OVERFLOW 2 Training Class

Introduction

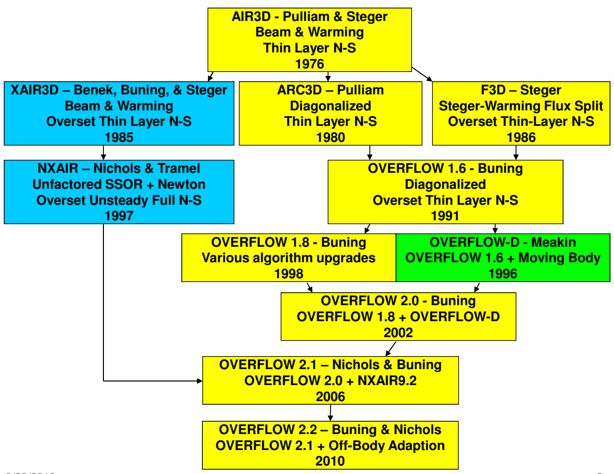
Pieter G. Buning
NASA Langley Research Center
pieter.g.buning@nasa.gov

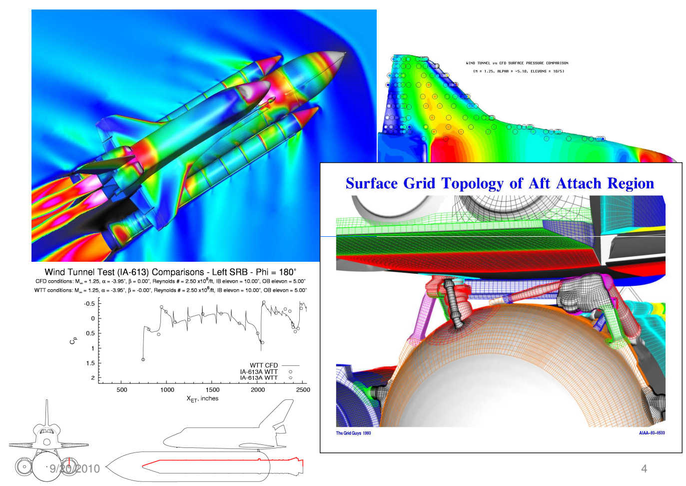
Robert H. Nichols
University of Alabama at Birmingham
CREATE-AV API Principal Developer
robert.nichols@arnold.af.mil

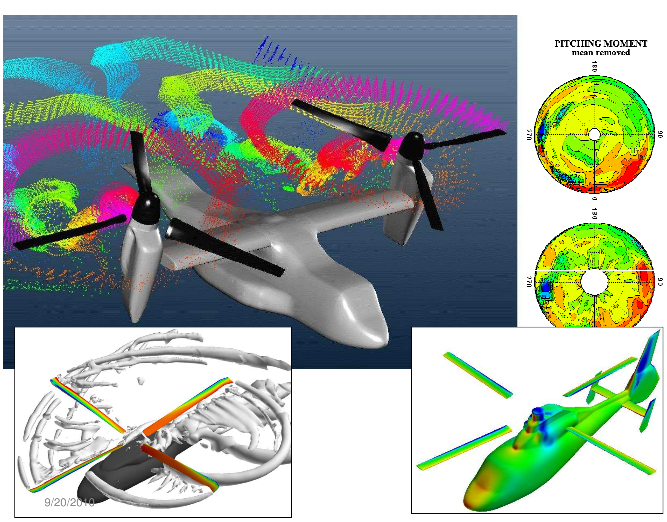
10th Symposium on Overset Composite Grids
& Solution Technology
NASA Ames Research Center
September 20-23, 2010

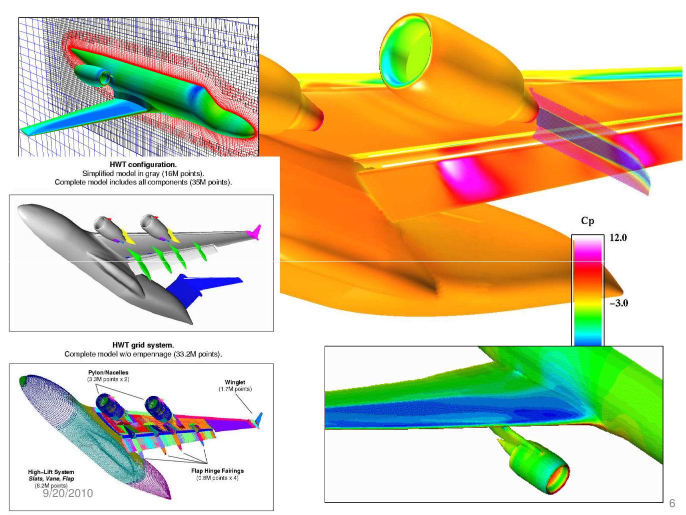
Origins of OVERFLOW 2

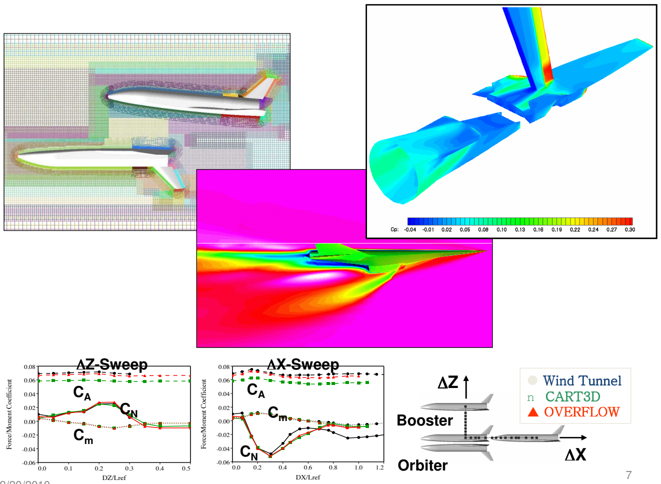
- Major roots
  - F3D (Steger)
  - ARC3D (Pulliam)
  - OVERFLOW-D (Meakin)
  - NXAIR (Nichols)
- Contributors
  - Joe Steger, Tom Pulliam, William Chan, Dennis Jespersen
  - Bob Meakin, Andrew Wissink, Mark Potsdam, Ing-Tsau Chiu
  - Bobby Nichols, Bob Tramel
  - Jeff Slotnick, Steve Krist, Kevin Renze, Shigeru Obayashi, Yehia Rizk
  - and *many* others…
- Major support
  - NASA Basic Research
  - NASA Space Shuttle Program
  - U.S. Army Aeroflightdynamics Division
  - NASA Subsonic Rotary Wing Project
  - NASA Advanced Subsonic Technology Program
  - NASA 2nd Generation Reusable Launch Vehicle Program/Space Launch Initiative
  - DoD High Performance Computing Modernization Program











Class Outline

Morning: Bobby Nichols

1. OVERFLOW 2.2 capabilities
2. CFD nomenclature overview
3. Running in OVERFLOW mode
4. NAMELIST Input
   - Inviscid fluxes
   - Implicit solvers
   - Boundary conditions
   - Species equations
   - Turbulence models
   - Unsteady flow outputs

Afternoon: Pieter Buning

1. Introduction/review
2. OVERFLOW-D mode without grid motion
3. OVERFLOW-D mode with grid motion
4. Solution adaption for off-body grids
5. Compiling and running OVERFLOW
6. Utilities and test cases
7. Future directions

Reference Material

- User's Manual for OVERFLOW Version 2.2
  - Code description and theory
  - NAMELIST input description
  - Instructions for 3 modes of code operation
  - File formats
- Turbulence Models and Their Application to Complex Flows
  - Turbulent flow theory
  - Turbulence model theory
  - Recommendations on turbulence model usage

OVERFLOW 2 Training Class

Morning Session

Robert H. Nichols
University of Alabama at Birmingham
CREATE-AV API Principal Developer
robert.nichols@arnold.af.mil

Pieter G. Buning
NASA Langley Research Center
pieter.g.buning@nasa.gov

10th Symposium on Overset Composite Grids & Solution Technology
NASA Ames Research Center
September 20-23, 2010

Class Outline - Morning

- OVERFLOW 2.2 capabilities
- CFD nomenclature overview
- Running in OVERFLOW mode
- NAMELIST Input
  - Inviscid fluxes
  - Implicit solvers
  - Boundary conditions
  - Species equations
  - Turbulence models
  - Unsteady flow outputs

Capabilities

Capabilities

- Implicit time marching solution algorithms
- Node-centered structured mesh
- OVERSET
- Gas models
  - Perfect gas
  - Variable gamma (currently not correct)
  - Low Mach number preconditioning (not all flux algorithms)
  - Multi-species (non-reacting)
- Moving body
  - Grid assembly (DCF)
  - Prescribed motion (GMP)
  - 6dof (GMP or internal or user specified)
  - Force integration (FOMOCO or USURP)
  - Collision modeling
  - Automatic off-body Cartesian grid generation (DCF)
  - Off-body grid refinement based on body motion or flow field
- Parallel performance enhancement
  - Parallel with MPI and/or OPENMP
  - Auto grid decomposition for load balance

Capabilities (cont.)

