Appendix A: Gambit and Fluent Journal Files

To automate execution of the Gambit and Fluent journal files, an executable script file was used as the simulation code for optimization within iSIGHT. The script file runs Gambit to create and mesh the solution domain, runs Fluent to iterate toward a converged solution, and performs some simple file management. Fluent is executed twice to enable solution on parallel processors. In the first Fluent session, the solution domain is divided into four nearly equal partitions. For the second Fluent session, the parallel solver is invoked to iterate toward a solution. Iteration on multiple processors reduces the amount of wall clock time to achieve a solution.

Begin Executable Script

```
rm run.*
rm vane.*
gambit -in gambit.jou
fluent 3d -64 -i fluent1.jou
rm meantemp.hist
rm meansquare.hist
fluent 3d -64 –t4 -i fluent2.jou
rm lastrun.*
cp meantemp.hist lastrunmt.hist
cp meansquare.hist lastrunms.hist
cp run.jou lastrun.jou
cp vane.cas lastrun.cas
cp vane.dat lastrun.dat
exit
```

End Executable Script

The Gambit journal file defines and creates the turbine vane and fillet geometry and meshes the solution domain. Several capabilities of the Gambit journal file made it possible to automate solution domain creation. The first and perhaps most important feature of Gambit is that it allows for the definition of variables within the journal file. Variables are denoted by a dollar sign before the variable name. Matrices can also be declared as variables and are particularly convenient for storing X, Y, Z coordinate information. The second key capability of the Gambit journal file is the use of algebraic expressions. Algebraic expressions enable a variable to be defined as a function of other variables. Gambit also features many built-in functions that can be used in expressions, including the trigonometric functions. Finally, Gambit has basic logic capability. The

use of logical expressions enabled much greater freedom in leading edge fillet geometry. These enabling capabilities are highlighted in the journal file excerpt below.

