Simulasi CFD Airfoil *Diamond* pada Mach 2 dengan Variasi Sudut Serang

Caesar Wiratama[1]
PT Tensor, Sleman, Yogyakarta, 55287, Indonesia

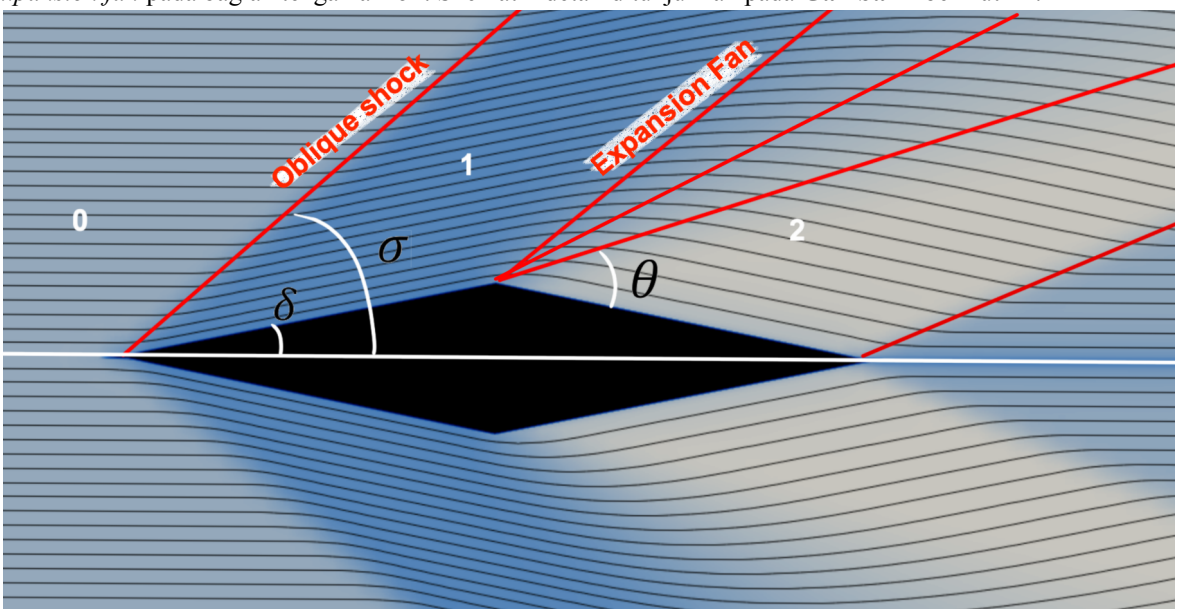
Pada pekerjaan ini, airfoil berbentuk diamond disimulasikan menggunakan metode *Computational Fluid Dynamics* (CFD) OpenFOAM dengan solver kompresibel pada Mach 2. Hasilnya kemudian dibandingkan dengan perhitungan analitis dengan asumsi *weak shock*. Hasil simulasi sesuai dengan perhitungan analisis dengan error hanya 0.127% untuk sudut gelombang kejut. Variasi sudut serang juga disimulasikan untuk melihat karakteristik aliran; gelombang kejut tajam (*oblique*) menghilang pada sudut serang di atas 8 derajat.

I. Nomenklature

δ	=	setengah sudut airfoil
σ	=	sudut gelombang kejut
Ma	=	Bilangan Mach
γ	=	Rasio kalor spesifik

II. Pengantar

Airfoil supersonik terdiri dari berbagai jenis, *double wedge* atau dikenal juga dengan *diamond* airfoil merupakan salah satu yang populer diantaranya dan banyak digunakan pada desain pesawat terbang. Aliran udara pada airfoil *diamond* memiliki karakteristik yang unik, seperti kemunculan gelombang kejut tajam (*oblique*) pada *leading edge* dan *expansion fan* pada bagian tengah airfoil. Skematik detail ditunjukkan pada **Gambar 1** berikut ini:



Gambar. 1 Skematik aliran pada sekitar airfoil diamond

[1] Application engineer.

Pada pekerjaan ini, airfoil *diamond* disimulasikan menggunakan metode *Computational Fluid Dynamics* (CFD) dan hasilnya dibandigkan dengan perhitungan analitis.