- Turbulence models
  - Baldwin-Lomax
  - Baldwin-Barth
  - Spalart-Allmaras (SA-DES, SA-DDES, SARC,ASARC)
  - k-ω
  - SST (SST-DES, SST-DDES, SST-MS, SSTRC,ASSTRC)
- Boundary conditions
  - Slip and no-slip wall
  - Constant temperature wall
  - Topology bc's (overlap, slit, polar axis)
  - Characteristic inflow/outflow
  - Nozzle inflow
  - Actuator Disk
  - Mass flow
  - Wall functions
  - And much, much more

Convergence Acceleration Methods

- Newton subiteration
- Dual time-stepping subiteration
- Multigrid
- Grid sequencing
- Local time step
- dq Limiter

CFD Nomenclature Overview

Navier-Stokes Equations

Differential form:

$$\frac{\partial \vec{q}}{\partial t} + \frac{\partial \vec{E}}{\partial \xi} + \frac{\partial \vec{F}}{\partial \eta} + \frac{\partial \vec{G}}{\partial \zeta} = 0$$

Conserved variables:

$$\vec{q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho\left(e + 1/2(u^2 + v^2 + w^2)\right) \end{bmatrix}$$

Implicit Discrete Unfactored Form (1st or 2nd Order Time):

LHS

$$\left[I + \frac{\Delta t}{1+\theta}\left(\delta_\xi A + \delta_\eta B + \delta_\zeta C\right) \right] \Delta q^{n+1} =$$

RHS

$$\left[\frac{\theta}{1+\theta} \Delta q^n - \frac{\Delta t}{1+\theta} RHS^n \right]$$

Inviscid and viscous flux terms

$$A = \frac{\partial \vec{E}}{\partial \vec{q}}, B = \frac{\partial \vec{F}}{\partial \vec{q}}, C = \frac{\partial \vec{G}}{\partial \vec{q}} \qquad RHS^n = \frac{\partial \vec{E}^n}{\partial \xi} + \frac{\partial \vec{F}^n}{\partial \eta} + \frac{\partial \vec{G}^n}{\partial \zeta}$$

LHS Approximations

ADI Factorization (block tridiagonal matrix system):

$$\left[I+\frac{\Delta t}{1+\theta}\partial_\xi A\right]\left[I+\frac{\Delta t}{1+\theta}\partial_\eta B\right]\left[I+\frac{\Delta t}{1+\theta}\delta_\zeta C\right]\Delta q^{n+1} = \left[\frac{\theta}{1+\theta}\Delta q^n - \frac{\Delta t}{1+\theta}RHS^n\right] + Error$$

Diagonalized Scheme (scalar pentadiagonal matrix scheme):

$$X_\xi\left[I+\frac{\Delta t}{1+\theta}\partial_\xi \Lambda_\xi\right]X_\xi^{-1}\, X_\eta\left[I+\frac{\Delta t}{1+\theta}\partial_\eta \Lambda_\eta\right]X_\eta^{-1}\, X_\zeta\left[I+\frac{\Delta t}{1+\theta}\delta_\zeta \Lambda_\zeta\right]X_\zeta^{-1}\Delta q^{n+1} = \left[\frac{\theta}{1+\theta}\Delta q^n - \frac{\Delta t}{1+\theta}RHS^n\right] + Error$$

X_ξ = Eigenvector of A Λ_ξ = Eigenvalues of A

Factorization Error:

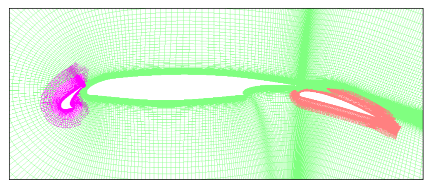
$$Error = \left[\left(\frac{\Delta t}{1+\theta}\right)^2\left(\delta_\xi A\delta_\eta B + \partial_\xi A\delta_\zeta C + \delta_\eta B\delta_\zeta C\right) + \left(\frac{\Delta t}{1+\theta}\right)^3\left(\partial_\xi A\delta_\eta B\delta_\zeta C\right)\right]\Delta q^{n+1}$$

LHS Approximation Summary

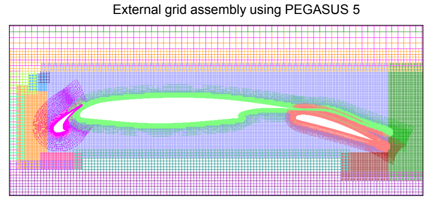
- Unfactored SSOR
  - More memory (store entire Jacobian matrix)
  - Relaxation method to invert matrix
    - Slower
    - Most stable
  - No factorization error
- ADI block tridiagonal
  - Fast (fits well in cache)
  - Small memory (solve 1 direction at a time)
  - Factorization error can cause instability for large time steps
- ADI diagonalized
  - Extremely fast
  - Smallest memory (solve 1 direction at a time)
  - Least stable
  - Factorization error can cause instability for large time steps

Running in OVERFLOW Mode

Overset Methodology



External grid assembly using PEGASUS 5



Internal grid assembly and Cartesian blocks generated using DCF

Required Input Files

- Single grid solution
  - **grid.in** plot3d grid file (single or multi grid format)
  - **over.namelist** namelist input file
  - Optional initial solution file **q.restart**
  - Optional force and moment files **mixsur.fmp**, **grid.ibi**, and **grid.ptv** (fomoco) or **panels_weight.dat** (usurp)
- Multiple grid solution
  - **grid.in** plot3d grid file (multi grid format)
  - **over.namelist** namelist input file
  - **INTOUT** or **XINTOUT** overset communication file
  - Optional initial solution file **q.restart**
  - Optional force and moment files **mixsur.fmp**, **grid.ibi**, and **grid.ptv** (fomoco) or **panels_weight.dat** (usurp)

Restart and Solution Files

- Normally solution written to **q.save** plot3d format file
  - **q.save** file **<u>is overwritten</u>** with latest output
- Option to write solution to **q.<istep>** plot3d format files
- Solution file for restart is named **q.restart**
  - Restart file now contains information needed for true 2nd order time restart if running 2nd order time
  - Restart file **q.restart** is <u>not</u> automatically generated on restart

Initializing Solution Files

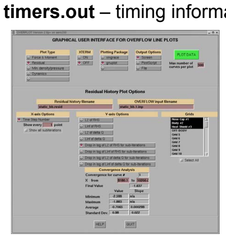
- Code will initialize to free stream input conditions in NAMELIST (initial **q.restart** not required)
- Users may write their on **q.restart** file
- Code will scale restart file when NAMELIST input values are different from those in the **q.restart** file (M_∞, α, β)
  - Allows start from existing solution file
  - Allows high speed solutions to be started from lower speed initial file

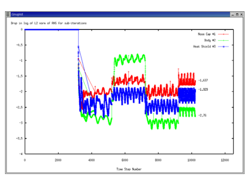
Running the Code

- Serial code
  - **./overflow**
- MPI code
  - **mpirun –np <ncpus> ./overflowmpi**
- Serial run script (Input file **basename.inp** or **basename.n.inp**)
  - **overrun basename n**
- MPI run script
  - **overrunmpi –np <ncpus> -machinefile <hostfile> basename n**
- Run scripts perform the following tasks
  - Move ***.save** output files to ***.restart** input files
  - Highlight warnings and errors
  - Creates a log file with time/date, machine name, executable name, and NAMELIST input file name
  - Concatenates output history files upon completion

Output Files

- Solution files
  - **q.save** or **q.<istep>**
  - **q.avg** time averaged solution file
- History files (view with **overplot**)
  - **resid.out** - residual history file
  - **rpmin.out** – min ρ, min p, γ, number of reverse flow points, number of supersonic points, and max μ_t
  - **turb.out** - residual history file for turbulence equations
  - **species.out** - residual history file for species equations
  - **fomoco.out** – force and moment history file
  - **timers.out** – timing information for run





Standard Out File

Compile information:

```
OVERFLOW/OVERFLOW-D -- OVERLAPPED GRID FLOW SOLVER
   ( MPI VERSION )
         VERSION 2.1s   8 July 2008


   Compiled for SINGLE PRECISION


Compile time: Tue Sep 16 14:07:40 CDT 2008

Code was compiled with the following:
F90     = mpif77
F90FLAGS= -Mnoopenmp -fastsse -Ktrap=fp
CC      = mpicc
CFLAGS  = -fastsse -Ktrap=fp
CPP     = /lib/cpp -traditional
CPPFLAGS= -DUSE_MPI -DNOCPU_TIME

Current time: Sep 19 01:33:54 2008
```


Standard Out File (cont.)

