SPACEMARCHING: MOTIVATION

- Flowfield 'Mainly Supersonic' for:
  - SSCT
  - Scramjets
  - Missiles
  - Re-Entry Vehicles

- If $Ma > 1$: Downstream Does Not Influence Upstream

  $\Rightarrow$ Can Obtain the Solution in One Pass

  $\Rightarrow$ Should Be Extremely Cost-Effective

REFERENCES

Among Others:

Kutler, Reinhardt and Warming (1973)

Schiff and Steger (1979): Viscous

Chakravarthy and Szema (1987)

Matus and Bender (1990): Direct Solver in Plane

Lawrence, Chaussee and Tannehill (1991): 3-D PNS

McGrory, Walters and Löhner (1991): Semi-Structured

Nakahashi and E. Saitoh (1996): Fully Unstructured

R. Löhner (1998): Macro-Blocking, Transient

SPACE-MARCHING OPTIONS

- Plane-by-Plane
    - Good for Structured/Semi-Structured Grids
    - Very Low Memory Overhead (2-3 Planes)
    - Subsonic Pockets Awkward

- Subregion-by-Subregion
    - Good for Fully Unstructured Grids
    - Based on (Any) Time-Marching Solver
    - Subsonic Pockets Easily Accommodated
    - Requires Complete Mesh

AREAS OF WORK

Areas of Work:

- Masking of Points and Edges
- Renumbering of Points and Edges
- Grouping to Avoid Memory Contention
- Extrapolation of the Solution
- Treatment of Subsonic Pockets
- Proper Measures For Convergence
- Transient Problems
- Blocking

MASKING OF POINTS AND EDGES (1)

Two Types of Loops:

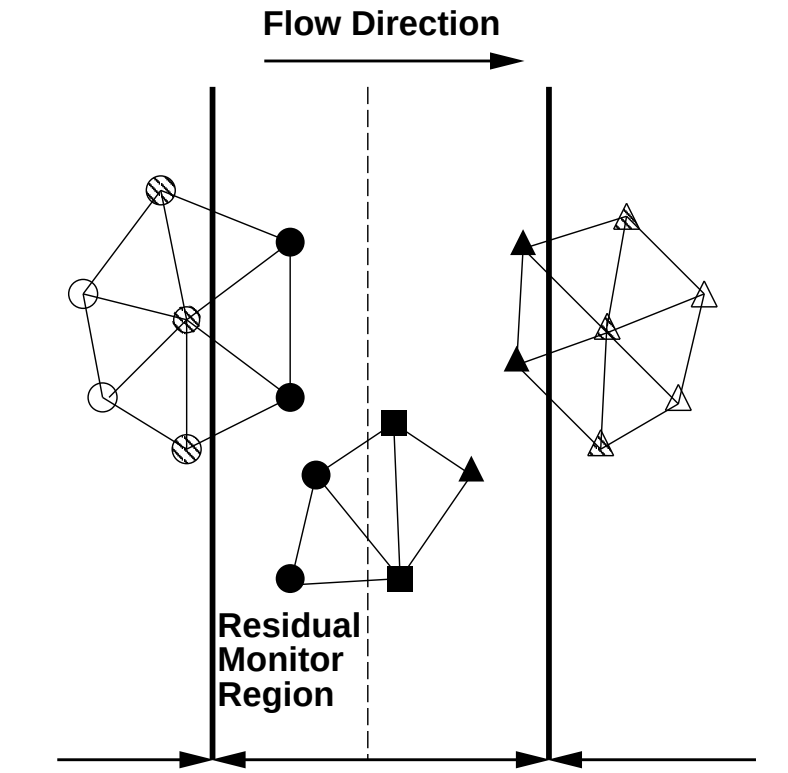
a) Point Loops (Updates, Point-Sum Initialization, ..)

```
do ipoin=1,npoin
   do work on the point level
enddo
```

b) Edge Loops (Flux Evaluations, ..)

```
do iedge=1,nedge
   gather point information
   do work on the edge level
   scatter-add edge results to points
enddo
```

MASKING OF POINTS AND EDGES (0)



	△	◬	▲	■	●	⊘	○
maskp	0	1	2	3	4	5	6

Figure 1 Masking of Points

MASKING OF POINTS AND EDGES (1)

Notation of **Nakahashi and Saito** (AIAA-96-0418)

`maskp=0:` Point in Downstream, Uncomputed Field;
`maskp=1:` Point in Downstream, Uncomputed Field, Connected to Active Domain;
`maskp=2:` Point in Active Domain;
`maskp=3:` Point of `maskp=2`, Connected to Points of `maskp=4`
`maskp=4:` Point in Residual-Monitor Subregion of Active Domain;
`maskp=5:` Point in Upstream Computed Field, Connected to Active Domain;
`maskp=6:` Point in Upstream Computed Field

`maske=0:` Inactive Edge
`maske=1:` Active Edge (At Least One Point With `0<maskp<6`)

MASKING OF POINTS AND EDGES (2)

Conversion of Time-Marching to Space-Marching:

Loop 1a:

```
do ipoin=1,npoin
   if(maskp(ipoin).gt.0.  and .
      maskp(ipoin).lt.6) then
      do work on the point level
   endif
enddo
```

Loop 2a:

```
do iedge=1,nedge
   if(maske(iedge).eq.1) then
      gather point information
      do work on the edge level
      scatter-add edge results to points
   endif
enddo
```

RENUMBERING OF POINTS AND EDGES (1)

Observation: Large Number of Missed `if`-Tests

⇒ Renumber Points According to Marching Direction

Define:

`npami:` The Minimum Point-nr. in the Active Region;
`npamx:` The Maximum Point-nr. in the Active Region;
`npdmi:` The Minimum Point-nr. Touched by Active Edges;
`npdmx:` The Maximum Point-nr. Touched by Active Edges;

Loop 1b:

```
do ipoin=npami,npamx
   if(maskp(ipoin).gt.0.  and .
      maskp(ipoin).lt.6) then
      do work on the point level
   endif
enddo
```

Re: Also Reduces Cache-Misses Considerably

RENUMBERING OF POINTS AND EDGES (2)

Observation: Large Number of Missed `if`-Tests

⇒ Renumber Edges According to Point-Nr.
(i.e. Marching Direction)

Define:

`neami` : The minimum active edge-nr.;
`neamx` : The maximum active edge-nr.;

