RAE 2822 Drag Divergence

James Grisham

December 2, 2013

Contents

List of Figures

Chapter 1

Introduction and Background

1.1 Background

The RAE 2822 is a supercritical airfoil meant to allow for more efficient transonic cruise by prolonging the drag divergence associated with wave drag to a higher Mach number. This airfoil was chosen for CFD simulations because of the availability of experimental data with which computational data can be compared [1]. For the simulations contained in this report, the CFD data were compared with experimental, wind tunnel data at $M = 0.745$, $\alpha = 3.19°$, and $Re_c = 2.7 \times 10^6$. The ratio of the max thickness to chord is 0.121.

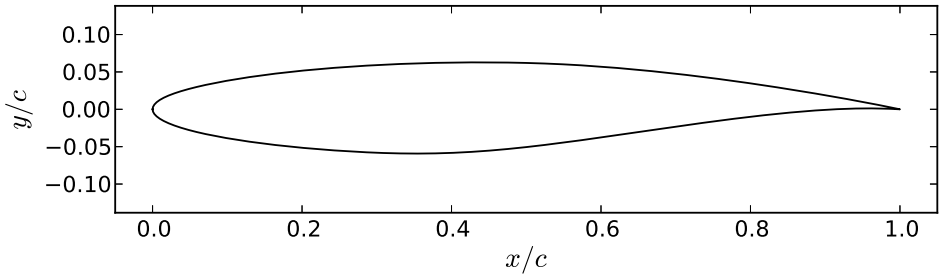


Figure 1.1. RAE 2822 airfoil profile.

1.2 Computational Capabilities

The parallel capabilities of the Lonestar Linux cluster at the Texas Advanced Computing Center (TACC) were used to run approximately 100 CFD simulations.

1.2.1 Grid generation

The AFLR2 grid generator from Mississippi State University was used to generate two-dimensional unstructured grids [2, 3]. This software uses advancing-front/local-reconnection schemes to generate high-quality, unstructured grids. Several programs from NASA Langley Research Center were also

used to wrap the AFLR2 grid generator. A Python script was developed to create the geometry files required by the Langley-developed wrappers (see the appendix for the code).

1.2.2 Flow solver

The FUN2D flow solver was used for simulations at various Mach numbers with and without viscosity [4]. FUN2D is a node-based, finite volume code that has capabilities too broad to be completely covered here. See the FUN3D website for more information. For this work, the steady, two-dimensional flow solver was used for inviscid and viscous flow solves. A Python script was developed that allowed the user to submit batch jobs to the queue (see appendix for code listing). Inputs to the script are the flowfield variable to increment and a vector of the desired values. The script creates the directories which are named according to the desired value, copies FUN2D input files, links the grid, increments the necessary values and submits the job to the queue.

1.2.3 Post-processing

Post-processing tasks were accomplished using Python and Tecplot. A script was developed to create and save all the necessary line plots (see appendix). Tecplot was used for more detailed flowfield visualization including contour plots of flow variables and streamlines.

References

[1] "Experimental Database for Computer Program Assessment," AGARD Advisory Report No. 138, 1979.

[2] Marcum, D.L., and Weatherhill, N.P., "Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection," AIAA Journal, Vol. 33, No. 9, pp. 1619-1625, 1995.

[3] Marcum, D.L., "Unstructured Grid Generation Using Automatic Point Insertion and Local Reconnection," The Handbook of Grid Generations, editted by J.F. Thompson, B. Soni, and N.P. Weatherhill, CRC Press, 1998.

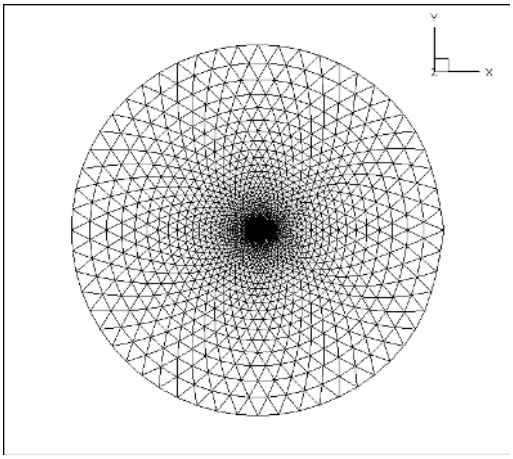
[4] NASA Langley Research Center, `fun3d.larc.nasa.gov`.

Chapter 2

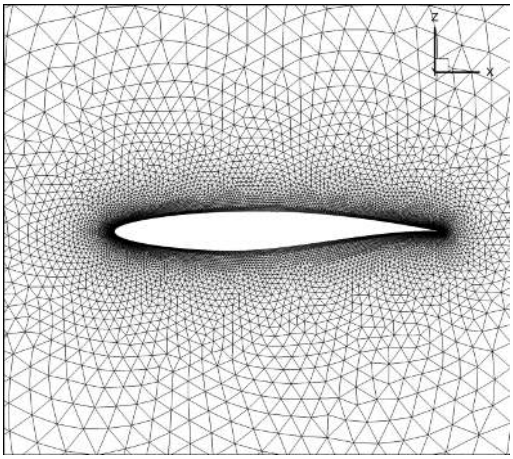
Inviscid Drag Divergence

2.1 Grid Convergence Study

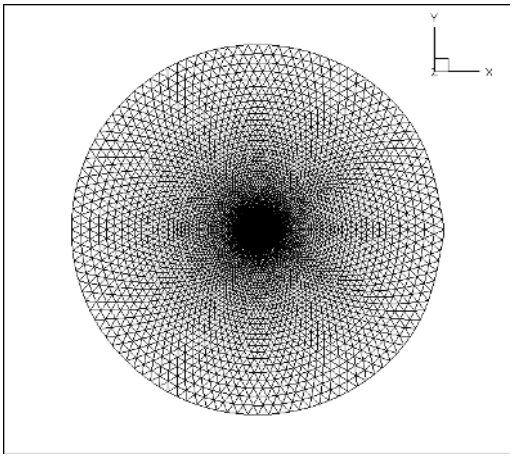
2.1.1 Grids



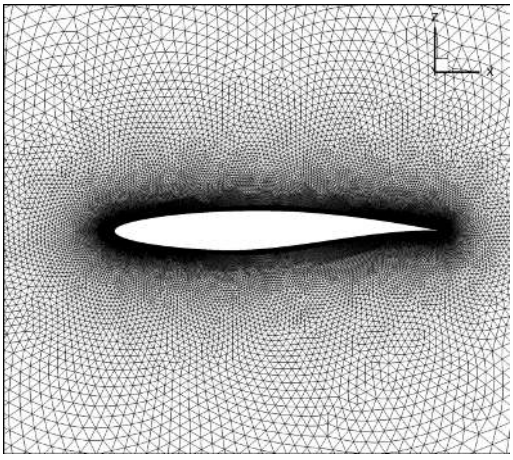
(a) Coarse grid.



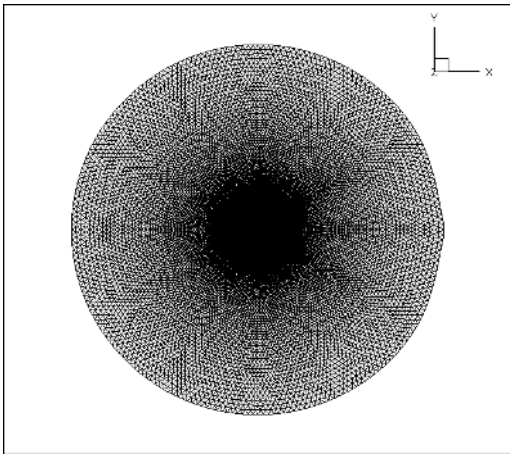
(b) Coarse grid close-up.



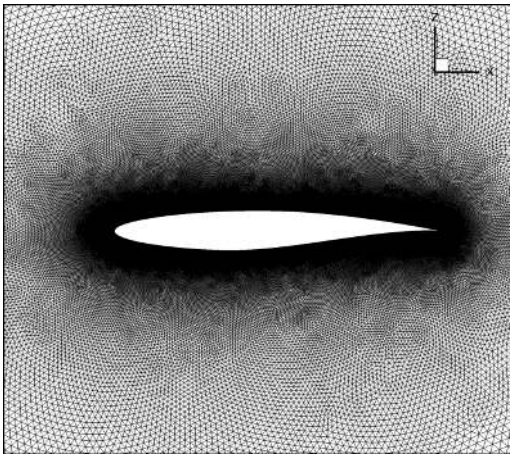
(c) Medium grid.



(d) Medium grid close-up.



(e) Fine grid.



(f) Fine grid close-up.

Figure 2.1. Computational meshes.

2.1.2 Results

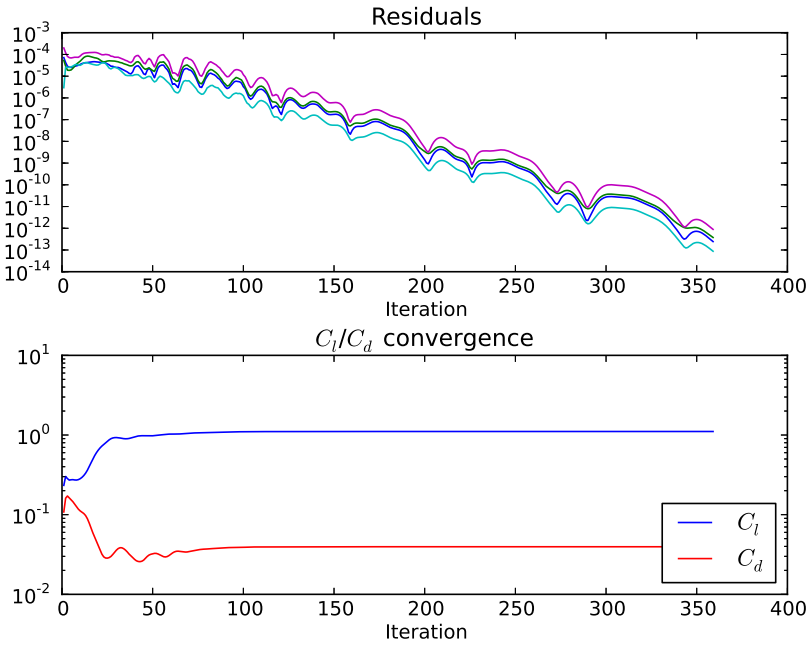


Figure 2.2. Coarse grid residuals.

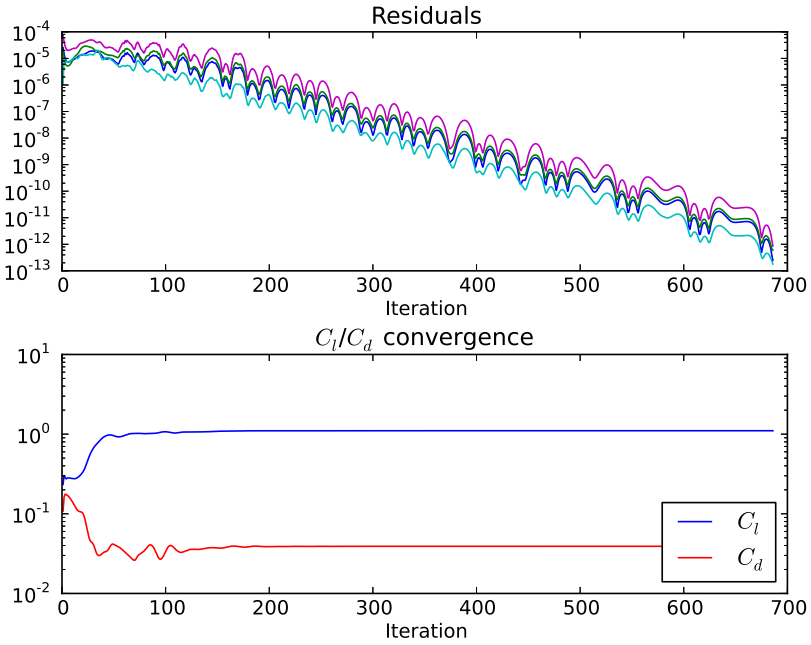


Figure 2.3. Medium grid residuals.

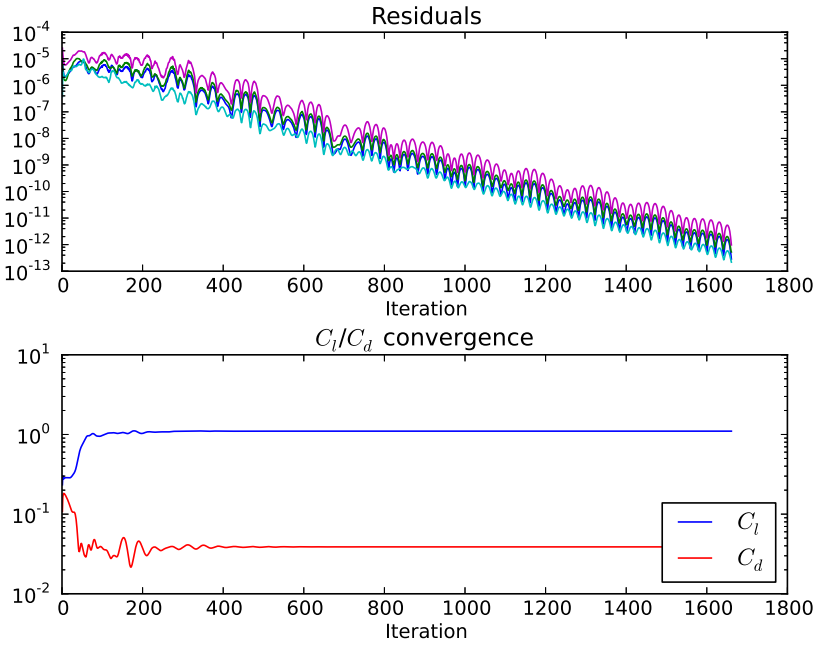


Figure 2.4. Fine grid residuals.

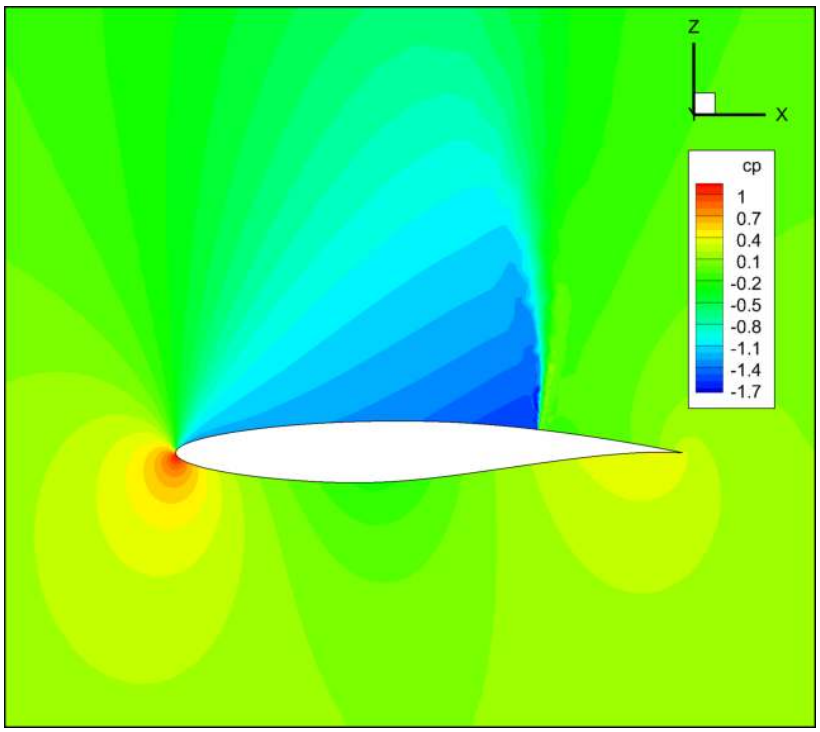


Figure 2.5. Coarse grid contours of C_p.

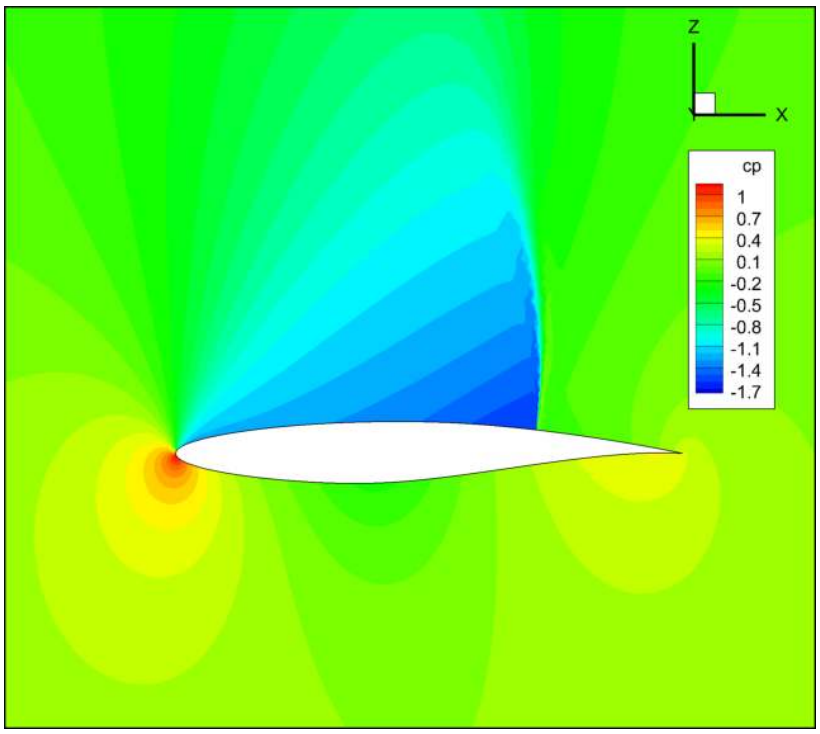


Figure 2.6. Medium grid contours of c_p.

