Motivation
- Weighted Least Squares FPMs
- Construction of Local Clouds
- Flow Solver
- Examples
- Conclusions and Outlook

MOTIVATION

- Perception: Grid Generation Difficult
  $\Rightarrow$ Generation of Points Easier (?)
- Perception: Higher Order Unstructured Difficult
  $\Rightarrow$ Higher Order Finite Point Easier (?)

RELATED WORK

- R.A. Nay and S. Utku - *Variational Methods Eng.* 1 (1972).

- J. Batina - *AIAA*-93-0333 (1993).

- A. Duarte and J.T. Oden - *TICAM-Rep.* 95-05 (1995).

- E. Oñate, S. Idelsohn, O.C. Zienkiewicz and R.L. Taylor - *Int. J. Num. Meth. Eng.* 39, 3839-3866 (1996).

- E. Oñate, S. Idelsohn, O.C. Zienkiewicz, R.L. Taylor and C. Sacco - *Comp. Meth. Appl. Mech. Eng.* 139, 315-346 (1996).

- E. Oñate and S. Idelsohn - *Computational Mechanics* 21, 283-292 (1998).

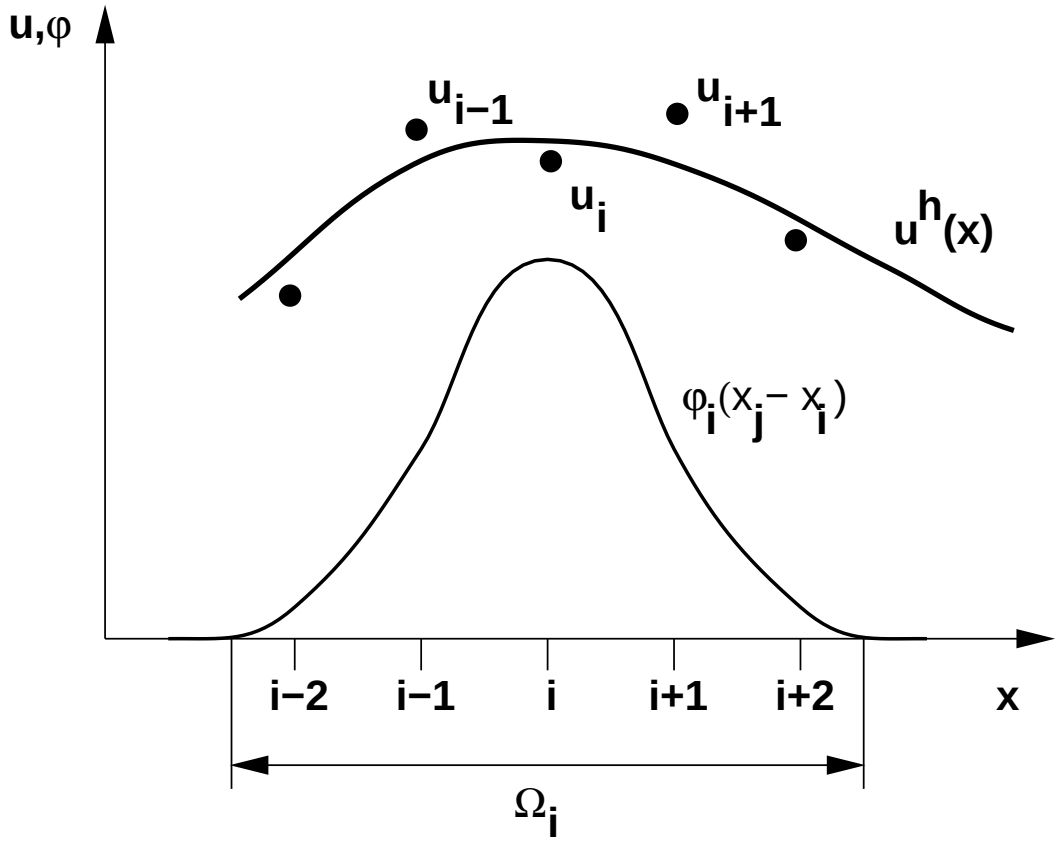
- E. Oñate C. Sacco and S. Idelsohn - *Comput. Visual. Sci.* 3, 67-75 (2000).

- R. Löhner, C. Sacco, E. Oñate and S. Idelsohn - *Int. J. Num. Meth. Eng.* 53, 1765-1779 (2002).

FINITE POINT METHODS

- Generate Global Cloud of Points
- For Each Point: Obtain Local Cloud
- Choose Local Approximation
  (Monomials + Least Squares)
- Introduce Local Approximation into PDE
- Weighted Residuals $\Rightarrow$ Algebraic Equations
  - Galerkin: Numerical Integration
    $\Rightarrow$ Need Grid
  - Collocation: Truly Mesh Free

WEIGHTED LEAST SQUARES (WLSQ)



WLSQ Procedure

WEIGHTED LEAST SQUARES (WLSQ)

Define:

$u(\mathbf{x})$: Function
Ω_i: Local Interpolation Domain of $u(\mathbf{x})$
Ω_i: $\mathbf{x}_j \in \Omega_i; j = 1, n$

Then:

$$u(\mathbf{x}) \approx u^h(\mathbf{x}) = p_k(\mathbf{x})\alpha_k = \mathbf{p}(\mathbf{x})^T \cdot \boldsymbol{\alpha} \quad , \quad k = 1, m$$

$\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_m]^T$
$\mathbf{p}(\mathbf{x})$: 'Base Interpolating Functions'
(e.g. Monomials)

MONOMIAL FUNCTIONS

$$\mathbf{p}_2 = [1, x, y, z, x^2, xy, xz, y^2, yz, z^2]^T \quad (10)$$

$$\mathbf{p}_{2.5} = [1, x, y, z, x^2, xy, xz, y^2, yz, z^2,$$
$$x^2y, x^2z, xy^2, xyz, xz^2, y^2z, yz^2]^T \quad (17)$$

$$\mathbf{p}_3 = [1, x, y, z, x^2, xy, xz, y^2, yz, z^2,$$
$$x^3, x^2y, x^2z, xy^2, xyz, xz^2,$$
$$y^3, y^2z, yz^2, z^3]^T \quad (20)$$

WLSQ Cont.