NAMELIST Input Summary with checks:

```
** ASSUMING THIS IS AN OVERFLOW-D RUN.

THIS IS A PARALLEL RUN WITH    16 GROUPS.

GLOBAL PARAMETERS                    ($GLOBAL)
  USE grdwghts.dat FOR LOAD BALANCING?   (GRDWTS) = F
  MAXIMUM GRID SIZE              (MAX_GRID_SIZE) =         0
  RUNNING CDISC INVERSE DESIGN?        (CDISC ) = F
  SUPPRESS WRITING q.bomb FILE?        (NOBOMB) = F
  CONSERVE MEMORY?                 (CONSERVE_MEM) = F
  DEBUG OPTION (0-3)                  (DEBUG ) =    0
  NUMBER OF STEPS                     (NSTEPS) =       12000
  READ RESTART FILE?                  (RESTRT) = T
  SAVE RESTART FILE EVERY              (NSAVE )         50 STEPS
  2ND-ORDER Q RESTART OPTION       (SAVE_HIORDER) =          2
  START Q AVERAGING AT STEP        (ISTART_QAVG) =        2300
  COMPUTE FORCE/MOMENT COEFS EVERY      (NFOMO )          10 STEPS
  TURBULENCE MODEL TYPE                  (NQT)   =  205
  NUMBER OF SPECIES                     (NQC)    =    0
  USE MULTIGRID?                     (MULTIG) = F
  USE FULL MULTIGRID?                   (FMG)    = F
  NO. OF GRID LEVELS (IF MULTIG=.T.)     (NGLVL)  =    3
  NO. OF FMG CYCLES (IF FMG=.T.)       (FMGCYC) =    0    0
```


Standard Out File (cont.)

Grid Summary with checks:

```
GRID SIZE FOR GRID     1:

  NUMBER OF POINTS IN J  (JD     ) =   201
                      K  (KD     ) =   111
                      L  (LD     ) =    51

CHECKING TIME STEP SPECIFICATION FOR GRID     1:

  RUNNING TIME-ACCURATE WITH NEWTON SUBITERATIONS
    WITH DTPHYS (BASED ON V_REF) =  0.20000
                (BASED ON C_INF) =  0.21053

CHECKING BOUNDARY CONDITIONS FOR GRID     1:

    1)  BOUNDARY CONDITION TYPE#     1     DIRECTION     3
    INVISCID ADIABATIC SOLID WALL (PRESSURE EXTRAPOLATION)
    DIR= 3   J-RANGE=    1   10    K-RANGE=    1  111    L-RANGE=    1    1
    2)  BOUNDARY CONDITION TYPE#     5     DIRECTION     3
    VISCOUS ADIABATIC SOLID WALL  (PRESSURE EXTRAPOLATION)
    DIR= 3   J-RANGE=   11   41    K-RANGE=    1  111    L-RANGE=    1    1
```

Standard Out File (cont.)

Grid Splitting Summary:

```
NEAR-BODY/OFF-BODY GRID LEVEL SUMMARY:

   Level    #Grids    First    Last
  ----------------------------------------
  near-body       2        1        2

................ START GROUPR .....................


  Target (weighted) near-body grid size from grouping:          51179
  Checking near-body grids...
  Original number of near-body grids:    2
    Splitting grid     1 at J =   101
    Splitting grid     1 at K =    56
    Splitting grid     1 at J =    52
    Splitting grid     1 at K =    30
    Splitting grid     1 at J =    28
    Splitting grid     2 at J =    61
    Splitting grid     2 at L =    41
    Splitting grid     2 at J =    32
    Splitting grid     2 at K =    26
    Splitting grid     3 at K =    56
```


Standard Out File (cont.)

Load Balance Summary by Groups:

```
Load balance will be based on grid size.

   Summary of work distribution for    16 groups:

   Group     Kpts      %load   Grid list
     1       151       100   20   13   11
     2       151       100   35   12    2
     3       151       100   32    3    8
     4       151       100   37   16   22
     5       151       100   43   17   26
     6       151       100   46   19   39
     7       151       100   45   36   24
     8       151       100   48   18    9
     9       150       100    1   34   47
    10       150       100   29   33   40
    11       150       100   28    7   41
    12       150       100    5    4   38
    13       150       100   42   21   27
    14       150       100   30   44   23
    15       150       100   31   15   10
    16       150       100    6   14   25

   Predicted parallel efficiency is   100%,

    based on a maximum of   151K grid points per group
    compared to an average of   150K points (weighted)

   Estimated parallel speedup is    16.0
```


Standard Out File (cont.)

Residual Summary:

```
    FOR GRID    1  AT STEP      12030   L2NORM =  0.80239815E-05
 FLOW SOLVER/TURB MODEL/SPECIES CONVERGENCE RATES:
0.9995    1.0214    0.0000
    FOR GRID    2  AT STEP      12030   L2NORM =  0.92307528E-05
 FLOW SOLVER/TURB MODEL/SPECIES CONVERGENCE RATES:
1.0041    0.9737    0.0000
    FOR GRID    1  AT STEP      12040   L2NORM =  0.76590641E-05
 FLOW SOLVER/TURB MODEL/SPECIES CONVERGENCE RATES:
0.9954    1.0133    0.0000
    FOR GRID    2  AT STEP      12040   L2NORM =  0.91043430E-05
 FLOW SOLVER/TURB MODEL/SPECIES CONVERGENCE RATES:
0.9986    1.0114    0.0000
    FOR GRID    1  AT STEP      12050   L2NORM =  0.78810453E-05
 FLOW SOLVER/TURB MODEL/SPECIES CONVERGENCE RATES:
1.0029    0.9666    0.0000
    FOR GRID    2  AT STEP      12050   L2NORM =  0.85000529E-05
 FLOW SOLVER/TURB MODEL/SPECIES CONVERGENCE RATES:
0.9932    1.0180    0.0000
   Wrote 2nd-order restart file -- q.save
   Elapsed simulation time (based on V_ref):   0.2410016E+04
   Wrote restart file -- q.avg
   Elapsed simulation time (based on V_ref):   0.2410016E+04
   q.avg data collected over        9751 steps
```


Standard Out File (cont.)

Timer Summary on Completion of Run:

```
--------------------------------------------
TIMED FOR    12000 LEVEL-1 STEPS
--------------------------------------------


--------------------------------------------
BREAKDOWN FOR TOTAL      (TIMER NUMBER   1)
--------------------------------------------
  N    TIMER          %           TIME
 --- ----------    ---------    ------------
  1  TOTAL         100.00      1.8575E+05
  2  OVERGL           0.00      2.4827E-03
  3  OVERSZ           0.01      1.0730E+01
  4  OVERST           0.00      3.2468E+00
  5  OVERFL          99.99      1.8573E+05
  6  OVERDO           0.00      1.4612E+00
  7  OTHER           0.00      7.1377E-06
  8  test           0.00      0.0000E+00
  9  test           0.00      0.0000E+00

--------------------------------------------
BREAKDOWN FOR OVERFL     (TIMER NUMBER   5)
--------------------------------------------
  N    TIMER          %        TIME/STEP          MAX/STEP
 --- ----------    ---------    -----------        -----------
  5  OVERFL        100.00      1.5478E+01         1.5478E+01
 11  CBCXCH           8.45      1.3080E+00         1.4018E+00
 12  FLOW_SOLVE      87.85      1.3597E+01          1.3706E+01
 13  FOMOCO           0.00      0.0000E+00         0.0000E+00
 14  FLOW_Idle       2.08      3.2176E-01         4.7814E-01
 15  SAVE            1.42      2.1947E-01         2.5384E-01
 16  SIXDOF           0.00      0.0000E+00         0.0000E+00
```


Standard Out File (cont.)