Begin gambit.jou

```
$pitch=0.457000
$midspan=0.274320
$chord=0.594000
$exitalpha=78.01
$manfil=0.01143
$sloth=0.016000
$steph=0.009000

/ Limits for $smaxps and $smaxss
/ -0.514281 < $smaxps < -0.053447
/ 0.064534 < $smaxss < 0.337139
$smaxps=-0.514281
$smaxss=0.337139

/ Variable for fillet height.
$maxh=0.095

/ Variable for fillet extent.
$maxd=0.095

/ Location of maximum fillet height.
/ -0.403365 < $smaxh < 0.170696
$smaxh=0.068523

/ Location of maximum fillet extent.
/ -0.403365 < $smaxd < 0.170696
$smaxd=0.034262

/ Angle of vane surface outward normals.
declare $angle[1:146]

$angle[1]=67.228892
$angle[2]=74.594053
------------------------------------------------------------------------------------------------------------------------
$angle[146]=67.228892

/ Length along the vane surface.
declare $s[1:146]

$s[1]=-0.530284
$s[2]=-0.529575
------------------------------------------------------------------------------------------------------------------------
$s[146]=0.766221

declare $v[1:146,1:3]
declare $fe[1:146,1:3]

$v[1,1]= 0.284760
$v[1,2]= 0.442228
$v[1,3]=0.000000+$manfil
vertex create coordinates $v[1,1] $v[1,2] $v[1,3]
```

```
$fe[1,1]= 0.284760+$manfil*COS($angle[1])
$fe[1,2]= 0.442228+$manfil*SIN($angle[1])
$fe[1,3]=0.000000
vertex create coordinates $fe[1,1] $fe[1,2] $fe[1,3]
vertex create coordinates $fe[1,1] $fe[1,2] $v[1,3]

$v[2,1]=0.284097
$v[2,2]=0.442434
$v[2,3]=0.000000+$manfil
vertex create coordinates $v[2,1] $v[2,2] $v[2,3]

$fe[2,1]=0.284097+$manfil*COS($angle[2])
$fe[2,2]=0.442434+$manfil*SIN($angle[2])
$fe[2,3]=0.000000
vertex create coordinates $fe[2,1] $fe[2,2] $fe[2,3]
vertex create coordinates $fe[2,1] $fe[2,2] $v[2,3]
-------------------------------------------------------------------------------------------------------------------------------
$v[17,1]=0.274747
$v[17,2]=0.432330
$v[17,3]=0.000000+$manfil
vertex create coordinates $v[17,1] $v[17,2] $v[17,3]

$fe[17,1]=0.274747+$manfil*COS($angle[17])
$fe[17,2]=0.432330+$manfil*SIN($angle[17])
$fe[17,3]=0.000000
vertex create coordinates $fe[17,1] $fe[17,2] $fe[17,3]
vertex create coordinates $fe[17,1] $fe[17,2] $v[17,3]

$v[18,1]=0.273238
$v[18,2]=0.428924
$v[18,3]=0.000000+$manfil

$fe[18,1]=0.273238+$manfil*COS($angle[18])
$fe[18,2]=0.428924+$manfil*SIN($angle[18])
$fe[18,3]=0.000000

IF COND ($s[18] .LE. $smaxps)
  vertex create coordinates $v[18,1] $v[18,2] $v[18,3]
  vertex create coordinates $fe[18,1] $fe[18,2] $fe[18,3]
  vertex create coordinates $fe[18,1] $fe[18,2] $v[18,3]
ENDIF

$v[18,1]=0.273238
$v[18,2]=0.428924
$v[18,3]=0.000000+$manfil+0.5*($maxh-$manfil)*(COS(180*($smaxh -$s[18])/($smaxh -$smaxps))+1)
$fe[18,1]=0.273238+($manfil+0.5*($maxd-$manfil)*(COS(180*($smaxd-$s[18])/($smaxd-
          $smaxps))+1))*COS($angle[18])
$fe[18,2]=0.428924+($manfil+0.5*($maxd-$manfil)*(COS(180*($smaxd-$s[18])/($smaxd-
          $smaxps))+1))*SIN($angle[18])
$fe[18,3]=0.000000

IF COND ($s[18] .GT. $smaxps)
  vertex create coordinates $v[18,1] $v[18,2] $v[18,3]
  vertex create coordinates $fe[18,1] $fe[18,2] $fe[18,3]
  vertex create coordinates $fe[18,1] $fe[18,2] $v[18,3]
```

```
ENDIF
-----------------------------------------------------------------------------------------------------------------------------
$v[32,1]=0.225278
$v[32,2]=0.333095
$v[32,3]=0.000000+$manfil+0.5*($maxh-$manfil)*(COS(180*($smaxh -$s[32])/($smaxh -$smaxps))+1)
$fe[32,1]=0.225278+($manfil+0.5*($maxd-$manfil)*(COS(180*($smaxd-$s[32])/($smaxd-
        $smaxps))+1))*COS($angle[32])
$fe[32,2]=0.333095+($manfil+0.5*($maxd-$manfil)*(COS(180*($smaxd-$s[32])/($smaxd-
        $smaxps))+1))*SIN($angle[32])
$fe[32,3]=0.000000

IF COND ($s[32] .LE. $smaxd)
        IF COND ($s[32] .LE. $smaxh)
                vertex create coordinates $v[32,1] $v[32,2] $v[32,3]
                vertex create coordinates $fe[32,1] $fe[32,2] $fe[32,3]
                vertex create coordinates $fe[32,1] $fe[32,2] $v[32,3]
        ENDIF
ENDIF

$v[32,3]=0.000000+$manfil+0.5*($maxh-$manfil)*(COS(180*($s[32]-$smaxh)/($smaxss-$smaxh))+1)

IF COND ($s[32] .LE. $smaxd)
        IF COND ($s[32] .GT. $smaxh)
                vertex create coordinates $v[32,1] $v[32,2] $v[32,3]
                vertex create coordinates $fe[32,1] $fe[32,2] $fe[32,3]
                vertex create coordinates $fe[32,1] $fe[32,2] $v[32,3]
        ENDIF
ENDIF

$v[32,3]=0.000000+$manfil+0.5*($maxh-$manfil)*(COS(180*($smaxh-$s[32])/($smaxh-$smaxps))+1)
$fe[32,1]=0.225278+($manfil+0.5*($maxd-$manfil)*(COS(180*($s[32]-$smaxd)/($smaxss-
        $smaxd))+1))*COS($angle[32])
$fe[32,2]=0.333095+($manfil+0.5*($maxd-$manfil)*(COS(180*($s[32]-$smaxd)/($smaxss-
        $smaxd))+1))*SIN($angle[32])

IF COND ($s[32] .GT. $smaxd)
        IF COND ($s[32] .LE. $smaxh)
                vertex create coordinates $v[32,1] $v[32,2] $v[32,3]
                vertex create coordinates $fe[32,1] $fe[32,2] $fe[32,3]
                vertex create coordinates $fe[32,1] $fe[32,2] $v[32,3]
        ENDIF
ENDIF

$v[32,3]=0.000000+$manfil+0.5*($maxh-$manfil)*(COS(180*($s[32]-$smaxh)/($smaxss-$smaxh))+1)

IF COND ($s[32] .GT. $smaxd)
        IF COND ($s[32] .GT. $smaxh)
                vertex create coordinates $v[32,1] $v[32,2] $v[32,3]
                vertex create coordinates $fe[32,1] $fe[32,2] $fe[32,3]
                vertex create coordinates $fe[32,1] $fe[32,2] $v[32,3]
        ENDIF
ENDIF
-----------------------------------------------------------------------------------------------------------------------------
$v[76,1]=0.003193
$v[76,2]=-0.063184+$pitch
$v[76,3]=0.000000+$manfil+0.5*($maxh-$manfil)*(COS(180*($smaxh -$s[76])/($smaxh -$smaxps))+1)
```

```
$fe[76,1]=0.003193+($manfil+0.5*($maxd-$manfil)*(COS(180*($smaxd-$s[76])/($smaxd-
          $smaxps))+1))*COS($angle[76])
$fe[76,2]=-0.063184+$pitch+($manfil+0.5*($maxd-$manfil)*(COS(180*($smaxd-$s[76])/($smaxd-
          $smaxps))+1))*SIN($angle[76])
$fe[76,3]=0.000000

IF COND ($s[76] .LE. $smaxd)
        IF COND ($s[76] .LE. $smaxh)
                vertex create coordinates $v[76,1] $v[76,2] $v[76,3]
                vertex create coordinates $fe[76,1] $fe[76,2] $fe[76,3]
                vertex create coordinates $fe[76,1] $fe[76,2] $v[76,3]
        ENDIF
ENDIF

$v[76,3]=0.000000+$manfil+0.5*($maxh-$manfil)*(COS(180*($s[76]-$smaxh)/($smaxss-$smaxh))+1)

IF COND ($s[76] .LE. $smaxd)
        IF COND ($s[76] .GT. $smaxh)
                vertex create coordinates $v[76,1] $v[76,2] $v[76,3]
                vertex create coordinates $fe[76,1] $fe[76,2] $fe[76,3]
                vertex create coordinates $fe[76,1] $fe[76,2] $v[76,3]
        ENDIF
ENDIF

$v[76,3]=0.000000+$manfil+0.5*($maxh-$manfil)*(COS(180*($smaxh-$s[76])/($smaxh-$smaxps))+1)
$fe[76,1]=0.003193+($manfil+0.5*($maxd-$manfil)*(COS(180*($s[76]-$smaxd)/($smaxss-
          $smaxd))+1))*COS($angle[76])
$fe[76,2]=-0.063184+$pitch+($manfil+0.5*($maxd-$manfil)*(COS(180*($s[76]-$smaxd)/($smaxss-
          $smaxd))+1))*SIN($angle[76])

IF COND ($s[76] .GT. $smaxd)
        IF COND ($s[76] .LE. $smaxh)
                vertex create coordinates $v[76,1] $v[76,2] $v[76,3]
                vertex create coordinates $fe[76,1] $fe[76,2] $fe[76,3]
                vertex create coordinates $fe[76,1] $fe[76,2] $v[76,3]
        ENDIF
ENDIF

$v[76,3]=0.000000+$manfil+0.5*($maxh-$manfil)*(COS(180*($s[76]-$smaxh)/($smaxss-$smaxh))+1)

IF COND ($s[76] .GT. $smaxd)
        IF COND ($s[76] .GT. $smaxh)
                vertex create coordinates $v[76,1] $v[76,2] $v[76,3]
                vertex create coordinates $fe[76,1] $fe[76,2] $fe[76,3]
                vertex create coordinates $fe[76,1] $fe[76,2] $v[76,3]
        ENDIF
ENDIF
-------------------------------------------------------------------------------------------------------------------------------
$v[146,1]=0.284760
$v[146,2]=0.442228+$pitch
$v[146,3]=0.000000+$manfil
vertex create coordinates $v[146,1] $v[146,2] $v[146,3]

$fe[146,1]=0.284760+$manfil*COS($angle[146])
$fe[146,2]=0.442228+$pitch+$manfil*SIN($angle[146])
$fe[146,3]=0.000000
```

```
vertex create coordinates $fe[146,1] $fe[146,2] $fe[146,3]
vertex create coordinates $fe[146,1] $fe[146,2] $v[146,3]

declare $vmid[1:146,1:3]

$vmid[1,1]= 0.284760
$vmid[1,2]= 0.442228
$vmid[1,3]=0.000000+$midspan

$vmid[2,1]=0.284097
$vmid[2,2]=0.442434
$vmid[2,3]=0.000000+$midspan
-------------------------------------------------------------------------------------------------------------------------------
$vmid[146,1]=0.284760
$vmid[146,2]=0.442228+$pitch
$vmid[146,3]=0.000000+$midspan

declare $ext[1:30,1:3]

$ext[1,1]=$vmid[68,1]-$chord
$ext[1,2]=$vmid[68,2]+0.200000*SIN($angle[68])
$ext[1,3]=0.000000

$ext[2,1]=$vmid[68,1]+0.200000*COS($angle[68])
$ext[2,2]=$vmid[68,2]+0.200000*SIN($angle[68])
$ext[2,3]=0.000000
-------------------------------------------------------------------------------------------------------------------------------
$ext[18,1]=$fe[146,1]+1.5*$chord*COS($exitalpha)+0.1*$chord
$ext[18,2]=$fe[146,2]+1.5*$chord*SIN($exitalpha)
$ext[18,3]=0.000000+$midspan

/ Vertices at midspan.
vertex create coordinates $vmid[1,1] $vmid[1,2] $vmid[1,3]
vertex create coordinates $vmid[2,1] $vmid[2,2] $vmid[2,3]
-------------------------------------------------------------------------------------------------------------------------------
vertex create coordinates $vmid[146,1] $vmid[146,2] $vmid[146,3]

/ Vertices defining the extent of the solution domain.
vertex create coordinates $ext[1,1] $ext[1,2] $ext[1,3]
vertex create coordinates $ext[2,1] $ext[2,2] $ext[2,3]
-------------------------------------------------------------------------------------------------------------------------------
vertex create coordinates $ext[18,1] $ext[18,2] $ext[18,3]

/ Create edges.
edge create nurbs "vertex.1" "vertex.4" "vertex.7" "vertex.10" "vertex.13" \
  "vertex.16" "vertex.19" "vertex.22" "vertex.25" "vertex.28" "vertex.31" \
  "vertex.34" "vertex.37" "vertex.40" "vertex.43" "vertex.46" "vertex.49" \
  interpolate
edge create nurbs "vertex.49" "vertex.52" "vertex.55" "vertex.58" "vertex.61" \
  "vertex.64" "vertex.67" "vertex.70" "vertex.73" "vertex.76" "vertex.79" \
  "vertex.82" "vertex.85" "vertex.88" "vertex.91" "vertex.94" "vertex.97" \
  "vertex.100" "vertex.103" "vertex.106" "vertex.109" "vertex.112" \
  "vertex.115" "vertex.118" "vertex.121" "vertex.124" "vertex.127" \
  "vertex.130" "vertex.133" "vertex.136" "vertex.139" "vertex.142" \
  "vertex.145" "vertex.148" "vertex.151" "vertex.154" "vertex.157" \
  "vertex.160" "vertex.163" "vertex.166" "vertex.169" "vertex.172" \
```

```
  "vertex.175" "vertex.178" "vertex.181" "vertex.184" "vertex.187" \
  "vertex.190" "vertex.193" "vertex.196" "vertex.199" "vertex.202" \
  interpolate
-------------------------------------------------------------------------------------------------------------------------------
edge create straight "vertex.506" "vertex.202"

/ Define fillet profile edges.
edge create center "vertex.207" major "vertex.206" onedge "vertex.205" start \
  0 end 90 ellipse
edge create center "vertex.210" major "vertex.209" onedge "vertex.208" start \
  0 end 90 ellipse
-------------------------------------------------------------------------------------------------------------------------------
edge create center "vertex.3" major "vertex.2" onedge "vertex.1" start 0 end \
  90 ellipse

/ Define domain extent edges.
edge create straight "vertex.203" "vertex.586"
edge create straight "vertex.586" "vertex.585"
-------------------------------------------------------------------------------------------------------------------------------
edge create straight "vertex.590" "vertex.599"

/  Create faces.
face create uedges "edge.7" "edge.1" udirections 0 0 vedges "edge.172" \
  "edge.171" "edge.170" "edge.169" "edge.168" "edge.167" "edge.166" \
  "edge.165" "edge.164" "edge.163" "edge.162" "edge.161" "edge.160" \
  "edge.159" "edge.158" "edge.157" "edge.156" vdirections 0 0 0 0 0 0 0 0 0 0 \
  0 0 0 0 0 0 0 net
face create uedges "edge.8" "edge.2" udirections 0 0 vedges "edge.156" \
  "edge.155" "edge.154" "edge.153" "edge.152" "edge.151" "edge.150" \
  "edge.149" "edge.148" "edge.147" "edge.146" "edge.145" "edge.144" \
  "edge.143" "edge.142" "edge.141" "edge.140" "edge.139" "edge.138" \
  "edge.137" "edge.136" "edge.135" "edge.134" "edge.133" "edge.132" \
  "edge.131" "edge.130" "edge.129" "edge.128" "edge.127" "edge.126" \
  "edge.125" "edge.124" "edge.123" "edge.122" "edge.121" "edge.120" \
  "edge.119" "edge.118" "edge.117" "edge.116" "edge.115" "edge.114" \
  "edge.113" "edge.112" "edge.111" "edge.110" "edge.109" "edge.108" \
  "edge.107" "edge.106" "edge.105" vdirections 0 0 0 0 0 0 0 0 0 0 0 0 0 0 \
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 net
-------------------------------------------------------------------------------------------------------------------------------
face create wireframe "edge.13" "edge.14" "edge.183" "edge.184" "edge.185" \
  "edge.186" "edge.187" "edge.15" "edge.16" "edge.17" "edge.18" "edge.188" \
  "edge.189" "edge.190" "edge.191" "edge.192" "edge.193" "edge.194" real

/ Create volume.
volume create stitch "face.1" "face.2" "face.3" "face.4" "face.5" "face.6" \
  "face.7" "face.8" "face.9" "face.10" "face.11" "face.12" "face.13" \
  "face.14" "face.15" "face.16" "face.17" "face.18" "face.19" "face.20" \
  "face.21" "face.22" "face.23" "face.24" "face.25" "face.26" virtual

/ Link faces for meshing.
face link "face.13" "face.17" edges "edge.201" "edge.187" vertices \
  "vertex.594" "vertex.599" reverse
face link "face.14" "face.16" edges "edge.184" "edge.204" vertices \
  "vertex.594" "vertex.599" reverse
face link "face.24" "face.18" edges "edge.194" "edge.195" vertices \
  "vertex.595" "vertex.600" reverse
```

```
face link "face.23" "face.19" edges "edge.193" "edge.196" vertices \
  "vertex.596" "vertex.601" reverse
face link "face.22" "face.20" edges "edge.199" "edge.190" vertices \
  "vertex.596" "vertex.601" reverse

/ Mesh edges.
edge mesh "edge.7" "edge.8" "edge.1" "edge.2" "edge.172" "edge.156" \
  "edge.105" "edge.26" "edge.14" "edge.13" "edge.24" "edge.25" "edge.200" \
  "edge.194" successive ratio1 1 size 0.004
edge mesh "edge.9" "edge.3" "edge.10" "edge.4" "edge.42" "edge.19" "edge.15" \
  "edge.16" "edge.21" "edge.59" "edge.11" "edge.5" "edge.17" "edge.91" \
  "edge.22" "edge.12" "edge.6" "edge.18" successive ratio1 1 size 0.004
edge mesh "edge.173" "edge.174" "edge.175" successive ratio1 1 size 0.004
edge mesh "edge.201" "edge.202" firstlast ratio1 0.3 intervals 35
edge mesh "edge.183" firstlast ratio1 0.25 intervals 28
edge mesh "edge.184" "edge.185" successive ratio1 1 size 0.0125
edge mesh "edge.182" "edge.193" lastfirst ratio1 0.2 intervals 85
edge mesh "edge.181" "edge.199" "edge.192" "edge.198" "edge.180" "edge.191" \
  successive ratio1 1 size 0.02

/ Mesh faces.
face mesh "face.1" "face.2" "face.3" "face.4" "face.5" "face.6" "face.8" \
  "face.9" "face.10" "face.11" triangle size 1
face mesh "face.7" "face.12" map size 1
face mesh "face.14" "face.13" triangle size 1
face mesh "face.23" "face.24" "face.22" triangle size 1
face mesh "face.21" "face.25" "face.15" "face.26" triangle size 1

/ Mesh volume.
volume mesh "v_volume.1" tetrahedral size 1

/ Select flow solver and apply boundary types.
solver select "FLUENT 5"
physics create "fluid.1" ctype "FLUID" volume "v_volume.1"
physics create "inlet" btype "VELOCITY_INLET" face "face.15"
physics create "pressure_side" btype "WALL" face "face.7" "face.8"
physics create "suction_side" btype "WALL" face "face.9" "face.10" "face.11" "face.12"
physics create "pressure_fillet" btype "WALL" face "face.1" "face.2"
physics create "suction_fillet" btype "WALL" face "face.3" "face.4" "face.5" "face.6"
physics create "endwall" btype "WALL" face "face.25"
physics create "midspan" btype "SYMMETRY" face "face.26"
physics create "outflow" btype "OUTFLOW" face "face.21"

save name "run"

export uns "vane.msh"
```