Notation:

$u_j = u(\mathbf{x}_j)$, $u_j^h = u^h(\mathbf{x}_j)$
$\mathbf{p}_j = \mathbf{p}(\mathbf{x}_j)$
$\varphi_{ij} = \varphi(\mathbf{x}_j - \mathbf{x}_i)$

WLSQ $\Rightarrow$ Minimize:

$$J_i = \varphi_{ij}(u_j - u_j^h)^2 = \varphi_{ij}(u_j - \mathbf{p}_j^T \cdot \boldsymbol{\alpha})^2 \quad . \quad j = 1, ..., n$$

Remarks:

- Always Require: $n \geq m$
- $n = m \Rightarrow$ Interpolation

WLSQ Cont.

Minimization of $J_i(\alpha_j) \Rightarrow$

$$\mathbf{A} \cdot \boldsymbol{\alpha} = \mathbf{B} \cdot \mathbf{u}$$

i.e.

$$\boldsymbol{\alpha} = \mathbf{C} \cdot \mathbf{u} \quad , \quad \mathbf{C} = \mathbf{A}^{-1} \cdot \mathbf{B}$$

With

$$\mathbf{A} = \sum_{j=1}^{n} \varphi_{ij} (\mathbf{p}_j \otimes \mathbf{p}_j^T)$$

$$\mathbf{B} = [\varphi_{1i}\mathbf{p}_1, \varphi_{2i}\mathbf{p}_2, ..., \varphi_{ni}\mathbf{p}_n]$$

SHIFT TO LOCAL FRAME (1)

Idea: Shift Origin to $\mathbf{x}_i$

$\Rightarrow$ Can Easily Obtain Derivatives from $\boldsymbol{\alpha}$

$$u_i^h = \alpha_1$$

$$\nabla u_i^h = (\alpha_2, \alpha_3, \alpha_4)$$

$$\nabla^2 u_i^h = 2 * (\alpha_5 + \alpha_8 + \alpha_{10})$$

SHIFT TO LOCAL FRAME (2)

Same As:

$$u^h\big|_i = C^{1j} u_j$$

$$\left.\frac{\partial u^h}{\partial x_l}\right|_i = D_l^{ij} u_j \quad , \quad D_l^{ij} = C^{qj} \quad , \quad q = l + 1$$

$$\nabla^2 u_i^h = L^{ij} u_j \quad , \quad L^{ij} = 2 * (C^{5j} + C^{8j} + C^{10j})$$

GENERATION OF LOCAL CLOUDS

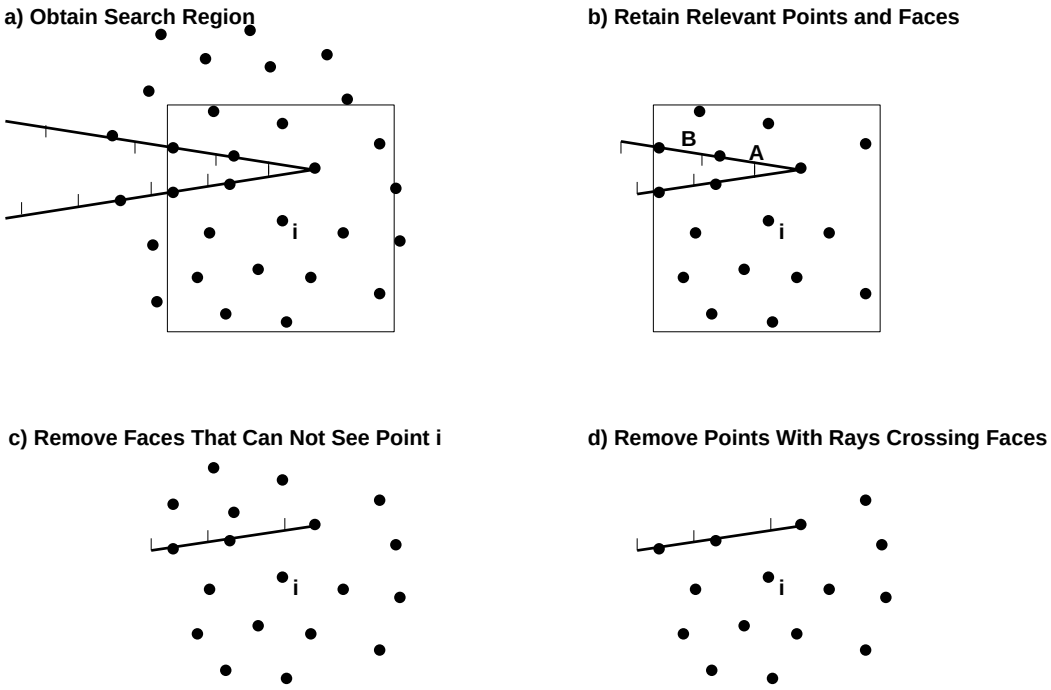
Many Possibilities:

- Spheres
- Octants + Minimum Nr.
- Local Delauney + Layers (Used Here)

Input Info:

- Cloud of Points
- Surface Triangulation (With B.C.)