Loop 2b:

```
do iedge=neami,neamx
   if(maske(iedge).eq.1) then
      gather point information
      do work on the edge level
      scatter-add edge results to points
   endif
enddo
```

Re: Also Reduces Cache-Misses Considerably

RENUMBERING OF POINTS AND EDGES

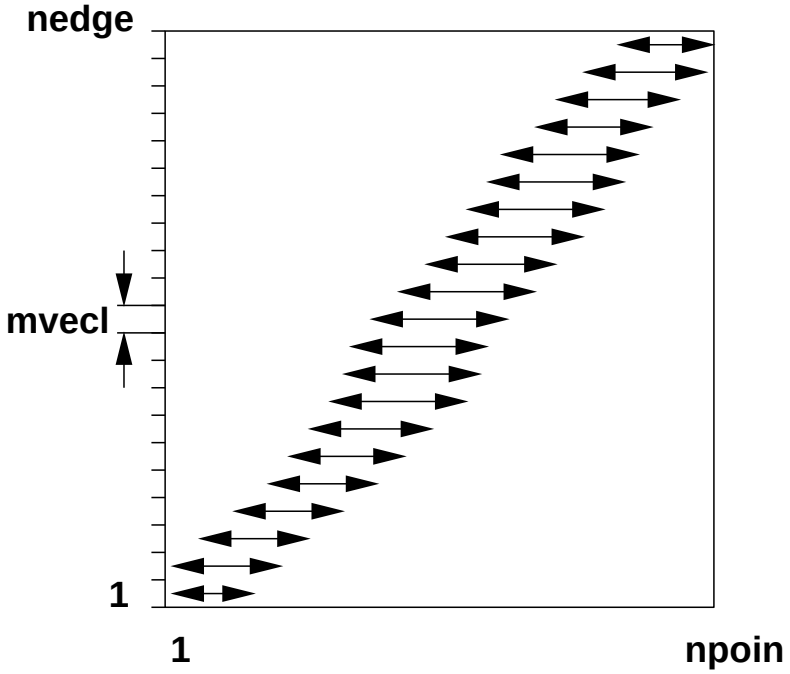


Figure 2 Near-Optimal Point-Range Access of Edge-Groups

GROUPING TO AVOID MEMORY CONTENTION (1)

Observation: Memory Contention Possible

⇒ Renumber Edges According to Groups

Loop 2c

```
    do ipass=1,npass
       nedg0=edpas(ipass)+1
       nedg1=edpas(ipass+1)
c$ dir ivdep                !  Pipelining directive
       do iedge=nedg0,nedg1
          if(maske(iedge).eq.1) then
             gather point information
             do work on the edge level
             scatter-add edge results to points
          endif
       enddo
    enddo
```

GROUPING TO AVOID MEMORY CONTENTION (1)

Identify 'Active Groups'

Loop 2d

```
    do ipass=1,npass
       nedg0=abs(edpas(ipass))+1
       nedg1=    edpas(ipass+1)
       if(nedg1.gt.0) then
c$ dir ivdep                !  Pipelining directive
          do iedge=nedg0,nedg1
             gather point information
             do work on the edge level
             scatter-add edge results to points
          enddo
       endif
    enddo
```

Re: Same Inner Loop As Original Code

EXTRAPOLATION OF THE SOLUTION

For New Active Points:

- Keep Values at $t = t_0$
- Extrapolate

Extrapolation Better

- Use 'Most Upstream' (Angle) Criterion
- Do Several Passes, Starting Upstream

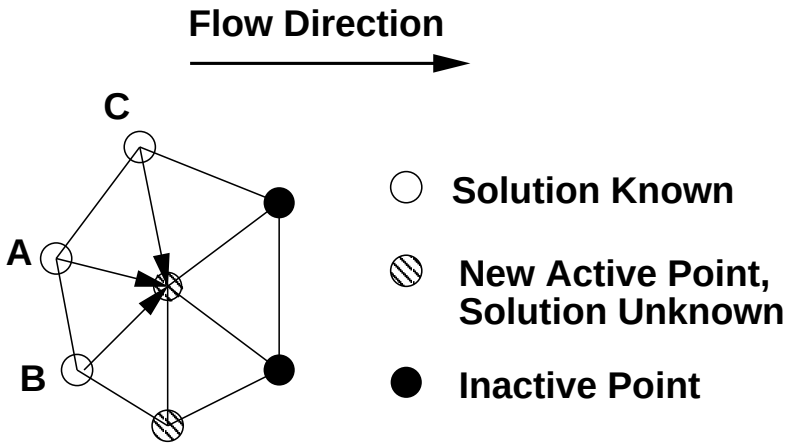


Figure 3 Extrapolation of the Solution

TREATMENT OF SUBSONIC POCKETS (1)

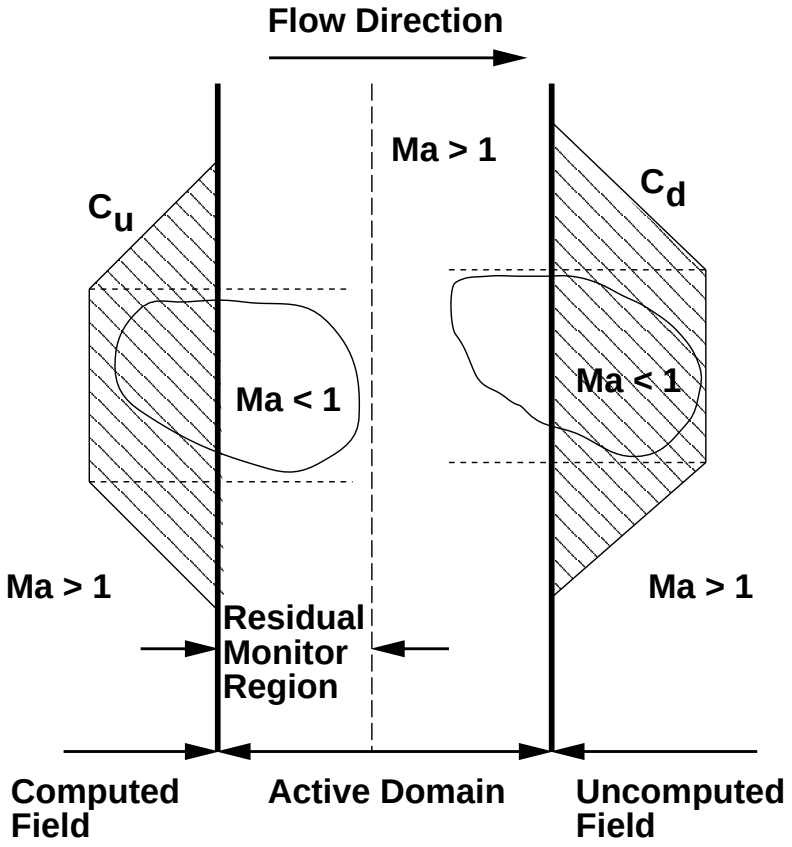


Figure 4 Treatment of Subsonic Pockets

TREATMENT OF SUBSONIC POCKETS (2)