End gambit.jou

The first Fluent journal file imports the mesh file created in Gambit and defines the solver formulation. In addition to the solver formulation, the turbulence modeling approach, fluid properties, discretization schemes, and convergence criteria are also

specified. Finally, the solution domain is partitioned to enable parallel solution on multiple processors.

Begin fluent1.jou

```
;Read in the mesh file
file read-case vane.msh

;Define the solver to be utilized
define/models/solver segregated
;Enable segregated solver? [yes]
yes

;Specify turbulence model.
define/models/viscous rng-ke
;Enable the RNG k-epsilon turbulence model? [no]
yes
define/models/viscous noneq-wall-fn?
;Enable non-equilibrium wall functions? [no]
yes

;Turn on the energy equation
define/models energy?
;Enable energy model? [no]
yes
;Compute viscous energy dissipation? [no]
no
;include pressure work in energy equation? [no]
no
;include kinetic energy in energy equation? [no]
no
;Include diffusion at inlets? [yes]
yes

;Specify fluid properties and pertinent property models.
define/materials change-create air
;material name [air]
air
;air is a fluid
;Change density? [no]
yes
;new method [constant]
incompressible-ideal-gas
;change Cp (Specific Heat)? [no]
no
;change Thermal Conductivity? [no]
no
;change Viscosity? [no]
no
;change Molecular Weight? [no]
no
;change L-J Characteristic Length? [no]
no
;change L-J Energy Parameter? [no]
no
;change Thermal Expansion Coefficient? [no]
```

```
no
;change Degrees of Freedom? [no]
no

;Set operating pressure to 101325 Pa.
define/operating-conditions operating-pressure
;operating pressure (pascal) [101325]
95500

;Set reference pressure location.
define/operating-conditions reference-pressure-location
;x-coordinate (m) [0]
-0.594305
;y-coordinate (m) [0]
0.284861
;z-coordinate (m) [0]
0.137160

;Make periodic boundaries.
grid/modify-zones make-periodic
;Periodic zone [()]
periodic1
;Shadow zone [()]
periodic11
;Rotational periodic? (if no,translational) [yes]
no
;Create periodic zones? [yes]
yes

grid/modify-zones make-periodic
;Periodic zone [()]
periodic2
;Shadow zone [()]
periodic22
;Rotational periodic? (if no,translational) [yes]
no
;Create periodic zones? [yes]
yes

grid/modify-zones make-periodic
;Periodic zone [()]
periodic3
;Shadow zone [()]
periodic33
;Rotational periodic? (if no,translational) [yes]
no
;Create periodic zones? [yes]
yes

grid/modify-zones make-periodic
;Periodic zone [()]
periodic4
;Shadow zone [()]
periodic44
;Rotational periodic? (if no,translational) [yes]
no
```

```
;Create periodic zones? [yes]
yes

grid/modify-zones make-periodic
;Periodic zone [()]
periodic5
;Shadow zone [()]
periodic55
;Rotational periodic? (if no,translational) [yes]
no
;Create periodic zones? [yes]
yes

;Select equations to solve.
solve/set/equations flow
;Solve Flow equation(s)? [yes]
yes
solve/set/equations temperature
;Solve Energy equation(s)? [yes]
yes
solve/set/equations ke
;Solve Turbulence equation(s)? [yes]
yes

;Set the discretization schemes.\
solve/set/discretization-scheme pressure
;Convective discretization scheme for Pressure [10]
10
solve/set/discretization-scheme flow
;Convective discretization scheme for Pressure-Velocity Coupling [20]
20
solve/set/discretization-scheme mom
;Convective discretization scheme for Momentum [0]
1
solve/set/discretization-scheme temperature
;Convective discretization scheme for Energy [0]
1
solve/set/discretization-scheme k
;Convective discretization scheme for Turbulence Kinetic Energy [0]
1
solve/set/discretization-scheme epsilon
;Convective discretization scheme for Turbulence Dissipation Rate [0]
1

;Set underrelaxation factors.
solve/set/under-relaxation pressure
;Underrelaxation factor for Pressure [0.3]
0.3
solve/set/under-relaxation mom
;Underrelaxation factor for Momentum [0.7]
0.7
solve/set/under-relaxation temperature
;Underrelaxation factor for Energy [1]
1
solve/set/under-relaxation k
;Underrelaxation factor for Turbulence Kinetic Energy [0.8]
```

```
0.8
solve/set/under-relaxation epsilon
;Underrelaxation factor for Turbulence Dissipation Rate [0.8]
0.8
solve/set/under-relaxation turb-viscosity
;Underrelaxation factor for Viscosity [1]
1
solve/set/under-relaxation density
;Underrelaxation factor for Density [1]
1
solve/set/under-relaxation body-force
;Underrelaxation factor for Body Forces [1]
1

;Set the convergence criteria.
solve/monitors/residual convergence-criteria
;continuity residual convergence criterion [0.001]
0.00001
;x-velocity residual convergence criterion [0.001]
0.00001
;y-velocity residual convergence criterion [0.001]
0.00001
;z-velocity residual convergence criterion [0.001]
0.00001
;energy residual convergence criterion [1e-06]
1e-06
;k residual convergence criterion [0.001]
0.00001
;epsilon residual convergence criterion [0.001]
0.00001

;Partition the grid
grid/partition/set merge
;attempt to merge small connected regions? [no]
yes
;maximum merge iterations [3]
10
grid/partition/set smooth
;smooth during partitioning? [yes]
yes
;maximum smoothing iterations [3]
10
grid/partition/set pre-test
;pre-test coordinate direction before bisection? [no]
yes
grid/partition/set verbosity
;0 for quiet, 1 for limited, 2 for noisy [1]
2
grid/partition/set across-zones
;partition across zone boundaris? [yes]
yes
grid/partition method
;partition by>
cartesian-x
;number of partitions [1]
4
```

```
;Write settings to a case file.
file/write-case
;case file name [""]
"vane.cas"

;Exit Fluent.
exit
```

End fluent1.jou

To invoke the parallel solver, Fluent must be launched with the proper parallel command. In the second Fluent journal file, inlet boundary conditions are applied and the partitioned solution domain is distributed to multiple processors for solution.

Begin fluent2.jou

```
;Read in the case file
file read-case vane.cas

;Read in inlet boundary profile.
file read-profile inlet.prof

;Set boundary conditions at inlet.
define/boundary-conditions velocity-inlet
;zone id/name [inlet]
inlet
;Velocity Specification Method: Magnitude and Direction [no]
no
;Velocity Specification Method: Components [no]
yes
;Reference Frame: Absolute [yes]
yes
;Coordinate System: Cartesian (X, Y, Z) [yes]
yes
;Use Profile for X-Velocity? [no]
yes
;profile name [""]
"inlet-profile"
;data name [""]
"u"
;Use Profile for Y-Velocity? [no]
no
;Y-Velocity (m/s) [0]
0
;Use Profile for Z-Velocity? [no]
no
;Z-Velocity (m/s) [0]
0
;Use Profile for Temperature? [no]
yes
;profile name [""]
"inlet-profile"
;data name [""]
"temp"
```

```
;Turbulence Specification Method: K and Epsilon [yes]
no
;Turbulence Specification Method: Intensity and Length Scale [no]
yes
;Turbulence intensity (%) [10]
1
;Turbulence Length Scale (m) [1]
0.1

;Set reference values.
report/reference-values/compute velocity-inlet
;Zone id/name [inlet]
inlet

;Initialize solution domain to velocity inlet conditions.
solve/initialize/compute-defaults velocity-inlet
;Zone id/name [inlet]
inlet

solve/monitors/residual plot?
;Plot residuals? [no]
yes

solve iterate
;Number of iterations [1]
700

;Write case and data files.
file write-case-data
;case file name [""]
"vane.cas"
;The following files already exist:
;     " vane.cas"
;OK to overwrite? [no]
yes

;Exit Fluent.
exit
```

End fluent2.jou

Appendix B: Vane Geometry and Data

This appendix contains the geometric coordinates of the turbine vane, provided by Pratt and Whitney (Johnson, 1996). In addition, the Mach number, temperature, and pressure distributions around the vane were also provided.