LOCAL CLOUD



Finding Points for Local Cloud

LOCAL DELAUNEY CLOUD

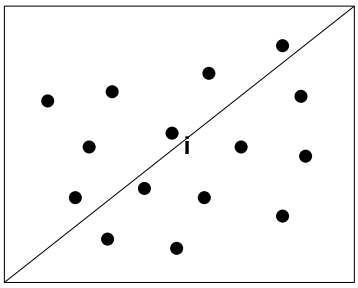
`do`: For each point `ipoin`:
Initialize search region for `ipoin`;
`while`: Not enough close points:
Enlarge search region;
Obtain points in search region;
Obtain boundary faces in region;
Remove faces that can not see `ipoin`;
Remove points whose ray
`ipoin:jpoin` intersects a face;
`endwhile`
Produce Delauney grid with local points;
Initialize local cloud list with first layer of nearest neighbours;
If local cloud acceptable: exit;
`do`: For all points, according to layers:
Add a point to local cloud;
If local cloud acceptable: exit;
`enddo`
No proper local cloud found ⇒ increase search region;
`enddo`

DATA STRUCTURES

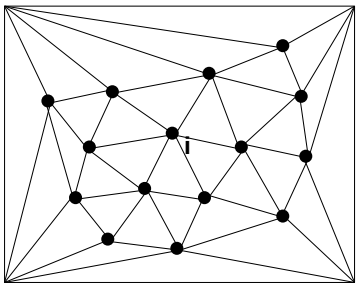
- Search for Close Points:
    - Octrees
    - Enlarge Search Region 30%, Reduce 15%
- Search for Close Faces:
    - Modified Octree (Spill-Over)
    - Search With Bounding Box
    - Removal of Repeated Faces:
      Hashing Techniques

DELAUNEY MESHING

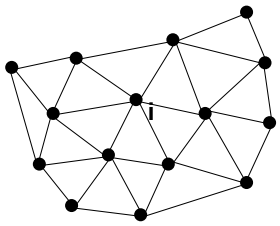
a) Place Points in Macro−Triangles



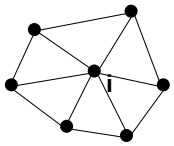
b) Delauney−Mesh of Local Cloud



c) Retain Elements of Original Points



d) Retain First Layer of Nearest Neighbours



Formation of Local Cloud

DELAUNEY MESHING

- Enclose Region (Box of 5/6 Tetrahedra)
- `do`: For All Close Points:
  Find Host Element
  Evaluate Proper Star-Shaped Domain
  Eliminate Elements in Star-Shaped
  Domain ⇒ Faces
  Introduce Point + New Elements
  Update Arrays
  `enddo`
- Remove External Elements
- Order Points According to Layers

(Bowyer/George)

ACCEPTABLE CLOUD CRITERIA

Observation: Not All Local Clouds Good

Reason: Singular Approximation Matrix $\mathbf{A}$

Accept If:

- $\mathbf{A}$ Invertible
- $\mathbf{A} \cdot \mathbf{A}^{-1} \approx 1$
- $|\mathbf{A}^{-1}| < c_b$
- Take Known Function and Obtain Derivatives
  - $u = x + y + z \Rightarrow \nabla u = (1, 1, 1)$
  - $u = x^2 + y^2 + z^2 \Rightarrow \nabla^2 u = 6$

  Difference $< 10^{-10}$

APPROXIMATION ORDER OF LOCAL CLOUD

Points Per Local Cloud:

- Tet Mesh: 15
- Cart Mesh: 27
- Empirical: 18-20

$\Rightarrow$ Could Approximate to Higher Order

- Test $\mathbf{A}$ of Higher Order Polynomials as Before
- Accept if Better

FLOW SOLVER (1)

Compressible Navier-Stokes:

$$\mathbf{u}_{,t} + \nabla \cdot \mathbf{F} = 0$$

WLSQ:

$$\nabla \cdot \mathbf{F}\big|_i \approx \mathbf{r}^i = D_l^{ij} \mathbf{F}_j^l$$

Key Idea: Symmetrize

$$\mathbf{r}^i = D_l^{ij}\Big|_{j \neq i} (\mathbf{F}_j^l + \mathbf{F}_i^l) + (D_l^{ii} - D_l^{ij}\Big|_{j \neq i}) \mathbf{F}_i^l$$

or:

$$\mathbf{r}^i = D_l^{ij}\Big|_{j \neq i} (\mathbf{F}_j^l + \mathbf{F}_i^l) + \tilde{D}_l^{ii} \mathbf{F}_i^l$$

FLOW SOLVER (2)

Rewrite Inner Product Over Dimensions l:

$$\mathbf{r}^i = d^{ij}\mathcal{F}_{ij} + \tilde{d}^{ii}\tilde{\mathbf{f}}_i = d^{ij}(\mathbf{f}_i + \mathbf{f}_j) + \tilde{d}^{ii}\tilde{\mathbf{f}}_i$$

$\mathbf{f}_i$: 'fluxes along edges/directions'

$$\mathbf{f}_i = S_l^{ij}\mathbf{F}_i^l \quad , \quad S_l^{ij} = \frac{D_l^{ij}}{d^{ij}} \quad , \quad d^{ij} = \sqrt{D_l^{ij}D_l^{ij}}$$

$$\tilde{\mathbf{f}}_i = \tilde{S}_l^{ii}\mathbf{F}_i^l \quad , \quad \tilde{S}_l^{ii} = \frac{\tilde{D}_l^{ii}}{\tilde{d}^{ii}} \quad , \quad \tilde{d}^{ii} = \sqrt{\tilde{D}_l^{ii}\tilde{D}_l^{ii}}$$