Space-Marching ⇒ Need to Have Active Region Completely Enclosed By Supersonic Flow

- Shift Planes (Nakahashi and Saito)
    - Inefficient for Small Pockets
    - Large Already Converged Region
- Construct 'Conical Regions' (Here)
    - Efficient for Small Pockets

Avoid Repeated Markings by Adding 'Safety Zone'

MEASURING CONVERGENCE (1)

Why: Need to Know When to Stop

Explicit Scheme

$$\mathbf{M}_l \Delta \mathbf{u}^n = \mathbf{R}^n$$

Typical Convergence Measure:

$$r^n = \int_\Omega |\Delta \mathbf{u}^n| d\Omega \approx \sum_i \mathbf{M}_l^i \ |\Delta \hat{\mathbf{u}}_i^n|$$

- $r^n << r^1 \Rightarrow$ Convergence
  - Relative Measure
  - Depends on Domain
- Space-Marching Region Changes
  - Need Absolute Measure

MEASURING CONVERGENCE (2)

a) Absolute Measure:

$$max_i(|\Delta\hat{\mathbf{u}}_i^n|) < \epsilon_0$$

b) Take Out Dimensionality:

$$\frac{max_i(|\Delta\hat{\mathbf{u}}_i^n|)}{max_i(\hat{\mathbf{u}}_i)} < \epsilon_1$$

c) Take Out Courant-Nr. (Δt) Dependency

$$\frac{max_i(|\Delta\hat{\mathbf{u}}_i^n|)}{CFL\ max_i(\hat{\mathbf{u}}_i)} < \epsilon_2$$

- Quite Reliable
- Possible 'Hang-Up' for Limitors

EXTENSION TO TRANSIENT PROBLEMS

Implicit Time-Marching Scheme:

$$\frac{3}{2}\mathbf{M}_l^{n+1}\mathbf{u}^{n+1} - 2\mathbf{M}_l^{n}\mathbf{u}^{n} + \frac{1}{2}\mathbf{M}_l^{n-1}\mathbf{u}^{n-1} + \Delta t\mathbf{R}^{n+1} = 0$$

Solved Via Pseudo-Timestep Approach:

$$\frac{d}{d\tau}\mathbf{u} + \mathbf{R}^{*} = 0 \quad , \qquad (*)$$

With:

$$\mathbf{R}^{*} = \frac{3}{2}\mathbf{M}_l^{n+1}\mathbf{u}^{n+1} - 2\mathbf{M}_l^{n}\mathbf{u}^{n} + \frac{1}{2}\mathbf{M}_l^{n-1}\mathbf{u}^{n-1} + \Delta t\mathbf{R}^{n+1}$$

Solve (*) Via Space-Marching

MACRO-BLOCKING (1)

Why:

- Space-Marcher Requires Whole Computational Domain
  ⇒ Memory Limitation for Large Problems
- Extent of Supersonic Region Usually Known A Priori

⇒ Subdivide Domain and Operate on Sub-Domain Only

MACRO-BLOCKING (2)

Steps:

- Generate Mesh for Complete Domain

  ⇒ No Need to Change CAD Database

- Subdivide Mesh Into `nblock` Blocks

- Add 1-2 Layers of Elements

- Then:

```
do iblock=1,nblock
   Interpolate From Block iblock-1
   Converge the Solution
   Store Data
enddo
```

Use Space-Marching in Each Subdomain

MACRO-BLOCKING (3)

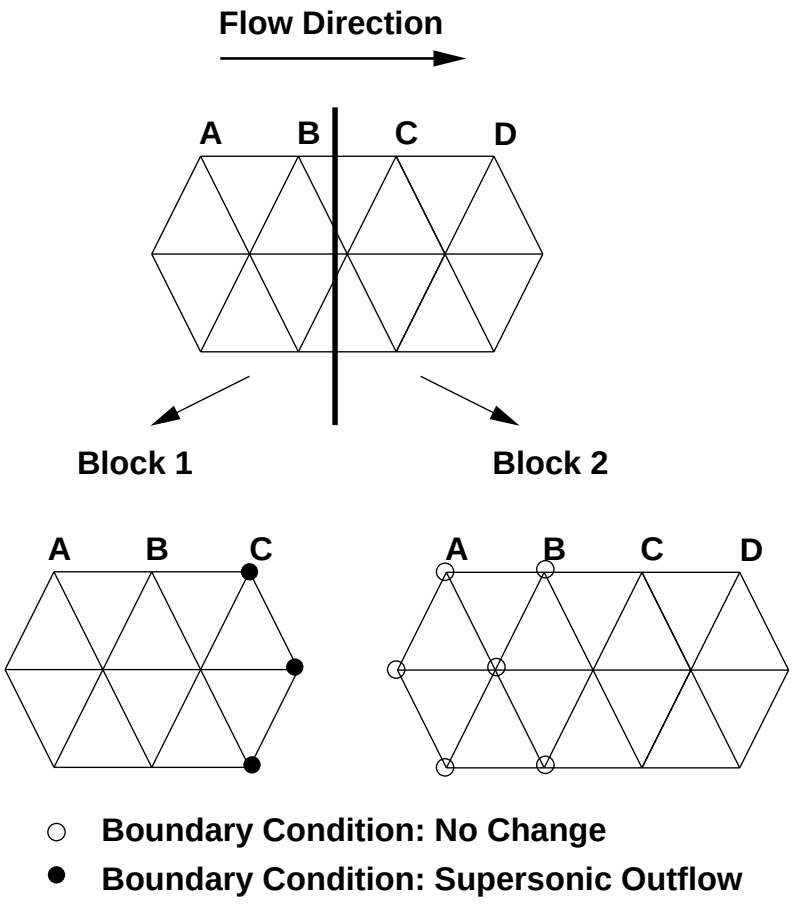


Figure 5 Macro-Blocking With 2 Layers of Overlap

MACRO-BLOCKING (4)