Stagnation Temperature: 1666K					
Stagnation Pressure: 103.4E+05 Pa					
Gamma: 1.31					
Pressure surface					
X (cm)	**Y (cm)**	**s/C**	**p/p_o**	**T/T_o**	**Ma**
-1.4635	0.8329	-0.0018	1.0001	1.0000	0.0038
-1.4300	0.7381	-0.0135	0.9995	0.9999	0.0197
-1.3881	0.6467	-0.0287	0.9988	0.9998	0.0388
-1.3429	0.5565	-0.0440	0.9984	0.9997	0.0455
-1.2974	0.4669	-0.0592	0.9978	0.9996	0.0535
-1.2502	0.3777	-0.0745	0.9968	0.9993	0.0663
-1.2012	0.2880	-0.0900	0.9960	0.9991	0.0744
-1.1491	0.1956	-0.1061	0.9949	0.9989	0.0841
-1.0937	0.1006	-0.1227	0.9937	0.9986	0.0937
-1.0351	0.0036	-0.1399	0.9924	0.9983	0.1034
-0.9731	-0.0955	-0.1576	0.9909	0.9980	0.1132
-0.9078	-0.1958	-0.1758	0.9893	0.9977	0.1232
-0.8390	-0.2977	-0.1943	0.9875	0.9973	0.1332
-0.7673	-0.4006	-0.2134	0.9856	0.9968	0.1435
-0.6922	-0.5044	-0.2328	0.9835	0.9963	0.1540
-0.6142	-0.6091	-0.2525	0.9811	0.9958	0.1647
-0.5334	-0.7145	-0.2727	0.9786	0.9952	0.1757
-0.4501	-0.8207	-0.2931	0.9758	0.9946	0.1870
-0.3642	-0.9279	-0.3139	0.9728	0.9939	0.1987
-0.2766	-1.0358	-0.3350	0.9694	0.9931	0.2110
-0.1875	-1.1448	-0.3563	0.9657	0.9923	0.2238
-0.0975	-1.2543	-0.3778	0.9615	0.9913	0.2374
-0.0081	-1.3637	-0.3992	0.9569	0.9903	0.2517
0.0803	-1.4735	-0.4206	0.9517	0.9891	0.2670
0.1676	-1.5829	-0.4418	0.9459	0.9877	0.2832
0.2532	-1.6927	-0.4629	0.9394	0.9862	0.3005
0.3371	-1.8026	-0.4838	0.9320	0.9845	0.3191
0.4188	-1.9126	-0.5046	0.9238	0.9825	0.3387
0.4983	-2.0226	-0.5252	0.9147	0.9804	0.3596
0.5756	-2.1323	-0.5455	0.9047	0.9779	0.3816
0.6502	-2.2418	-0.5656	0.8937	0.9753	0.4046
0.7224	-2.3510	-0.5854	0.8818	0.9723	0.4284
0.7920	-2.4592	-0.6049	0.8691	0.9692	0.4530
0.8588	-2.5667	-0.6240	0.8556	0.9658	0.4781
0.9228	-2.6726	-0.6428	0.8415	0.9622	0.5036
0.9843	-2.7770	-0.6612	0.8268	0.9584	0.5293
1.0429	-2.8796	-0.6791	0.8118	0.9544	0.5551
1.0991	-2.9799	-0.6965	0.7964	0.9503	0.5807

X (cm)	Y (cm)	s/C	p/p_o	T/T_o	Ma
1.1521	-3.0777	-0.7134	0.7810	0.9461	0.6060
1.2027	-3.1727	-0.7297	0.7656	0.9419	0.6309
1.2504	-3.2647	-0.7453	0.7503	0.9376	0.6552
1.2957	-3.3531	-0.7604	0.7354	0.9333	0.6788
1.3378	-3.4379	-0.7747	0.7208	0.9291	0.7015
1.3774	-3.5184	-0.7884	0.7068	0.9250	0.7232
1.4143	-3.5946	-0.8012	0.6935	0.9210	0.7438
1.4483	-3.6662	-0.8132	0.6809	0.9172	0.7632
1.4796	-3.7330	-0.8243	0.6691	0.9135	0.7815
1.5077	-3.7945	-0.8346	0.6580	0.9101	0.7985
1.5334	-3.8504	-0.8439	0.6479	0.9069	0.8140
1.5560	-3.9002	-0.8522	0.6381	0.9037	0.8290
1.5758	-3.9444	-0.8595	0.6272	0.9002	0.8458
1.5926	-3.9822	-0.8658	0.6156	0.8963	0.8638
1.6063	-4.0132	-0.8710	0.6054	0.8929	0.8796
1.6172	-4.0371	-0.8749	0.5942	0.8891	0.8970
1.6256	-4.0559	-0.8780	0.5846	0.8858	0.9121
1.6289	-4.0627	-0.8792	0.5808	0.8844	0.9181
1.6322	-4.0693	-0.8803	0.5745	0.8823	0.9279
1.6365	-4.0754	-0.8814	0.5844	0.8857	0.9124
1.6413	-4.0810	-0.8826	0.5960	0.8897	0.8942
1.6472	-4.0856	-0.8837	0.6019	0.8917	0.8851
1.6533	-4.0897	-0.8848	0.6056	0.8930	0.8794
1.6599	-4.0930	-0.8859	0.6069	0.8934	0.8773
1.6670	-4.0952	-0.8870	0.6071	0.8935	0.8771
1.6744	-4.0965	-0.8882	0.6099	0.8944	0.8727
1.6817	-4.0968	-0.8893	0.6156	0.8964	0.8637
1.6891	-4.0963	-0.8904	0.6240	0.8991	0.8509
1.6965	-4.0945	-0.8915	0.6243	0.8992	0.8503
Suction surface					
X (cm)	**Y (cm)**	**s/C**	**p/p_o**	**T/T_o**	**Ma**
-1.4877	0.9307	0.0170	0.9996	0.0164	0.0164
-1.5019	1.0305	0.0323	0.9984	0.0450	0.0450
-1.5062	1.1311	0.0476	0.0000	0.0626	0.0626
-1.5006	1.2316	0.0628	0.9964	0.0705	0.0705
-1.4849	1.3310	0.0781	0.9849	0.1472	0.1472
-1.4595	1.4282	0.0933	0.9997	0.1550	0.1550
-1.4247	1.5235	0.1086	0.9795	0.1720	0.1720
-1.3805	1.6139	0.1239	0.9652	0.2256	0.2256
-1.3277	1.6995	0.1392	0.9548	0.2580	0.2580
-1.2667	1.7798	0.1544	0.9408	0.2968	0.2968
-1.1976	1.8529	0.1697	0.9221	0.3427	0.3427
-1.1184	1.9152	0.1849	0.8864	0.4193	0.4193
-1.0287	1.9611	0.2002	0.8509	0.4867	0.4867
-0.9319	1.9891	0.2154	0.8241	0.5340	0.5340
-0.8319	1.9995	0.2307	0.8023	0.5709	0.5709
-0.7313	1.9939	0.2460	0.7849	0.5997	0.5997
-0.6482	1.9787	0.2588	0.7766	0.6132	0.6132
-0.5616	1.9535	0.2724	0.7693	0.6250	0.6250
-0.4702	1.9169	0.2874	0.7605	0.6391	0.6391
-0.3749	1.8694	0.3035	0.7513	0.6536	0.6536