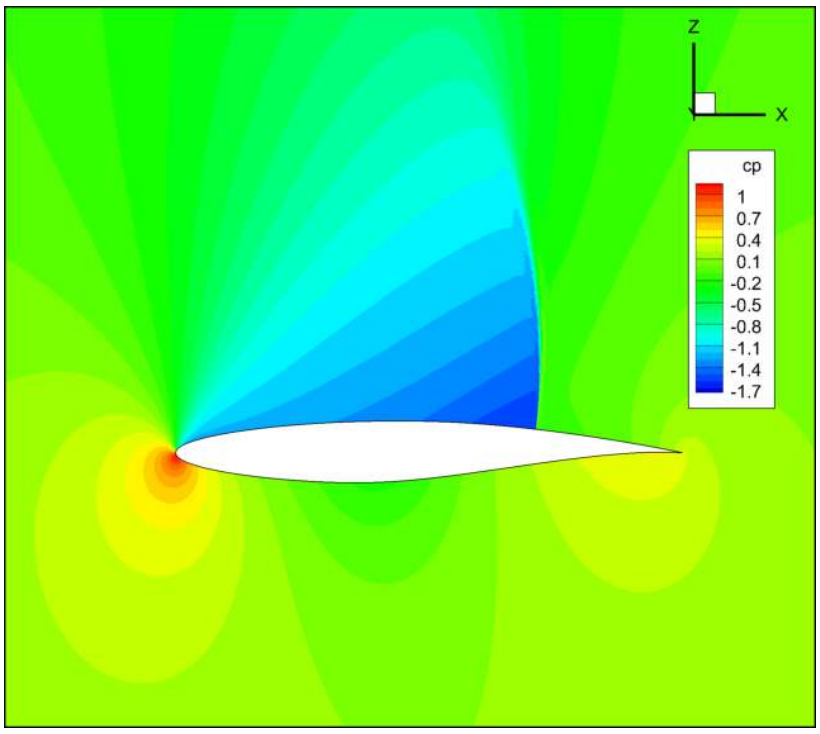


Figure 2.7. Fine grid contours of c_p.

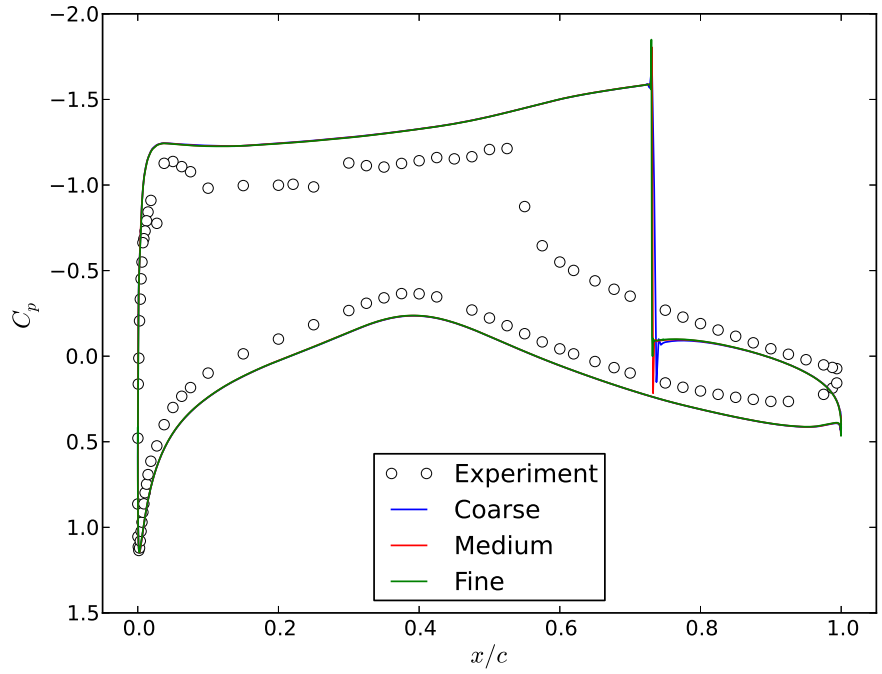


Figure 2.8. c_p comparison.

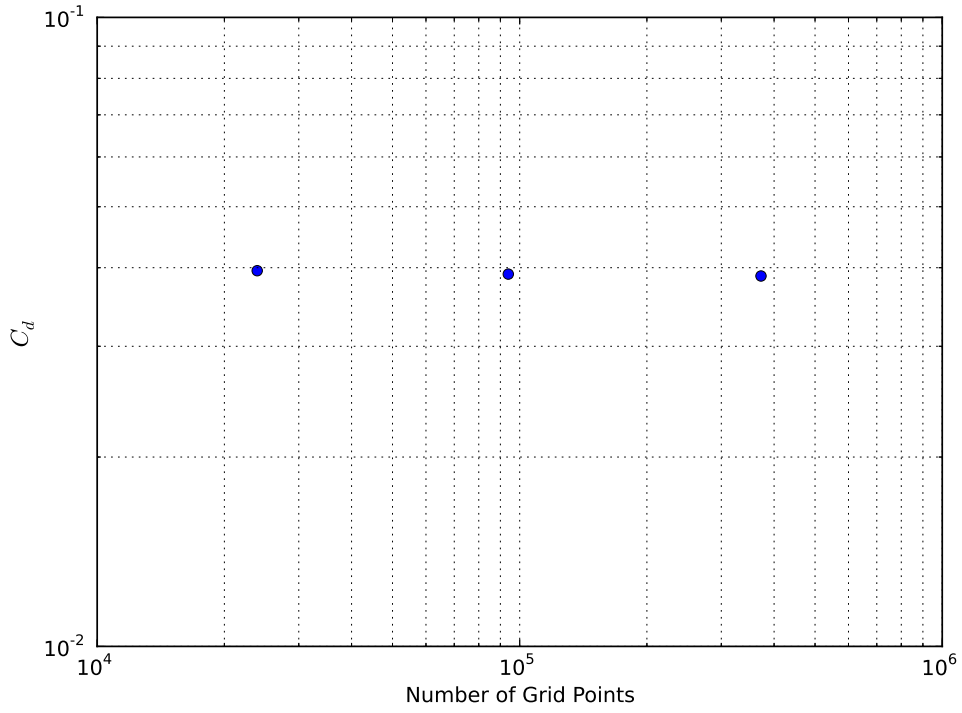


Figure 2.9. C_d vs number of grid points.

Because the main concern is the surface pressure distribution, and the lack of sensitivity of the drag coefficient to grid size, the coarse grid will be used for simulations.

2.2 Drag divergence

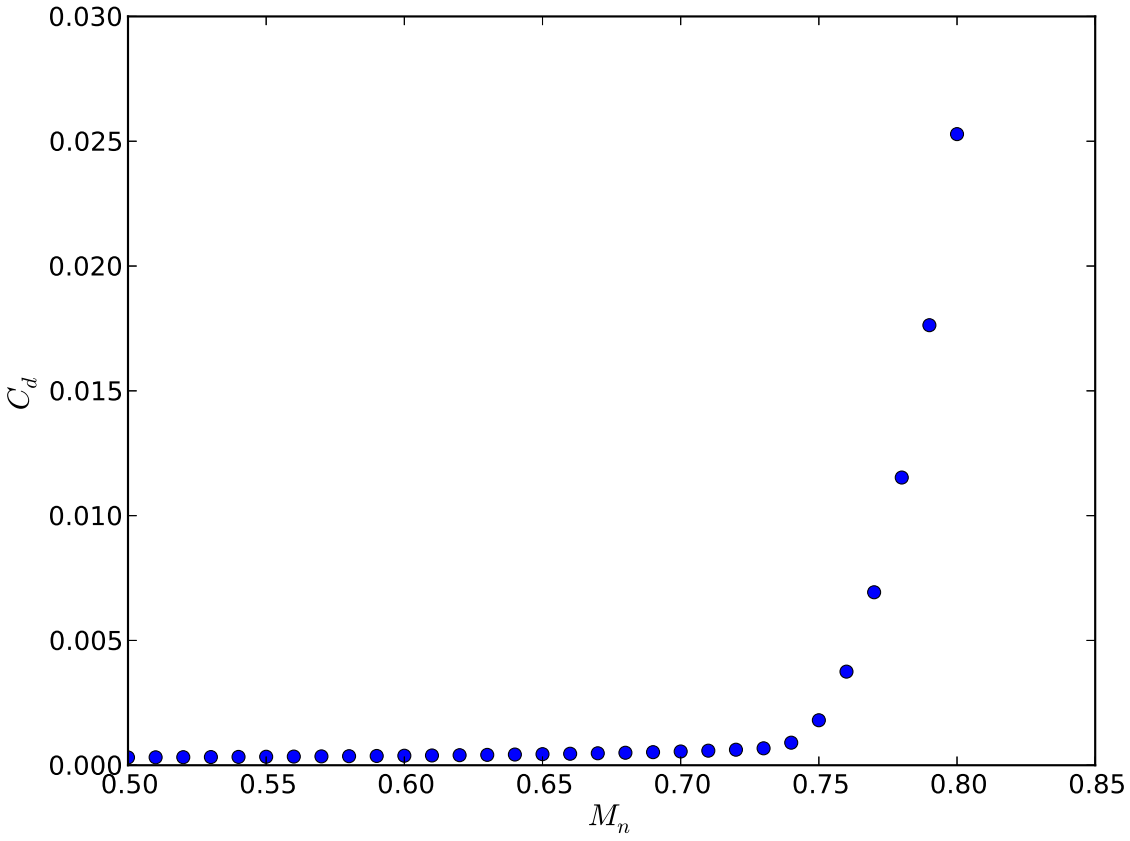
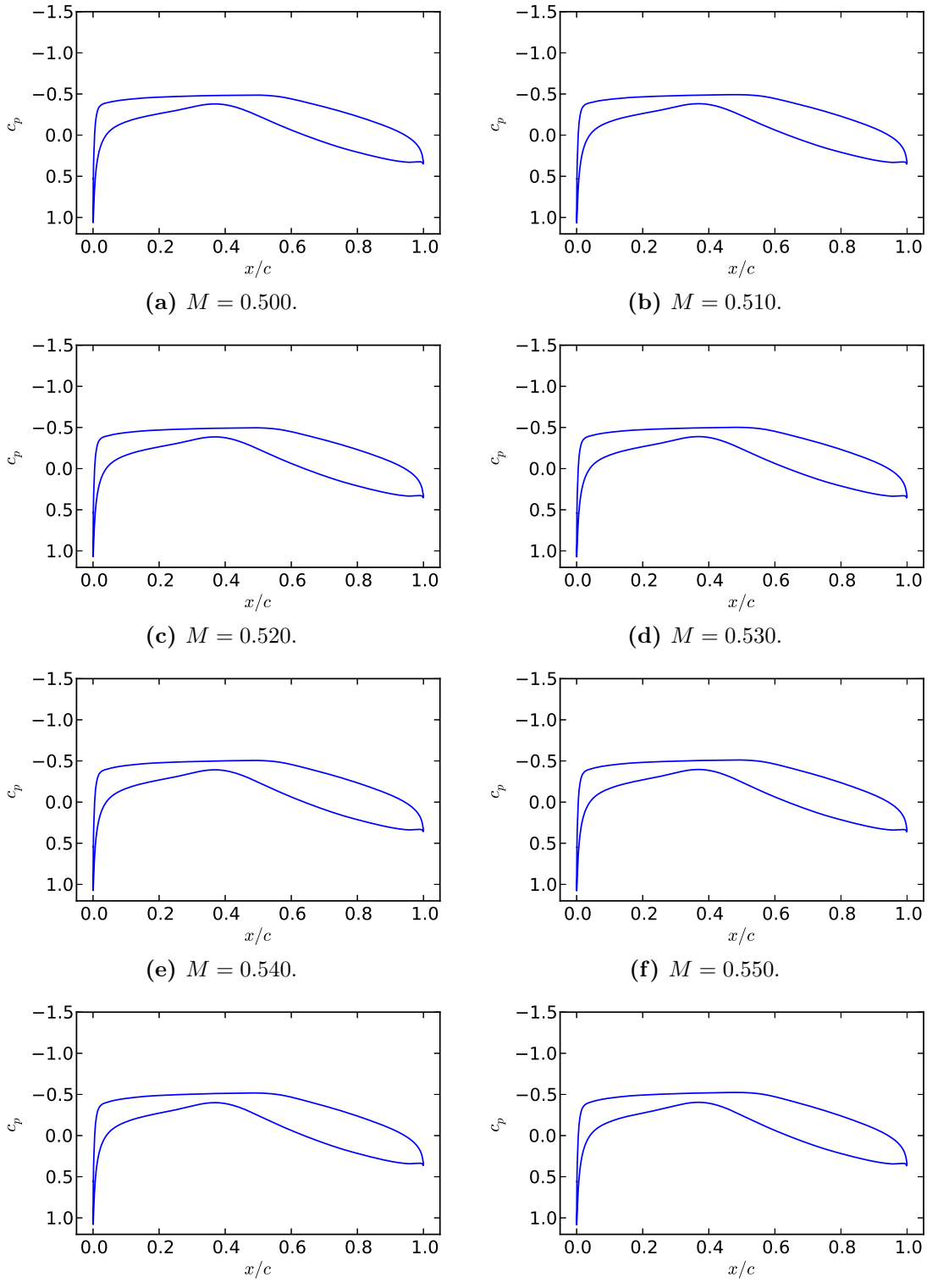


Figure 2.10. Inviscid drag divergence of the RAE 2822 airfoil.

2.2.1 Pressure coefficient distributions



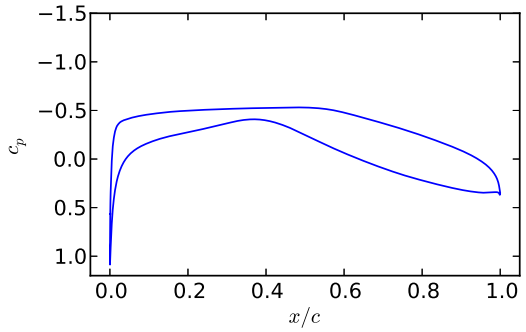
(a) $M = 0.500$. **(b)** $M = 0.510$.

(c) $M = 0.520$. **(d)** $M = 0.530$.

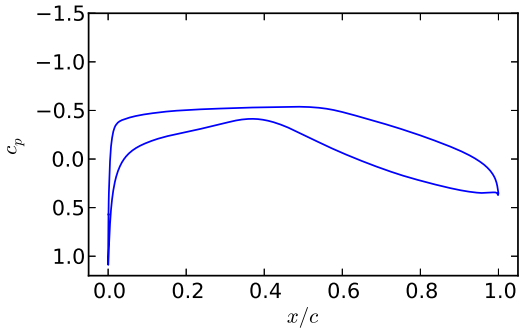
(e) $M = 0.540$. **(f)** $M = 0.550$.

(g) $M = 0.560$. **(h)** $M = 0.570$.

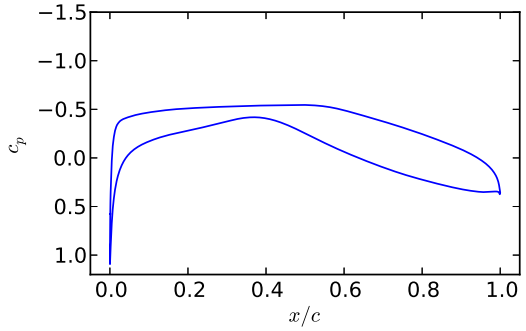
Figure 2.11. Inviscid pressure coefficient distributions.



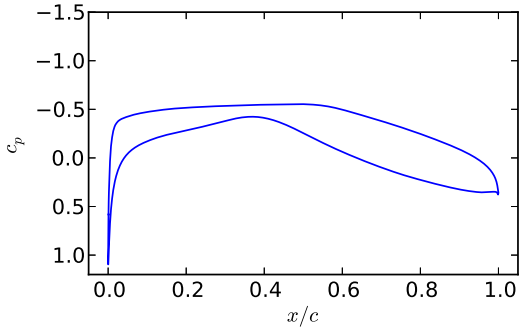
(a) $M = 0.580$.



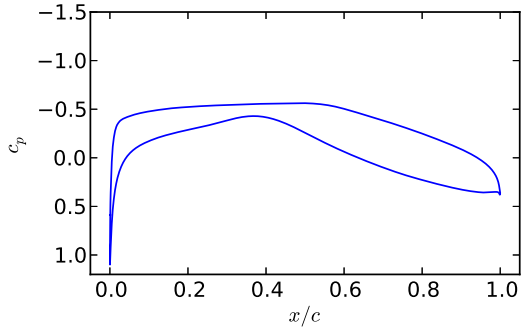
(b) $M = 0.590$.