Ramp

$\alpha = 15^o$
$Ma_\infty = 3$

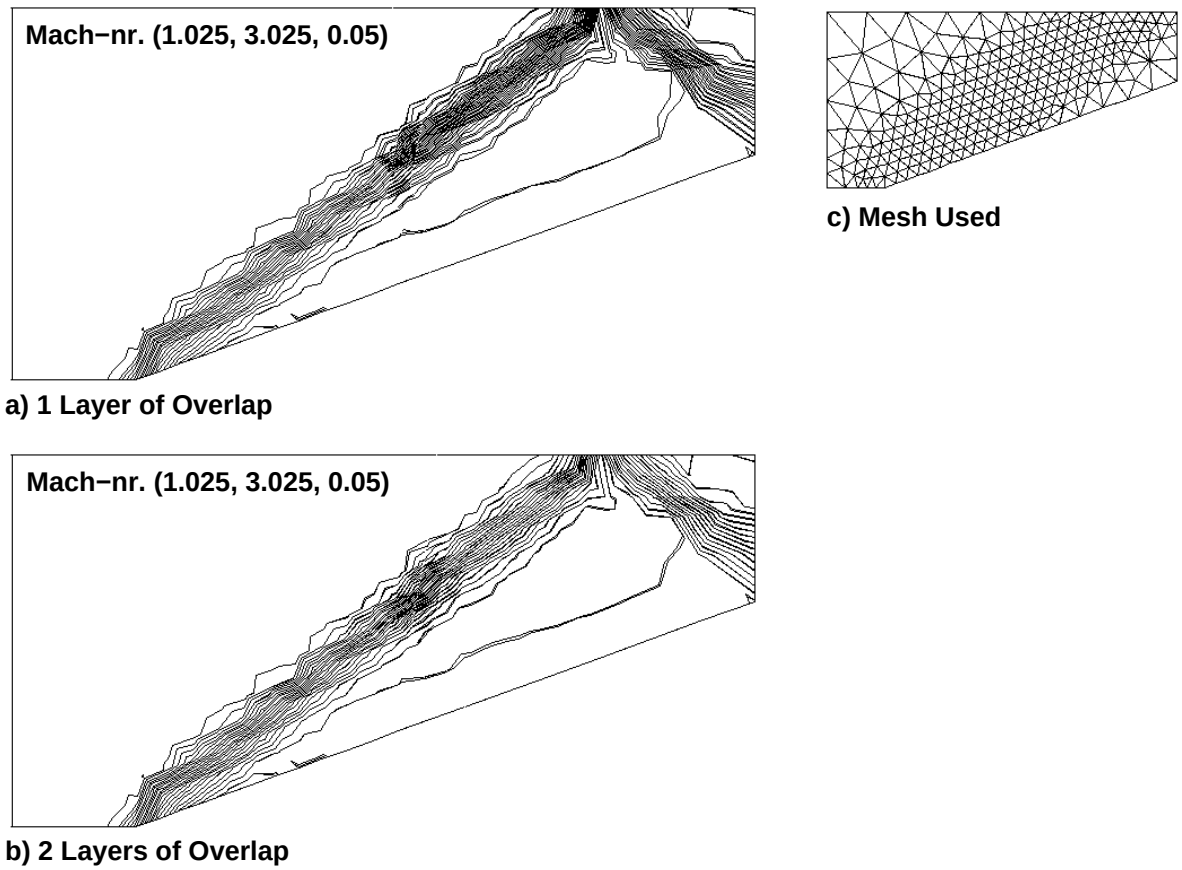


Figure 6 Macro-Blocking For Ramp

SUPERSONIC INLET (1)

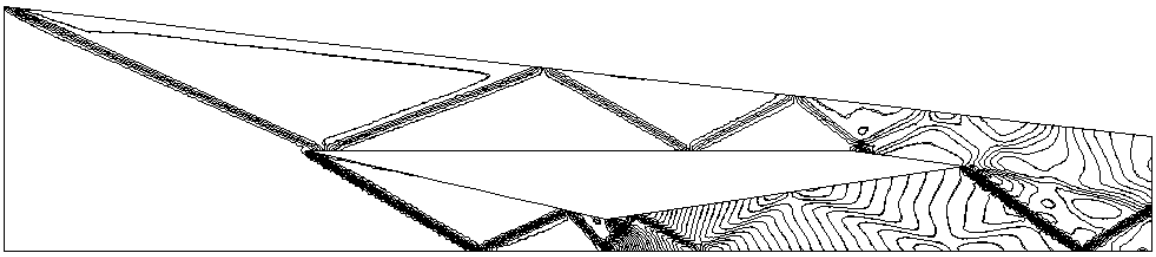


Figure 7a Supersonic Inlet: Mach-Number (Usual vs Marching)

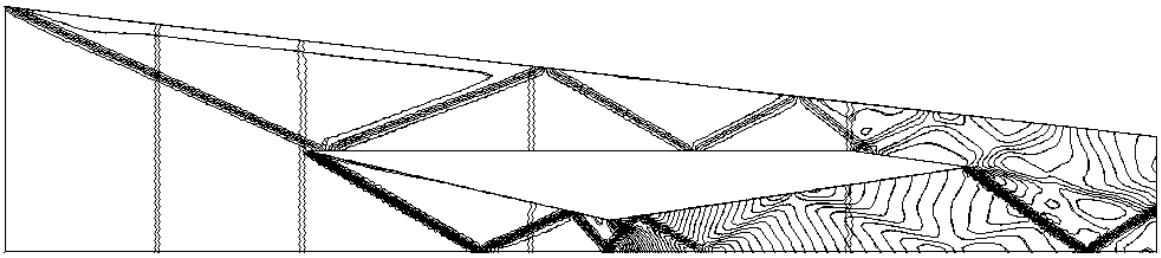


Figure 7b Supersonic Inlet: Mach-Number (Usual vs Block)

SUPERSONIC INLET (2)