X (cm)	Y (cm)	s/C	p/p_o	T/T_o	Ma
-0.2774	1.8100	0.3208	0.7415	0.6692	0.6692
-0.1783	1.7396	0.3392	0.7299	0.6873	0.6873
-0.0787	1.6586	0.3586	0.7162	0.7088	0.7088
0.0201	1.5672	0.3790	0.7001	0.7337	0.7337
0.1176	1.4658	0.4003	0.6814	0.7624	0.7624
0.2129	1.3559	0.4224	0.6602	0.7950	0.7950
0.3056	1.2372	0.4452	0.6363	0.8319	0.8319
0.3950	1.1107	0.4687	0.6097	0.8729	0.8729
0.4803	0.9774	0.4927	0.5818	0.9165	0.9165
0.5611	0.8374	0.5172	0.5549	0.9589	0.9589
0.6368	0.6914	0.5421	0.5322	0.9954	0.9954
0.7076	0.5387	0.5676	0.5176	1.0191	1.0191
0.7734	0.3797	0.5936	0.5154	1.0227	1.0227
0.8344	0.2151	0.6203	0.5232	1.0100	1.0100
0.8913	0.0460	0.6473	0.5309	0.9975	0.9975
0.9446	-0.1265	0.6746	0.5307	0.9979	0.9979
0.9949	-0.3018	0.7023	0.5231	1.0102	1.0102
1.0422	-0.4788	0.7301	0.5133	1.0263	1.0263
1.0871	-0.6571	0.7579	0.5055	1.0390	1.0390
1.1300	-0.8357	0.7857	0.5014	1.0459	1.0459
1.1709	-1.0142	0.8135	0.5003	1.0477	1.0477
1.2101	-1.1918	0.8410	0.5016	1.0456	1.0456
1.2476	-1.3680	0.8683	0.5048	1.0403	1.0403
1.2837	-1.5425	0.8954	0.5099	1.0319	1.0319
1.3185	-1.7150	0.9220	0.5165	1.0210	1.0210
1.3520	-1.8844	0.9482	0.5242	1.0085	1.0085
1.3840	-2.0505	0.9738	0.5321	0.9956	0.9956
1.4150	-2.2131	0.9988	0.5396	0.9835	0.9835
1.4448	-2.3713	1.0232	0.5464	0.9725	0.9725
1.4729	-2.5250	1.0469	0.5527	0.9625	0.9625
1.5001	-2.6739	1.0698	0.5587	0.9529	0.9529
1.5260	-2.8174	1.0919	0.5648	0.9432	0.9432
1.5507	-2.9548	1.1131	0.5709	0.9336	0.9336
1.5740	-3.0858	1.1333	0.5767	0.9244	0.9244
1.5959	-3.2106	1.1524	0.5819	0.9163	0.9163
1.6165	-3.3279	1.1705	0.5861	0.9098	0.9098
1.6355	-3.4379	1.1874	0.5894	0.9045	0.9045
1.6533	-3.5397	1.2031	0.5923	0.9000	0.9000
1.6693	-3.6335	1.2175	0.5954	0.8951	0.8951
1.6838	-3.7183	1.2306	0.5989	0.8897	0.8897
1.6970	-3.7943	1.2422	0.6026	0.8839	0.8839
1.7082	-3.8603	1.2523	0.6058	0.8791	0.8791
1.7178	-3.9162	1.2610	0.6073	0.8767	0.8767
1.7252	-3.9614	1.2679	0.6097	0.8729	0.8729
1.7313	-3.9954	1.2732	0.6081	0.8755	0.8755
1.7358	-4.0241	1.2775	0.6057	0.8792	0.8792
1.7371	-4.0315	1.2787	0.6069	0.8773	0.8773
1.7379	-4.0389	1.2798	0.6015	0.8857	0.8857
1.7376	-4.0465	1.2809	0.6062	0.8783	0.8783
1.7363	-4.0538	1.2820	0.6116	0.8700	0.8700
1.7341	-4.0610	1.2832	0.6141	0.8661	0.8661

X (cm)	Y (cm)	s/C	p/p_o	T/T_o	Ma
1.7308	-4.0676	1.2843	0.6147	0.8652	0.8652
1.7267	-4.0739	1.2854	0.6140	0.8662	0.8662
1.7219	-4.0795	1.2866	0.6143	0.8658	0.8658
1.7165	-4.0846	1.2877	0.6161	0.8630	0.8630
1.7104	-4.0886	1.2888	0.6183	0.8596	0.8596
1.7038	-4.0922	1.2899	0.6235	0.8517	0.8517
1.6965	-4.0945	1.2911	0.6243	0.8503	0.8503

Appendix C: Calculating Lateral Average Effectiveness

Two different Matlab programs were developed to calculate lateral average adiabatic effectiveness from experimental and computational results. Due to the lack of thermal periodicity observed in the experiments, inner and outer passage experimental data were processed separately, yielding two different axial distributions of lateral average effectiveness. For the computational results, a more straightforward routine could be used as there is only one vane passage. The program and subfunctions used for processing experimental results is presented below, followed by the program used for the computational results.

Begin LatAve.m

```
function LatAve(filename,fileout1,fileout2)

[PS,SS,ISS,OPS] = VaneCoords('pressurecoords.txt','suctioncoords.txt');

[Zone1,ZoneA,ZoneB,ZoneC,ZoneD,ZoneE,pas1_Upp,pas1_Lpp,pas2_Upp,pas2_Lp
p] = DataSorter(PS,SS,ISS,OPS, ...
    filename);

Chord = 0.594;
Pitch = 0.457;

XoverC = 0:0.02:0.48;
XoverP = XoverC*Chord/Pitch;

Passage1_UL = ppval(pas1_Upp,XoverP);
Passage1_LL = ppval(pas1_Lpp,XoverP);
Passage2_UL = ppval(pas2_Upp,XoverP);
Passage2_LL = ppval(pas2_Lpp,XoverP);

[ETA_AVE_Passage1,ETA_ALL_Passage1] =
LateralAverageEta(ZoneC,Passage1_UL,Passage1_LL,XoverP);
[ETA_AVE_Passage2,ETA_ALL_Passage2] =
LateralAverageEta(ZoneD,Passage2_UL,Passage2_LL,XoverP);

XoverC2 = -0.10:0.02:-0.02;
XoverP2 = XoverC2*Chord/Pitch;

Pass1_UL = [0.9 0.9 0.9 0.9 0.9];
Pass1_LL = Pass1_UL - 0.9;
Pass2_UL = [0.0 0.0 0.0 0.0 0.0];
Pass2_LL = Pass2_UL -1;

[ETA_AVE_UpstreamPass1,ETA_ALL_UpPass1] =
LateralAverageEta2(ZoneA,0.9,-0.1,XoverP2);
[ETA_AVE_UpstreamPass2,ETA_ALL_UpPass2] =
LateralAverageEta2(ZoneB,0.0,-1.0,XoverP2);
```

```
XoverC = [XoverC2 XoverC]';
XoverP = [XoverP2 XoverP]';
ETA_LAT_AVE_Pass1 = [ETA_AVE_UpstreamPass1 ETA_AVE_Passage1]';
ETA_LAT_AVE_Pass2 = [ETA_AVE_UpstreamPass2 ETA_AVE_Passage2]';
Pass1_UL = [Pass1_UL Passage1_UL]';
Pass1_LL = [Pass1_LL Passage1_LL]';
Pass2_UL = [Pass2_UL Passage2_UL]';
Pass2_LL = [Pass2_LL Passage2_LL]';

OUT1 = [XoverC XoverP ETA_LAT_AVE_Pass1 Pass1_UL Pass1_LL
ETA_LAT_AVE_Pass2 Pass2_UL Pass2_LL];
save(fileout1,'OUT1','-ASCII','-TABS');

OUT2 = [ETA_ALL_UpPass1 ETA_ALL_Passage1 ETA_ALL_UpPass2
ETA_ALL_Passage2];
save(fileout2,'OUT2','-ASCII','-TABS');
```

End LatAve.m

Begin VaneCoords.m

```
function [PS,SS,ISS,OPS] = VaneCoords(PressSide,SuctSide)

% Test arguments
Args = nargin;
switch Args
case 0
    PS = load('pressurecoords.txt');
    SS = load('suctioncoords.txt');
case 1
    PS = load(PressSide);
    SS = load('suctioncoords.txt');
otherwise
    PS = load(PressSide);
    SS = load(SuctSide);
end

%Inner vane suction surface definition.
ISS = [SS(:,1), SS(:,2) + 0.457];
%Outer vane pressure surface definition.
OPS = [PS(:,1), PS(:,2) - 0.457];
```

End VaneCoords.m

Begin DataSorter.m

```
function [Zone1,ZoneA,ZoneB,ZoneC,ZoneD,ZoneE,pas1_Upp,pas1_Lpp,
pas2_Upp,pas2_Lpp] = DataSorter(PS,SS,ISS,OPS,DataFile)

XYETA=load(DataFile);

chord = 0.594;
pitch = 0.457;

% Nondimensionalize vane coordinates.
PS = PS/pitch;
SS = SS/pitch;
ISS = ISS/pitch;
OPS = OPS/pitch;
```

```
XLE=min(PS(:,1));
XTE=max(PS(:,1));
XYETA=sortrows(XYETA,[1 2]);

% Zone1 includes all data before vane LE.
Zone1 = [];
% Zone2 includes all data from LE to TE.
Zone2 = [];
% Zone3 includes all data aft of the TE.
Zone3 = [];

i = 1; j = 1; k = 1; n = 1;
for i = 1:length(XYETA)
    if XYETA(i,1) >=-0.15 & XYETA(i,1) <= XLE
        Zone1(j,:) = XYETA(i,:);
        j = j + 1;
    elseif XYETA(i,1) > XLE & XYETA(i,1) <= XTE
        Zone2(k,:) = XYETA(i,:);
        k = k + 1;
    elseif XYETA(i,1) > XTE
        Zone3(n,:) = XYETA(i,:);
        n = n + 1;
    end
end

clear XYETA i j k n XTE XLE;

% Define upper and lower limits in Y for Zone1 data.
maxyin = 0.9;
minyout = -1.0;
minyin = maxyin - 1;
maxyout = minyout + 1;

% Sort Zone1 data into inner(ZoneA) and outer(ZoneB) passages.
Zone1=sortrows(Zone1,2);

ZoneA = [];
ZoneB = [];

i = 1; j = 1; k = 1;
while Zone1(i,2) <= maxyin
    YLoc = Zone1(i,2);
    if YLoc >= minyout & YLoc < minyin
        ZoneB(j,:) = Zone1(i,:);
        j = j + 1;
    elseif YLoc >= minyin & YLoc <= maxyout
        ZoneA(k,:) = Zone1(i,:);
        ZoneB(j,:) = Zone1(i,:);
        j = j + 1;
        k = k + 1;
    elseif YLoc > maxyout
        ZoneA(k,:) = Zone1(i,:);
        k = k + 1;
    end
    i = i + 1;
    if i > length(Zone1)
```

```
        break
    end
end

% Spline Upper and Lower Vane Passage boundaries.
pas1_Upp = spline(ISS(:,1),ISS(:,2));
pas1_Lpp = spline(PS(:,1),PS(:,2));
pas2_Upp = spline(SS(:,1),SS(:,2));
pas2_Lpp = spline(OPS(:,1),OPS(:,2));

%Seperate passage data into inner and outer passages.
n = 1; m = 1;
for i=1:length(Zone2)
        vp1L = ppval(pas1_Lpp,Zone2(i,1));
        if Zone2(i,2) > vp1L
            vp1U = ppval(pas1_Upp,Zone2(i,1));
            if Zone2(i,2) < vp1U
                ZoneC(n,:) = Zone2(i,:);
                n = n + 1;
            end
        end
        vp2L = ppval(pas2_Lpp,Zone2(i,1));
        if Zone2(i,2) > vp2L
            vp2U = ppval(pas2_Upp,Zone2(i,1));
            if Zone2(i,2) < vp2U
                ZoneD(m,:) = Zone2(i,:);
                m = m + 1;
            end
        end
end

% Calculate flexible wall Y/P values for Zone3 data.
Angle = 75;
AngleRad = pi/180*Angle;
Slope = tan(AngleRad);

YoP = Slope*Zone3(:,1)-Slope*OPS(1,1) - OPS(1,2);

% Eliminate data points outside of flexible wall.
i = 1; j = 1;
for i = 1:length(Zone3)
    if Zone3(i,2) >= YoP(i)
        ZoneE(j,:) = Zone3(i,:);
        j = j + 1;
    end
end
```