Group Timing Summary on Completion of Job:

```
GROUP TIMING SUMMARY   (Time each group spent in OVERFL)
 (*) Flow Solver, (/) Chimera BC, (a) Adapt, (D) DCFCRT, (s) Grid & Q save

          0         25         50         75        100
          |----------|----------|----------|----------|
Group:  1  |**********************************************////s   98%
Group:  2  |**********************************************////    98%
Group:  3  |*********************************************////     97%
Group:  4  |*********************************************////     97%
Group:  5  |*********************************************////s    97%
Group:  6  |*********************************************////     97%
Group:  7  |**********************************************////    97%
Group:  8  |*********************************************////     98%
Group:  9  |**********************************************////    98%
Group: 10  |**********************************************////    98%
Group: 11  |**********************************************////    98%
Group: 12  |**********************************************////    99%
Group: 13  |**********************************************////    98%
Group: 14  |**********************************************////    99%
Group: 15  |**********************************************////    99%
Group: 16  |**********************************************////    99%

 Overall Measured Parallel Efficiency:  97.9%
------------------------------------------------------------


 Current time: Sep 21 05:09:40 2008
```


Debug Options

- Turbulence model diagnostics (**DEBUG**=1)
  - Baldwin Lomax **q.turb** file

Q Value	*Q1*	*Q2*	*Q3*	*Q4*	*Q5*
(J,K,LS,_)	F_{max}	y^+	$\|\Omega\|$	μ_w	-
(J,K,L,_)	$F(y)$	y	$\|\Omega\|$	μ_t	F_{max} *location*

  - Spalart Allmaras **q.turb** file

Q Value	*Q1*	*Q2*	*Q3*	*Q4*	*Q5*
(J,K,LS,_)	f_{v1}	y^+	$\|\Omega\|$	μ_t	*Turbulence index*
(J,K,L,_)	f_{v1}	y	$\|\Omega\|$	μ_t	*Transition factor*

  - SST **q.turb** file

Q Value	*Q1*	*Q2*	*Q3*	*Q4*	*Q5*
(J,K,LS,_)	ω	y^+	F_1	μ_t	k
(J,K,L,_)	ω	y	F_1	μ_t	k

- Time step **q.time** diagnostics file (**DEBUG**=2)

Q Value	*Q1*	*Q2*	*Q3*	*Q4*	*Q5*
(J,K,L,_)	Δt	CFL_j	CFL_k	CFL_l	CFL_{max}

- Residual **q.resid** diagnostics file (**DEBUG**=3)

Q Value	*Q1*	*Q2*	*Q3*	*Q4*	*Q5*
(J,K,L,_)	r_{Q1}	r_{Q2}	r_{Q3}	r_{Q4}	r_{Q5}

Namelist Inputs

NAMELIST Input

&GLOBAL /	&METPRM
&FLOINP /	&TIMACU /
&VARGAM /	&SMOACU /
&OMIGLIB /	&VISINP /
&DCFGLIB /	&BCINP /
&GBRICK /	&SCEINP /
&BRKINP /	&ADSNML /
&GROUPS /	&SPLITM /
&XRINFO /	&CDNTNS /
&GRDNAM /	&EROTOR /
&NITERS /	&SIXINP /

Color Codes:

Required once per run.

Required once per run for OVERFLOW-D mode. May be omitted if not using bricks and DCF.

Required for every grid.

Optional output control. Once per run.

Optional rotorcraft coupling input.

Required for every grid for moving body runs using internal 6DOF. May be omitted for static grid cases, GMP, or prescribed motion problems.

NAMELIST Inputs

- NAMELIST must follow standard format specifications
  - Old format $NAME $END
  - New format &NAME /
- All NAMELIST values have defaults
- Input values can be inherited from the previous grid
- Order of grids in NAMELIST must correspond to order of grids in **grid.in!!!**

Namelist Input for Euler 2D Wing

```
&GLOBAL  NSTEPS = 500,
/
&FLOINP  ALPHA  = 2.0, FSMACH = 0.8,
/

&GRDNAM  NAME   = 'WING',   /
&NITERS /
&METPRM /
&TIMACU /
&SMOACU /
&VISINP /
&BCINP
        IBTYP  =  1, 47, 10, 21,
        IBDIR  =  2, -2,  1,  3,
        JBCS   =  1,  1,  1,  1,
        JBCE   = -1, -1,  1, -1,
        KBCS   =  1, -1,  1,  1,
        KBCE   =  1, -1, -1, -1,
        LBCS   =  1,  1,  1,  1,
        LBCE   = -1, -1, -1,  1,
/
&SCEINP /
```


Inheriting Defaults in Namelist

```
&GRDNAM  NAME   = 'Grid1',  /
&NITERS /
&METPRM IRHS = 5, /
&TIMACU /
&SMOACU /
&VISINP /
&BCINP
…
/
&SCEINP /

&GRDNAM  NAME   = 'Grid2',  /
&NITERS /
&METPRM /
&TIMACU /
&SMOACU /
&VISINP /
&BCINP
…
/
&SCEINP /
```

Grid1 sets IRHS=5

Grid2 uses IRHS=5

```
&GRDNAM  NAME   = 'Grid3',  /
&NITERS /
&METPRM IRHS = 4, /
&TIMACU /
&SMOACU /
&VISINP /
&BCINP
…
/
&SCEINP /

&GRDNAM  NAME   = 'Grid4',  /
&NITERS /
&METPRM /
&TIMACU /
&SMOACU /
&VISINP /
&BCINP
…
/
&SCEINP /
```

Grid3 sets IRHS=4

Grid4 uses IRHS=4

Inviscid Flux Algorithms

RHS Options

- Central Difference (2^{nd} – 6^{th})(IRHS=0)
- Symmetric Yee (2^{nd})(IRHS=2)
- Upwind AUSM$^+$ (3^{rd} – 5^{th}) (IRHS=3)
- Upwind Roe (3^{rd} – 5^{th}) (IRHS=4)
- Upwind HLLC (3^{rd} – 5^{th}) (IRHS=5)
- Upwind HLLE++ (3^{rd} – 5^{th}) (IRHS=6)
- Low Mach preconditioning for 2^{nd} order central, HLLC, and Roe

Central Difference Scheme (2^{nd})

- Smoothing required because of odd-even decoupling
  - 4^{th} order smoothing away from shocks (DIS4)
  - 2^{nd} order smoothing near shocks (DIS2)
- Smoothing options:
  - F3D dissipation scheme (IDISS=1)
  - ARC3D dissipation scheme (IDISS=2)
  - TLNS3D dissipation scheme (IDISS=3)
  - Matrix dissipation scheme (IDISS=4)

```
&METPRM
          IRHS = 0, IDISS=3,
/
&SMOACU
          DIS2 = 2.0, DIS4 = 0.04, FSO = 2.0,
/
```


Roe Scheme

- Up to 5^{th} order upwind in space (FSO)
- Flux limiter options:
  - Koren (ILIMIT=1)
  - Minmod (ILIMIT=2)
  - Van Albada (ILIMIT=3)
  - WENO
- Limiter fix for carbuncles and strong shocks (DELTA)
- Preconditioned option

```
&METPRM
          IRHS = 4,
/
&SMOACU
          DELTA = 1.0, FSO = 3.0,
/
```


HLLC Scheme

- Up to 5^{th} order upwind in space (FSO)
- Flux limiter options:
  - Koren (ILIMIT=1)
  - Minmod (ILIMIT=2)
  - Van Albada (ILIMIT=3)
  - WENO
- Limiter fix for strong shocks (DELTA)
- Preconditioned option

```
&METPRM
          IRHS = 5, ILIMIT = 1,
/
&SMOACU
          DELTA = 1.0, FSO = 3.0,
/
```


HLLE++ Scheme

- Up to 5^{th} order upwind in space (FSO)
- Flux limiter options:
  - Koren (ILIMIT=1)
  - Minmod (ILIMIT=2)
  - Van Albada (ILIMIT=3)
  - WENO
- Limiter fix for strong shocks (DELTA)

```
&METPRM
          IRHS = 6, ILIMIT = 1,
/
&SMOACU
          DELTA = 1.0, FSO = 3.0,
/
```


High Order Upwind Schemes

- 5^{th} order upwind in space (FSO=5.0)
- Based on WENO
- Use with AUSM, Roe, HLLC, or HLLE++
- Flux limiter options:
  - WENO (ILIMIT = anything but 4)
  - Mapped WENO (ILIMIT = 4)
- Limiter fix for strong shocks (DELTA)
- Requires triple fringe interpolation boundaries

```
&METPRM
          IRHS = 5, ILIMIT = 4,
/
&SMOACU
          DELTA = 1.0, FSO = 5.0,
/
```

High Order Central Schemes

- 4^{th} or 6^{th} order in space
- Use smoothing or filtering
- Smoothing controlled using DIS2 or DIS4
- Requires triple fringe interpolation boundaries

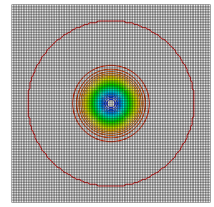
FSO = 2	**2^{nd} order central with 4/2 dissipation**
FSO = 3	**4^{th} order central with 4/2 dissipation**
FSO = 4	**4^{th} order central with 6/2 dissipation**
FSO = 5	**6^{th} order central with 6/2 dissipation**
FSO = 6	**6^{th} order central with 8/2 dissipation**

Smoothing:
&METPRM
IRHS = 0, IDISS = 2, /
&SMOACU
DISS2 = 0.0, DISS4 = 0.001,
FSO = 6.0, SMOO = 0, /

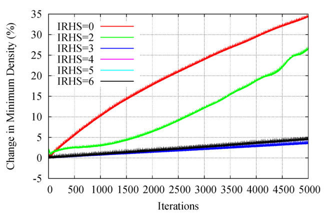
Filtering:
&METPRM
IRHS = 0, IDISS = 2,/
&SMOACU
DISS2 = 0.0, DISS4 = 0.0,
FSO = 6.0, SMOO = 0,
FILTER = 5,/