FLOW SOLVER (3)

Consider First Term:

- Equivalent to Galerkin WRM
- Equivalent to Central Differences
- ⇒ Unstable
- ⇒ Add Stabilizing Terms
    - Higher Order Damping
      (Requires Length)
    - Upwinding (No Length)
    - Used Here: vanLeer, Roe, AUSM+

ROE SOLVER

First Order:

$$\mathcal{F}_{ij} = \mathbf{f}_i + \mathbf{f}_j - |\mathbf{A}^{ij}|(\mathbf{u}_i - \mathbf{u}_j)$$

$|\mathbf{A}^{ij}|$: Roe Matrix in Direction D^{ij}

HIGHER ORDER SCHEMES (1)

Aim: Reduce Difference $\mathbf{u}_i - \mathbf{u}_j$

How: Reconstruct Solution at Edge Mid-Point
$\mathbf{u}_j^-, \mathbf{u}_i^+$

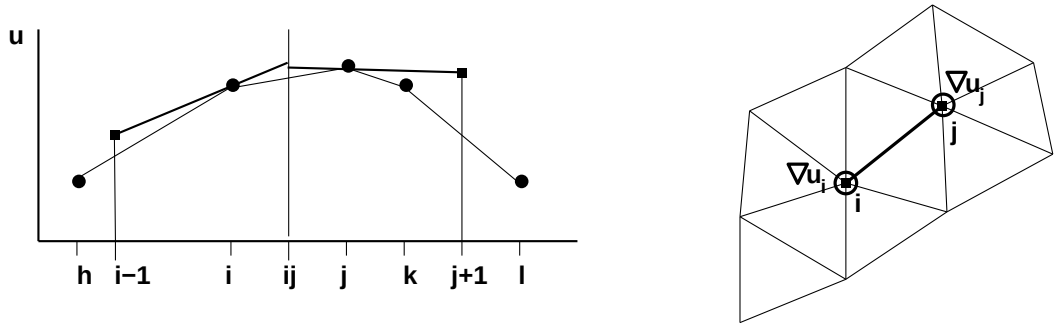
$$\mathcal{F}_{ij} = \mathbf{f}^+ + \mathbf{f}^- - \mid A(\mathbf{u}_i^+, \mathbf{u}_j^-) \mid (\mathbf{u}_j^- - \mathbf{u}_i^+)$$

where

$$\mathbf{f}^+ = \mathbf{f}(\mathbf{u}_i^+), \qquad \mathbf{f}^- = \mathbf{f}(\mathbf{u}_j^-)$$

HIGHER ORDER SCHEMES (2)

- MUSCL Upwind-Biased Interpolations
- Via Gradients
- Limiting (e.g. vanAlbada)
    - Conservative/Non-Conservative/Characteristic Variables



Higher Order Approximations

DISCRETIZATION IN TIME

WLSQ + Euler/NS ⇒ System of ODEs:

$$\mathbf{M}_c \frac{d\mathbf{u}}{dt} = \mathbf{r}$$

For Each Node i:

$$C_i^{1j} \frac{d\mathbf{u}_j}{dt} = \mathbf{r}^i = d^{ij}\mathcal{F}_{ij} + \tilde{d}^{ii}\mathbf{f}_i$$

Explicit Runge-Kutta:

$$\mathbf{M}_c \Delta\mathbf{u}^j = \mathbf{M}_c(\mathbf{u}^{n+j} - \mathbf{u}^n) = \alpha_j \Delta t \mathbf{r}(\mathbf{u}^{n+j-1}) = \mathbf{r}^j \quad , j = 1, m$$

α_j: Chosen To:
- Achieve High Temporal Accuracy
- Fast Damping of Higher Order Modes (Steady-State)

SOLUTION OF CONSISTENT MASS SYSTEM

Each Runge-Kutta Step:

$$\mathbf{M}_c \Delta \mathbf{u} = \mathbf{r}$$

Iterate:

$$\mathbf{M}_l \Delta \mathbf{u}^i = \mathbf{r} + (\mathbf{M}_l - \mathbf{M}_c)\Delta \mathbf{u}^{i-1} \quad , \quad i = 1, n$$