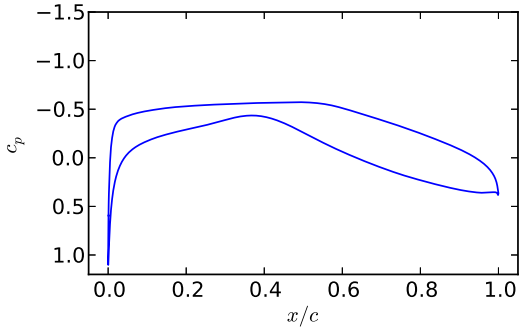
(c) $M = 0.600$.



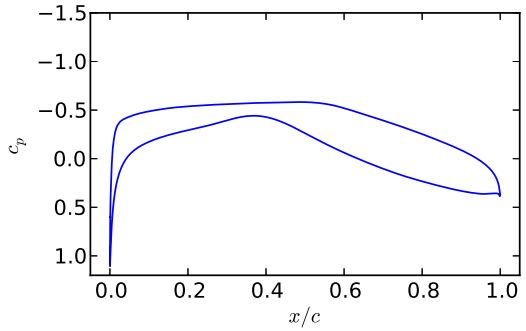
(d) $M = 0.610$.



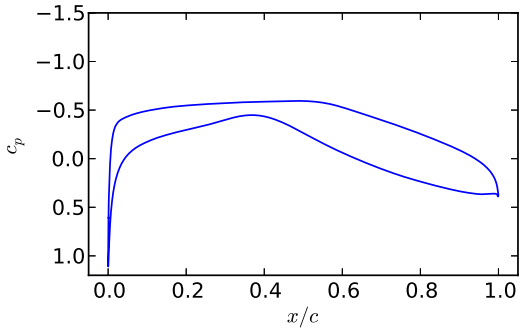
(e) $M = 0.620$.



(f) $M = 0.630$.

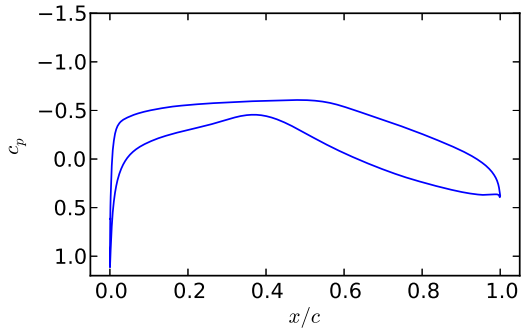


(g) $M = 0.640$.

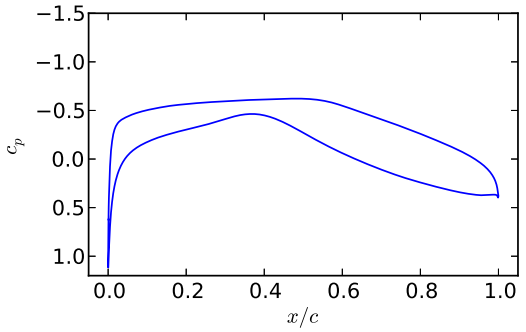


(h) $M = 0.650$.

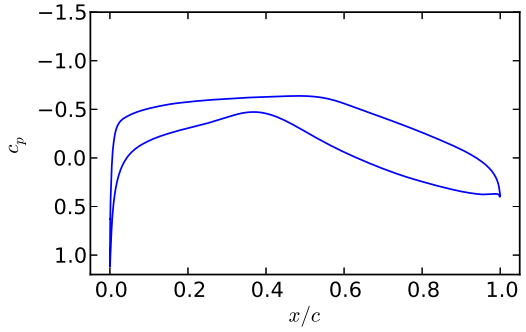
Figure 2.12. Inviscid pressure coefficient distributions, cont.



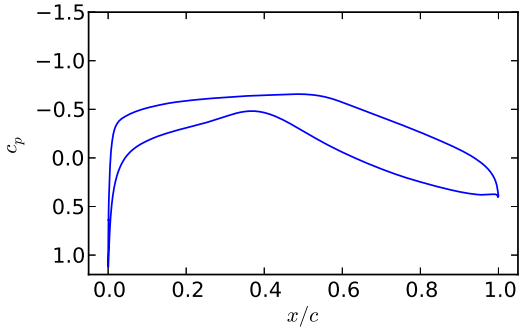
(a) $M = 0.660$.



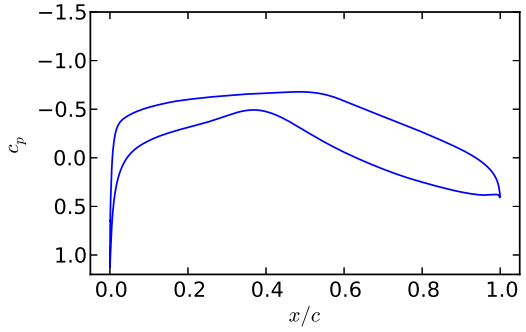
(b) $M = 0.670$.



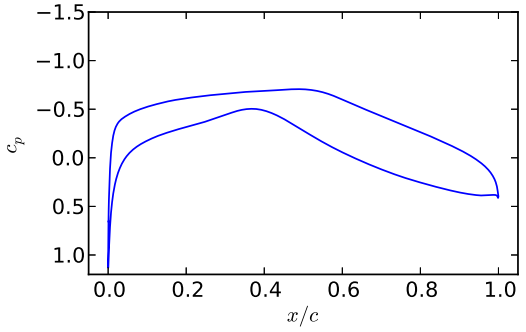
(c) $M = 0.680$.



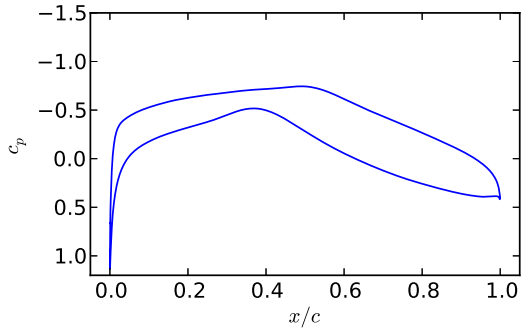
(d) $M = 0.690$.



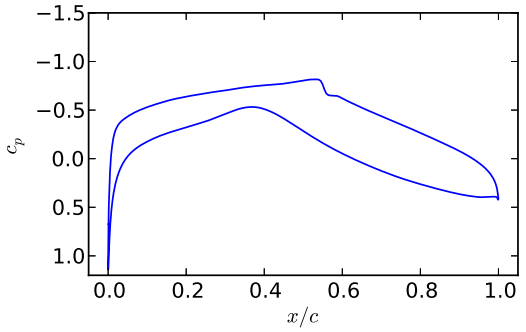
(e) $M = 0.700$.



(f) $M = 0.710$.

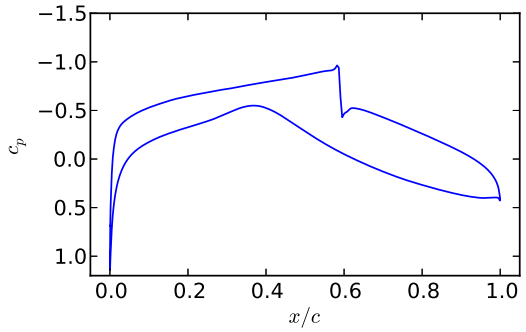


(g) $M = 0.720$.

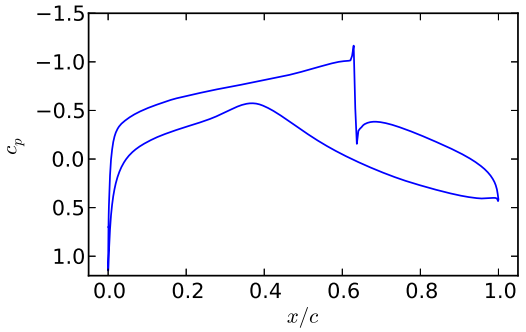


(h) $M = 0.730$.

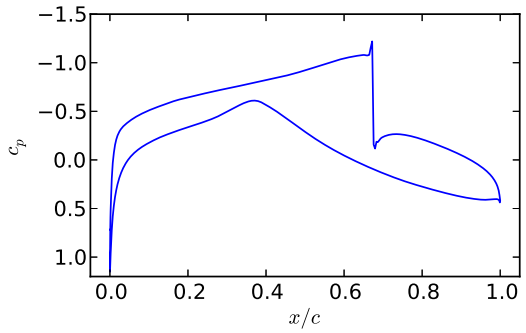
Figure 2.13. Inviscid pressure coefficient distributions, cont..



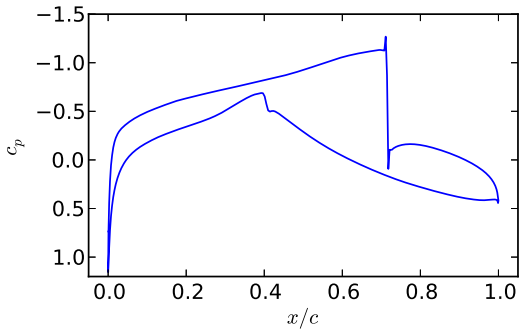
(a) $M = 0.740$.



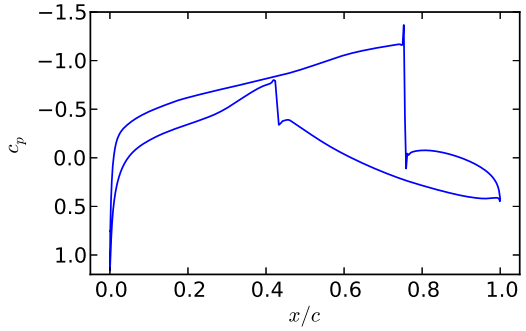
(b) $M = 0.750$.



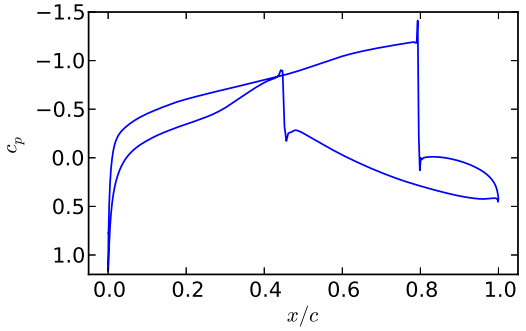
(c) $M = 0.760$.



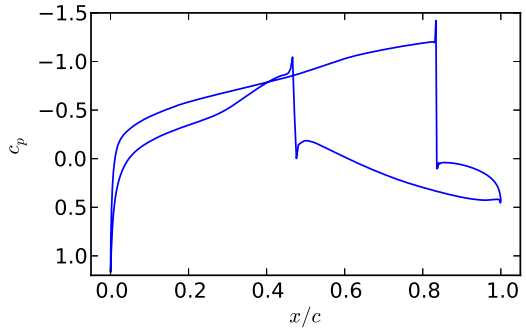
(d) $M = 0.770$.



(e) $M = 0.780$.



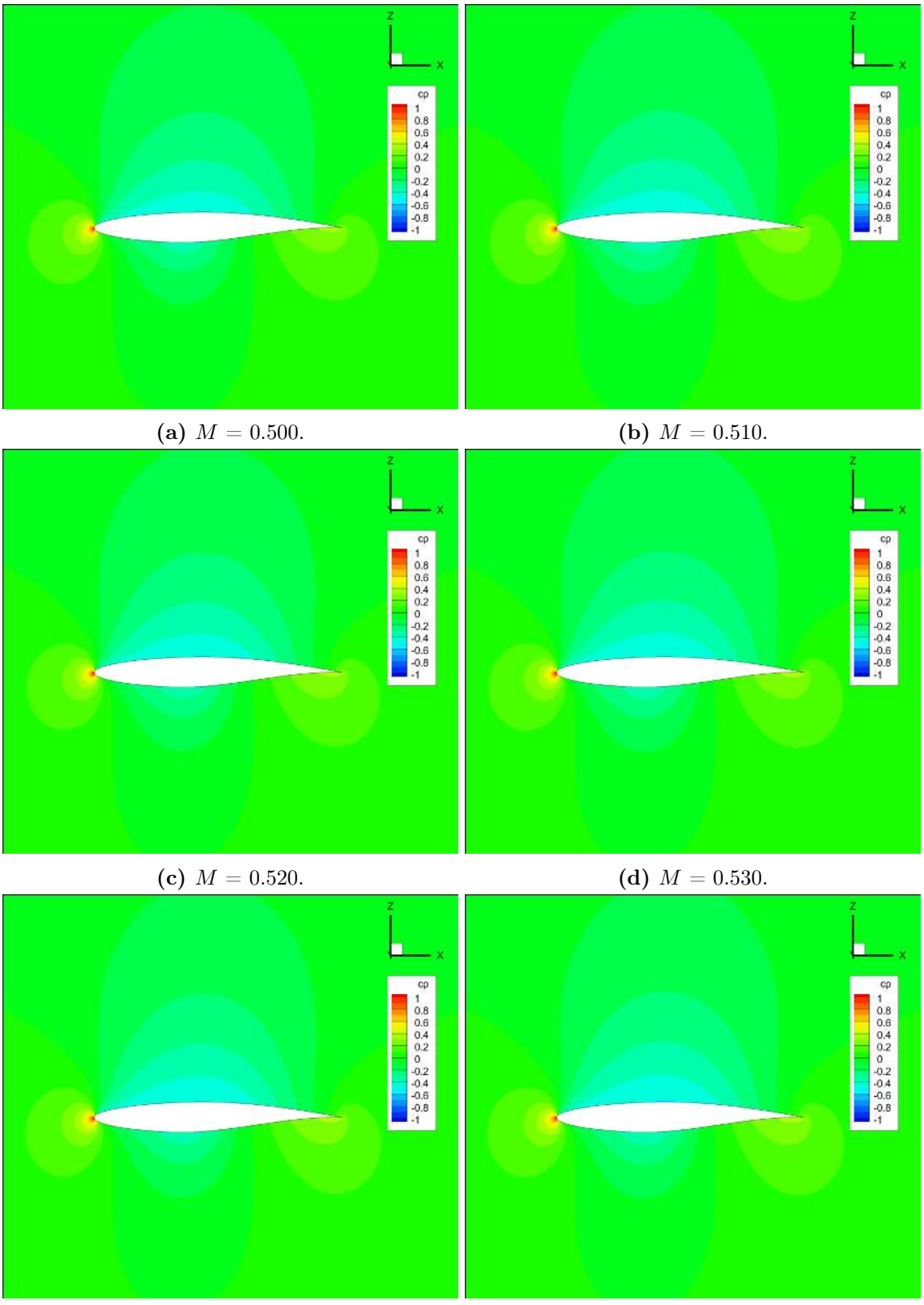
(f) $M = 0.790$.



(g) $M = 0.800$.

Figure 2.14. Inviscid pressure coefficient distributions, cont...

2.2.2 Pressure coefficient contours

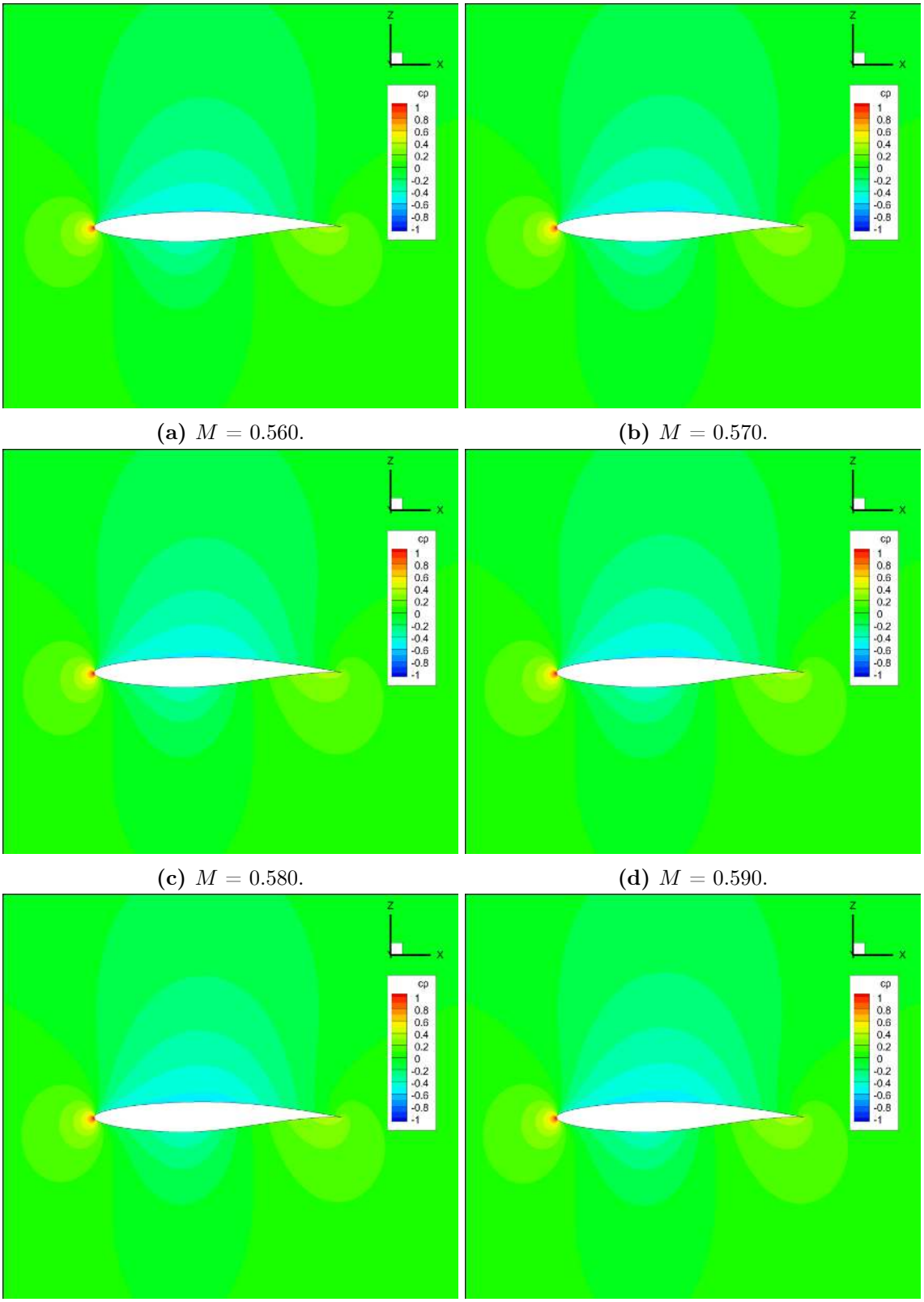


(a) $M = 0.500$. **(b)** $M = 0.510$.