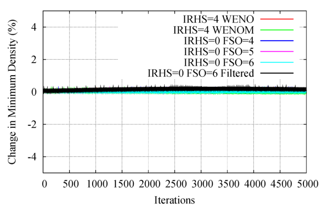
Isentropic Vortex Convection

- Prescribed starting vortex
- Free stream Mach = 0.5
- Periodic BC's in flow direction
- DTPHYS=0.01
- 1000 time steps per grid cycle
- 2^{nd} order time with 10 Newton subiterations

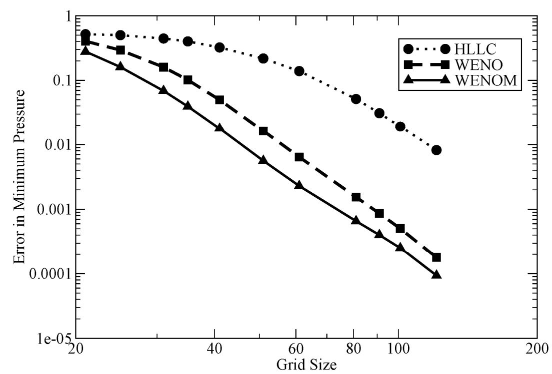

121x121 Grid

Flux Scheme Dissipation

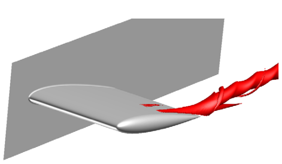


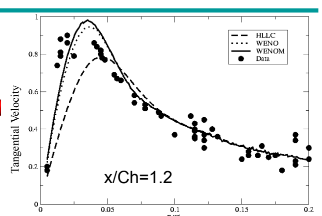


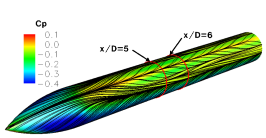
Grid Convergence

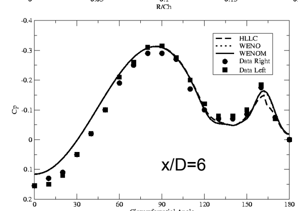


3D Vortex Preservation



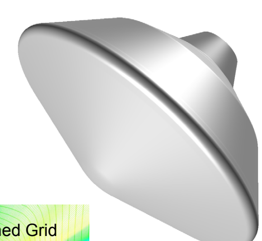


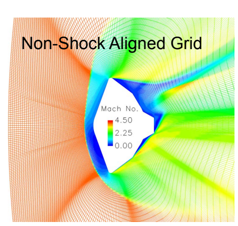




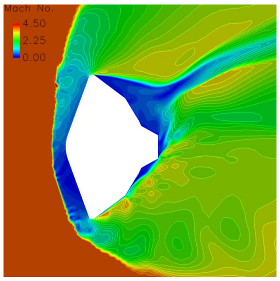
3D Capsule Test Case

- M_∞=4.02
- γ=1.246
- Re=2.49×10^5
- α=16°
- SST Turbulence Model

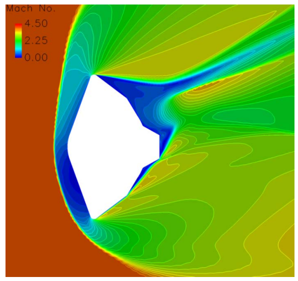




3D Capsule Symmetry Plane

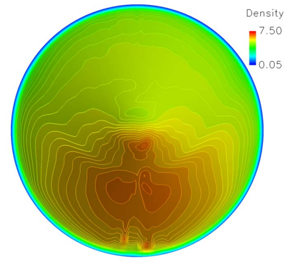


HLLC Delta=1

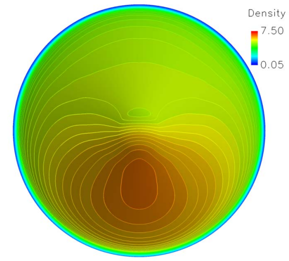


HLLE++ Delta=5

3D Capsule Windward Surface

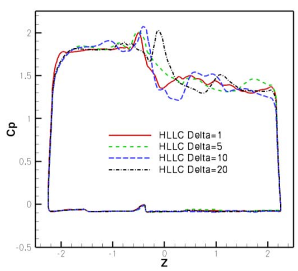


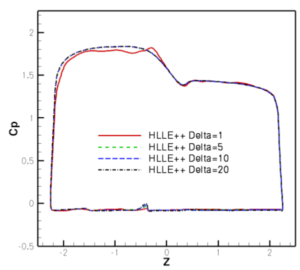
HLLC Delta=1



HLLE++ Delta=5

3D Capsule Symmetry Plane





Low Mach Number Preconditioning

Low Mach Number Preconditioning

- Eigenvalues of the inviscid fluxes (u,u+a,u-a) become stiff as u→ 0
- Modify equation set with preconditioning matrix Γ_p to rescale eigenvalues
- Must also modify RHS fluxes to use scaled eigenvalues
- Must use dual time step for time accurate simulations

$$\Gamma_p \frac{\partial \vec{q}_p}{\partial \tau} + \frac{\partial \vec{q}}{\partial t} + \frac{\partial \vec{E}}{\partial x} + \frac{\partial \vec{F}}{\partial y} + \frac{\partial \vec{G}}{\partial z} = 0$$

Eigenvalues

Navier-Stokes:

$$\lambda = \begin{bmatrix} U \\ U \\ U \\ U+C \\ U-C \end{bmatrix}$$

Smith-Weiss Preconditioned:

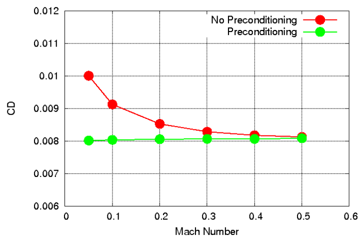
$$\beta_{\min} = 3M_{ref}^2$$

$$\beta = MAX\left[MIN\left(M^2,1\right),\beta_{\min}\right]$$

$$\lambda = \begin{bmatrix} U \\ U \\ U \\ 0.5U(\beta+1) - \sqrt{\left(0.5U(\beta-1)^2 + \beta C^2\right)} \\ 0.5U(\beta+1) + \sqrt{\left(0.5U(\beta-1)^2 + \beta C^2\right)} \end{bmatrix}$$

NACA 0012

- Re_c=1.0x10^7, α=0°
- IRHS = 5
- FSO = 3.0
- ILHS = 4
- Default BIMIN for preconditioned results



Flux Schemes Relative Timings*

2nd and 3rd Order Methods

Central IRHS=0	Yee IRHS=2	AUSM IRHS=3	Roe IRHS=4	HLLC IRHS=5	HLLE++ IRHS=6
1.00	1.68	1.91	1.38	1.58	1.71

High Order Methods

Central FSO=3	Central FSO=4	Central FSO=5	Central FSO=6	WENO	WENOM
1.04	1.06	1.15	1.19	2.66	3.63

*Sensitive to processor choice

RHS Hints

1. Default explicit smoothing (DIS4=0.04) is high for central algorithm for stability – accuracy can often be improved by reducing DIS4.
2. HLLE++ is the best choice for high speed flows with non-grid aligned grids.
3. DELTA can normally be set to 1.0 (default = no fix) for upwind algorithms. If help is needed, try setting DELTA ~ 2-10 for supersonic and hypersonic flows.
4. Preconditioning can improve accuracy of the code for low speed applications (M<0.25). Preconditioning destroys time accuracy – must use dual time stepping for time accurate applications. Sensitive to choice of BIMIN.
5. Riemann solvers are happy to have shock in the first cell off the wall. Two approaches to push off shock:
   a.) Run central difference with grid sequencing for a few steps and then switch to upwind.
   b.) Start solution with transonic Mach and increase free stream Mach to desired value.