$$\Delta \mathbf{u}^0 = 0$$

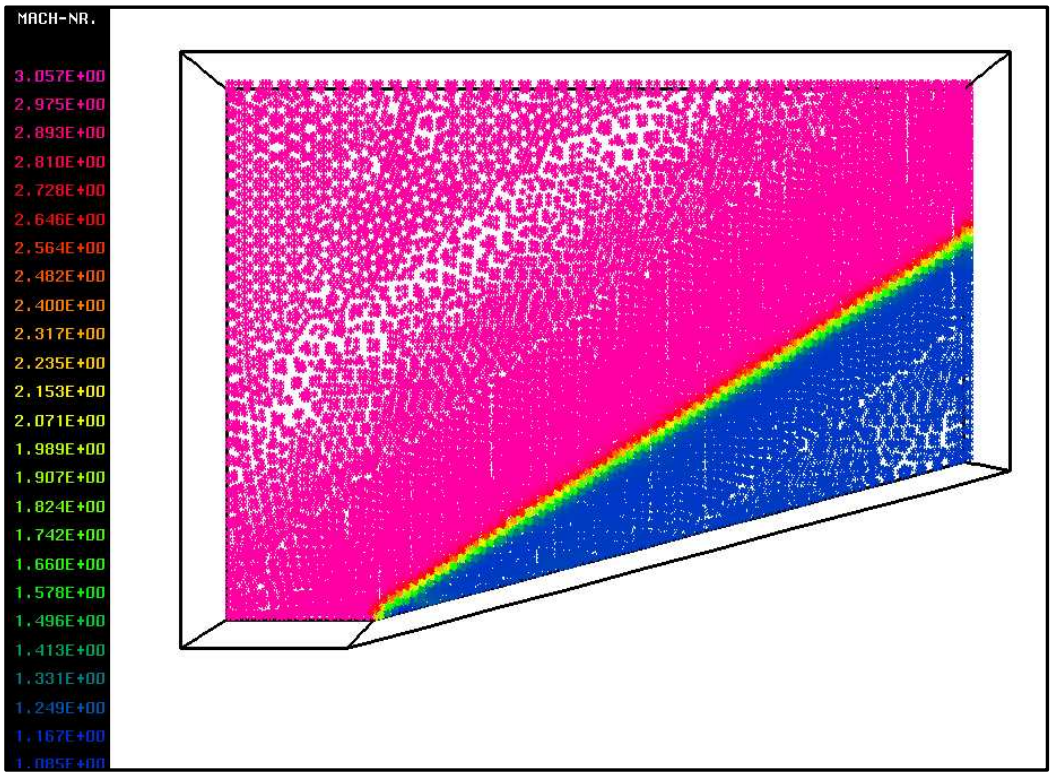
- $\mathbf{M}_l$: Unity Vector
- Typically $n = 3$

LAPLACIAN: BASIC LOOP

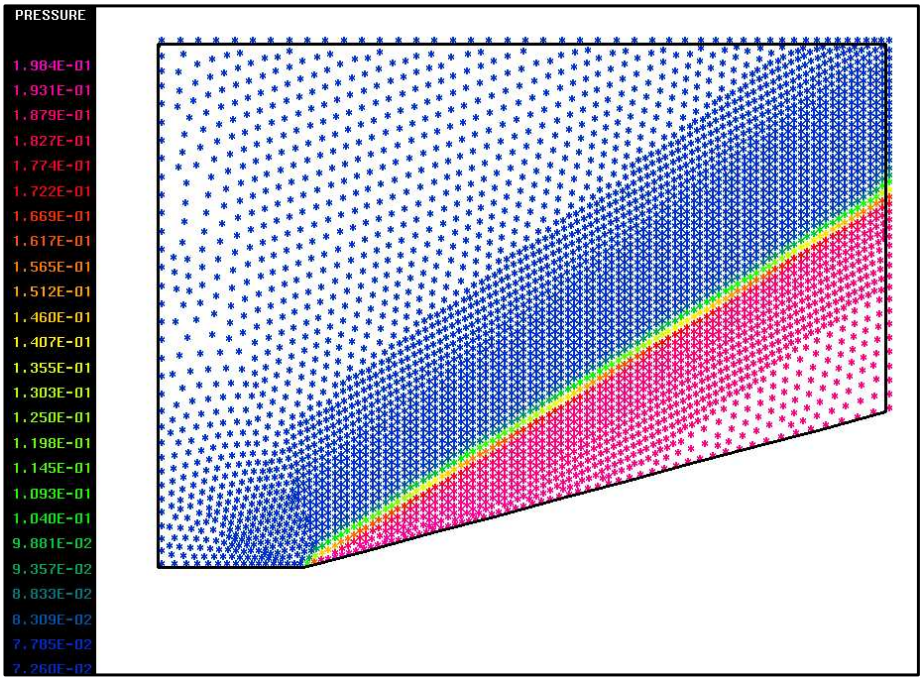
```
do ipoin=1,npoin
clapl=geopo(ipoin)*unkno(ipoin)
iedg0=esup2(ipoin)+1
iedg1=esup2(ipoin+1)
   do iedge=iedg0,iedg1
   jpoin=esup1(iedge)
   clapl=clapl+geoed(iedge)*unkno(jpoin)
   enddo
lapla(ipoin)=clapl
enddo
```

SUPERSONIC FLOW PAST A WEDGE (1)

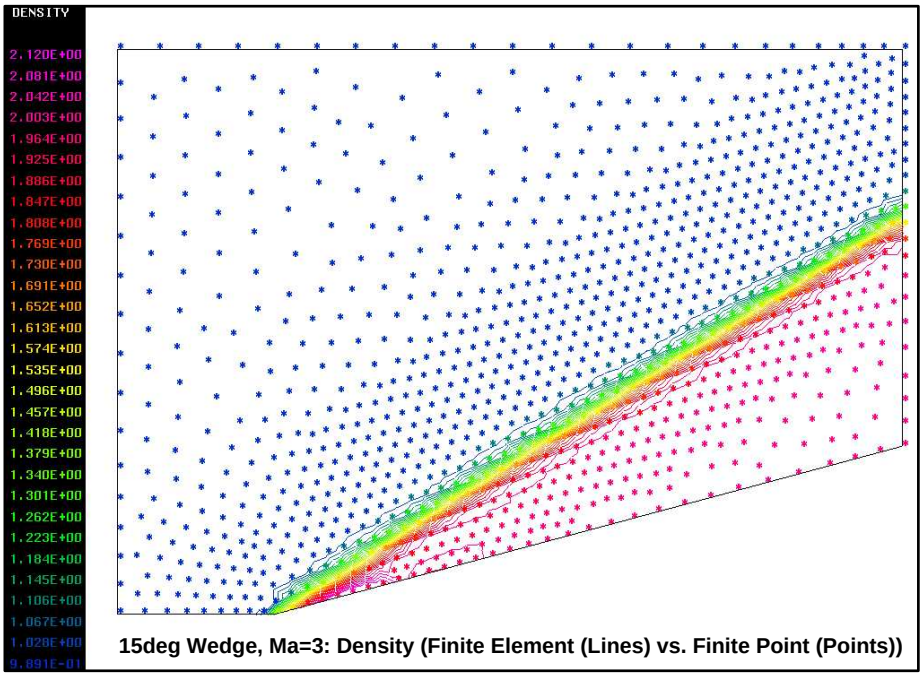
- Test Conservation/Shock Capturing Properties
- Compare to FEM/FVM
- $Ma = 3.0$, Wedge: 15^o
- vanLeer + vonAlbada