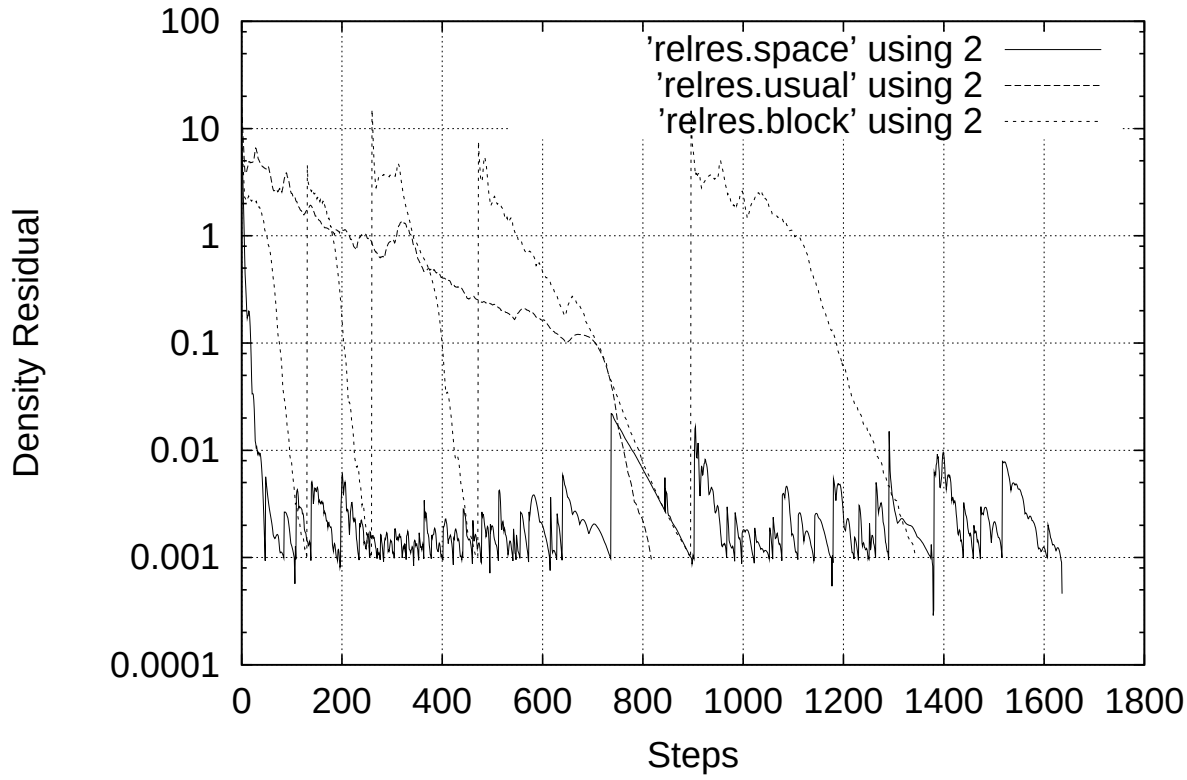


Figure 7b Supersonic Inlet: Mach-Number (Usual vs Block)

SUPERSONIC INLET (3)

- 543KTetra, 106KPt
- 3-Stage R-K, Roe + MUSCL
- CFL=1.0, RSMOO=3
- $\epsilon_2 = 10^{-3}$
- Blocking: 108, 103, 126, 113, 119KTetra

Table 1: Timings for Duct Problem

`dxmar`	`dxsaf`	CPU (min)	Speedup
Usual		400	1.00
0.05	0.20	160	2.50
0.10	0.40	88	4.54
0.10	0.60	66	6.06
Block		140	2.85

F117 FIGHTER (1)

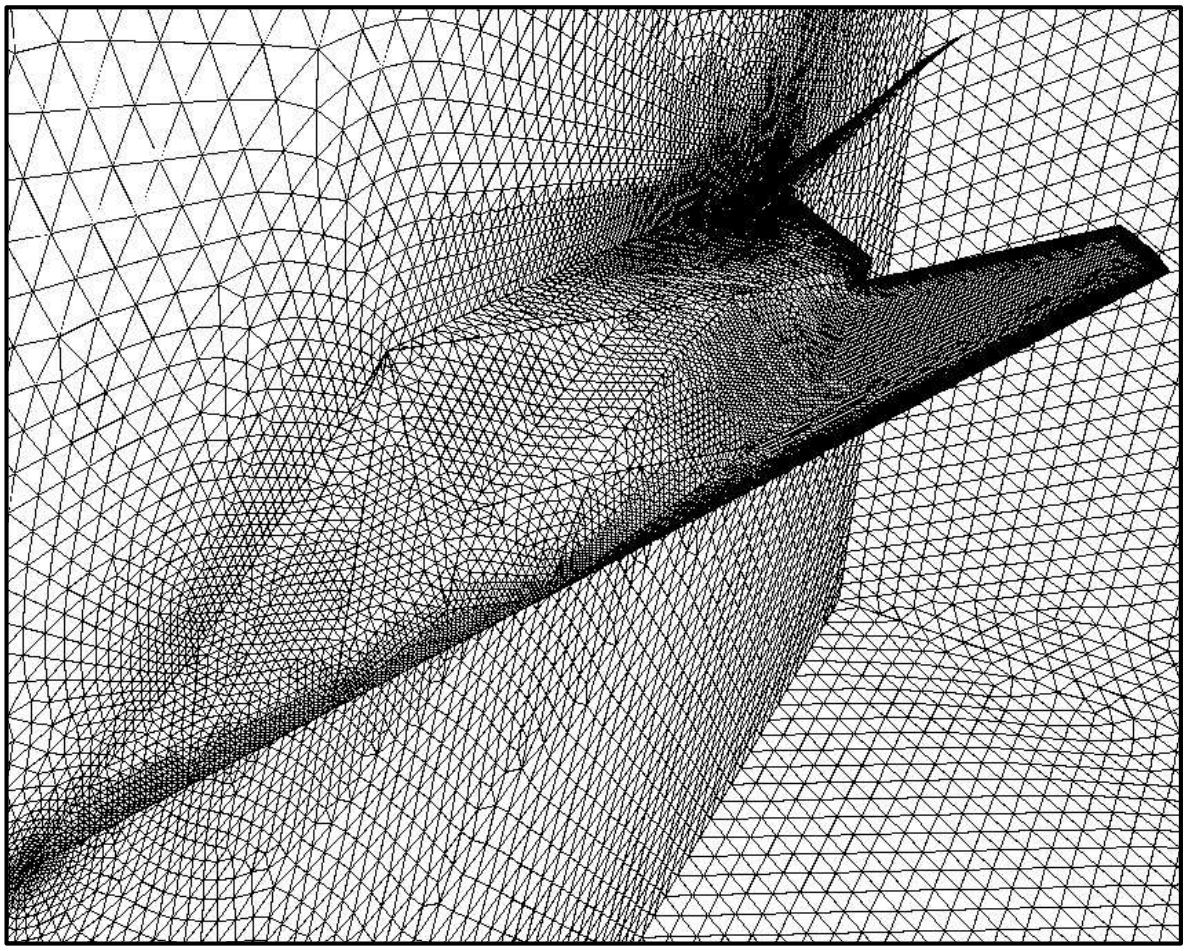


Figure 8a F117: Surface Mesh

F117 FIGHTER (2)