(c) $M = 0.520$. **(d)** $M = 0.530$.

(e) $M = 0.540$. **(f)** $M = 0.550$.

Figure 2.15. Inviscid contours of pressure coefficient.

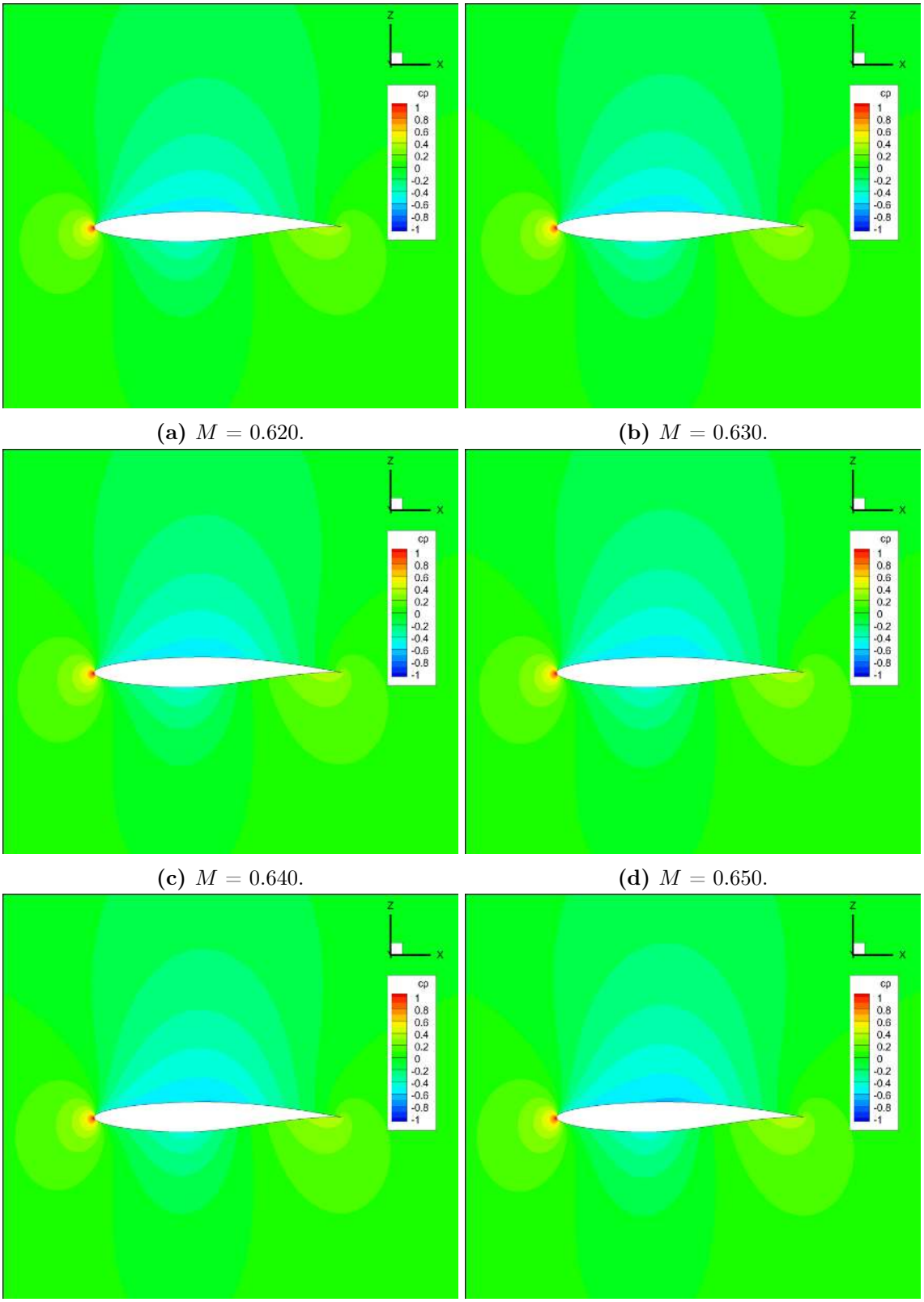


(a) $M = 0.560$. **(b)** $M = 0.570$.

(c) $M = 0.580$. **(d)** $M = 0.590$.

(e) $M = 0.600$. **(f)** $M = 0.610$.

Figure 2.16. Inviscid contours of pressure coefficient.

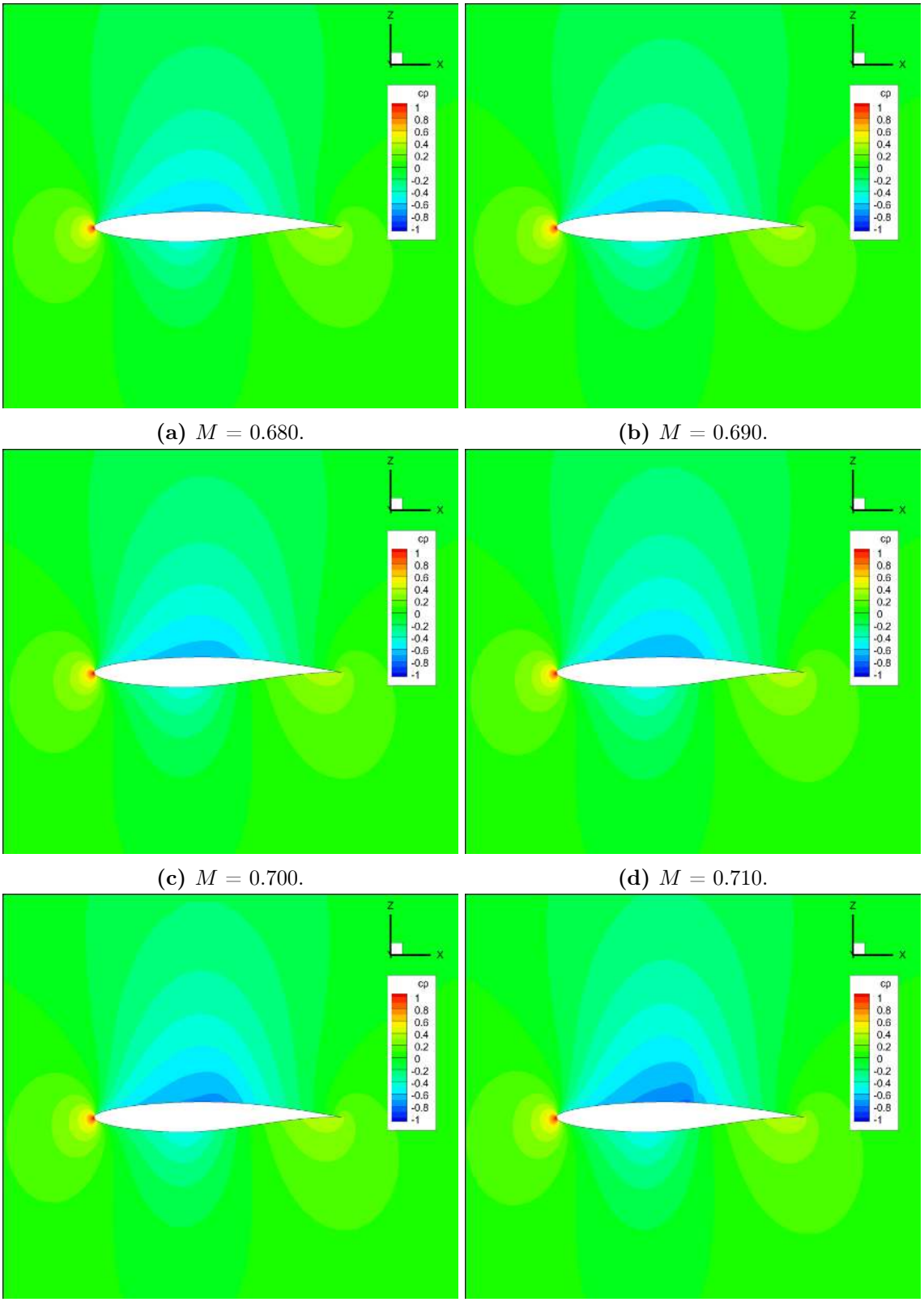


(a) $M = 0.620$. **(b)** $M = 0.630$.

(c) $M = 0.640$. **(d)** $M = 0.650$.

(e) $M = 0.660$. **(f)** $M = 0.670$.

Figure 2.17. Inviscid contours of pressure coefficient.

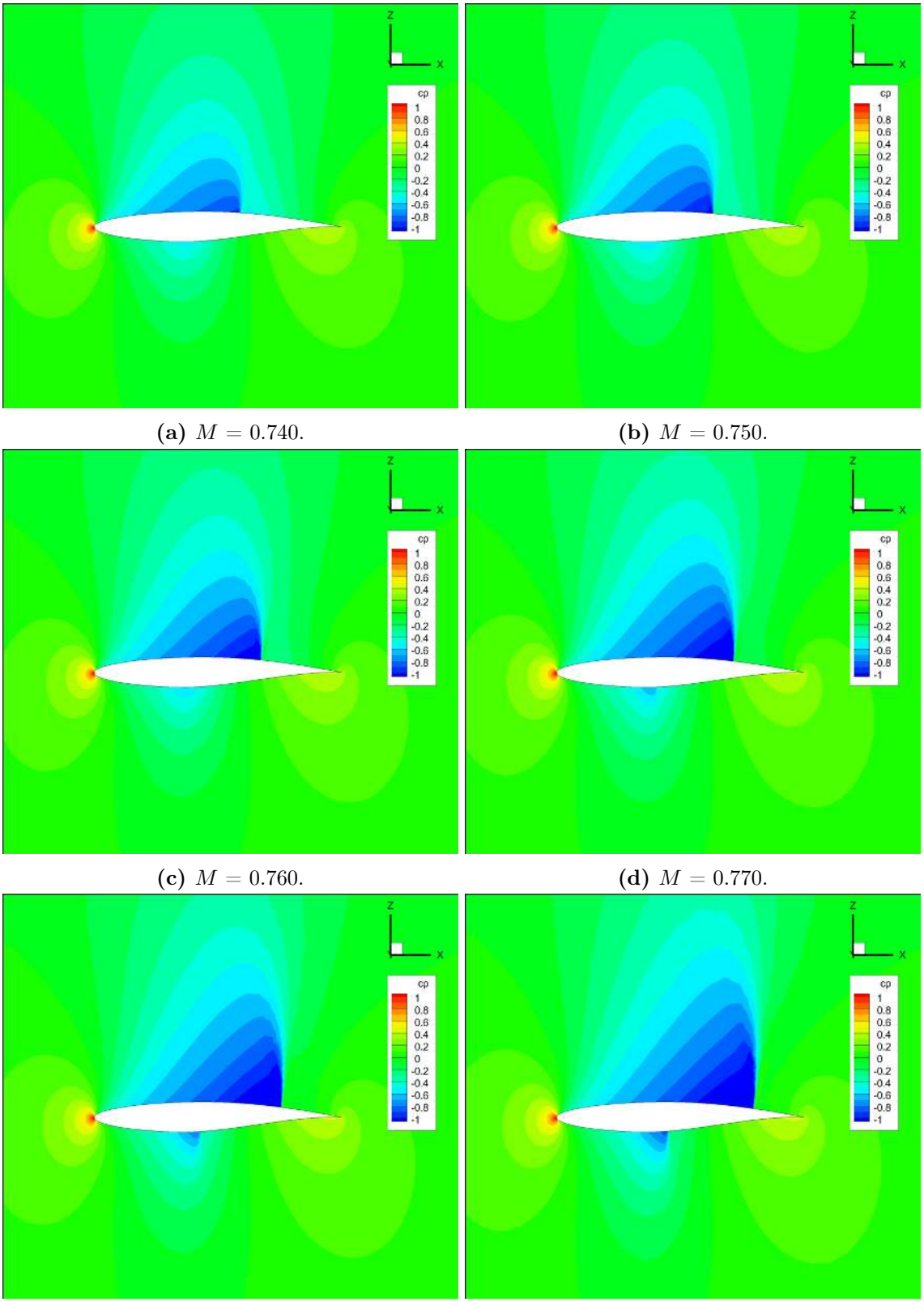


(a) $M = 0.680$. **(b)** $M = 0.690$.

(c) $M = 0.700$. **(d)** $M = 0.710$.

(e) $M = 0.720$. **(f)** $M = 0.730$.

Figure 2.18. Inviscid contours of pressure coefficient.



(a) $M = 0.740$. **(b)** $M = 0.750$.

(c) $M = 0.760$. **(d)** $M = 0.770$.

(e) $M = 0.780$. **(f)** $M = 0.790$.

Figure 2.19. Inviscid contours of pressure coefficient.

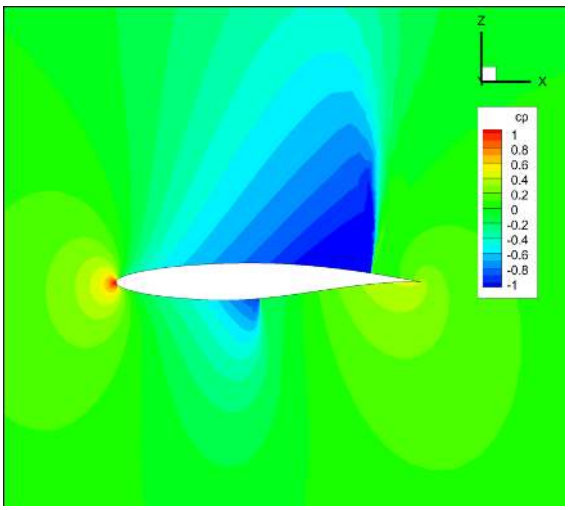


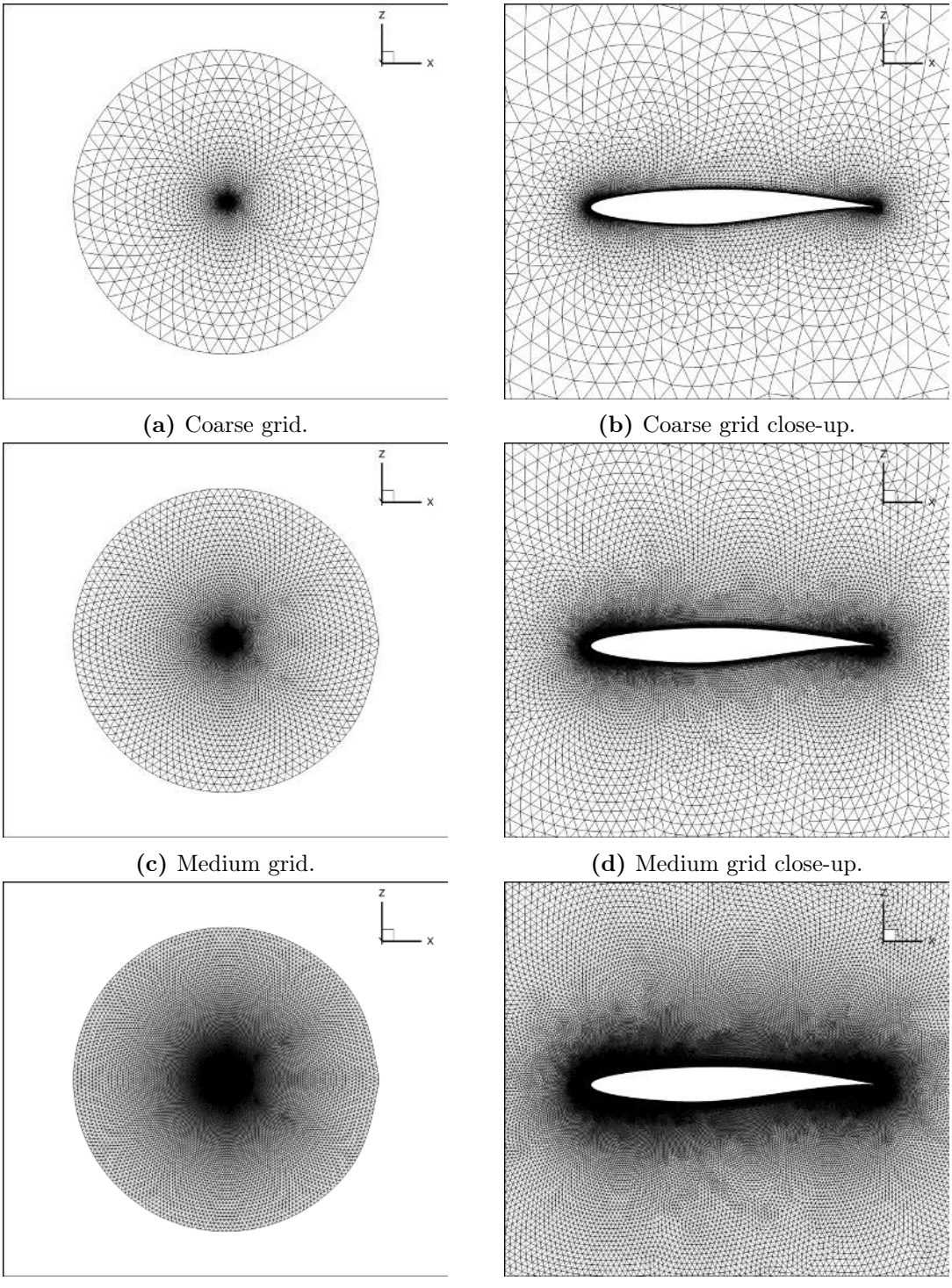
Figure 2.20. Inviscid contours of pressure coefficient ($M = 0.800$).