III. Metodologi

3.1. Solver dan Skema

Solver yang digunakan pada project ini adalah OpenFOAM CFD dengan modul solver Fluid, yaitu solver untuk *steady* atau *transient* turbulen, untuk aliran kompresibel dengan transfer kalor [1]. Meskipun kita tidak membutuhkan data terhadap waktu pada pekerjaan ini, skema transient PIMPLE (kombinasi dari skema SIMPLE dan PISO) digunakan karena permasalahan stabilitas konvergensi pada skema *steady*. Berikut adalah lampiran dari skema yang digunakan pada pekerjaan ini:

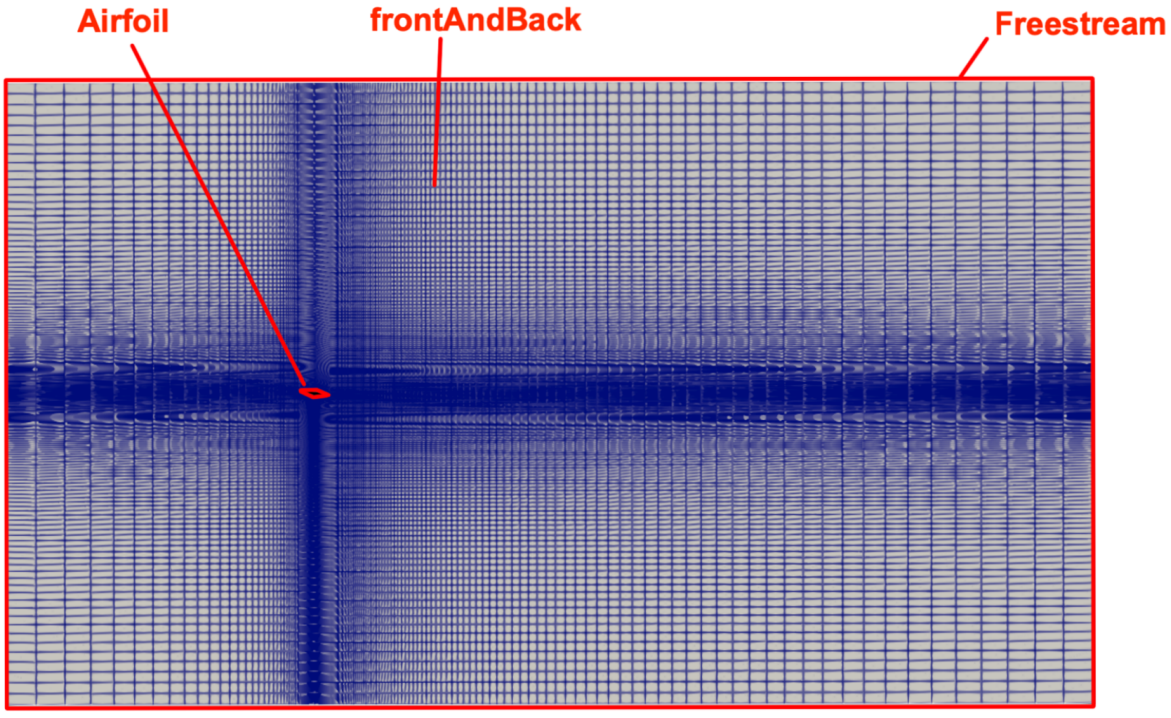
Tabel 1. Rangkuman skema komputasi pada simulasi CFD

Variable	Scheme	Description
Pressure	GAMG	*Generalized Geometric-Algebraic MultiGrid Solver*: yang memiliki karakteristik matriks simetris dan asimetris, smoother waktu running yang bisa dipilih, transportasi informasi yang efisien antar domain solusi, dan skala paralelisasi yang moderat.
Density, velocity, thermal energy	PBiCGStab	*Preconditioned bi-conjugate gradient*: yang memiliki karakteristik matriks simetri dan asimetri, precondiitioner waktu running yang bisa disesuaikan, dan skala paralelisasi yang baik.

Skema-skema di atas dipilih berdasarkan trade-off antara kecepatan dan stabilitas, karena sensitivitas dari kasus aliran kompresibel pada kasus ini. Dengan total iterasi 5000, menghasilkan nilai residual sebagai berikut: p = 1.453e-05, Ux = 3.129e-05, Uy = 1.858e-05, e = 2.552e-06, k = 2.860e-06, omega = 7.318e-05.