Wedge: Mach-Number in Mid-Plane



Wedge: Surface Pressure



Wedge: Results from FE (lines) and FP (point)

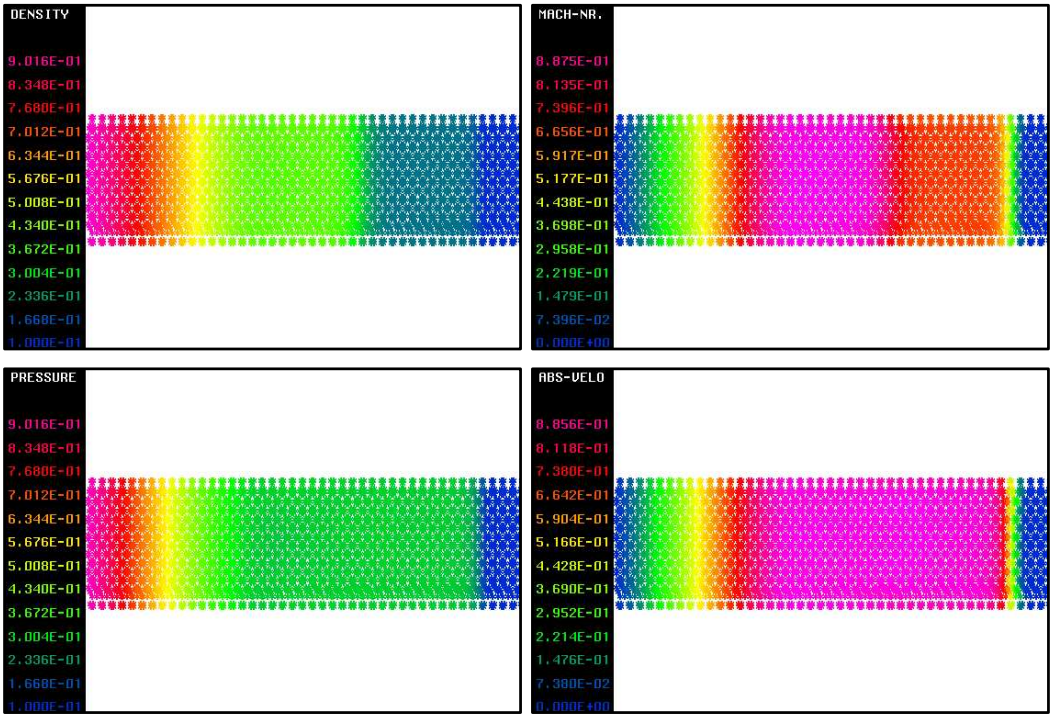
SOD SHOCK TUBE (1)