End DataSorter.m

Begin LateralAverageEta.m

```
function [ETA_AVE,ETA_ALL] =
LateralAverageEta(Zone,Passage_UL,Passage_LL,XoverP)

ETA_AVE = [];
ETA_ALL = [];
numberofpoints = 50;
```

```
for i=1:length(XoverP)

    YoverP = linspace(Passage_LL(i),Passage_UL(i),numberofpoints);

    [XI,YI,ETAI] =
griddata(Zone(:,1),Zone(:,2),Zone(:,3),XoverP(i),YoverP);

    ETA_ALL = [ETA_ALL XI YI ETAI];

    NANS = isnan(ETAI);
    Indicies = find(NANS);
    missing = length(Indicies);

    if missing <= numberofpoints/10
        if missing > 0
            j = 1;
            while NANS(j) == 1
                j = j + 1;
            end
            FirstNum = ETAI(j);

            k = length(NANS);
            while NANS(k) == 1
                k = k - 1;
            end
            LastNum = ETAI(k);

            for m = 1:length(Indicies)
                if Indicies(m) < j
                    ETAI(Indicies(m)) = FirstNum;
                elseif Indicies(m) > k
                    ETAI(Indicies(m)) = LastNum;
                end
            end
        end
        ETA_AVE(i) = trapz(YI,ETAI)/(max(YI)-min(YI));
    else
        ETA_AVE(i) = NaN;
    end

end
```

End Lateral AverageEta.m

Begin LateralAverageEta2.m

```
function [ETA_AVE,ETA_ALL] =
LateralAverageEta2(Zone,Passage_UL,Passage_LL,XoverP)

ETA_AVE = [];
ETA_ALL = [];
numberofpoints = 50;

YoverP = linspace(Passage_LL,Passage_UL,numberofpoints);

% figure
for i=1:length(XoverP)
```

```
    [XI,YI,ETAI] =
griddata(Zone(:,1),Zone(:,2),Zone(:,3),XoverP(i),YoverP);

    ETA_ALL = [ETA_ALL XI YI ETAI];
    NANS = isnan(ETAI);
    Indicies = find(NANS);
    missing = length(Indicies);

    if missing <= numberofpoints/10
        if missing > 0
            j = 1;
            while NANS(j) == 1
                j = j + 1;
            end
            FirstNum = ETAI(j);

            k = length(NANS);
            while NANS(k) == 1
                k = k - 1;
            end
            LastNum = ETAI(k);

            for m = 1:length(Indicies)
                if Indicies(m) < j
                    ETAI(Indicies(m)) = FirstNum;
                elseif Indicies(m) > k
                    ETAI(Indicies(m)) = LastNum;
                end
            end
        end
        ETA_AVE(i) = trapz(YI,ETAI)/(max(YI)-min(YI));
    else
        ETA_AVE(i) = NaN;
    end

end
```

End EXPLatAve.m

The Matlab program used to calculate lateral average effectiveness for the CFD results is presented below.

Begin CFDLatAve.m

```
clear all
close all
%Turn off duplicate data point warning.
warning off MATLAB:griddata:DuplicateDataPoints

%Load data file X/P, Y/P, ETA.
XYETA = load('Case2 OffStag.dat');

%Determine maximum limits of the solution domain.
XoverPMIN = min(XYETA(:,1))
XoverPMAX = max(XYETA(:,1))
YoverPMIN = min(XYETA(:,2))
YoverPMAX = max(XYETA(:,2))
```

```
Chord = 0.594;
Pitch = 0.457;

%Define axial locations for lateral average eta evaluation.
XoverC = -0.06:0.02:0.50;
XoverP = XoverC*Chord/Pitch;

ETA_AVE = [];
ETA_ALL = [];
numberofpoints = 1000;

for i=1:length(XoverP)
    %Create an evenly-spaced array of points in the Y/P direction.
    YoverP = linspace(-0.25,2.25,numberofpoints);
    %Interpolate the CFD data onto the evenly-spaced points.
    [XI,YI,ETAI] =
griddata(XYETA(:,1),XYETA(:,2),XYETA(:,3),XoverP(i),YoverP);
    %Record values to a matrix.
    ETA_ALL = [ETA_ALL XI YI ETAI];
    %Remove all NaNs from ETAI.
    ETAI = ETAI(~isnan(ETAI));
    %Calculate the lateral average as the mean of ETAI.
    ETA_AVE(i) = mean(ETAI);
    clear XI YI ETAI;
end

%Format results for output.
OUT1 = [XoverC' XoverP' ETA_AVE'];
%Write the results to a text file.
save('Case2 OffStag Eta Ave.dat','OUT1','-ASCII','-TABS');
```

End CFDlatave.m

Appendix D: Infrared Image Processing Programs

This appendix outlines the image processing procedures and presents the various Fortran and Matlab codes used in the process. Infrared images were taken at 14 different imaging locations. To get a good average, five identical images were taken at each imaging location. The first step in processing the images is to extract the temperature matrix using the Thermonitor Lite software. Before the temperature matrix is extracted, the surface emissivity and background temperature are adjusted to achieve the best possible agreement between the images and endwall thermocouple data. Thermonitor Lite is also used to determine the pixel coordinates of the endwall markers in each image. Following extraction of the temperature matrices, the average of 5 images is calculated at each imaging location. The Fortran program used for averaging the five images follows.

Begin avgLX.f

```
C*********************************************************************
C
C  THIS PROGRAM READS IN PIXEL TEMPERATURE DATA OUTPUT FROM THE
C  THERMONITOR PROGRAM FOR 5 SIMILAR IMAGES AND AVERAGES THEM.
C
C*********************************************************************
      PROGRAM IMAGEAVGR
      IMPLICIT REAL *8 (A-H,O-Z)
      DIMENSION T1(300,300),T2(300,300),T3(300,300),T4(300,300)
      DIMENSION T5(300,300),AVGTEMP(300,300)
      INTEGER NUMX, NUMY
C*********************************************************************
C     INITIALIZE MATRICES TO ZERO.
C*********************************************************************
      DO 5 I=1,300
        DO 4 J=1,300
          T1(I,J)=0.0
          T2(I,J)=0.0
          T3(I,J)=0.0
          T4(I,J)=0.0
          T5(I,J)=0.0
          AVGTEMP(I,J)=0.0
   4    CONTINUE
   5  CONTINUE
C*********************************************************************
C     OPEN INPUT FILES AND OUTPUT FILE.
C*********************************************************************
      OPEN(UNIT=10, FILE="LX_1.txt", STATUS="OLD")
      OPEN(UNIT=11, FILE="LX_2.txt", STATUS="OLD")
      OPEN(UNIT=12, FILE="LX_3.txt", STATUS="OLD")
      OPEN(UNIT=13, FILE="LX_4.txt", STATUS="OLD")
      OPEN(UNIT=14, FILE="LX_5.txt", STATUS="OLD")
      OPEN(UNIT=15, FILE="LX.txt", STATUS="UNKNOWN")
```

```
C*********************************************************************
C     SPECIFY THE SIZE OF THE IMAGE MATRIX. THE IMAGE IS 255 PIXELS
C     WIDE BY 206 PIXELS TALL.
C*********************************************************************
      NUMY=255
      NUMX=206
C*********************************************************************
C     READ PIXEL TEMPERATURE VALUES INTO THE MATRICES T#(I,J).
C*********************************************************************
      DO 13 I=1,NUMX
        READ(10,*) (T1(I,J),J=1,NUMY)
        READ(11,*) (T2(I,J),J=1,NUMY)
        READ(12,*) (T3(I,J),J=1,NUMY)
        READ(13,*) (T4(I,J),J=1,NUMY)
        READ(14,*) (T5(I,J),J=1,NUMY)
  13  CONTINUE
C*********************************************************************
C     AVERAGE THE IMAGES AND WRITE THE RESULTING AVERAGE TO THE OUTPUT
C     TEXT FILE, LX.TXT.
C*********************************************************************
      DO 55 I=1,NUMX
        DO 54 J=1,NUMY
          AVGTEMP(I,J)=0.2*(T1(I,J)+T2(I,J)+T3(I,J)+T4(I,J)+T5(I,J))
  54    CONTINUE
  55  CONTINUE
      DO 65 I=1,NUMX
        WRITE(15,8) (AVGTEMP(I,J),J=1,NUMY)
  65  CONTINUE
   8  FORMAT(255(F5.2,2X))
      STOP
      END
```

End avgLX.f

After calculating the average at each imaging location, the average temperature matrix and alignment marker pixel coordinates are used as input for a second Fortran code that converts the pixel data into global coordinate data. This mapping can be expressed as follows.

$$i, j, T_{aw} \rightarrow X, Y, T_{aw}$$

This program was revised and modified from a previous code written by Atul Kohli (1997) to automatically determine image orientation.