3.2. Kondisi Batas

Berikut adalah kondisi batas (*boundary conditions*) yang digunakan pada pekerjaan ini:



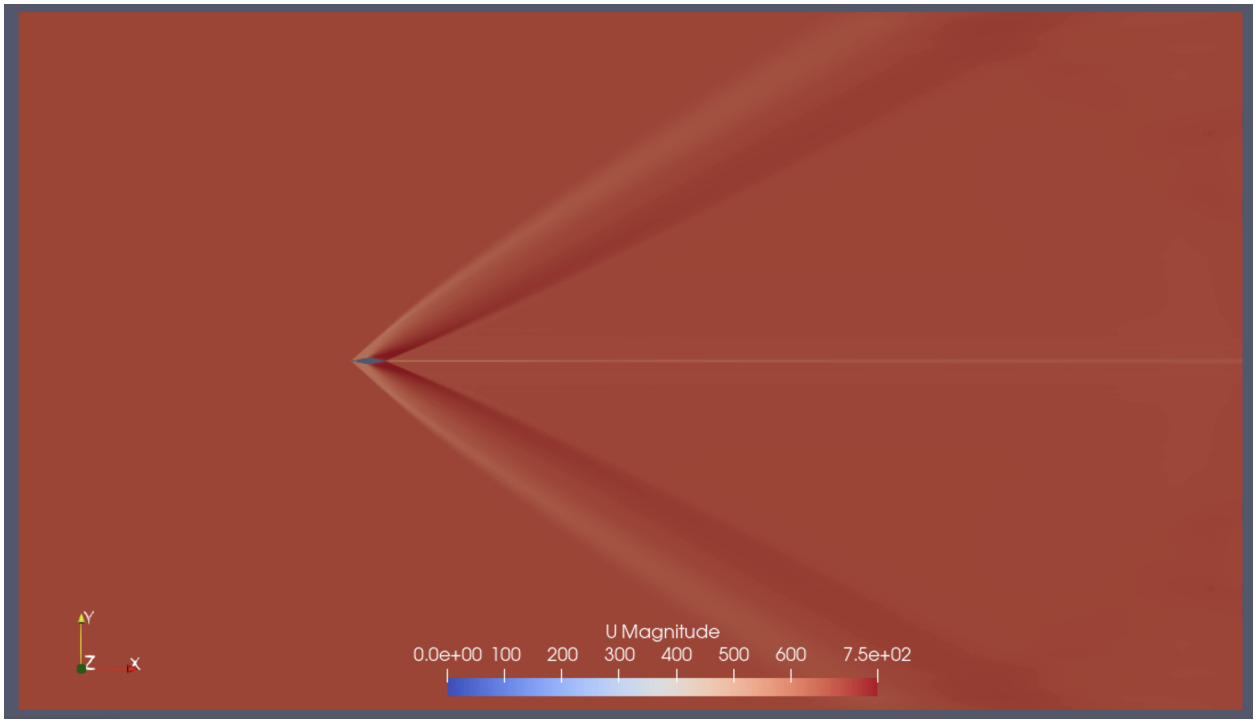
Gambar. 2 Lokasi kondisi batas

Tabel 2. Rangkuman Kondisi Batas

Lokasi	**Pressure**	**Velocity**
FreeStream	freestreamPressure	freestreamVelocity
Airfoil	zeroGradient	noSlip
FrontAndBack	empty	empty

3.3. Griding dan *grid independence test*

Pada pekerjaan ini, airfoil *diamond* dengan panjang chord 1 meter dan ketebalan 0.2 meter dimodelkan di dalam domain komputasi dengan ukuran 35 meter x 20 meter dengan cara *trial error* hingga sudut *shock-wave* tidak berubah lagi terhadap ukuran domain, serta "kekuatan" *shock wave* mulai berkurang yang mengindikasikan gradien tekanan tidak lagi mempengaruhi pola aliran obyek sekitarnya.