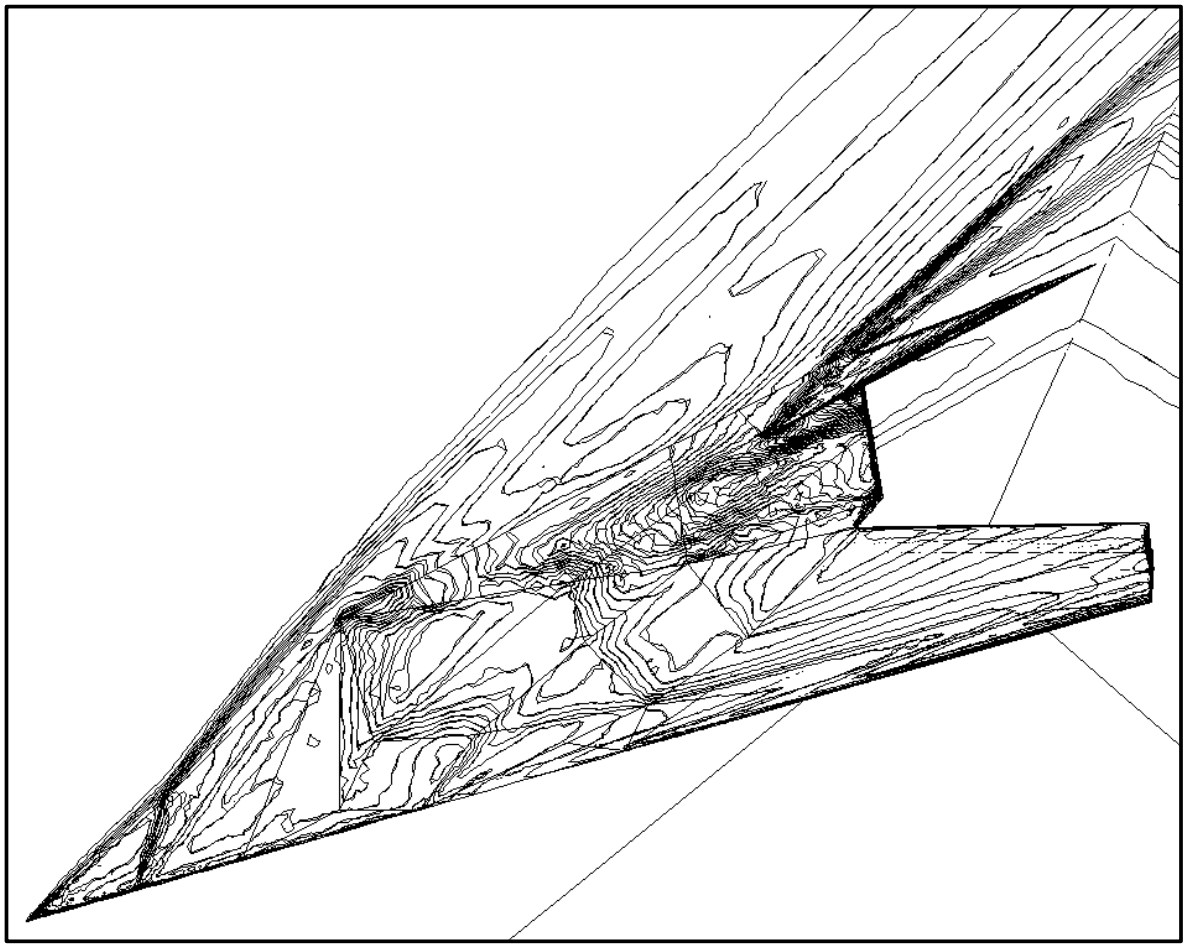


Figure 8b F117: Surface Mach-Number (Usual vs Marching)

F117 FIGHTER (3)