- Classic Test Case for Transient Flows

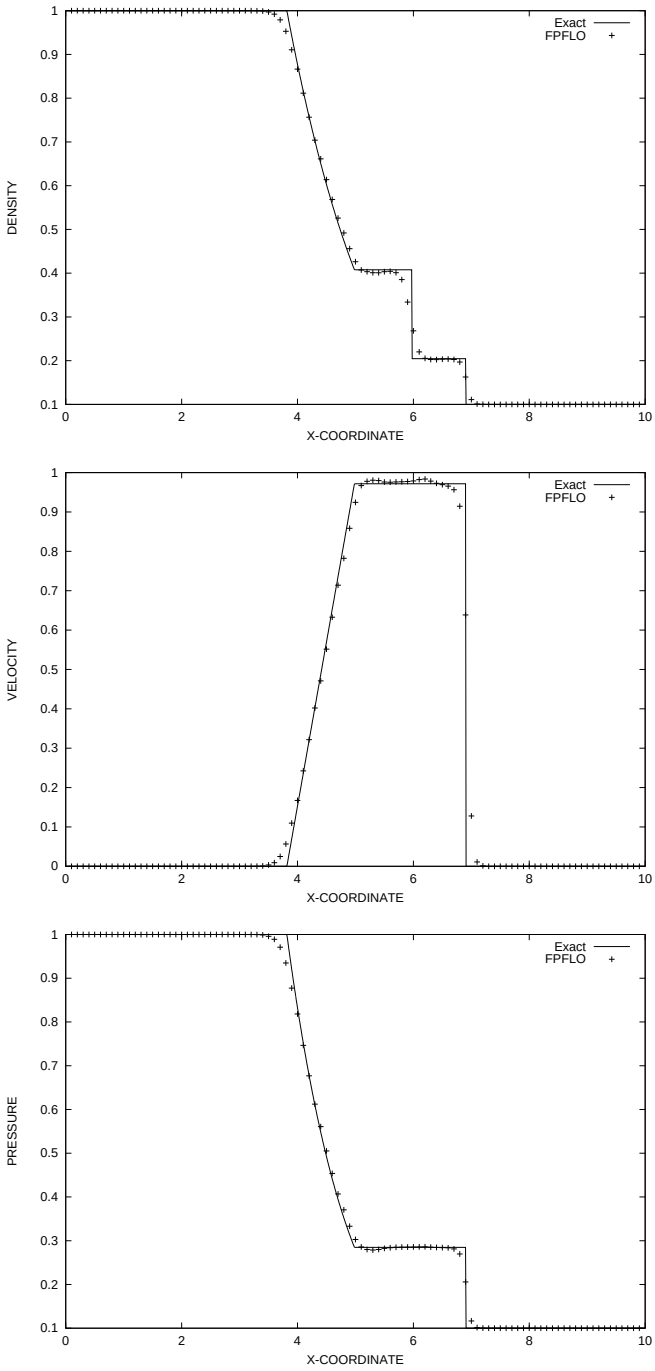
- Compare to FEM/FVM

- $\rho_1 = p_1 = 1.0$ and $\rho_2 = p_2 = 0.1$

- Roe + vonAlbada



Sod Shock Tube



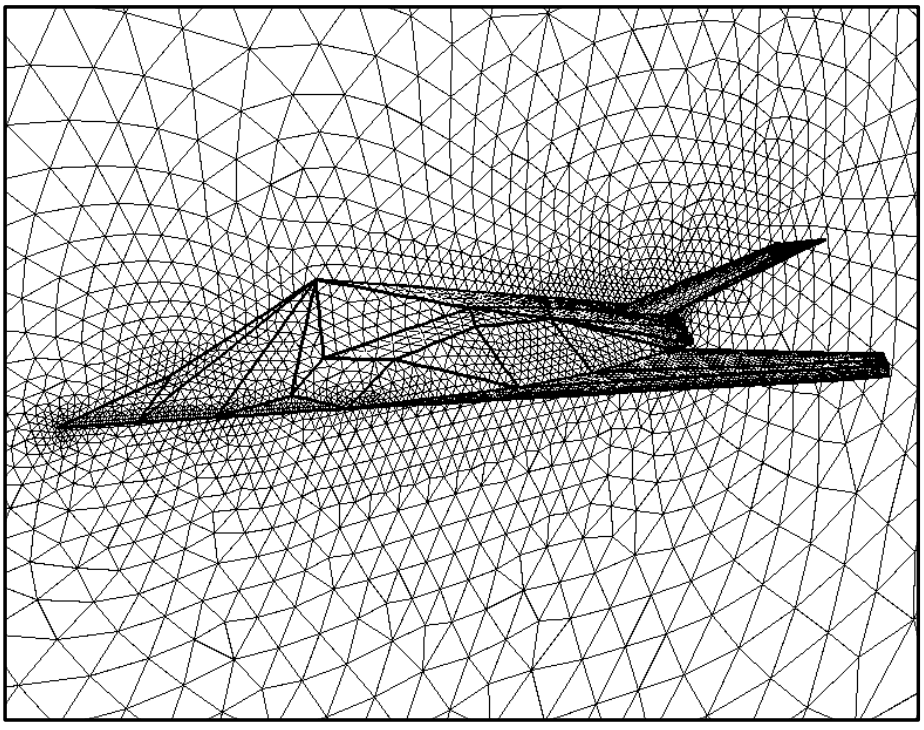
Sod Shock Tube: Comparison to Exact Solution

F117 (1)