Viscous Fluxes

Viscous Terms

- Option to include selected thin layer terms
- Option to include only the cross terms for selected thin layer terms
- Option to include all viscous terms (VISC=.TRUE.)
- Option to select wall functions (automatic)

```
&VISINP
        VISC = .TRUE., WALLFUN = .TRUE.,
/
```


Implicit Solvers

Implicit Solvers

- ADI block tridiagonal solver (Beam & Warming)
  - Central difference inviscid flux jacobians + 2nd order smoothing (ILHS=0)
  - Steger-Warming inviscid upwind flux jacobians (ILHS=5)
- F3D solver (ILHS=1)
- ADI Pulliam-Chaussee pentadiagonal solver (ILHS=2)
  - Central difference + 2nd/4th order smoothing
  - Preconditioned
- LU-SGS solver (ILHS=3)
- D3ADI diagonalized solver (ILHS=4)
  - Preconditioned
- NXAIR unfactored SSOR solver (ILHS=6,7)
  - Preconditioned

Implicit Solvers

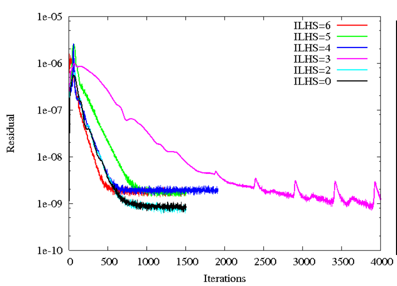
Tridiagonal ADI:
```
&METPRM
   ILHS = 0,
/
```

Diagonal ADI - Central:
```
&METPRM
   ILHS = 2,
/
```

SSOR:
```
&METPRM
   ILHS = 6, ILHSIT = 10,
/
```

Diagonal ADI - Upwind:
```
&METPRM
   ILHS = 2, IDISS =2,
/
&SMOACU
  DIS2 = 10.0, DIS4 = 0.1,
  SMOO = 0.0
/
```


Axisymmetric Bump Convergence



ILHS	IRHS	Time to Converge (sec.)
0	0	30.5
2	0	21.0
3	0	83.3
4	5	10.0
5	5	32.5
6	5	33.4

Solver Relative Timings*

B&W C ILHS=0	F3D ILHS=1	Diag. ILHS=2	LU-SGS ILHS=3	D3ADI ILHS=4	B&W U ILHS=5	SSOR ILHS=6	SSOR ILHS=7
3.94	5.31	1.27	1.00	3.41	4.08	9.38	9.85

*Sensitive to processor and grid size

*51x51x51 Grid

*Full Viscous Terms

Subiterations

Subiteration Strategies

$$\left[I+\frac{\Delta t}{1+\theta}\left(\delta_\xi A+\delta_\eta B+\delta_\zeta C\right)\right]\Delta q^{n+1,m+1}=$$

$$-\left[\left(q^{n+1,m}-q^{n}\right)-\frac{\theta}{1+\theta}\Delta q^{n}+\frac{\Delta t}{1+\theta}RHS^{n+1,m}\right]$$

Subiteration update of q

n = iteration counter

m = subiteration counter

Recalculate RHS each subiteration with latest q

Newton Method:

Δt = Constant

Dual Time Stepping:

- Use local Δt
- Locally converge inner iteration (m)

Subiteration NAMELIST Input

Dual Time Stepping:

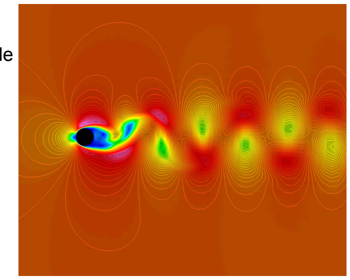
```
&GLOBAL
   NITNWT=5, FSONWT=2.0, DTPHYS=1.0,
   ORDNWT = 0,
/
&TIMACU
     ITIME=1, DT=0.1, CFLMIN=10,
/
```

Newton Time Stepping:

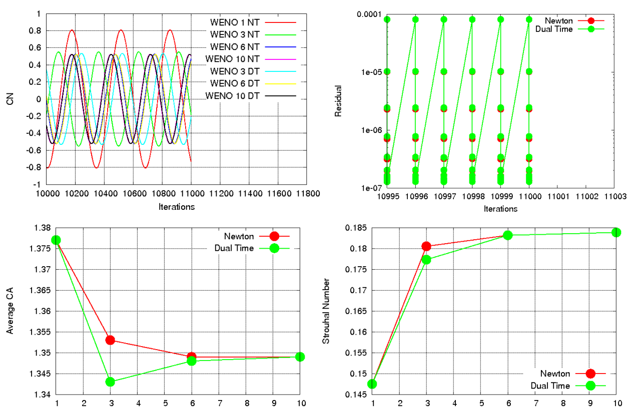
```
&GLOBAL
   NITNWT=5, FSONWT=2.0, DTPHYS=1.0,
   ORDNWT=3,
/
&TIMACU
     ITIME=0,DT=0.0,
/
```


Laminar Cylinder

- M_∞ = 0.2, Re_D =150
- DTPHYS = 0.02 (9.123×10^{-5} sec.)
  - 272 time steps per lift cycle
  - 136 time steps per drag cycle
- 5th order WENO inviscid fluxes
- Upwind Tridiagonal Solver
- 2nd order time
- 401x201 grid



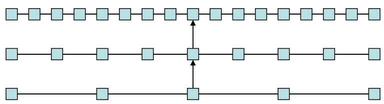
Laminar Cylinder



Convergence Acceleration

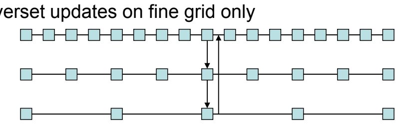
Grid Sequencing

- Get an improved initial solution quickly on coarser grids
- Works best for grids with $2^m n_{coarse}+1$ points
  - m = number of levels
  - n_{coarse} = number of points in coarse grid
- Not all grids will sequence
  - Too few points in fine grid
  - Highly skewed fine grid
- Overset updates on fine grid only
- Can jump over holes on coarser grids



Multigrid

- Construct solution at each time step from solutions on coarse and fine grid levels
- Currently use a "V" multigrid cycle
- Quickly dissipate low frequency error
- Transport turbulence models and species equations are not solved with multigrid
- Works best for grids with $2^m n_{coarse}+1$ points
  - m = number of levels
  - n_{coarse} = number of points in coarse grid
- Not all grids will sequence
  - Too few points in fine grid
  - Highly skewed fine grid
- Can jump over holes on coarser grids
- Overset updates on fine grid only



Convergence Acceleration NAMELIST Input

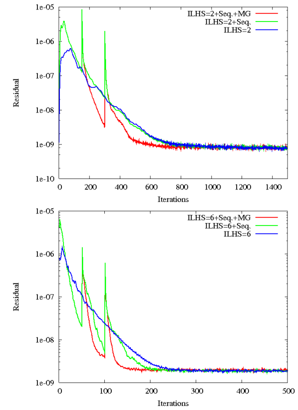
Grid Sequencing:

```
&GLOBAL
   FMG = .TRUE., NGLVL=3,
   FMGCYC = 150,150,
/
```

Multigrid:

```
&GLOBAL
   MULTIG = .TRUE., NGLVL=3,
/
```


Axisymmetric Bump Convergence



ILHS	Time to Converge (sec.)
2*	26.3
2 +Seq.	20.5
2+Seq.+MG	22.8
6**	51.3
6+Seq.	37.4
6+Seq.+MG	52.8

* Local time step, IRHS=0
** 3 Newtons, global time step, IRHS=5

LHS Hints

1. Recommended values for default diagonalized solver (ILHS=2) when used with upwind algorithms (IRHS=3,4,5): IDISS=2, SMOO=0.0, DIS2=10.0, DIS4=0.1
2. Recommend use of ILHS=2 or 4 (fastest) or ILHS=6 (most stable)
3. Use local time stepping and diagonalized solvers (ILHS=2,4) for low speed preconditioning. Use dual time stepping for time accurate solutions.
4. Use local time stepping and dual time stepping with diagonalized solvers. Recommended time step ITIME=1, DT=0.1, CFLMIN=10.
5. SSOR solver (ILHS=6) often does not need local time stepping. Best when used with Newton subiteration and second order time (FSONWT=2.0, NITNWT=3, DTPHYS=1.0)
6. Use grid sequencing (FMG=.TRUE., NGLVL=3, FMGCYC=150,150) when possible to accelerate solution convergence.
7. DT nondimensionalized by a_∞. DTPHYS nondimensionalized by V_∞. (DTPHYS = M_∞*DT)

Boundary Conditions

Boundary Condition Inputs

- Select BC type with IBTYP
- Specify direction with IBDIR
  - 1 for +J, 2 for +K, 3 for +L
  - -1 for –J, -2 for –K, -3 for -L
- Specify region with JBCS, JBCE, KBCS, KBCE, LBCS, LBCE
  - -1 for last point
  - -n for nend-n+1
- BCPAR1, BCPAR2 for BC specific values
- All boundaries must have a BC or be interpolated – if not the BC checker will stop the code

Boundary Condition Hints

- Walls (IBTYP=1-8)
  - Pressure extrapolation more stable and less sensitive to non-normal grid lines
  - Momentum equation more accurate (requires grid lines to be near normal to surface)
- Symmetry planes
  - IBTYP=11-13 requires reflection plane (implicit)
  - IBTYP=17 does not require reflection plane (explicit slip wall)
- Polar axis (IBTYP=14-16) $f_0=f_1+\frac{1}{2}\alpha(f_1-f_2)$
  - α=1 (default) is 1st order extrapolation (more accurate, less stable)
  - α=0 is 0th order extrapolation (less accurate, more stable)
  - α set by BCPAR1

Boundary Condition Hints (Cont.)