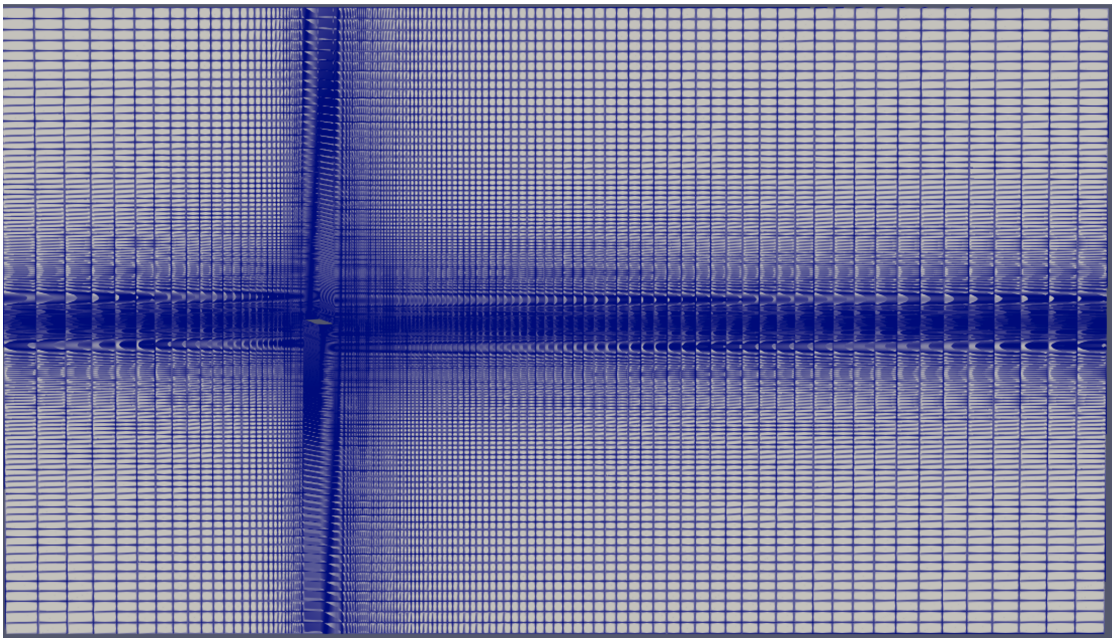


Gambar. 3 Kontur kecepatan pada domain fluida secara penuh

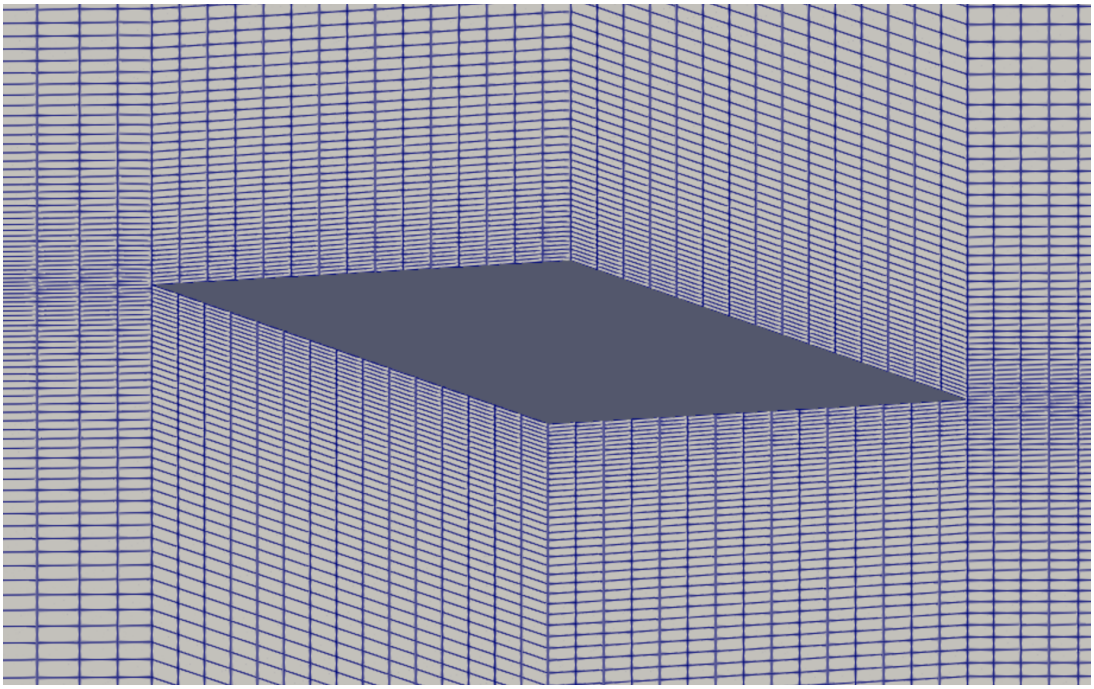
Grid di buat menggunakan program blockMesh di dalam OpenFOAM dengan detail sebagai berikut:

- 32 elemen searah chord
- 128 elemen searah aliran
- 256 elemen searah vertikal

dengan 30-*expansion ratio* kearah horizontal and 80 *expansion ratio* ke arah vertikal.