Chapter 3

Viscous Drag Divergence

3.1 Grid Convergence Study

3.1.1 Grids



(a) Coarse grid.

(b) Coarse grid close-up.

(c) Medium grid.

(d) Medium grid close-up.

(e) Fine grid.

(f) Fine grid close-up.

Figure 3.1. Computational meshes.

3.1.2 Results

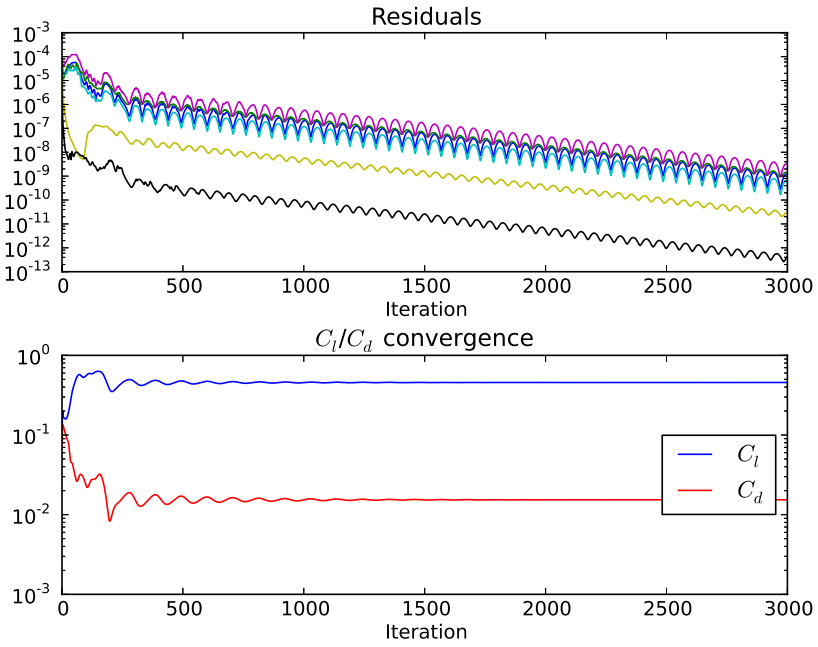


Figure 3.2. Coarse grid residuals.

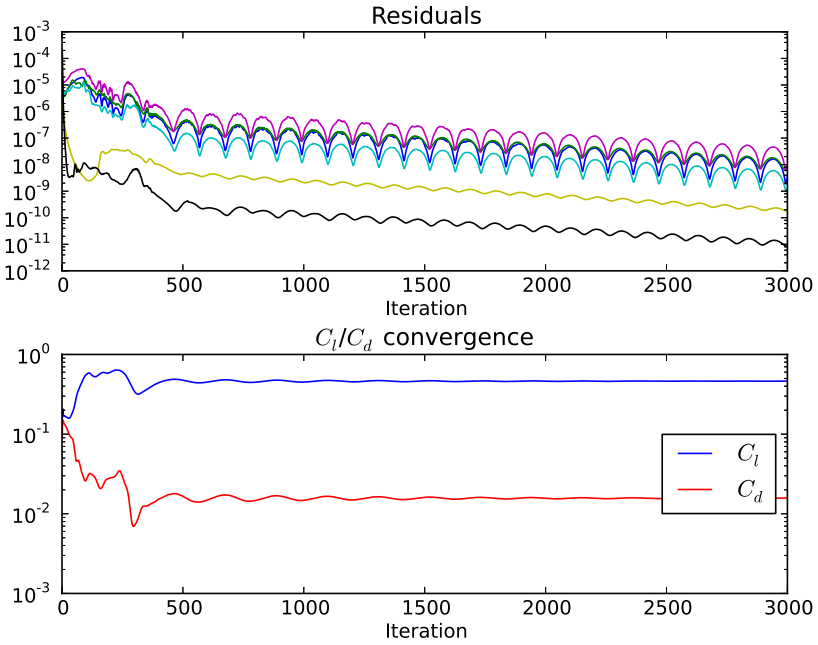


Figure 3.3. Medium grid residuals.

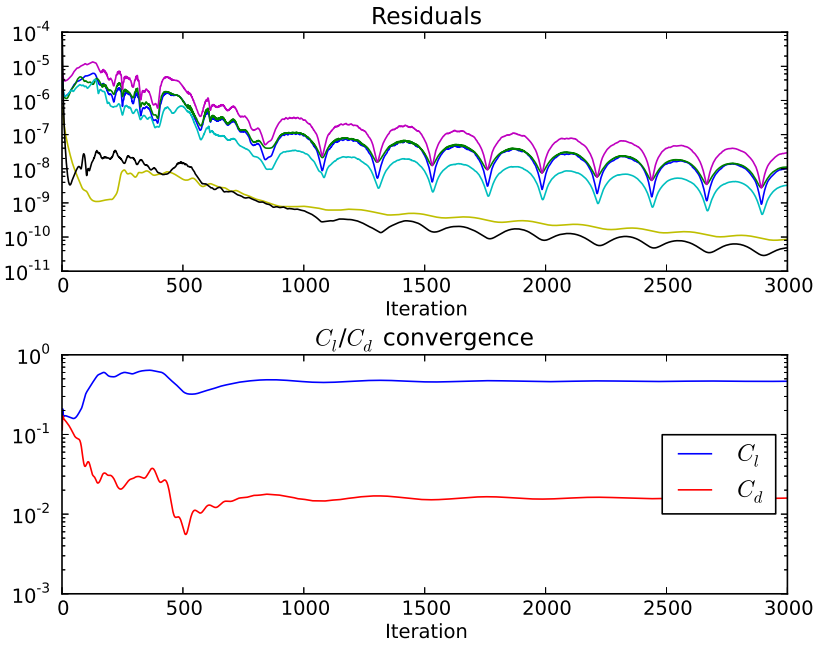


Figure 3.4. Fine grid residuals.

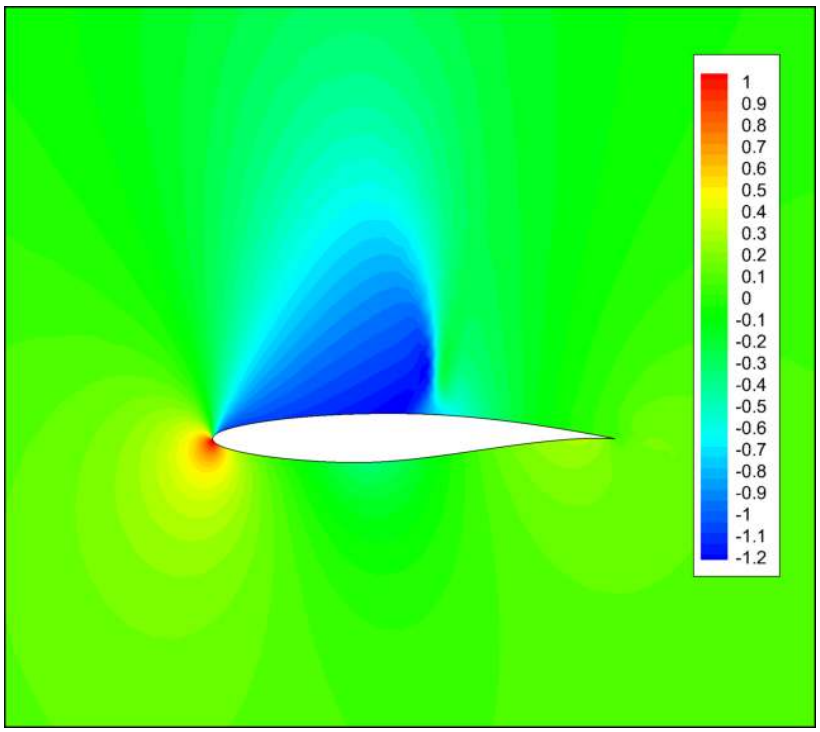


Figure 3.5. Coarse grid contours of c_p.

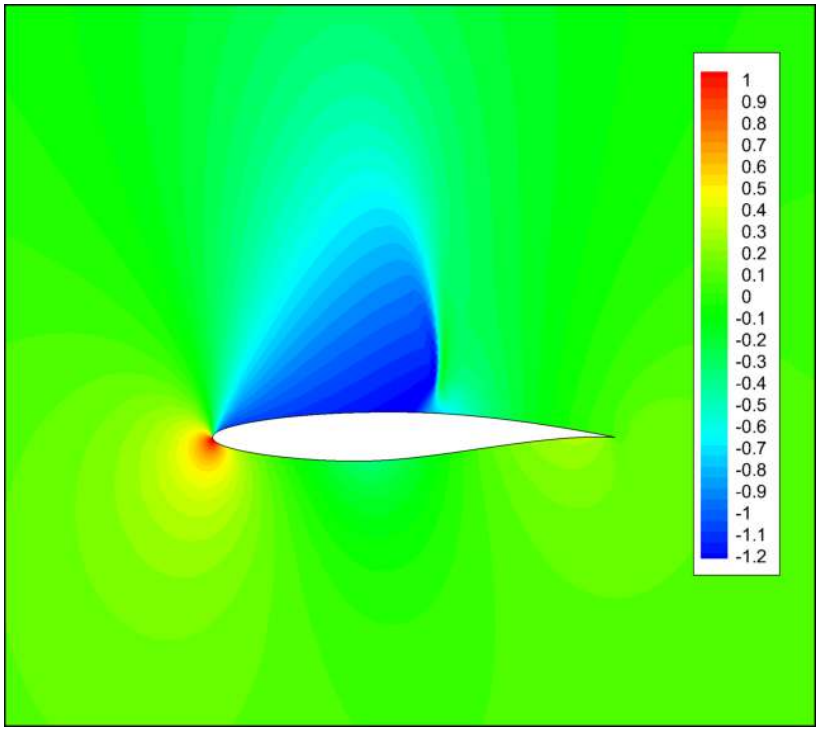


Figure 3.6. Medium grid contours of c_p.

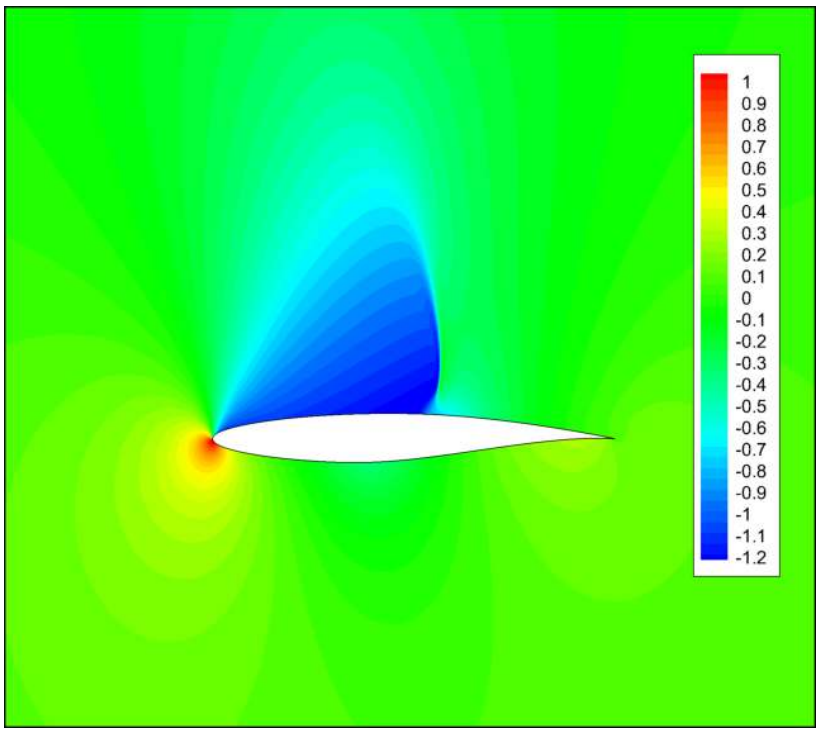


Figure 3.7. Fine grid contours of c_p.

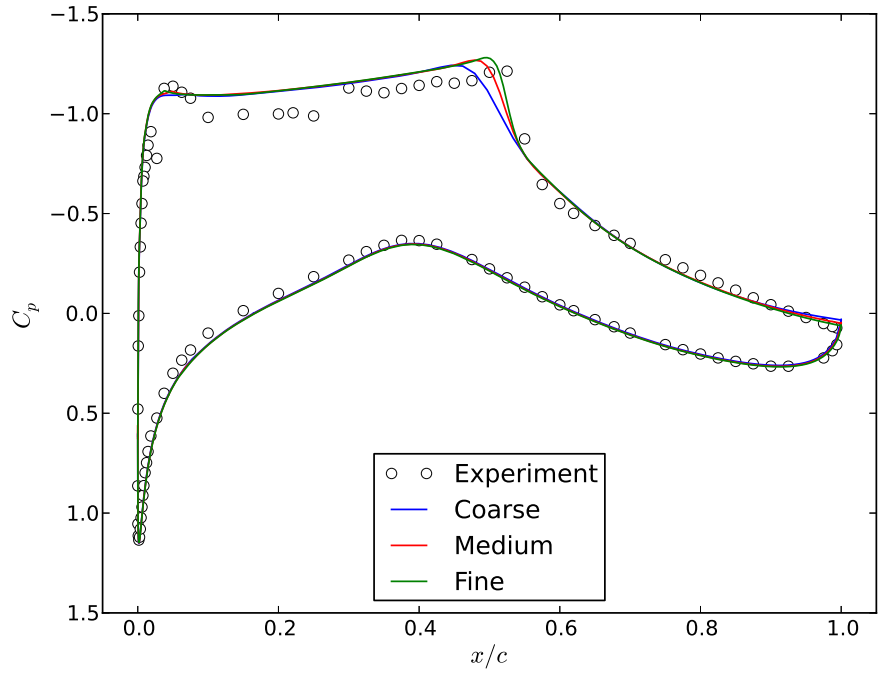


Figure 3.8. c_p comparison.

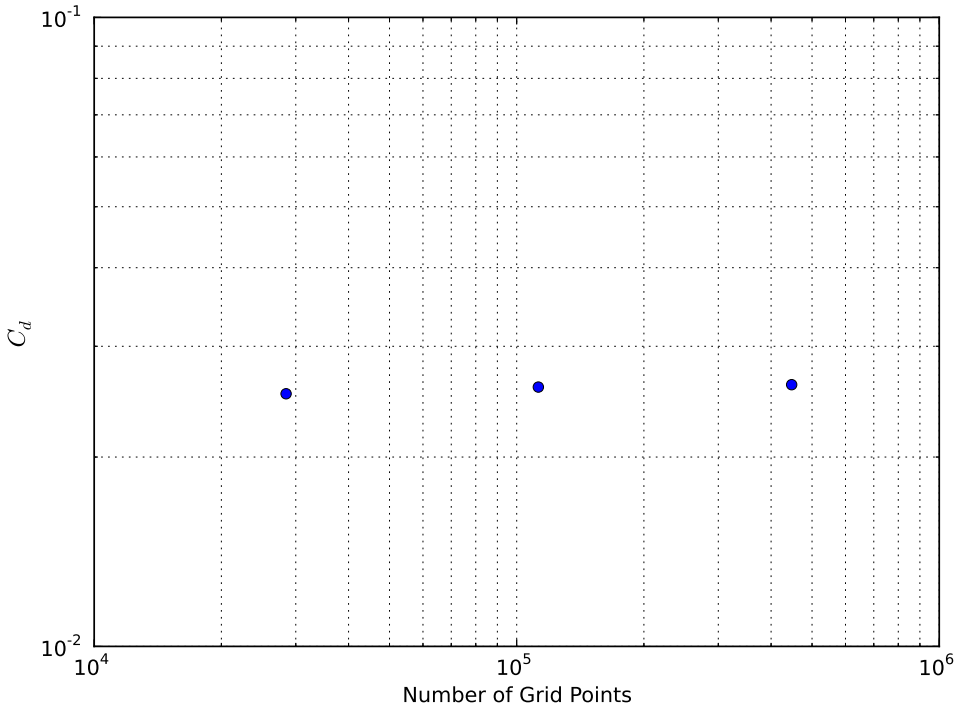


Figure 3.9. c_d vs number of grid points.

The fine grid is chosen for futher simulations because it resolves the shock better.