- IBTYP=21 - 2D – 3 planes ±1 in Y
- IBTYP=22 – Axi in Y, rotate about X ±1°
- IBTYP=41 – Nozzle inflow
  - BCPAR1=p_o/p_{oinf}
  - BCPAR2=T_o/T_{oinf}
- IBTYP=141 – Uniform nozzle inflow
  - BCPAR1=p_o/p_{oinf}
  - BCPAR2=T_o/T_{oinf}
- IBTYP=47 – Riemann outflow with free stream for incoming information
- IBTYP=51-59 C-grid and fold-over bcs
  - Be careful with topology

Boundary Condition Input

```
&BCINP
    IBTYP =  5,  17,  30,  141,  21,
    IBDIR  = -2,   2,   -1,     1,    3,
    JBCS  =  1,   1,   -1,     1,    1,
    JBCE  = -1,  -1,   -1,     1,   -1,
    KBCS =  -1,   1,    1,     1,    1,
    KBCE =  -1,   1,   -1,    -1,   -1,
    LBCS  =  1,   1,    1,     1,    1,
    LBCE  = -1,  -1,   -1,    -1,    1,
    BCPAR1(4) = 1.0,
    BCPAR2(4) = 1.0,
 /
```


Species Equations

Species Transport Equations

$$\frac{\partial \rho c_i}{\partial t}+\frac{\partial \rho U c_i}{\partial \xi}+\frac{\partial \rho V c_i}{\partial \eta}+\frac{\partial \rho W c_i}{\partial \zeta}=\frac{\partial}{\partial \xi}\left[\left(\frac{\mu}{\sigma_L}+\frac{\mu_t}{\sigma_T}\right)\left(\xi_x^2+\xi_y^2+\xi_z^2\right)\frac{\partial c_i}{\partial \xi}\right]+$$
$$\frac{\partial}{\partial \eta}\left[\left(\frac{\mu}{\sigma_L}+\frac{\mu_t}{\sigma_T}\right)\left(\eta_x^2+\eta_y^2+\eta_z^2\right)\frac{\partial c_i}{\partial \eta}\right]+\frac{\partial}{\partial \zeta}\left[\left(\frac{\mu}{\sigma_L}+\frac{\mu_t}{\sigma_T}\right)\left(\zeta_x^2+\zeta_y^2+\zeta_z^2\right)\frac{\partial c_i}{\partial \zeta}\right]+$$
$$\frac{\partial}{\partial \xi}\left[\left(\frac{\mu}{\sigma_L}+\frac{\mu_t}{\sigma_T}\right)\left(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z\right)\frac{\partial c_i}{\partial \eta}\right]+\frac{\partial}{\partial \xi}\left[\left(\frac{\mu}{\sigma_L}+\frac{\mu_t}{\sigma_T}\right)\left(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z\right)\frac{\partial c_i}{\partial \zeta}\right]+$$
$$\frac{\partial}{\partial \eta}\left[\left(\frac{\mu}{\sigma_L}+\frac{\mu_t}{\sigma_T}\right)\left(\xi_x\eta_x+\xi_y\eta_y+\xi_z\eta_z\right)\frac{\partial c_i}{\partial \xi}\right]+\frac{\partial}{\partial \eta}\left[\left(\frac{\mu}{\sigma_L}+\frac{\mu_t}{\sigma_T}\right)\left(\eta_x\zeta_x+\eta_y\zeta_y+\eta_z\zeta_z\right)\frac{\partial c_i}{\partial \zeta}\right]+$$
$$\frac{\partial}{\partial \zeta}\left[\left(\frac{\mu}{\sigma_L}+\frac{\mu_t}{\sigma_T}\right)\left(\xi_x\zeta_x+\xi_y\zeta_y+\xi_z\zeta_z\right)\frac{\partial c_i}{\partial \xi}\right]+\frac{\partial}{\partial \zeta}\left[\left(\frac{\mu}{\sigma_L}+\frac{\mu_t}{\sigma_T}\right)\left(\eta_x\zeta_x+\eta_y\zeta_y+\eta_z\zeta_z\right)\frac{\partial c_i}{\partial \eta}\right]$$

c_i = Species Mass Fraction

σ_L = Laminar Schmidt Number

σ_T = Turbulent Schmidt Number

Species Transport Equations

For each species c_i:

$$c_p - c_v = R$$

$$\frac{c_p}{R} = a_0 + a_1 T + a_2 T^2 + a_3 T^3 + a_4 T^4$$

Mass-Averaged Properties:

$$1 = \sum_{i=1}^{ngas} c_i \qquad R_{mix} = \sum_{i=1}^{ngas} c_i R_i \qquad c_{pmix} = \sum_{i=1}^{ngas} c_i c_{pi}$$

$$c_{vmix} = \sum_{i=1}^{ngas} c_i c_{vi} \qquad \gamma_{mix} = \frac{c_{pmix}}{c_{vmix}}$$

$$\overline{T} = \frac{T}{\gamma_\infty T_\infty} = \frac{\gamma_{mix}-1}{R_{mix}}\overline{e}_i = \frac{\overline{e}_i}{c_{vmix}}$$

Species Solution Options

- Differencing of species convection terms:
  - Central difference (IUPC=0 – Requires DIS2C and DIS4C)
  - Upwind difference (IUPC=1)
  - HLLC upwind difference (IUPC=2)
- Spatial order for convection terms:
  - 2 for central
  - 1-3 for upwind
  - 1-5 for HLLC
- LHS options:
  - ADI (ITLHIC = 1)
  - SSOR (ITLHIC > 1)

Constant γ, 1 Species

```
&GLOBAL
        NQC = 0,
/
&VARGAM
        IGAM=0,
/
&SCEINP
/
```

No species equation solved, so don't need &SCEINP

Variable γ, 1 Species

```
&GLOBAL
        NQC = 1,
/
&VARGAM
        IGAM=1,
        ALT0(1)=3.5,ALT1(1)=0,ALT2(1)=0,ALT3(1)=0,ALT4(1)=0,
        AUT0(1)=3.5,AUT1(1)=0,AUT2(1)=0,AUT3(1)=0,AUT4(1)=0,
/
&SCEINP
/
```

No species equation solved, so don't need &SCEINP

Variable γ, 2 Species

```
&GLOBAL
        NQC = 0,
/
&VARGAM
        IGAM=2,
        HT1=10.0,HT2=20.0,
        SCINF=1.0,0.0,
        SMW=1.0,1.2,
        ALT0(1)=3.5,ALT1(1)=0,ALT2(1)=0,ALT3(1)=0,ALT4(1)=0,
        AUT0(1)=3.5,AUT1(1)=0,AUT2(1)=0,AUT3(1)=0,AUT4(1)=0,
        ALT0(2)=3.5,ALT1(2)=0,ALT2(2)=0,ALT3(2)=0,ALT4(2)=0,
        AUT0(2)=3.5,AUT1(2)=0,AUT2(2)=0,AUT3(2)=0,AUT4(2)=0,
/
&SCEINP
/
```

No species equation solved, so don't need &SCEINP

Variable γ, ≥2 Species

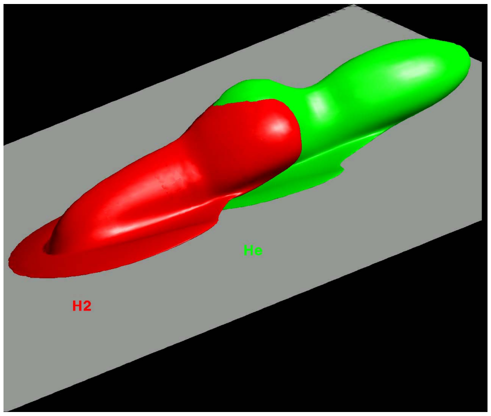
```
&GLOBAL
        NQC = 3, /
&VARGAM
        SCINF = 1.0,0.0,0.0,

        SMW(1) = 28.0, SIGL(1) = 1.0, SIGT(1) = 1.0,
        ALT0(1)=3.5,ALT1(1)=0,ALT2(1)=0,ALT3(1)=0,ALT4(1)=0,
        AUT0(1)=3.5,AUT1(1)=0,AUT2(1)=0,AUT3(1)=0,AUT4(1)=0,

        SMW(2) = 2.0, SIGL(2) = 2.0, SIGT(2) = 1.0,
        ALT0(2)=3.5,ALT1(2)=0,ALT2(2)=0,ALT3(2)=0,ALT4(2)=0,
        AUT0(2)=3.5,AUT1(2)=1,AUT2(2)=1,AUT3(2)=1,AUT4(2)=1,

        SMW(3) = 4.0, SIGL(3) = 0.9, SIGT(3) = 1.0,
        ALT0(3)=2.52,ALT1(3)=0,ALT2(3)=0,ALT3(3)=0,ALT4(3)=0,
        AUT0(3)=2.52,AUT1(3)=0,AUT2(3)=0,AUT3(3)=0,AUT4(3)=0,
/
&SCEINP
        ITLHIC = 10, IUPC=2,FSOC=3, /
```

N_2

H_2

He

H2 and HE Sonic Jets into N2



Species Hints

1. Recommend the HLLC flux with the SSOR solver (ITLHIC = 10, IUPC=2,FSOC=3)
2. Variable γ currently broken – pressure calculations inside code are not correct. Constant γ OK.
3. Free stream molecular weights and γ calculated based on NAMELIST inputs SCINF and SMW. SMW may be ratio or actual molecular weight. GAMINF is ignored.
4. Code initializes species to SCINF. Must modify initial q file or use specified input BC (BC 45) to get species into desired locations in grid.
5. Species concentration calculated based on local total enthalpy when using IGAM=2.
6. Most post-processors are constant γ. Use post-processing tool **vgplot** to get pressure, temperature, Mach number, enthalpy, γ, and species mass fraction (c_i) for plotting.