Gambar. 3 Grid dari domain fluida (tampak penuh)



Gambar. 4 Grid dari domain fluida (tampak sekitar airfoil)

Fenomena yang terjadi pada simulasi ini melibatkan bilangan Reynold yang tinggi, sehingga penulis mengasumsikan *boundary layer* yang terjadi sangatlah tipis, yang mana perhitungan dari kritera y+ pada mesh menjadi kurang relevan (untuk perhitungan sudut gelombang kejut).

Kemudian, mesh dibuat menggunakan parameterisasi berdasarkan ukuran mesh kearah chord dengan:

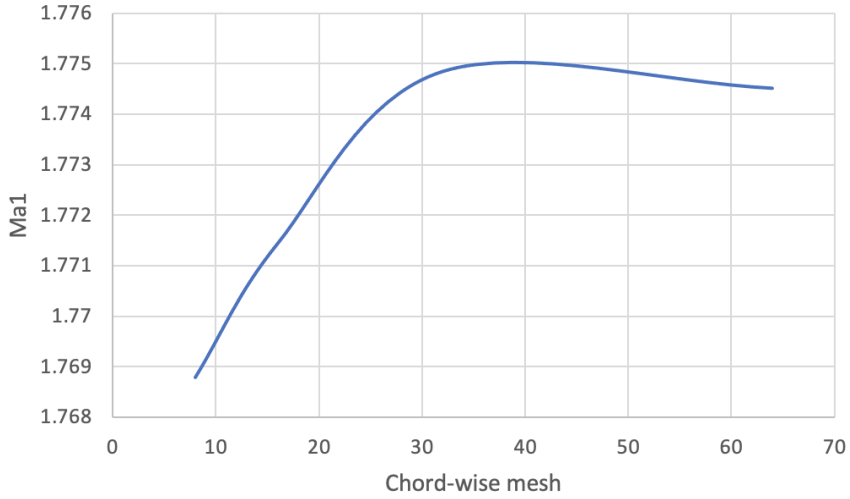
- Jumlah mesh ke arah vertikal = 16*chord mesh
- Jumlah mesh ke arah horizontal = 8* chord mesh

Yang diperoleh berdasarkan proporsionalitas ukuran chord dan domain serta *expansion ratio* yang digunakan. Dari parameter ini, dilakukan pula grid independence test dengan memvariasikan ukuran mesh searah chord terhadap nilai bilangan Mach pada daerah (1) pada Gambar 1 untuk sudut serang 0 derajat. Berikut adalah rangkuman dari *grid independence test*:

Tabel 2. Rangkuman hasil *Grid independence test*

Chord-wise mesh	Velocity (m/s)	Ma1
8	587.24	1.76879518
16	588.12	1.77144578
32	589.25	1.7748494
64	589.14	1.77451807

Kemudian, hasil pada **Tabel 2** diperlihatkan pada **Gambar 5** untuk melihat trend dari perubahan hasil solusi terhadap ukuran mesh. Mengubah resolusi chord dari 32 ke 64 tidak mengubah secara signifikan hasil solusi, sehingga 32 dipilih berdasarkan trade-off antara akurasi dan efisiensi.



Gambar. 5 Hasil *grid independence test*

3.3. Persamaan Analitis

Hasil simulasi kemudian dibandingkan dengan persamaan analitis dengan asumsi *weak shock* berbasiskan kemiringan obyek dengan persamaan sebagai berikut [2]:

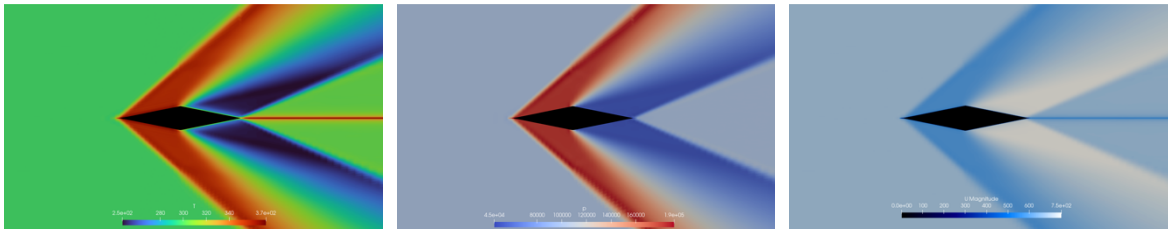
$$\tan tan\,\delta = 2\cot cot\,\sigma\left[\frac{Ma_0^2 sin^2\sigma - 1}{Ma_0^2(\gamma + coscos\,2\sigma) + 2}\right] \quad (1)$$

δ adalah setengah sudut dari airfoil, σ adalah sudut gelombang kejut, Ma0 adalah bilangan Mach *free-stream*, dan γ adalah rasio kalor spesifik.

IV. Hasil Simulasi dan Diskusi

4.1. Perbandingan dengan perhitungan analitis

Simulasi dengan sudut serang 0 derajat dibandingkan hasil perhitungan sudut gelombang kejut-nya dengan perhitungan analitis.



(a) (b) (c)

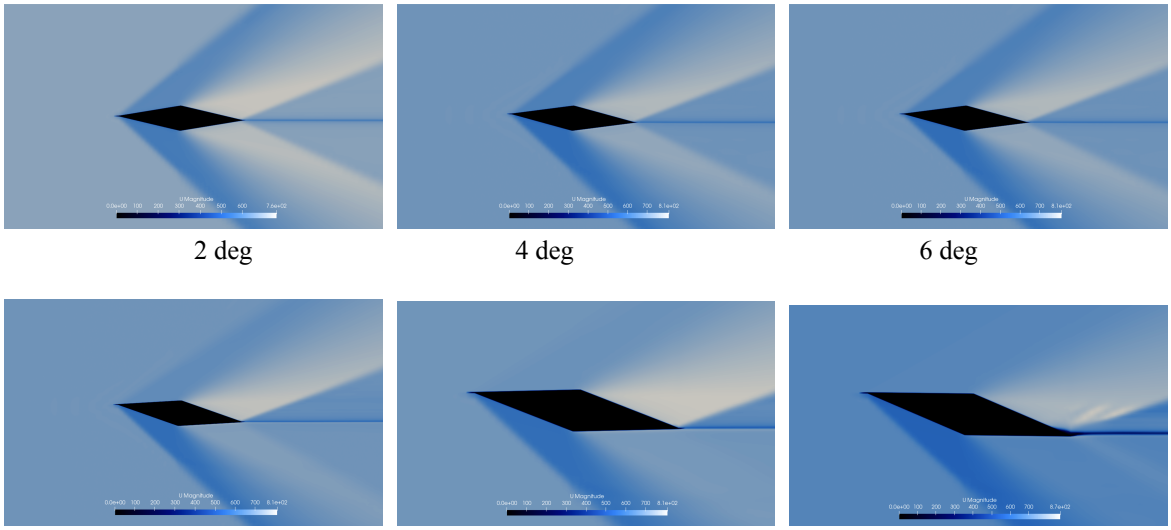
Fig. 6. Kontur hasil simulasi untuk 0 derajat dari (a) temperatur, (b) tekanan, (c) kecepatan

Dari hasil di atas, kita dapat mengukur sudut gelombang kejut, dengan hasil sebesar 41 derajat, dengan sudut setengah airfoil adalah 11 derajat, kemudian memasukkanya ke dalam persamaan (1); dengan bagian kiri persamaan adalah nilai eksak yang sudah ditentukan dari input geometri dan bagian kiri dari persamaan adalah nilai yang dihitung berdasarkan hasil simulasi, kemudian kedua sisi persamaan dibandingkan untuk melihat error nya. Berikut adalah hasil perbandingannya:

Tabel 3. Perbandingan persamaan (1) dengan hasil simulasi

Sisi kiri	Sisi kanan	Error
0.19438039	0.19413396	0.127%

Dengan setingan yang sama, kita dapat juga melakukan simulasi dengan variasi sudut serang dengan mentransformasikan blockMesh dari koordinat airfoil. Berikut adalah hasil kontur kecepatan pada sudut serang 2,4,6,8,10, dan 12 derajat:



2 deg 4 deg 6 deg

8 deg 10 deg 12 deg

Gambar. 7 Kontur kecepatan pada berbagai sudut serang

Dari **Gambar 7**, kita dapat melihat gelombang kejut tajam memiliki sudut yang hampir sama pada setiap sudut serang, namun kekuatanya cenderung berkurang dan hampir tidak terlihat pada sudut serang lebih dari 8 derajat, serta tidak terlihat sama sekali pada sudut serang 12 derajat.