3.2 Turbulence model comparison

Three turbulence models were chosen for comparison, namely, the Spallart-Allmaras (SA) model, the k-ϵ model, and the k-ω SST model. The computational data is compared with experimental results from AR-138. Some convergence issues were encountered with the k-ϵ model. Because of the convergence issues, only the SA and k-ω SST models are considered further. The residuals are shown in Figures 3.10 and 3.11.

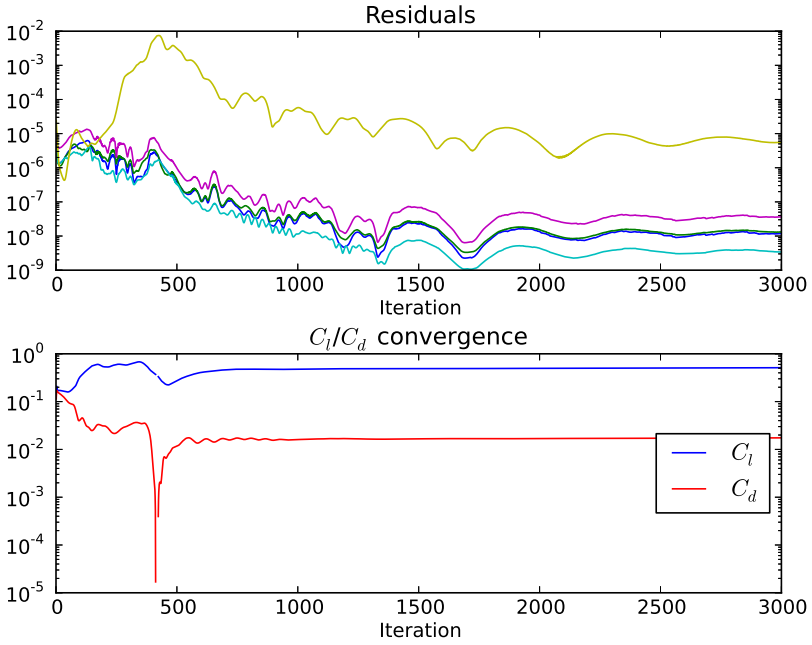


Figure 3.10. Spallart-Allmaras residuals.

It is well-known that the k-ϵ model does not perform well in cases of strong adverse pressure gradient [1]. Examining the contours of Mach number at the experiment conditions ($M = 0.745$, $\alpha = 3.19°$, and $Re_c = 2.7 \times 10^6$), it is evident that there is shock wave-boundary layer interaction (see Figure 3.12). The strong adverse pressure gradient caused by the shock induces boundary layer separation. Both the SA and k-ω SST turbulence models predict a separation bubble as is shown in Figures 3.13 and 3.14. Perhaps this shock-boundary layer interaction, subsequent flow separation and reattachment is the reason for the k-ϵ convergence issues.

The motivation for the development of the k-ω SST model was to more accurately model the physics of flows with strong pressure gradients and/or turbulent boundary layer separation [2]. Several sources confirm that the SST model is more accurate than the k-ϵ model when considering shocks and boundary layer separation [3]. A comparison between the c_p distributions from SA and k-ω SST models is given in Figure 3.15 and shows that the SST model matches the experimental data better. Because the SST model is capable of more accurately modeling flows that involve

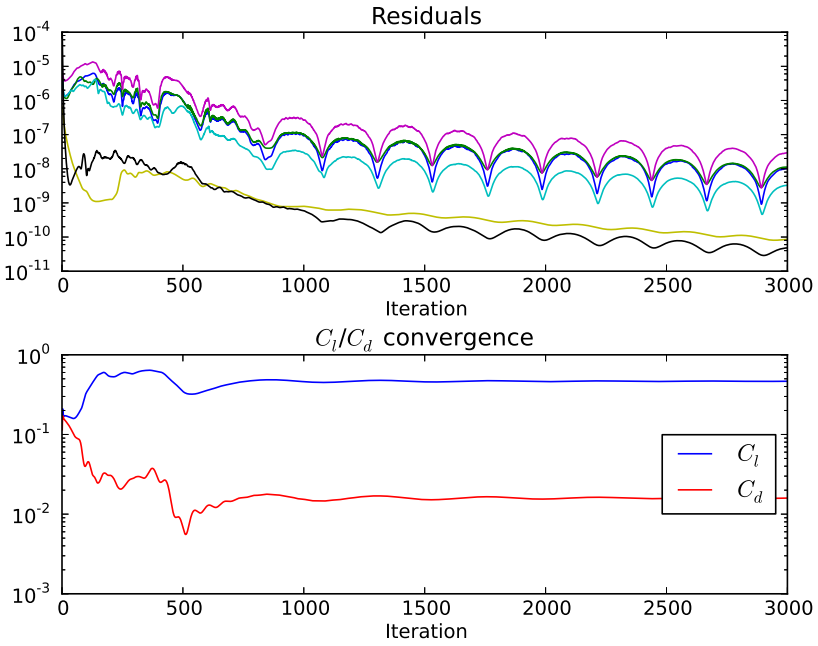
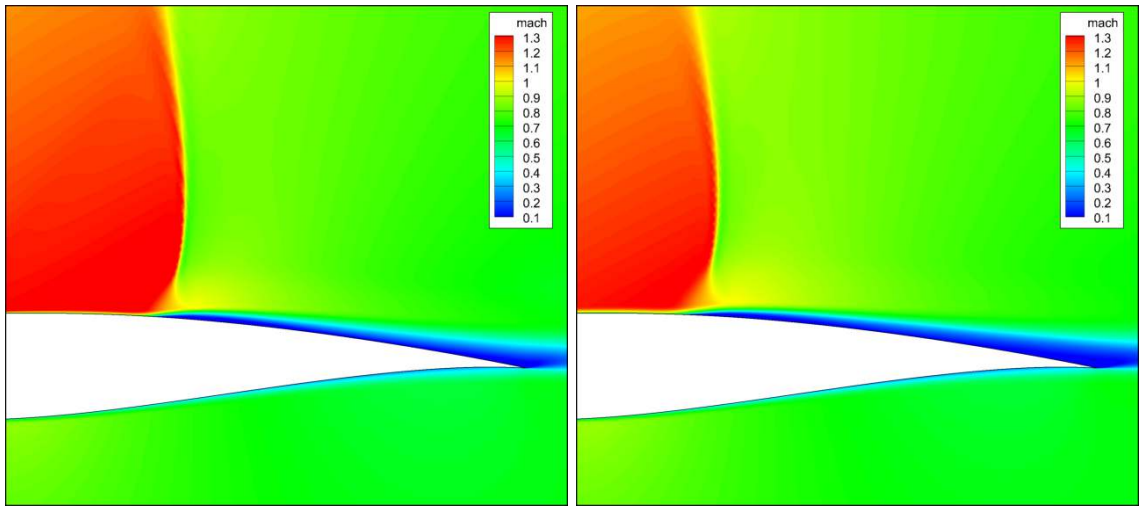


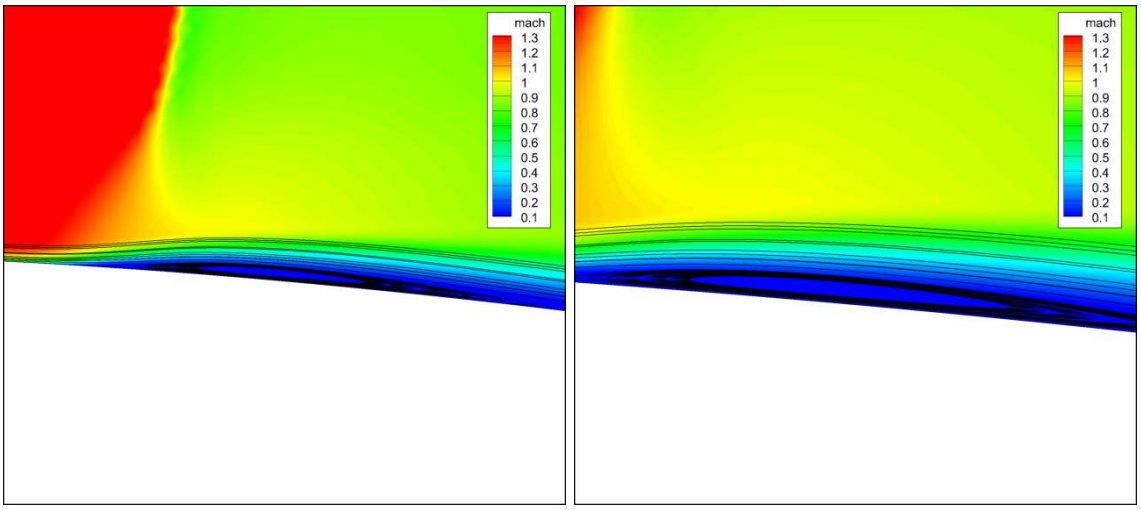
Figure 3.11. k-ω SST residuals.

strong adverse pressure gradients and/or flow separation, the k-ω SST turbulence model is chosen for further simulations.



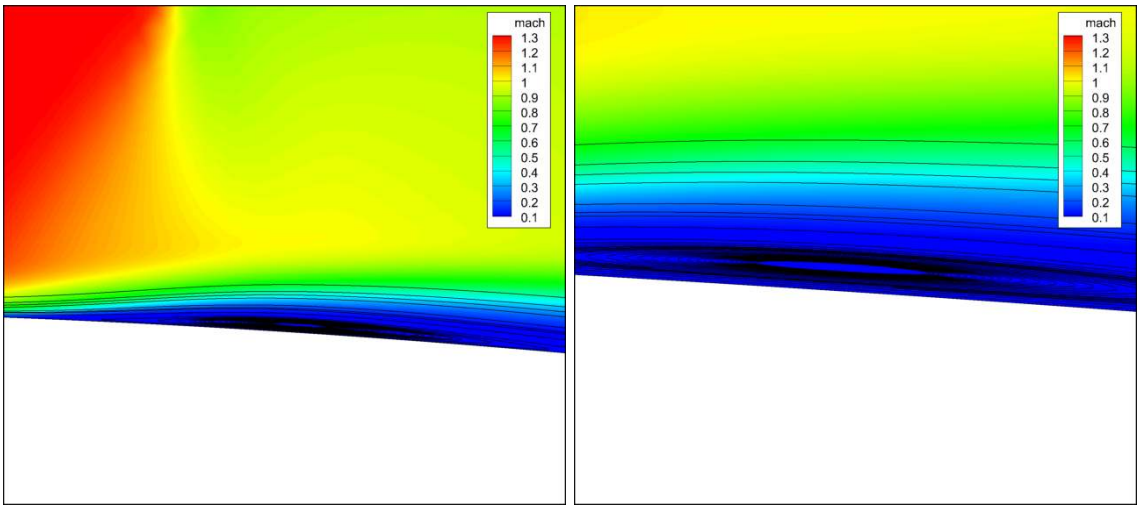
(a) SA model. **(b)** k-ω SST model.

Figure 3.12. Contours of Mach number.



(a) Separation bubble. **(b)** Close-up of separation bubble.

Figure 3.13. SA boundary layer separation and reattachment.



(a) Separation bubble. **(b)** Close-up of separation bubble.

Figure 3.14. k-ω SST boundary layer separation and reattachment.

Also, a comparison of force and moment coefficients is provided in Table 3.1. The percent difference is computed using the following equation

$$\% \text{ difference} = \left| \frac{E_1 - E_2}{\frac{1}{2}(E_1 + E_2)} \right| \times 100$$

where E_1 and E_2 are the two quantities being compared.

Table 3.1. Comparison between force and moment coefficients.

($Re_c = 2.7 \times 10^6$, $\alpha = 3.19°$, $M = 0.740$)

Coefficients	Experiment	SA	% difference	k-ω SST	% difference
c_n	0.733	0.857	15.6%	0.781	6.34%
c_m	-0.086	-0.097	12.0%	-0.083	3.55%
c_d	0.0188	0.0264	33.6%	0.0241	24.7%

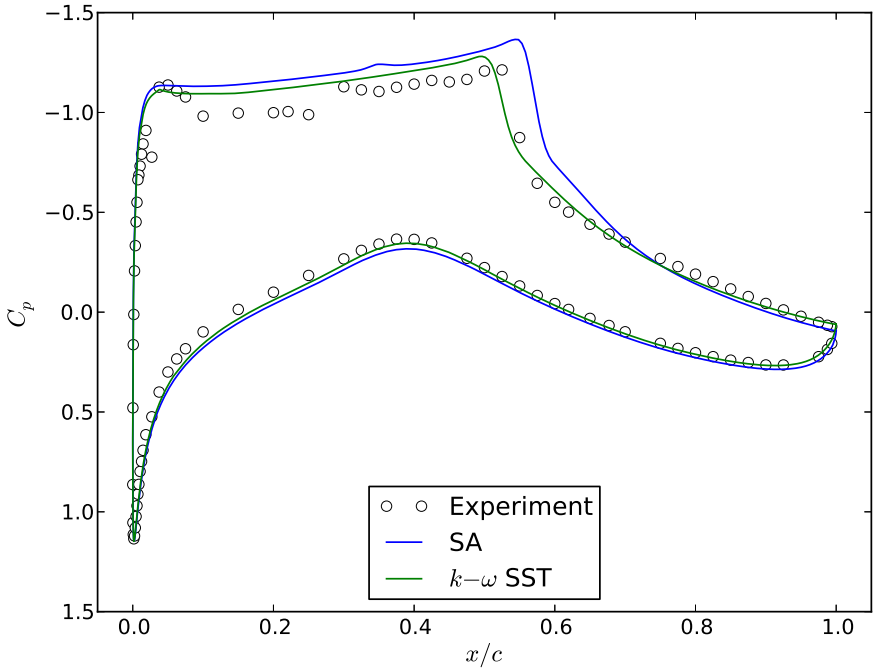


Figure 3.15. Comparison between c_p distributions for different turbulence models.

References

[1] Wilcox, D., "Turbulence Modeling for CFD," DCW Industries, Inc., La Cañada, CA, 1993.

[2] Menter, F. ,R., Kuntz, M., and Langtry, R., "Ten Years of Industrial Experience with the SST Turbulence Model," Turbulence, Heat and Mass Transfer 4, 2003.

[3] Wilcox, D., "Turbulence Modeling: An Overview," AIAA 2001-0724, 2001.

[4] Babinksy, H., and Harvey, J., "Shock Wave-Boundary-Layer Interactions," Cambridge University Press, 2011.