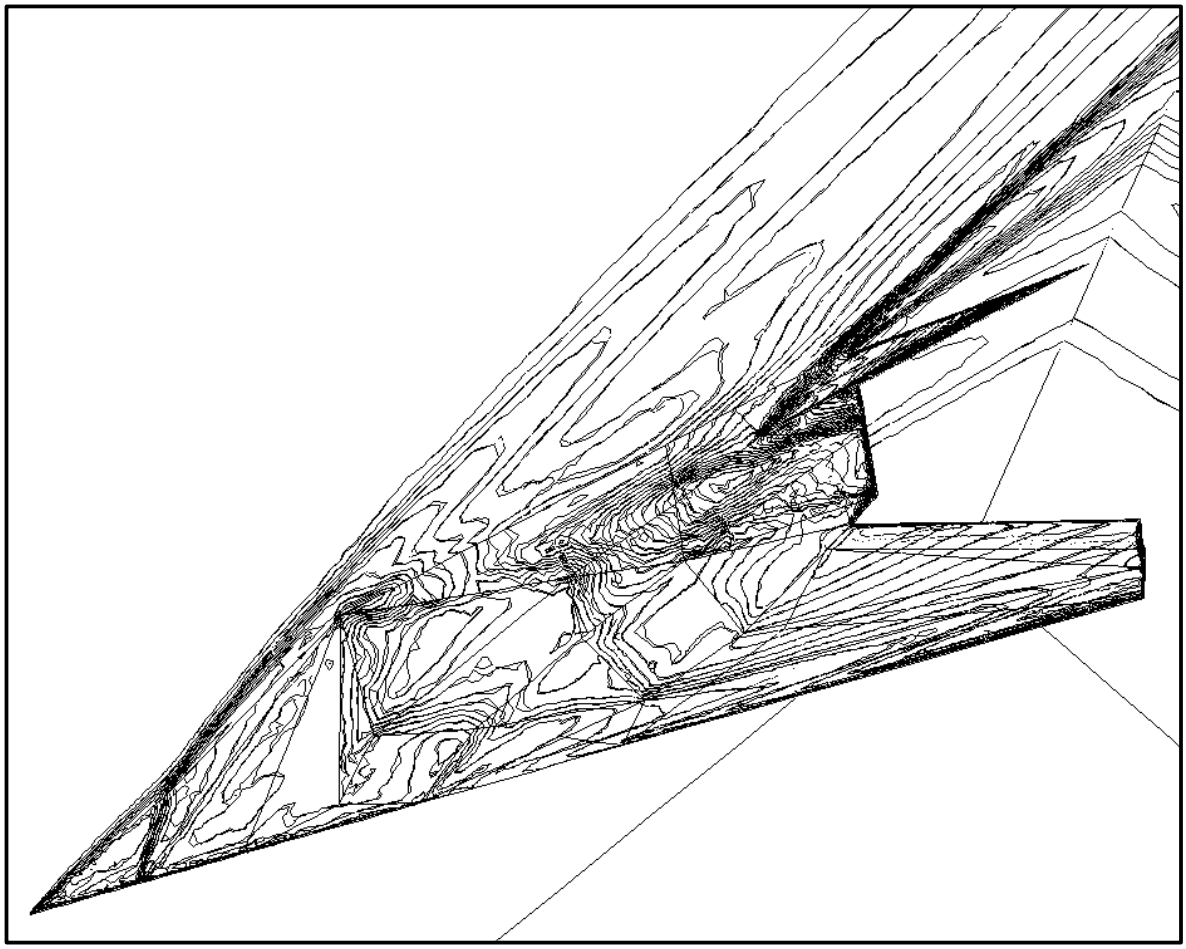


Figure 8c F117: Surface Mach-Number (Usual vs Block)

F117 FIGHTER (4)

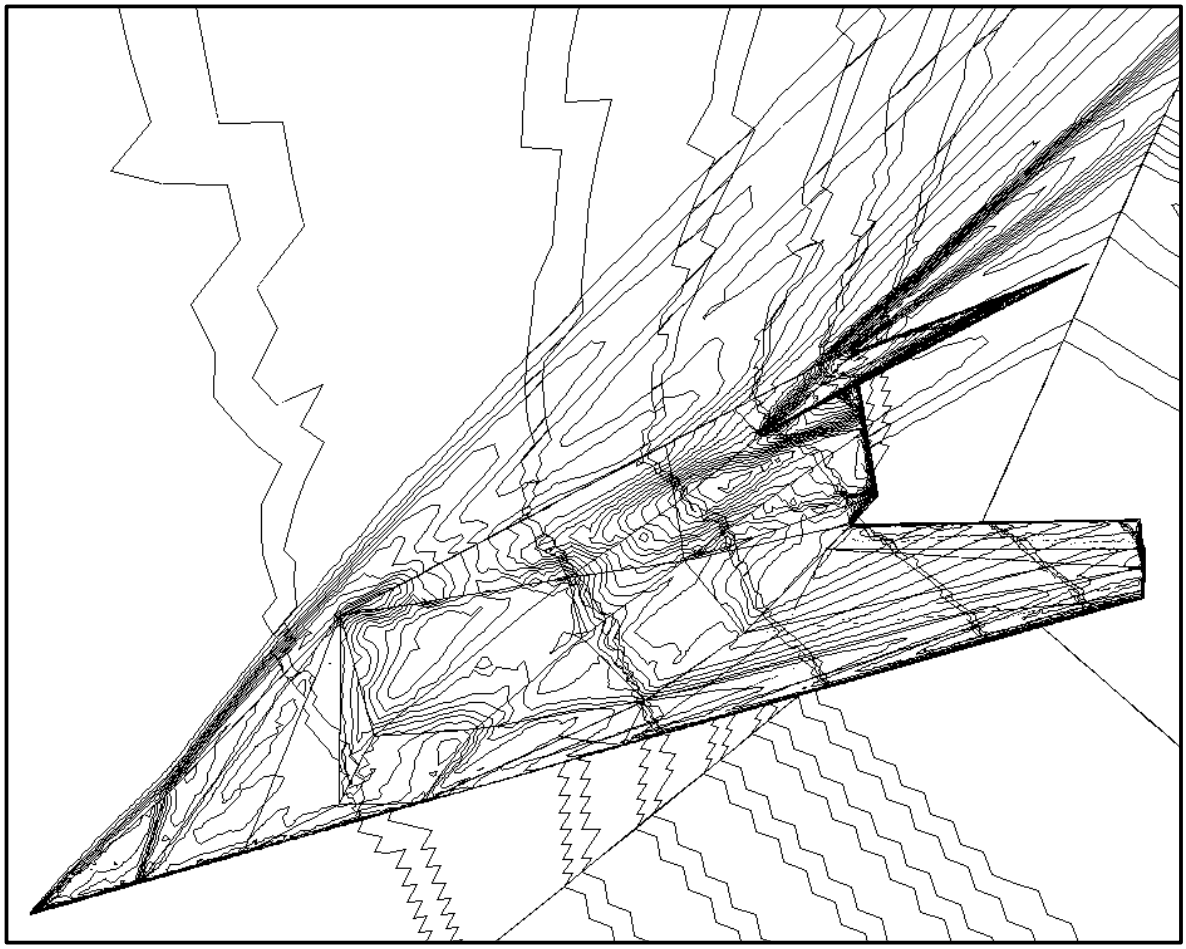


Figure 8d F117: Surface Mach-Number (Block

F117 FIGHTER (5)