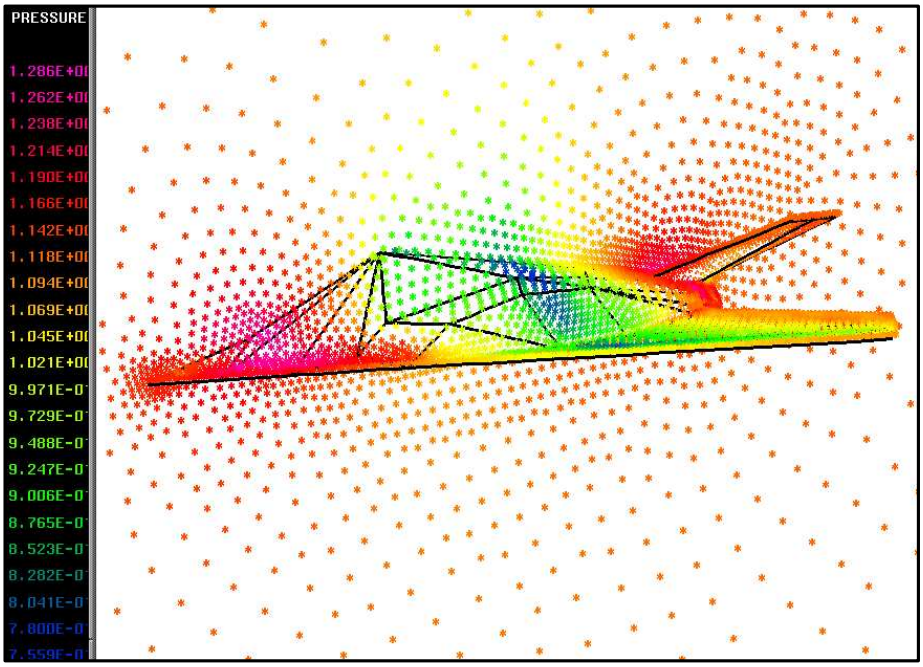
- More Realistic Case/Transonic Flow

- $Ma = 0.8$, $\alpha = 5^o$

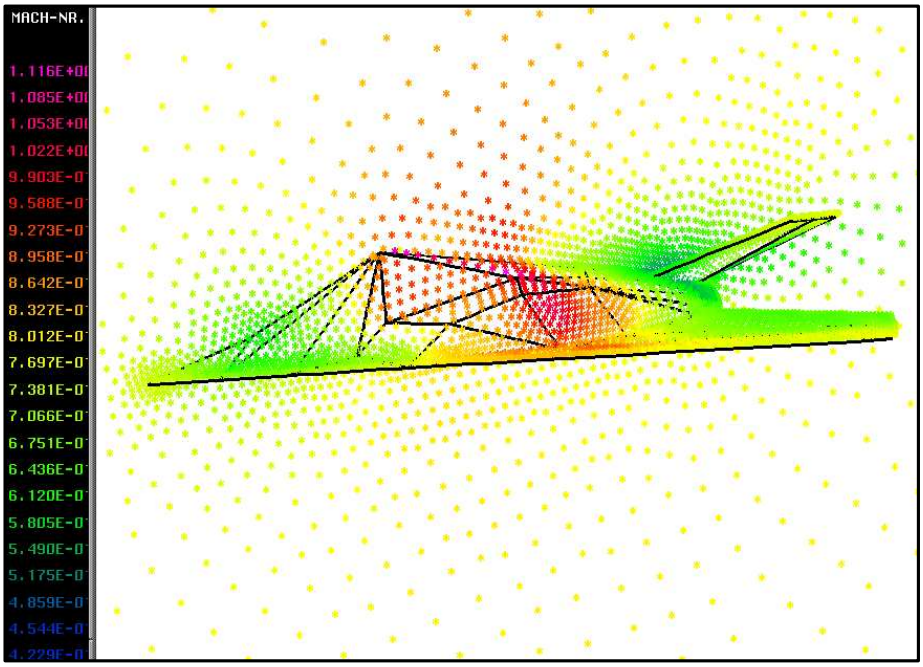
- Roe + vonAlbada



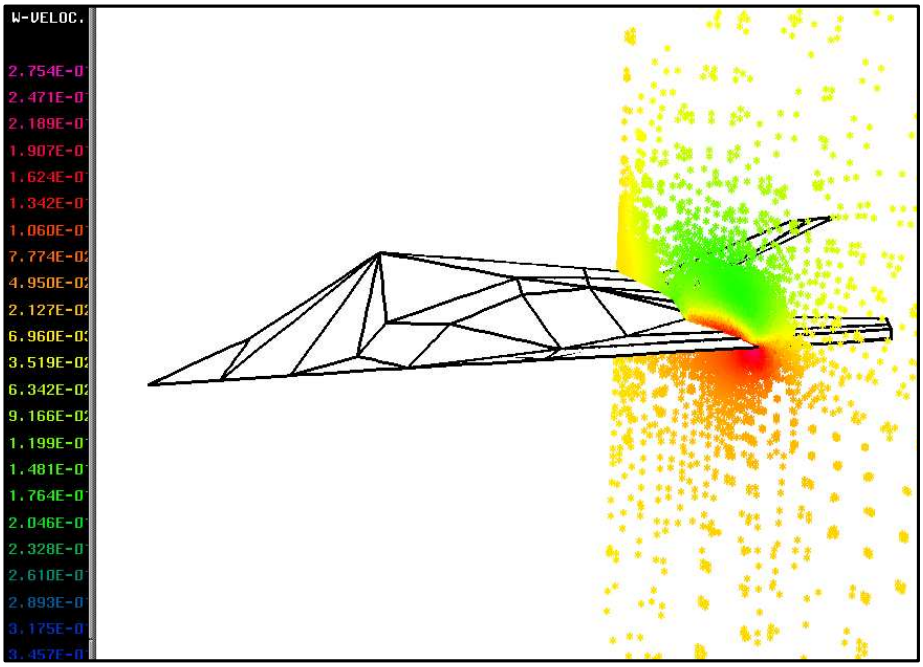
F117: Surface Mesh



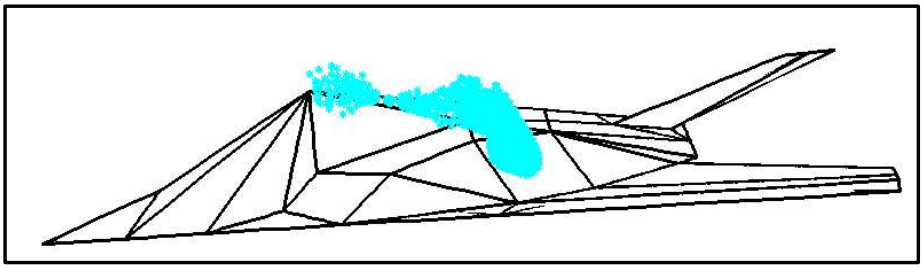
F117: Surface Pressures



F117: Surface Mach-Number



F117: Z-Velocity in Plane



F117: Sonic Surface

SUMMARY

- WLSQ Finite Point Method for Compressible Flow
- Key Ingredients:
    - Local Clouds via Delaunay
    - Quality Criteria
    - Riemann Solvers
    - Limitors
    - Edge-Based Data Structure
    - Fully Parallel (Shared)
- Accuracy Comparable to FEM/FVM
- Efficiency: 2-3 Times Slower Than FEM

OUTLOOK

- Improve Efficiencies
    - Reduction of Cache-Misses
    - Avoid Duplicate Info
- Higher-order Schemes