3.3 Drag divergence

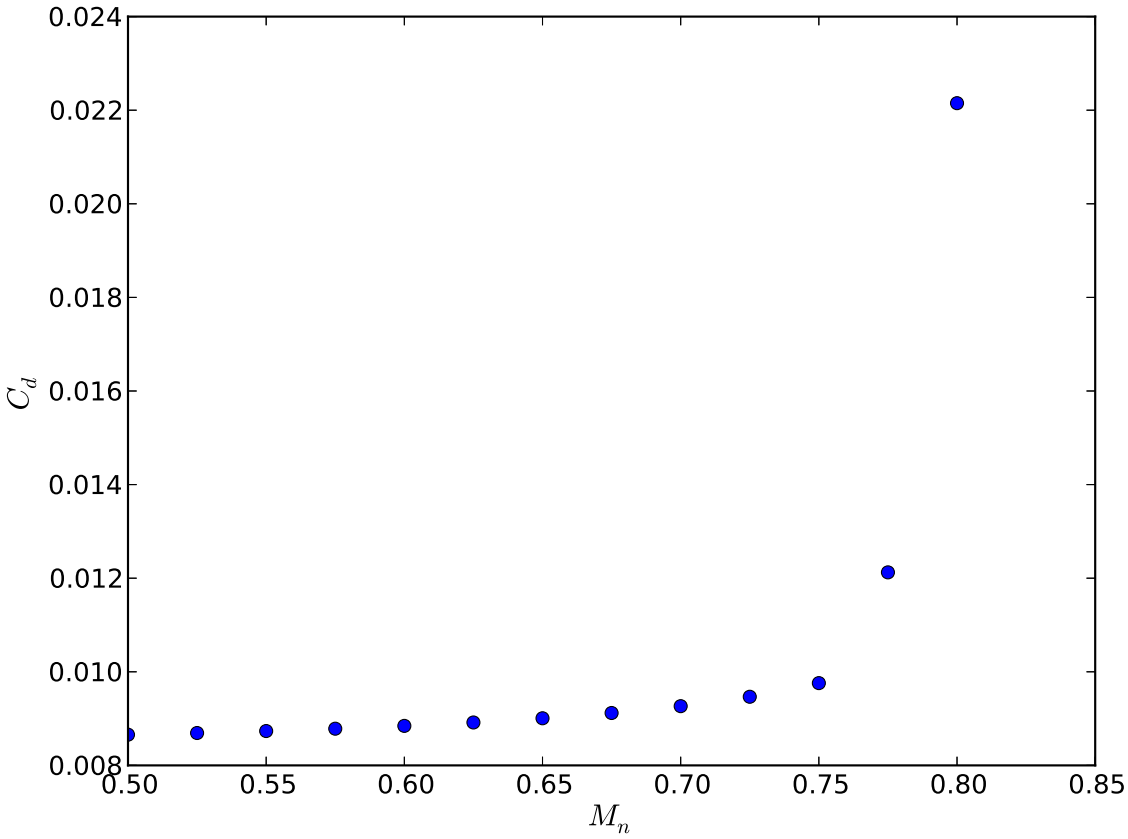
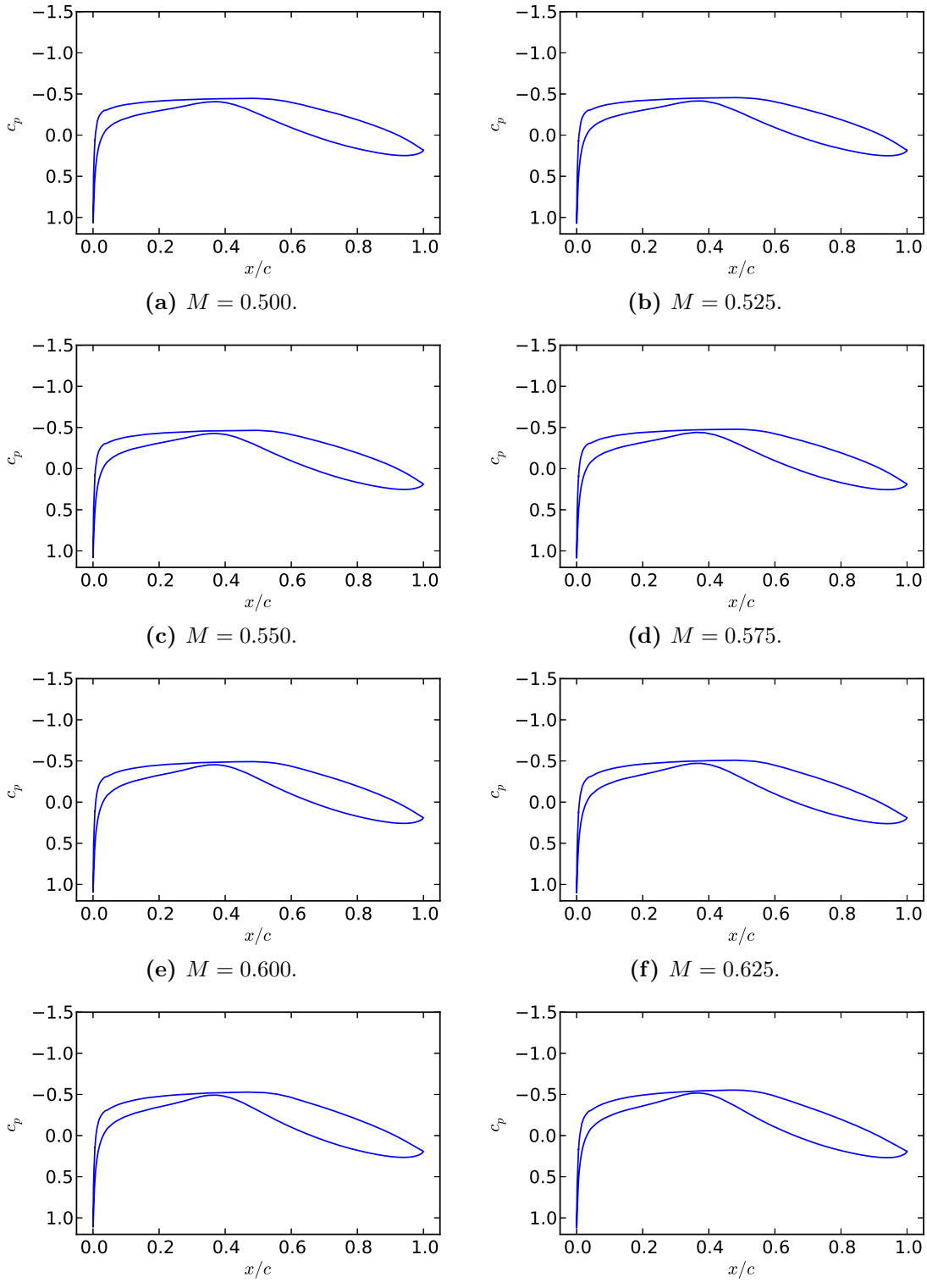


Figure 3.16. Viscous drag divergence of the RAE 2822 airfoil.

3.3.1 Pressure coefficient distributions



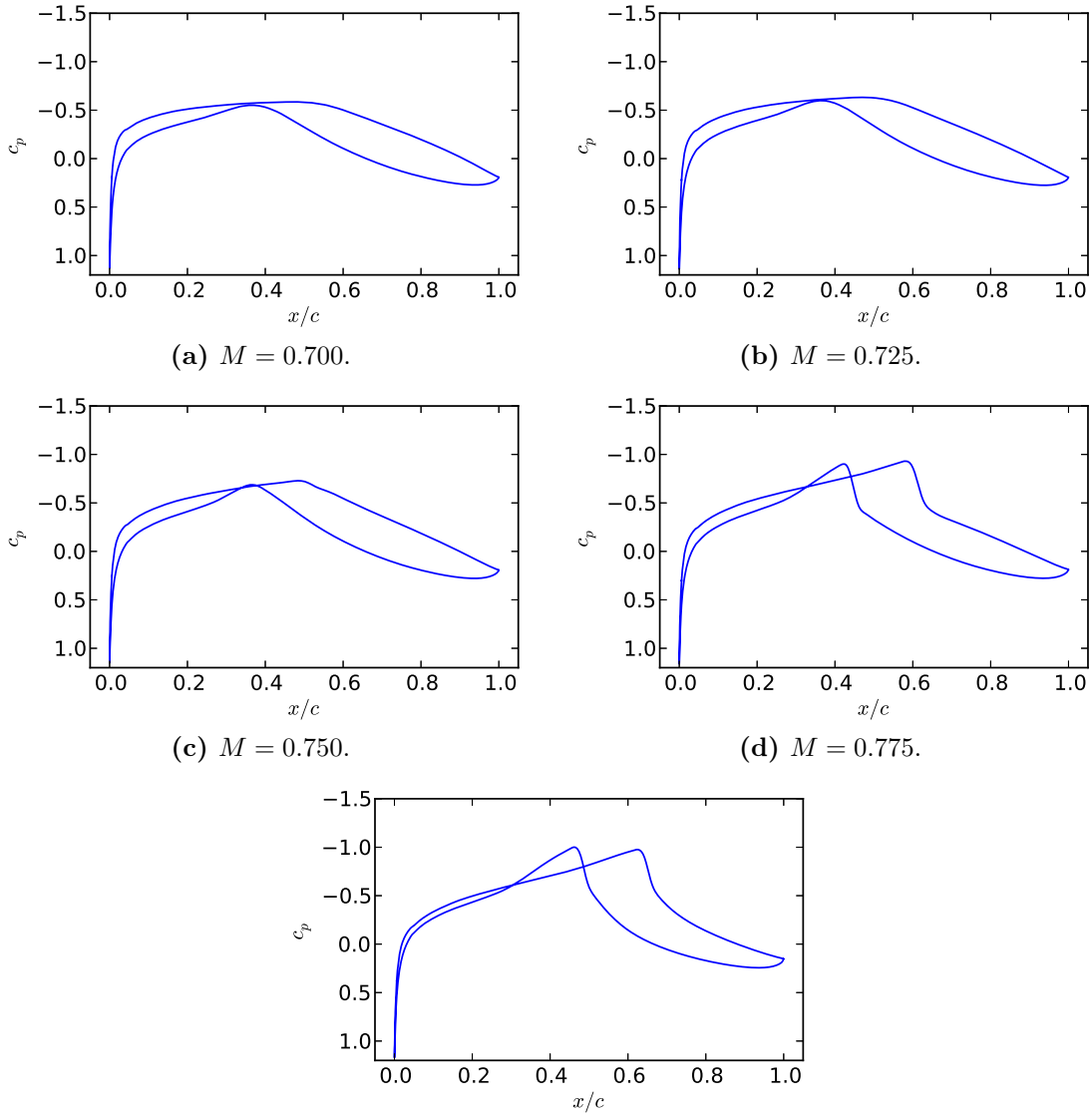
(a) $M = 0.500$. **(b)** $M = 0.525$.

(c) $M = 0.550$. **(d)** $M = 0.575$.

(e) $M = 0.600$. **(f)** $M = 0.625$.

(g) $M = 0.650$. **(h)** $M = 0.675$.

Figure 3.17. Viscous pressure coefficient distributions.



(a) $M = 0.700$.

(b) $M = 0.725$.

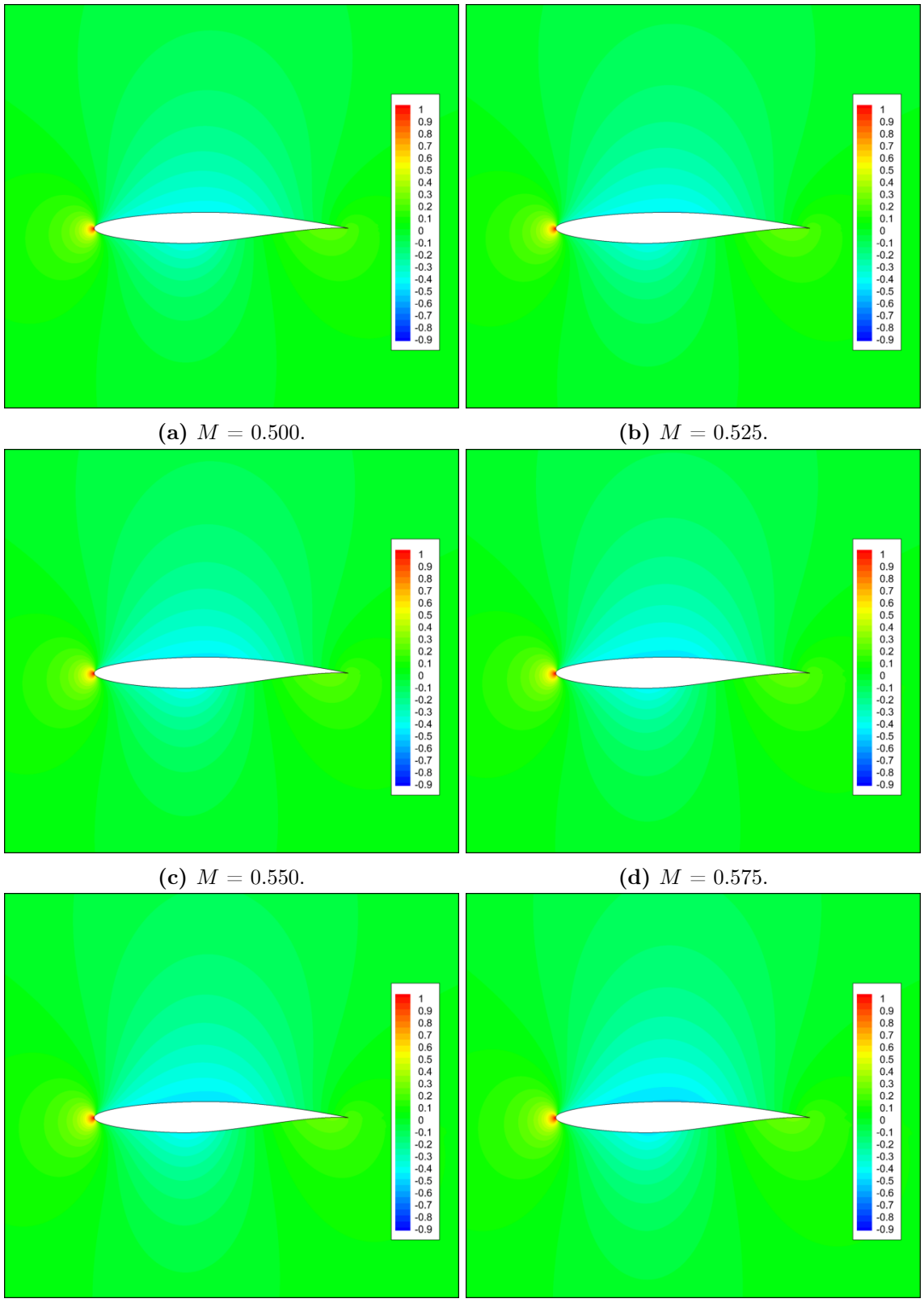
(c) $M = 0.750$.

(d) $M = 0.775$.

(e) $M = 0.800$.

Figure 3.18. Viscous pressure coefficient distributions.

3.3.2 Pressure coefficient contours

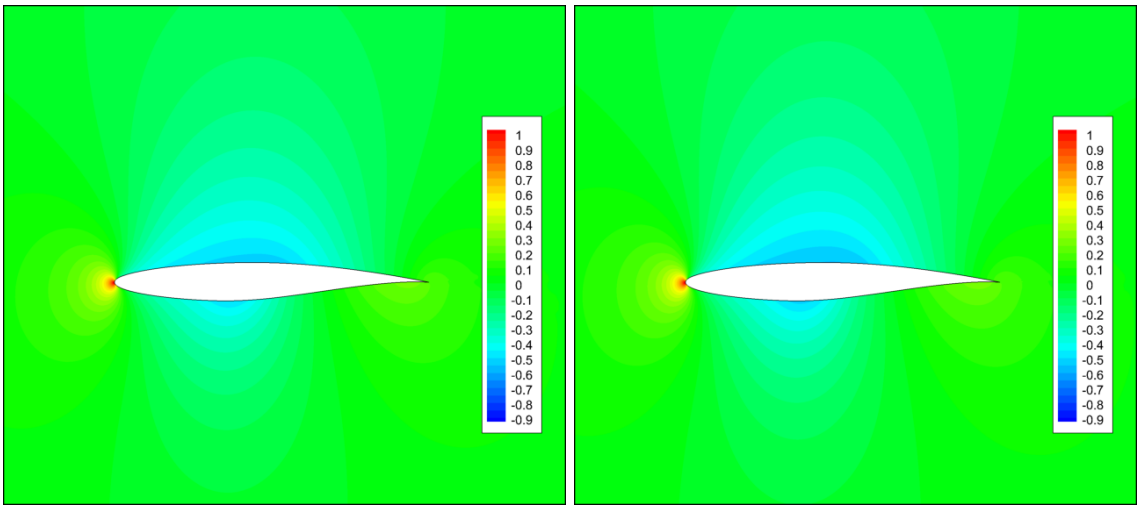


(a) $M = 0.500$. **(b)** $M = 0.525$.

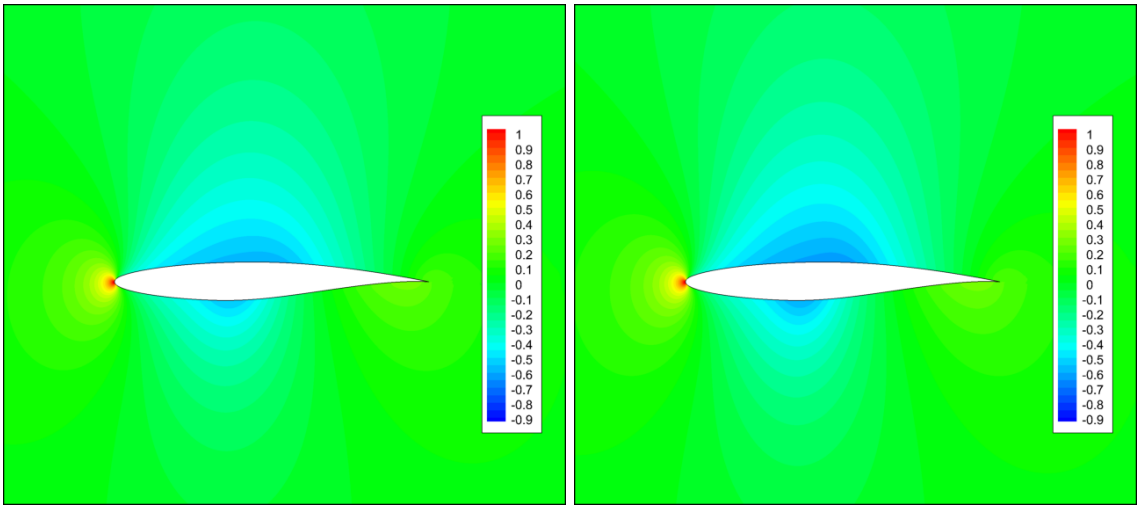
(c) $M = 0.550$. **(d)** $M = 0.575$.

(e) $M = 0.600$. **(f)** $M = 0.625$.

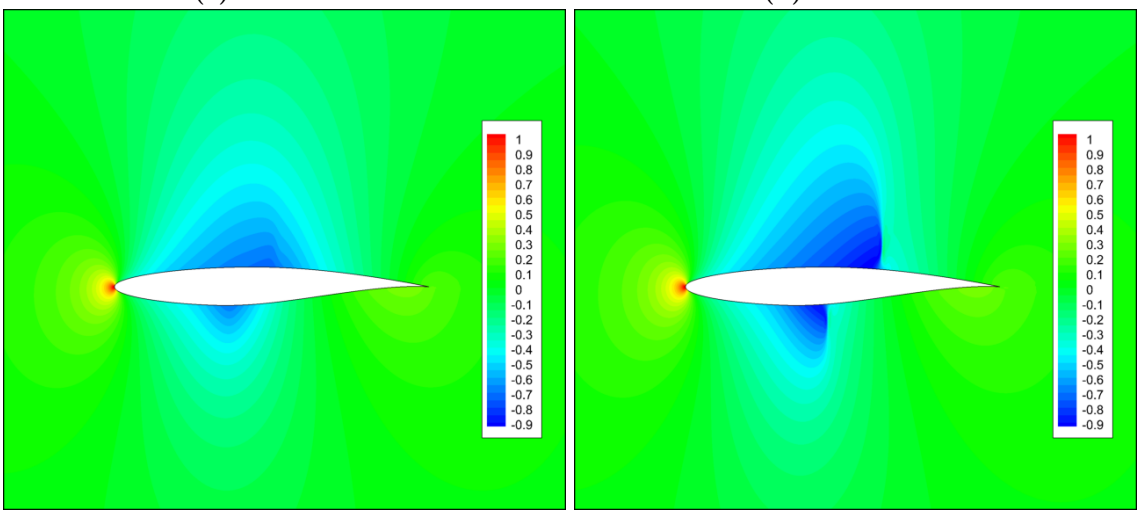
Figure 3.19. Viscous contours of pressure coefficient.