V. Kesimpulan

Simulasi di atas menunjukkan hasil yang memiliki kecocokan antara simulasi dan analitis, dengan error hanya 0.127 % pada perhitungan sudut gelombang kejut. Dari variasi AoA, gelombang kejut tajam masih dapat diamati hingga sudut serang 8 derajat.

Ucapan Terimakasih

Pekerjaan ini merupakan salah satu program dari pttensor.com untuk berkontribusi pada komunitas pengguna CFD memperoleh referensi yang memadahi. Anda dapat mengakses skrip dari OpenFOAM di akun GitHub penulis, yaitu Caesar3142 dengan tautan: https://github.com/Caesar3142/diamond_wing_compressible.git

Referensi

[1] openfoam.com

[2] Anderson, J.D. "*Fundamentals of Aerodynamics: Sixth Edition*". McGraw-Hill series in aeronautics and aerospace engineering. 2011.

APPENDIX: OPENFOAM SCRIPTS

BOUNDARY CONDITIONS

Velocity

```
internalField   uniform (680 0 0);

boundaryField
{
   freestream
   {
      type          freestreamVelocity;
      freestreamValue $internalField;
      value         $internalField;
   }

   airfoil
   {
      type          noSlip;
   }

   frontAndBack
   {
      type          empty;
   }
}
```

Pressure

```
internalField   uniform 1e5;

boundaryField
{
   freestream
   {
      type          freestreamPressure;
      freestreamValue uniform $pOut;
   }

   airfoil
   {
      type          zeroGradient;
   }

      frontAndBack
   {
      type          empty;
   }
}
```

Temperature

```
Tinlet        298;

dimensions     [0 0 0 1 0 0 0];

internalField   uniform $Tinlet;

boundaryField
{
   freestream
   {
      type          inletOutlet;
      inletValue     uniform $Tinlet;
      value         $inletValue;
   }
```

```
    airfoil
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}
```

PHYSICAL PROPERTIES

```
thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

{
    specie
    {
        molWeight   28.9;
    }
    thermodynamics
    {
        Cp          1005;
        Hf          0;
    }
    transport
    {
        mu          1.82e-05;
        Pr          0.71;
    }
}
```