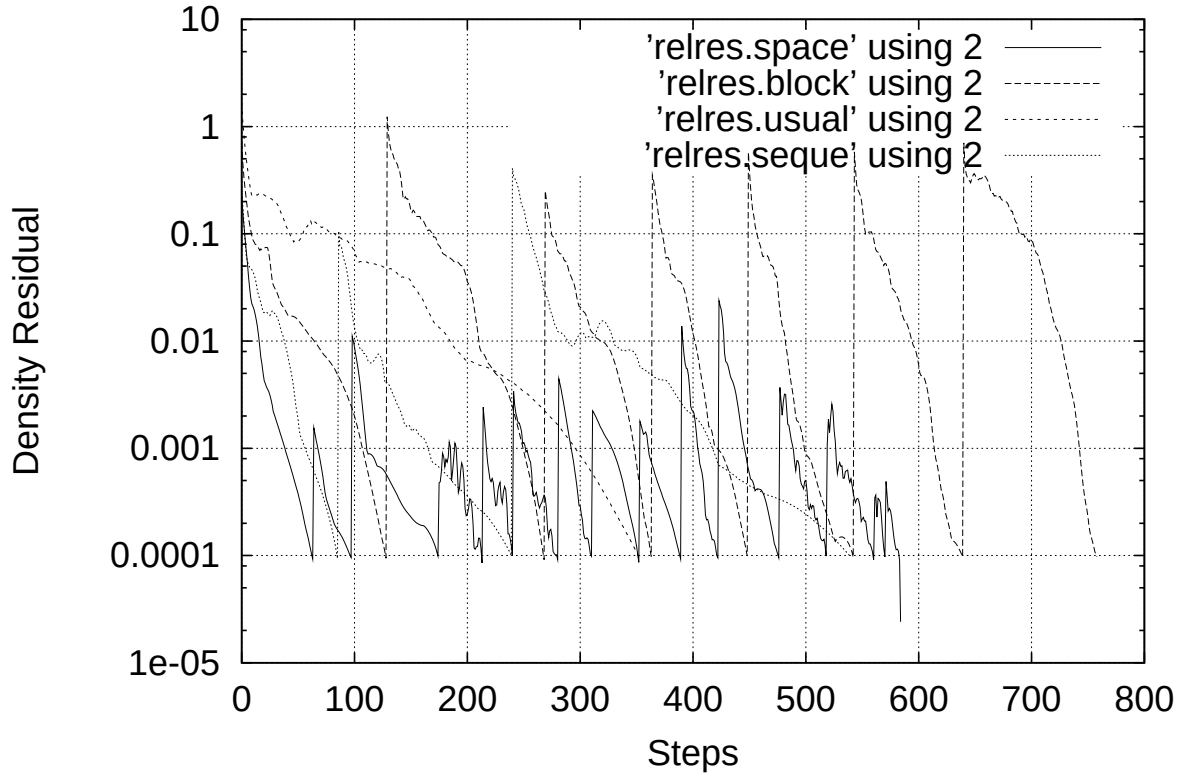


Figure 8e F117: Residuals

F117 FIGHTER (6)

- 2MTetra, 367KPt
- 3-Stage R-K, Roe + MUSCL
- CFL=1.0, RSMOO=3
- $\epsilon_2 = 10^{-4}$
- Blocking: 357, 323, 296, 348, 361, 385, 477Ktetra
- Grid Sequencing: 41, 281, 2056Ktetra

Table 2: Timings for F117 Problem

`dxmar`	`dxsaf`	CPU (min)	Speedup
Usual		611	1.00
Seque		518	1.17
10	30	227	2.69
20	30	218	2.80
Block	1	260	2.35

MOVING SUPERSONIC INLET (1)

- 543KTetra, 106KPt
- 3-Stage R-K, Roe + MUSCL
- CFL=1.0, RSMOO=3
- $\epsilon_2 = 10^{-3}$
- Movement of Inner Walls:

$$x_c = x_c^0 + a \cdot sin(\omega t)$$

  $x_c^0 = 4.2, a = 0.2, \omega = 0.05$
- Implicit Timestepping:
  - Nr. of Steps/Cycle: 40
  - $\Delta t = \pi \Rightarrow C \approx 300$

MOVING SUPERSONIC INLET (2)

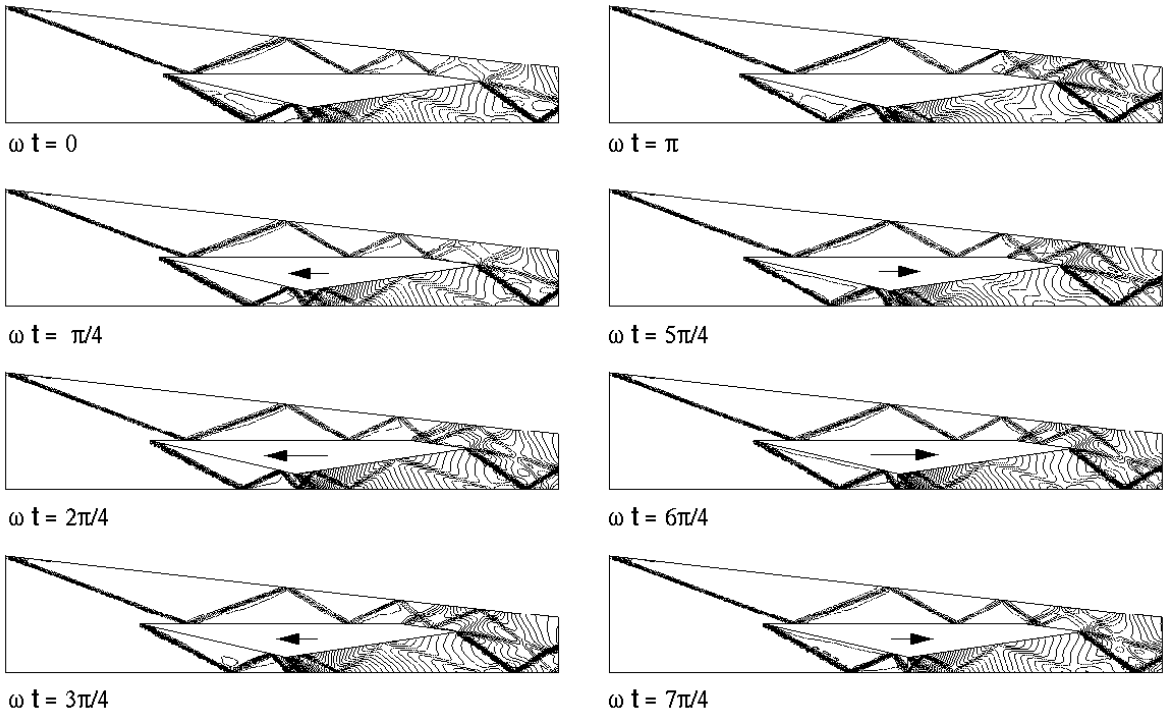


Figure 9 Moving Supersonic Inlet