(a) $M = 0.650$. **(b)** $M = 0.675$.



(c) $M = 0.700$. **(d)** $M = 0.725$.



(e) $M = 0.750$. **(f)** $M = 0.775$.

Figure 3.20. Viscous contours of pressure coefficient.

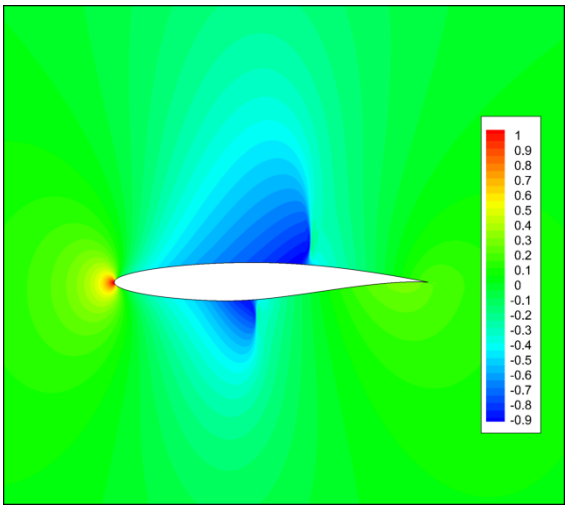


Figure 3.21. Viscous contours of pressure coefficient ($M = 0.800$).

3.3.3 Comparison between inviscid and viscous drag divergence

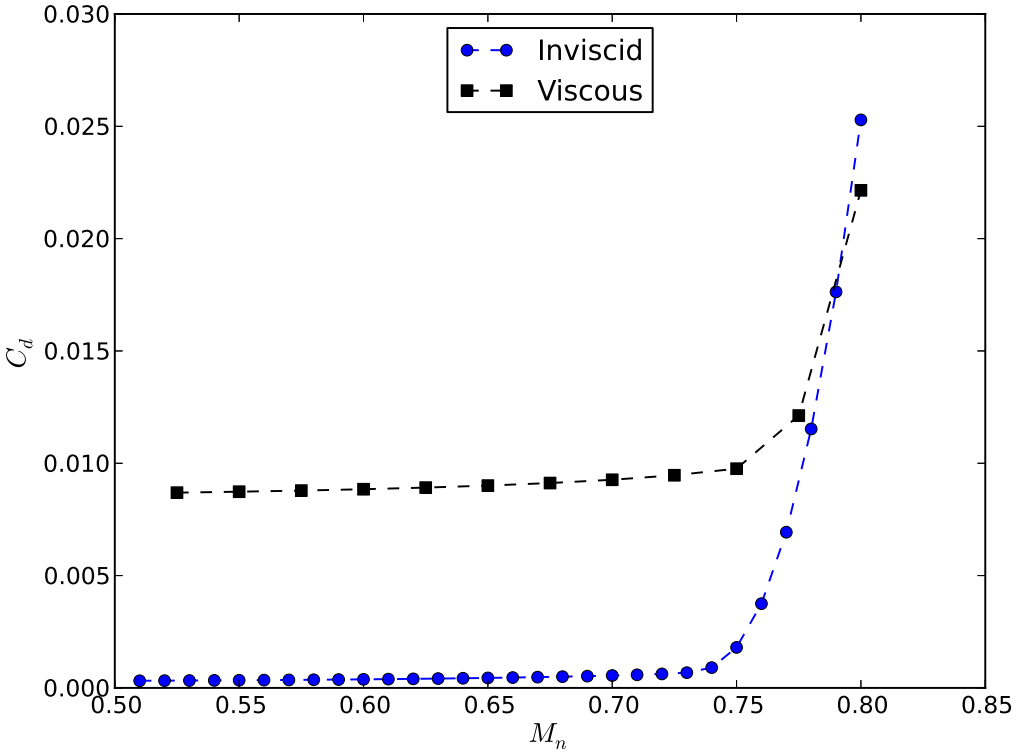


Figure 3.22. Comparison between inviscid and viscous drag divergence.

Appendices

Python geometry set-up script

```
#!/usr/bin/python

from StringIO import StringIO
from subprocess import call
import os
import sys
import numpy as np
import matplotlib.pyplot as plt

#====================================================
# Setting up
#====================================================

# Setting file name to read
dataFile = 'rae2822.dat'

# Setting file names to write
fName = 'data'

# Getting current path
mainPath = os.getcwd()

#====================================================
# Reading airfoil geometry file
#====================================================

# Read data
airfoil = np.genfromtxt(dataFile,skip_header=2)

# Separating upper and lower profiles
geomUpper = airfoil[0:65,:]
geomLower = airfoil[65:130,:]

# Sorting the lower portion so that the curve is
# continuous
geomLower = geomLower[np.argsort(-geomLower[:,0])]

# Separating upper and lower curves
x_u = geomUpper[:,0]
y_u = geomUpper[:,1]
x_l = geomLower[:,0]
```

```
y_l = geomLower[:,1]

# Plotting to double check
x = np.concatenate((x_u,x_l))
y = np.concatenate((y_u,y_l))

# Setting up the figure
fig = plt.figure()
figSize = fig.get_size_inches()
fig.set_size_inches((figSize[0],figSize[1]*0.4))

# Plotting
plt.plot(x,y,'-k')

# Adding labels
#plt.xlabel('$x/c$',fontsize=15)
#plt.ylabel('$y/c$',fontsize=15)

# Fixing the axes
plt.axis('equal')
plt.xlim(-0.05,1.05)
plt.ylim(-0.15,0.15)

# Adding labels
plt.xlabel('$x/c$',fontsize=15)
plt.ylabel('$y/c$',fontsize=15)
plt.gcf().subplots_adjust(bottom=0.25)

# Saving figure
fig.savefig('rae2822.eps')

#=====================================================
# Setting up data file for aflr2
#=====================================================

# Initializing list
data = []

# Number of curves
nCurves = 2
data.append(str(nCurves))

# Segments in curves
curve1seg = 2
curve2seg = 1
data.append(str(curve1seg))

# Knots on segment 1 and 2 of curve 1
curve1seg1knots = 65
curve1seg2knots = 65
data.append(str(curve1seg1knots))

#-----------------------------------------------------
# Segment 1
#-----------------------------------------------------

# Index x & y coordinates of knots
for n in range(1,curve1seg1knots+1):
    data.append(str(n) + ' ' + str(x_u[n-1]) + ' ' + str(y_u[n-1]))
```

```
101
102 # Other specs
103 nPts = 600
104 nKnotPts = 2
105 adaptTol = 1
106 nLap = 0
107 lapRelax = 0.5
108 data.append(str(nPts) +' '+ str(nKnotPts) +' '+ str(adaptTol) +' 1.0002 '+ str(nLap) +' ...
        '+ str(lapRelax))
109
110 # Knot control points
111 c = 0;
112 ind = [1,5,10,20,30,50,60,63]
113 spacParam = [5.0e-3,5.0e-2,5.0e-2,5.0e-2,5.0e-2,5.0e-2,5.0e-3,5.0e-3]
114 ind = [1,65]
115 spacParam = [1.0e-4,1.0e-6]
116 for i in ind:
117     data.append(str(i) + ' ' + str(spacParam[c]))
118     c += 1
119
120 #-----------------------------------------------------
121 # Segment 2
122 #-----------------------------------------------------
123
124 # Number of knots
125 data.append(str(curve1seg2knots))
126
127
128 # Index x & y coordinates of knots
129 for n in range(1,curve1seg2knots+1):
130     data.append(str(n) + ' ' + str(x_l[n-1]) + ' ' + str(y_l[n-1]))
131
132 # Other specs
133 adaptTol = 1
134 nLap = 0
135 lapRelax = 0.5
136 data.append(str(nPts) +' '+ str(nKnotPts) +' '+ str(adaptTol) +' 1.0002 '+ str(nLap) +' ...
        '+ str(lapRelax))
137
138 # Knot control points
139 c = 0;
140 #ind = [1,5,10,20,30,50,60,63]
141 spacParam = [1.0e-6,1.0e-4]
142 #spacParam = [5.0e-3,5.0e-2,5.0e-2,5.0e-2,5.0e-2,5.0e-2,5.0e-3,5.0e-3]
143 for i in ind:
144     data.append(str(i) + ' ' + str(spacParam[c]))
145     c += 1
146
147 #-----------------------------------------------------
148 # Outer boundary
149 #-----------------------------------------------------
150
151 # Number of curves in outer boundary
152 data.append(str(1))
153
154 # Number of knots
155 data.append(str(13))
156
157 # index, x, y coordinates of knots
```

```
158 data.append('1  30.00   0.00')
159 data.append('2  25.98 -15.00')
160 data.append('3  15.00 -25.98')
161 data.append('4   0.00 -30.00')
162 data.append('5 -15.00 -25.98')
163 data.append('6 -25.98 -15.00')
164 data.append('7 -30.00   0.00')
165 data.append('8 -25.98  15.00')
166 data.append('9 -15.00  25.98')
167 data.append('10   0.00  30.00')
168 data.append('11  15.00  25.98')
169 data.append('12  25.98  15.00')
170 data.append('13  30.00   0.00')
171
172 # Other stuff
173 data.append('57 2 2 1.00 0 0.5')
174 data.append('1 3.34')
175 data.append('13 3.34')
176
177 #====================================================
178 # Writing out data file
179 #====================================================
180
181 # Counter
182 n = 0
183
184 datafile = open(fName,'w')
185 for line in data:
186     datafile.write(data[n] + '\n')
187     n += 1
188
189 # Closing the file
190 datafile.close()
```

Python batch job script

```
1  #!/usr/bin/python
2
3  from StringIO import StringIO
4  from subprocess import call
5  import os
6  import sys
7  import shutil
8  import numpy as np
9
10 #====================================================
11 # Setting up
12 #====================================================
13
14 # Setting file names to write
15 fName = 'fun3d.nml'         # <--- Namelist input to FUN3D
16 fSub  = 'FUN3D_MPI'     # <--- Bash job submission script
17
18 # Getting current path
```

```
mainPath = os.getcwd()

# Name of baseline directory (contains fun3d.nml and FUN3D_MPI files)
blineDir = mainPath + '/Baseline/'

# Specifying Mach numbers to run
Mstart = 0.5
Mstop  = 0.8
M      = np.linspace(Mstart,Mstop,31)

# Specifying the project root name
rootname = 'RAE2822'

# Name of grid file
gridName = rootname + '.faces'

#=====================================================
# Iteratively setting up and submitting jobs
#=====================================================

i=0

for mNum in M:

    # Creating directories
    dirName = 'M=' + '%.3f' % M[i]
    os.mkdir(dirName)

    # Copying files to new directory
    shutil.copy('./Baseline/fun3d.nml',dirName)
    shutil.copy('./Baseline/FUN3D_MPI',dirName)

    # Changing to new directory
    os.chdir(dirName)

    #=====================================================
    # Opening fun3d.nml and incrementing the Mach number
    #=====================================================

    # Opening file and reading in lines
    with open(fName,'r') as file:
        nmlLines = file.readlines()

    # Overwriting old line that contains Mach number
    nmlLines[14] = '\tmach_number = ' + str(mNum) + '\n'

    # Writing new lines to file
    with open(fName,'w') as file:
        file.writelines(nmlLines)

        #=====================================================
    # Linking grid file
    #=====================================================

    # Grid
    lnCommand1 = blineDir + gridName
    lnCommand2 = os.getcwd() + '/' + gridName
    os.symlink(lnCommand1,lnCommand2)

```

```
78      #=====================================================
79      # Submitting job to queue
80      #=====================================================
81
82      # Creating job name
83      jobName = 'M' + '%.f' % (1000*M[i])
84
85      # Opening file
86      with open(fSub,'r') as file:
87          jobLines = file.readlines()
88
89      # Changing job name (line 6)
90      jobLines[5] = '#$ -N ' + jobName + '\n'
91
92      # Writing new line to file
93      with open(fSub,'w') as file:
94          file.writelines(jobLines)
95
96      # Submitting job
97      call('qsub FUN3D_MPI',shell=True)
98
99      # Going up one level
100     os.chdir(mainPath)
101
102     # Incrementing counter
103     i += 1
```

Python post-processing script

```
1   #!/usr/bin/python
2
3   from StringIO import StringIO
4   from subprocess import call
5   import re
6   import os
7   import sys
8   import numpy as np
9   import matplotlib.pyplot as plt
10  import itertools
11  from matplotlib import rcParams
12  import csv
13  rcParams.update({'figure.autolayout': True})
14
15  #=====================================================
16  # Setting up
17  #=====================================================
18
19  # Setting font size
20  font = {'size': 14}
21
22  # Getting current path
23  mainPath = os.getcwd()
24
25  # Specifying Mach numbers to run
```

```
Mstart = 0.5
Mstop  = 0.8
M      = np.linspace(Mstart,Mstop,31)

# Specifying the project root name
rootname = 'RAE2822'

# Initializing counter
i = 0

# Initializing list for Cd
Cd = []

for mNum in M:

    dirName   = 'M=' + '%.3f' % M[i]
    residFile = mainPath + '/' + dirName + '/' + rootname + '_hist.dat'
    cpFile    = mainPath + '/' + dirName + '/' + rootname + '_slice.dat'
    sliceFile = mainPath + '/' + dirName + '/' + rootname + '.sectional_forces'

    #=====================================================
    # Importing and plotting residuals
    #=====================================================

    # Importing
    data = np.genfromtxt(residFile,delimiter='  ',skip_header=3,usecols=(0,1,2,3,4,5,6,7))

    # Separating
    iter    = data[:,0]
    R1  = data[:,1]
    R2  = data[:,2]
    R3  = data[:,3]
    R4  = data[:,4]
    R5  = data[:,5]
    Clhist  = data[:,6]
    Cdhist  = data[:,7]

    # Creating figure
    fig = plt.figure()

    # First subplot
    ax1 = fig.add_subplot(211)
    plt.plot(iter,R1,label='R1')
    plt.plot(iter,R2,label='R2')
    plt.plot(iter,R3)
    plt.plot(iter,R4)
    plt.plot(iter,R5)
    plt.title('Residuals')
    plt.xlabel('Iteration')
    #plt.legend()
    ax1.set_yscale('log')

    # Second subplot
    ax2 = fig.add_subplot(212)
    plt.plot(iter,Clhist,'-b',label='$C_l$')
    plt.plot(iter,Cdhist,'-r',label='$C_d$')
    plt.legend(loc=4)
    plt.title('$C_l$/$C_d$ convergence')
    plt.xlabel('Iteration')
```

```
85      ax2.set_yscale('log')
86
87      # Adjusting
88      fig.subplots_adjust(wspace=0.35, hspace=0.35)
89
90      # Saving plot
91      pltName = 'M' + '%.f' % (1000*M[i]) + '_resids.eps'
92      fig.savefig('./Images/' + pltName)
93
94      #=====================================================
95      # Importing and plotting Cp slice
96      #=====================================================
97
98      # Counter
99      ii = 0
100
101     # Initializing lists
102     x  = []
103     Cp = []
104
105     # Importingnd strip data
106     f = open(cpFile,'r')
107     for line in itertools.islice(f,2,None):
108         if ii % 2 != 0:
109             line = line.strip()
110             columns = line.split()
111             x.append(float(columns[0]))
112             Cp.append(float(columns[3]))
113         ii += 1
114
115     # Closing data file
116     f.close()
117
118     # Normalize the x-data
119     xStart = min(x)
120     xStop = max(x)
121     xNorm = []
122     for xVal in x:
123         xNorm.append(((xVal - xStart)/(xStop - xStart)))
124
125     # Plotting
126     fig = plt.figure(figsize=(4.5,3))
127     plt.plot(xNorm,Cp)
128     plt.gca().invert_yaxis()
129     plt.xlim([-0.05,1.05])
130
131     # Adding labels
132     plt.xlabel('$x/c$')
133     plt.ylabel('$C_p$')
134     plt.tight_layout()
135
136     # Saving plot
137     pltName = 'M' + '%.f' % (1000*M[i]) + '_Cp.eps'
138     fig.savefig('./Images/' + pltName)
139
140     #=====================================================
141     # Extracting sectional Cd from slice
142     #=====================================================
143
```

```
144     # Open slice file and reading line
145     dline = open(sliceFile,'r').readlines()[30]
146 
147     # Extracting data
148     dline = dline.strip()
149     col = dline.split()
150     Cd.append(float(col[-1]))
151 
152     # Append value to list
153 
154     # Incrementing counter
155     i += 1
156 
157 
158 #=====================================================
159 # Plotting drag divergence
160 #=====================================================
161 
162 # Setting up figure
163 fig = plt.figure()
164 
165 # Plotting
166 plt.plot(M,Cd,'ob')
167 
168 # Adding labels
169 plt.xlabel('$M_n$',fontdict=font)
170 plt.ylabel('$C_d$',fontdict=font)
171 
172 # Saving plot
173 fig.savefig('./Images/2D_RAE2822_DragDiv.eps')
```