Begin imageX.f

```
C*********************************************************************
C
C  THIS PROGRAM READS IN AVERAGED PIXEL TEMPERATURE DATA OUTPUT FROM
C  THE MATLAB PROGRAM 'IRCAMERAAVERAGEDDATA', AND CONVERTS THE DATA
C  INTO GLOBAL X,Y COORDINATE AND TEMPERATURE DATA, WHILE MASKING IMAGE
C  REFERENCE MARKERS AND UP TO THREE ADDITIONAL IMAGE BLEMISH POINTS.
C
C*********************************************************************
      PROGRAM IMAGE1
```

```
      IMPLICIT REAL *8 (A-H,O-Z)
      DIMENSION XCOORD(300,300), YCOORD(300,300)
      DIMENSION TEMP(300,300), GLOBALX(300,300), GLOBALY(300,300)
      INTEGER NUMX, NUMY
      PI=3.1415927
C*********************************************************************
C     SPECIFY THE NUMBER OF POINTS TO BE SKIPPED IN CREATING THE FINAL
C     OUTPUT. ALSO, SPECIFY THE MASKING RADIUS ABOUT REFERENCE POINTS
C     AND OTHER BLEMISH POINTS DESIRED TO BE MASKED IN THE FINAL
C     OUTPUT.
C*********************************************************************
      NSKIP=7
      NPREF=3
C*********************************************************************
C     INITIALIZE MATRICES TO ZERO.
C*********************************************************************
      DO 5 I=1,300
        DO 4 J=1,300
          XCOORD(I,J)=0.0
          YCOORD(I,J)=0.0
          GLOBALX(I,J)=0.0
          GLOBALY(I,J)=0.0
          TEMP(I,J)=0.0
   4    CONTINUE
   5  CONTINUE
C*********************************************************************
C     OPEN INPUT FILES AND OUTPUT FILE.
C*********************************************************************
      OPEN(UNIT=10, FILE="LX.txt", STATUS="OLD")
      OPEN(UNIT=11, FILE="OUTLX.txt", STATUS="UNKNOWN")
      OPEN(UNIT=12, FILE="REF_LX.txt", STATUS="OLD")
C*********************************************************************
C     SPECIFY THE SIZE OF THE IMAGE MATRIX. THE IMAGE IS 255 PIXELS
C     WIDE BY 196 PIXELS TALL.
C*********************************************************************
      NUMY=255
      NUMX=206
C*********************************************************************
C     READ IN THE PIXEL INDICES AND GLOBAL X,Y COORDINATES OF IMAGE
C     REFERENCE POINTS. ALSO, READ IN THE PIXEL INDICES OF OTHER
C     BLEMISH POINTS DESIRED TO BE MASKED.
C*********************************************************************
      READ(12,*) DUMMY
C     READ IN THE PIXEL INDICES FOR THE FIRST REFERENCE POINT
C     **Note: IMAGE Y INDEX IS READ IN FIRST. THIS INDEX CAN
C             ASSUME VALUES RANGING FROM 0 TO 255. IMAGE X
C             INDEX IS READ IN SECOND AND ASSUMES VALUES
C             RANGING FROM 0 TO 206.
      READ(12,*) NPY1, NPX1
      READ(12,*) DUMMY
C     READ IN THE GLOBAL X,Y COORDINATES OF THE FIRST REFERENCE POINT.
      READ(12,*) X1, Y1
      READ(12,*) DUMMY
C     READ IN THE PIXEL INDICES FOR THE SECOND REFERENCE POINT.
      READ(12,*) NPY2, NPX2
      READ(12,*) DUMMY
C     READ IN THE GLOBAL X,Y COORDINATES OF THE SECOND REFERENCE POINT.
```

```
      READ(12,*) X2, Y2
      READ(12,*) DUMMY
C     READ IN NUMBER OF ADDITIONAL BLEMISH POINTS TO BE MASKED.
      READ(12,*) NUMBER
      IF(NUMBER.EQ.1) THEN
        READ(12,*) NYMASK1,NXMASK1
        NYMASK1=NYMASK1+1
        NXMASK1=NXMASK1+1
      ELSEIF(NUMBER.EQ.2) THEN
        READ(12,*) NYMASK1,NXMASK1
        READ(12,*) NYMASK2,NXMASK2
        NYMASK1=NYMASK1+1
        NXMASK1=NXMASK1+1
        NYMASK2=NYMASK2+1
        NXMASK2=NXMASK2+1
      ELSEIF(NUMBER.EQ.3) THEN
        READ(12,*) NYMASK1,NXMASK1
        READ(12,*) NYMASK2,NXMASK2
        READ(12,*) NYMASK3,NXMASK3
        NYMASK1=NYMASK1+1
        NXMASK1=NXMASK1+1
        NYMASK2=NYMASK2+1
        NXMASK2=NXMASK2+1
        NYMASK3=NYMASK3+1
        NXMASK3=NXMASK3+1
      ENDIF
C     REFERENCE POINT INDICES ARE ALL INDEXED BY 1 SINCE FORTRAN
C     MATRICES DO NOT HAVE 0,0 ELEMENTS.
      NPX1=NPX1+1
      NPY1=NPY1+1
      NPX2=NPX2+1
      NPY2=NPY2+1
C*********************************************************************
C     READ PIXEL TEMPERATURE VALUES INTO THE MATRIX TEMP(I,J).
C*********************************************************************
      DO 13 I=1,NUMX
        READ(10,*) (TEMP(I,J),J=1,NUMY)
  13  CONTINUE
C*********************************************************************
C     CONVERT I,J INDICES TO IMAGE LOCAL X,Y COORDINATES.
C*********************************************************************
C     CALCULATE PIXEL DISTANCE BETWEEN REFERENCE POINTS
      REFPIX=(FLOAT((NPX1-NPX2)**2+(NPY1-NPY2)**2))**0.5
C     CALCULATE PHYSICAL DISTANCE BETWEEN REFERENCE POINTS
      REFDIS=((X1-X2)**2+(Y1-Y2)**2)**0.5
C     CALCULATE PIXEL SCALE(INCHES/PIXEL)
      SCALE=REFDIS/REFPIX
C     ASSIGN LOCAL X,Y COORDINATES TO EACH PIXEL
      DO 20 I=1,NUMX
        DO 19 J=1,NUMY
          XCOORD(I,J)=(I-1)*SCALE
          YCOORD(I,J)=(J-1)*SCALE
  19    CONTINUE
  20  CONTINUE
   8  FORMAT(F7.3,2X,F7.3,2X,F4.1)
C*********************************************************************
C     TRANSFORM AXES TO GLOBAL COORDINATE SYSTEM
```

```
C*****************************************************************
C     CALCULATE ANGLE OF REF. PT. LINE RELATIVE TO IMAGE LOCAL Y AXIS.
      BETA=ATAN((FLOAT(NPX2-NPX1))/(FLOAT(NPY2-NPY1)))
C     CALCULATE ANGLE OF REFERENCE PT. LINE RELATIVE TO GLOBAL Y AXIS.
      GAMMA=ATAN((X2-X1)/(Y2-Y1))
C     CALCULATE ROTATION ANGLE OF THE IMAGE RELATIVE TO GLOBAL X,Y.
      ALPHA=BETA-GAMMA
C     DETERMINE IN WHICH QUADRANT OF THE GLOBAL COORDINATE SYSTEM THE
C     IMAGE WAS TAKEN
      DELTAY=Y2-Y1
      DELTAX=X2-X1
      SCALEX=DELTAX/(FLOAT(NPY2-NPY1))
      SCALEY=DELTAY/(FLOAT(NPY2-NPY1))
      IF(BETA.LT.0) THEN
        IF(DELTAY.LT.0) THEN
          IF(DELTAX.LT.0) THEN
            ALPHA=PI+ALPHA
          ENDIF
          IF(DELTAX.GE.0) THEN
            IF(ABS(SCALEY).GT.SCALE) ALPHA=PI+ALPHA
            IF(ABS(SCALEY).LE.SCALE) ALPHA=-PI+ALPHA
          ENDIF
        ENDIF
        IF(DELTAY.GE.0) THEN
          ALPHA=ALPHA
        ENDIF
      ENDIF
      IF(BETA.GE.0) THEN
        IF(DELTAY.LT.0) THEN
          IF(DELTAX.LT.0) THEN
            IF(ABS(SCALEY).GT.SCALE) ALPHA=-PI+ALPHA
            IF(ABS(SCALEY).LE.SCALE) ALPHA=PI+ALPHA
          ENDIF
          IF(DELTAX.GE.0) THEN
            ALPHA=-PI+ALPHA
          ENDIF
        ENDIF
        IF(DELTAY.GE.0) THEN
          ALPHA=ALPHA
        ENDIF
      ENDIF
C     PERFORM ROTATIONAL TRANSFORMATION TO CONVERT LOCAL X,Y TO GLOBAL.
      DO 35 I=1,NUMX
        DO 34 J=1,NUMY
          GLOBALX(I,J)=XCOORD(I,J)*COS(ALPHA)-YCOORD(I,J)*SIN(ALPHA)
          GLOBALY(I,J)=XCOORD(I,J)*SIN(ALPHA)+YCOORD(I,J)*COS(ALPHA)
  34    CONTINUE
  35  CONTINUE
C     CALCULATE X,Y TRANSLATIONAL OFFSET.
      XOFFSET=X1-GLOBALX(NPX1,NPY1)
      YOFFSET=Y1-GLOBALY(NPX1,NPY1)
C     PERFORM TRANSLATIONAL TRANSFORMATION.
      DO 45 I=1,NUMX
        DO 44 J=1,NUMY
          GLOBALX(I,J)=GLOBALX(I,J)+XOFFSET
          GLOBALY(I,J)=GLOBALY(I,J)+YOFFSET
  44    CONTINUE
```

```
  45  CONTINUE
C*********************************************************************
C     THIS SECTION OF THE CODE ENSURES THAT POINTS DESIRED TO BE MASKED
C     DO NOT GET WRITTEN TO THE FINAL OUTPUT FILE. THIS IS DONE BY
C     CALCULATING THE PIXEL DISTANCE BETWEEN EACH POTENTIAL OUTPUT
C     POINT AND THE POINTS DESIRED TO BE MASKED. IF THE POTENTIAL
C     OUTPUT POINT IS WITHIN THE SPECIFIED MASKING PIXEL RADIUS, IT IS
C     DISCARDED.
C*********************************************************************
      NCOUNT=0
      DO 55 I=1,NUMX,NSKIP
        DO 54 J=1,NUMY,NSKIP
          NSWITCH=0
          NRAD1=((I-NPX1)**2+(J-NPY1)**2)**0.5
          NRAD2=((I-NPX2)**2+(J-NPY2)**2)**0.5
        IF((NRAD1.LE.NPREF).OR.(NRAD2.LE.NPREF)) THEN
          NSWITCH=1
        ENDIF
        IF((NUMBER.EQ.1).AND.(NSWITCH.EQ.0)) THEN
          NRAD3=((I-NXMASK1)**2+(J-NYMASK1)**2)**0.5
          IF(NRAD3.LE.NPREF) NSWITCH=1
        ELSEIF((NUMBER.EQ.2).AND.(NSWITCH.EQ.0)) THEN
          NRAD3=((I-NXMASK1)**2+(J-NYMASK1)**2)**0.5
          NRAD4=((I-NXMASK2)**2+(J-NYMASK2)**2)**0.5
          IF((NRAD3.LE.NPREF).OR.(NRAD4.LE.NPREF)) NSWITCH=1
        ELSEIF((NUMBER.EQ.3).AND.(NSWITCH.EQ.0)) THEN
          NRAD3=((I-NXMASK1)**2+(J-NYMASK1)**2)**0.5
          NRAD4=((I-NXMASK2)**2+(J-NYMASK2)**2)**0.5
          NRAD5=((I-NXMASK3)**2+(J-NYMASK3)**2)**0.5
          IF((NRAD3.LE.NPREF).OR.(NRAD4.LE.NPREF).OR.(NRAD5.LE.NPREF))
     /       NSWITCH=1
        ENDIF
        IF(NSWITCH.EQ.0) THEN
           WRITE(11,8) GLOBALX(I,J),GLOBALY(I,J),TEMP(I,J)
           NCOUNT=NCOUNT+1
        ENDIF
  54    CONTINUE
  55  CONTINUE
      STOP
      END
```