SYSTEMS

BlockMesh

```
convertToMeters 1;
a 0;
C 0.5; //half chord
t 0.1; //half thickness

cosC #calc "$C*cos(degToRad($a))";
sinC #calc "$C*sin(degToRad($a))";
cost #calc "$t*cos(degToRad($a))";
sint #calc "$t*sin(degToRad($a))";

mincosC #calc "-$C*cos(degToRad($a))";
minsinC #calc "-$C*sin(degToRad($a))";
mincost #calc "-$t*cos(degToRad($a))";
minsint #calc "-$t*sin(degToRad($a))";

plusY 10;
minY -10;

plusX 25;
minX -10;
```

```
vertices
(
   ($minX $minY 0) //0
   ($mincosC $minY 0) //1
   ($minsint $minY 0) //2
   ($cosC $minY 0) //3
   ($plusX $minY 0) //4
   ($plusX 0 0) //5
   ($cosC $minsinC 0) //6
   ($minsint $mincost 0) //7
   ($mincosC $sinC 0) //8
   ($minX 0 0) //9
   ($minX $plusY 0) //10
   ($mincosC $plusY 0) //11
   ($sint $plusY 0) //12
   ($cosC $plusY 0) //13
   ($plusX $plusY 0) //14
   ($sint $cost 0) //15

   ($minX $minY 0.02) //0
   ($mincosC $minY 0.02) //1
   ($minsint $minY 0.02) //2
   ($cosC $minY 0.02) //3
   ($plusX $minY 0.02) //4
   ($plusX 0 0.02) //5
   ($cosC $minsinC 0.02) //6
   ($minsint $mincost 0.02) //7
   ($mincosC $sinC 0.02) //8
   ($minX 0 0.02) //9
   ($minX $plusY 0.02) //10
   ($mincosC $plusY 0.02) //11
   ($sint $plusY 0.02) //12
   ($cosC $plusY 0.02) //13
   ($plusX $plusY 0.02) //14
   ($sint $cost 0.02) //15
);

c 15; //half chord mesh
y #calc "8 * $c";
x1 #calc "2 * $c";
x2 #calc "6 * $c";


blocks
(
   hex (0 1 8 9 16 17 24 25) ($x1 $y 1) simpleGrading (0.04 0.0125 1) //1
   hex (1 2 7 8 17 18 23 24) ($c $y 1) simpleGrading (1 0.0125 1) //2
   hex (2 3 6 7 18 19 22 23) ($c $y 1) simpleGrading (1 0.0125 1) //3
   hex (3 4 5 6 19 20 21 22) ($x2 $y 1) simpleGrading (30 0.0125 1) //4
   hex (6 5 14 13 22 21 30 29) ($x2 $y 1) simpleGrading (30 80 1) //5
   hex (15 6 13 12 31 22 29 28) ($c $y 1) simpleGrading (1 80 1) //6
   hex (8 15 12 11 24 31 28 27) ($c $y 1) simpleGrading (1 80 1) //7
   hex (9 8 11 10 25 24 27 26) ($x1 $y 1) simpleGrading (0.04 80 1) //8
);

boundary
(
   airfoil
   {
      type wall;
      faces
      (
         (8 15 31 24)
         (15 6 22 31)
         (8 7 23 24)
         (7 6 22 23)
```

```
    );
  }
  freestream
  {
    type patch;
    faces
    (
      (0 9 25 16)
      (9 10 26 25)
      (0 1 17 16)
      (1 2 18 17)
      (2 3 19 18)
      (3 4 20 19)
      (10 11 27 26)
      (11 12 28 27)
          (12 13 29 28)
      (13 14 30 29)
            (14 5 21 30)
            (4 5 21 20)
    );
  }

  frontAndBack
  {
    type empty;
    faces
    (
      (0 1 8 9)
      (1 2 7 8)
      (2 3 6 7)
      (3 4 5 6)
      (9 8 11 10)
      (8 15 12 11)
      (15 6 13 12)
      (6 5 14 13)

      (16 17 24 25)
      (17 18 23 24)
      (18 19 22 23)
      (19 20 21 22)
      (25 24 27 26)
      (24 31 28 27)
      (31 22 29 28)
      (22 21 30 29)
    );
  }
);
```

Schemes

```
ddtSchemes
{
  default       Euler;
}

gradSchemes
{
  default       Gauss linear;

  limited       cellLimited Gauss linear 1;
  grad(U)       $limited;
  grad(k)       $limited;
  grad(omega)    $limited;
}

divSchemes
```

```
{
    default           none;

    div(phi,U)        Gauss linearUpwind limited;

    turbulence        Gauss linearUpwind limited;
    energy            Gauss linearUpwind limited;

    div(phi,k)        $turbulence;
    div(phi,omega)    $turbulence;

    div(phi,e)        $energy;
    div(phi,K)        $energy;
    div(phi,Ekp)      $energy;
    div(phi,(p|rho))  $energy;

    div((phi|interpolate(rho)),p) Gauss upwind;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default       Gauss linear corrected;
}

interpolationSchemes
{
    default       linear;
}

snGradSchemes
{
    default       corrected;
}

wallDist
{
    method meshWave;
}
```

Solution

```
solvers
{
    p
    {
        solver        GAMG;
        smoother      GaussSeidel;
        tolerance     1e-6;
        relTol        0.01;
    }

    pFinal
    {
        $p;
        relTol        0;
    }

    "(rho|U|k|omega|e)"
    {
        solver        PBiCGStab;
        preconditioner  DILU;
        tolerance     1e-6;
        relTol        0.1;
    }
```

```
    "(rho|U|k|omega|e)Final"
    {
        $U;
        relTol          0;
    }
}

SIMPLE
{
    residualControl
    {
        p               1e-4;
        U               1e-4;
        "(k|omega|e)"   1e-4;
    }

    nNonOrthogonalCorrectors 0;
}

PIMPLE
{
    nCorrectors             2;
    nNonOrthogonalCorrectors 1;
    nOuterCorrectors        1;
}

relaxationFactors
{
    equations
    {
        ".*"     1;
    }
}
```