End imageX.f

After mapping each of the images into the global coordinate system, the individual image data were combined using the following Matlab routine.

Begin combine.m

```
clear
clc
data1=load('OUTL1.txt');
data2=load('OUTL2.txt');
data3=load('OUTL3.txt');
data4=load('OUTL4.txt');
data5=load('OUTL5.txt');
data6=load('OUTL6.txt');
data7=load('OUTL7.txt');
```

```
data8=load('OUTL8.txt');
data9=load('OUTL9.txt');
data10=load('OUTL10.txt');
data11=load('OUTL11.txt');
data12=load('OUTL12.txt');
data13=load('OUTL13.txt');
data14=load('OUTL14.txt');
data_append=[data1;data2;data3;data4;data5;data6;data7;data8;data9;
             data10;data11;data12;data13;data14];
save 'data.txt' data_append -ascii;
```

End combine.m

Due to overlap between adjacent images, the combined output is finally passed through a smoothing program. The smoothing program identifies regions of overlap and averages the results in these regions. It is important to note that this program is units sensitive, requiring global coordinates X and Y in units of cm. After smoothing the regions of overlap, the data are ready for plotting.

Begin smooth.f

```
C**********************************************************************
C     THIS PROGRAM READS IN CAMERA DATA AND SMOOTHS THE REGIONS
C     OF OVERLAP BETWEEN FRAMES
C**********************************************************************
      PROGRAM SMOOTH
      IMPLICIT REAL *8 (A-H,O-Z)
      DIMENSION X(25000), Y(25000), STANTON(25000)
      DIMENSION MASK(25000), XTEMP(25000), YTEMP(25000), STATEMP(25000)
      DIMENSION XAVG(25000), YAVG(25000), STANAVG(25000)
      DO 5 I=1,10000
        X(I)=0.0
        Y(I)=0.0
        STANTON(I)=0.0
        MASK(I)=0.0
        XTEMP(I)=0.0
        YTEMP(I)=0.0
        STATEMP(I)=0.0
        XAVG(I)=0.0
        YAVG(I)=0.0
        STANAVG(I)=0.0
  5   CONTINUE
      RAD=0.4
      NUM=14380
C*****************************************
C     OPEN DATA FILE AND READ X, Y, STANTON
C*****************************************
      OPEN(UNIT=10, FILE="data.txt", STATUS="OLD")
      OPEN(UNIT=11, FILE="output.dat", STATUS="UNKNOWN")
      WRITE(6,*) 'NUMBER OF POINTS =', NUM
      WRITE(6,*) 'READING DATA FILE!'
        DO 10 I=1,NUM
          READ(10,*) X(I), Y(I), STANTON(I)
  10    CONTINUE
```

```
      WRITE(6,*) 'FINISHED READING DATA FILES!'
      CLOSE(10)
C*********************************************
C     SMOOTH DATA
C*********************************************
      WRITE(6,*) 'NOW SMOOTHING DATA'
      NTOTAL=0
      DO 30 I=1,NUM
      NSMOOTH=1
        DO 29 J=1,NUM
          RADTEST=SQRT((X(I)-X(J))**2+(Y(I)-Y(J))**2)
          IF((RADTEST.LT.RAD).AND.(J.NE.I).AND.(MASK(J).NE.1)) THEN
             MASK(I)=1
             MASK(J)=1
             NSMOOTH=NSMOOTH+1
             XTEMP(NSMOOTH)=X(J)
             YTEMP(NSMOOTH)=Y(J)
             STATEMP(NSMOOTH)=STANTON(J)
          ENDIF
  29    CONTINUE
      IF(NSMOOTH.GT.1) THEN
        NTOTAL=NTOTAL+1
        DO 27 K=1,NSMOOTH
          XAVG(NTOTAL)=XAVG(NTOTAL)+XTEMP(NSMOOTH)
          YAVG(NTOTAL)=YAVG(NTOTAL)+YTEMP(NSMOOTH)
          STANAVG(NTOTAL)=STANAVG(NTOTAL)+STATEMP(NSMOOTH)
  27    CONTINUE
        XAVG(NTOTAL)=XAVG(NTOTAL)/FLOAT(NSMOOTH)
        YAVG(NTOTAL)=YAVG(NTOTAL)/FLOAT(NSMOOTH)
        STANAVG(NTOTAL)=STANAVG(NTOTAL)/FLOAT(NSMOOTH)
       ENDIF
  30  CONTINUE
      WRITE(6,*) 'NUMBER OF SMOOTHED POINTS =', NTOTAL
      WRITE(6,*) 'WRITING SMOOTHED DATA'
      DO 35 I=1,NUM
       IF(MASK(I).NE.1) THEN
          WRITE(11,8) X(I), Y(I), STANTON(I)
       ENDIF
  35  CONTINUE
c     WRITE(11,*) '***************'
    8  FORMAT(F7.3,2X,F7.3,2X,F4.1)
      DO 36 I=1,NTOTAL
        WRITE(11,8) XAVG(I), YAVG(I), STANAVG(I)
  36  CONTINUE
      CLOSE(11)
      STOP
      END
```

End smooth.f

Appendix E: Image Distortion Correction Program

The leading edge fillet resulted in perspective distortion which prevented direct assembly of the multiple thermal images into a composite image of the endwall. To enable the images to be assembled, a distortion correction program was written in Matlab which projects the fillet surface temperature results onto the endwall. Calibration images were taken to determine the pixel coordinate to global coordinate mapping for each distorted image, and measures were taken to ensure precise camera positioning for all subsequent measurements. The inputs to the Matlab program include the matching pixel and global coordinate pairs and the thermal image temperature matrix. The output of the program is a 3 column matrix corresponding to global coordinates X and Y, and adiabatic wall temperature, T_{aw}.

Begin undistort.m

```
function undistort

    %Load temperature matrices.
    data1=load('LX_1.txt');
    data2=load('LX_2.txt');
    data3=load('LX_3.txt');
    data4=load('LX_4.txt');
    data5=load('LX_5.txt');

    %Average temperature matrices.
    T=(data1+data2+data3+data4+data5)/5;
    clear data1 data2 data3 data4 data5

    %Load reference pixel and global coordinate pairs.
    IJXY=load('REF_LX.txt');

    %Create individual coordinate vectors for interpolation function.
    I=IJXY(:,1);
    J=IJXY(:,2);
    X=IJXY(:,3);
    Y=IJXY(:,4);
    clear IJXY n

    %Create pixel coordinate matrices for interpolation function.
    [JI,II]=meshgrid(0:254,0:205);

    %Interpolate global coordinate values at all pixel locations.
    XG=griddata(I,J,X,II,JI);
    YG=griddata(I,J,Y,II,JI);
    clear I J X Y JI II

    %Subsample the resulting matrices of global X, Y, and Taw.
    j=1;
```

```
step=7;
s=size(T);
for i=1:step:s(1)
    XGNEW(j,:)=XG(i,1:step:s(2));
    YGNEW(j,:)=YG(i,1:step:s(2));
    TNEW(j,:)=T(i,1:step:s(2));
    j=j+1;
end
clear i j s step

%Create 3column X,Y,Taw matrix.
OUT(:,1)=XGNEW(:);
OUT(:,2)=YGNEW(:);
OUT(:,3)=TNEW(:);
clear XG YG T XGNEW YGNEW TNEW

%Remove NaNs from 3 column matrix due to pixels outside the convex
%hull of the reference coordinate pairs.
NAN=isnan(OUT);
TEST=sum(NAN');
clear NAN
m=0;
OUT2=[];
for i=1:length(OUT(:,1))
    if TEST(i)==0
        OUT2 = [OUT2; OUT(i,:)];
    else
        m=m+1;
    end
end
clear TEST i

%Warn of lost data points.
if length(OUT2(:,1))+m ~= length(OUT(:,1))
    NumLost=length(OUT(:,1))-length(OUT2(:,1))-m;
    disp(['ERROR: Filtering process has lost ',num2str(NumLost),'
          data points.'])
end

%Write output matrix to file.
save 'OUTLX.txt' OUT2 -ascii;
```

End undistort.m