Turbulence Models

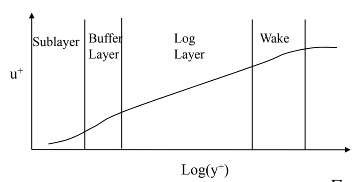
Turbulence Models

- Baldwin-Lomax (with wake model) (NQT=0)
- Baldwin-Barth 1 equation transport model (NQT=100)
- Spalart-Allmaras 1 equation transport model with trip line specification (NQT=101)
- Spalart-Allmaras 1 equation transport model (NQT=102)
- SA-DES hybrid RANS/LES (NQT=103)
- k-ω (1988) 2 equation transport model
  - DDADI LHS(NQT=202)
  - SSOR LHS (NQT=203)
- SST with compressibility correction 2 equation transport model
  - DDADI LHS (NQT=204)
  - SSOR LHS (NQT=205)
- Hybrid RANS/LES (SA and SST)
  - DES (IDES=1)
  - DDES (IDES=2)
  - MS (IDES=3)
- Rotational and curvature corrections (SA and SST)
  - SARC and SSTRC (IRC=1)
  - ASARC and ASSTRC (IRC=2)
- SST-MS hybrid RANS/LES (NQT=207)
- Wall functions for BB, SA, k-ω, and SST transport models

Wall Distance Calculations

- NAMELIST **$GLOBAL** parameters **WALLDIST** and **NWALL** used to control how and when wall distance is calculated
- **WALLDIST** options:
  - **WALLDIST = 0** – Read precomputed wall distance from file *walldist.dat* (PLOT3D function file format)
  - **WALLDIST = +/-1** – Simple computation of wall distance (only uses walls contained within the grid, ignores iblank)
  - **WALLDIST = +/-2** – Global wall distance computation
  - If **WALLDIST** is negative, write wall distance file *walldist.dat*
- **NWALL** – Recompute wall distance every **NWALL** steps
  - Currently ignored
  - Global wall distance currently computed on startup and after grid adaptation

Baldwin Lomax - Walls



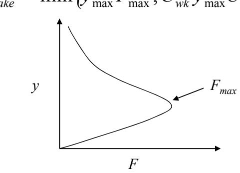
$$(\mu_t)_{inner} = \rho L_m^2 |\Omega|$$

$$(\mu_t)_{outer} = \rho K C_{cp} F_{wake} F_{kleb}$$

$$F_{kleb}(y) = \left[1 + 5.5\left(\frac{C_{kleb} y}{y_{\max}}\right)^6\right]^{-1}$$

$$F_{wake} = \min\{y_{\max} F_{\max}, C_{wk} y_{\max} U_{diff}^2 / F_{\max}\}$$

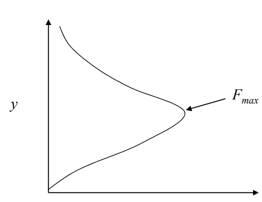
$$F(y) = y|\Omega|\left[1 - \exp\left(\frac{-y^+}{A^+}\right)\right]$$



Baldwin Lomax – Shear Layers

$$\mu_t = \rho K C_{cp} F_{wake} F_{kleb}$$

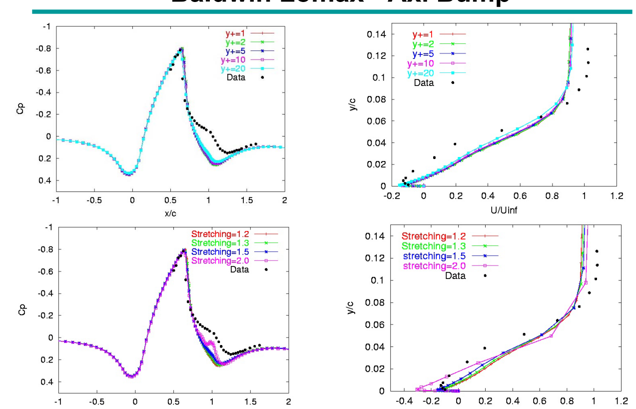
$$F(y) = y|\Omega|$$



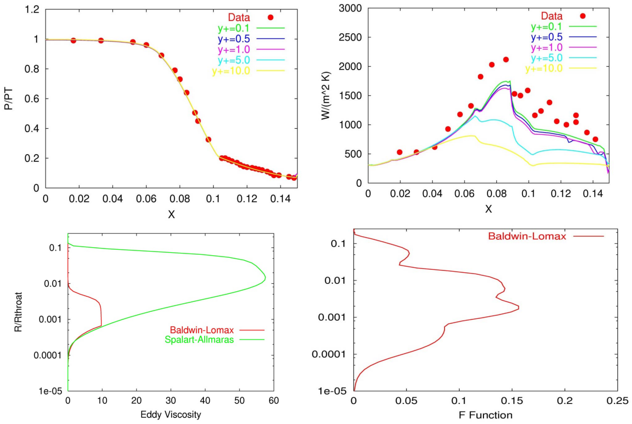
$$F_{Kleb} = \left(1 + 5.5\left(C_{Kleb} \frac{\left|y - y_{|\Omega|_{\max}}\right|}{\left(\frac{u_{dif}}{|\Omega|_{\max}}\right)}\right)^6\right)^{-1}$$

$$F_{wake} = \frac{u_{dif}^2}{|\Omega|_{\max}}$$

Baldwin Lomax - Axi Bump



Baldwin Lomax - Nozzle



Viscous Regions

- Used with Baldwin-Lomax to define where to apply the model
- Used with transport modes to specify boundary layer transition location
- Usage similar to boundary condition specification

Turbulent Region Types (ITTYP)

Type	*Description*
1	Baldwin-Lomax boundary layer model
11	Baldwin-Lomax shear layer model
102	**1- or 2-equation laminar region (zero production)**
103	Spalart-Allmaras boundary layer trip line

Baldwin Lomax Viscous Region

```
&GLOBAL
        NQT = 0,
/
&VISINP
        VISC = .TRUE.,
        ITTYP   =   1,   11,    11,
        ITDIR    =   2,    2,     2,
        JTLS     =  20,    1,  -19,
        JTLE     = -20,   19,    1-,
        KTLS      =  1,    1,     1,
       KTLE      =  -1,   -1,    -1,
        LTLS      =  1,    1,     1,
        LTLE      = -1,   -1,    -1,
        TLPAR1 =  -1,    1,     1,
/
```


Baldwin-Lomax Application Hints

1. **The Baldwin-Lomax model requires that the *F* function be well defined. This normally requires that at least three points be located within the sublayer (y^+<10).**
2. **The *F* function should be determined on lines normal to the flow direction.**
3. **The first point off the wall should be located about y^+<5 for pressure distributions, y^+<2 to obtain reasonable skin friction values, and y^+<0.5 for heat transfer.**
4. **The grid stretching normal to the wall should not exceed 1.3.**
5. **Improved heat transfer results can be obtained by using a constant spacing for the first three cells off the wall.**
6. **In order to reduce the probability of finding a second peak well off the wall, it is usually good to limit the number of points over which the *F* function is calculated.**
7. **Care should be taken not to divide viscous regions such as boundary layers when dividing the computational domain for blocked or chimera applications since the entire velocity profile is required to properly define the F_{max} and U_{diff} quantities.**

Spalart-Allmaras One-Equation Model

Advection — Diffusion

$$\frac{\partial \tilde{\nu}}{\partial t} + U_i \frac{\partial \tilde{\nu}}{\partial x_i} = \frac{1}{\sigma}\left[\nabla \cdot \left((\nu + \tilde{\nu})\nabla \tilde{\nu}\right) + C_{b2}\left(\nabla \tilde{\nu}\right)^2\right] + P(\tilde{\nu}) - D(\tilde{\nu})$$

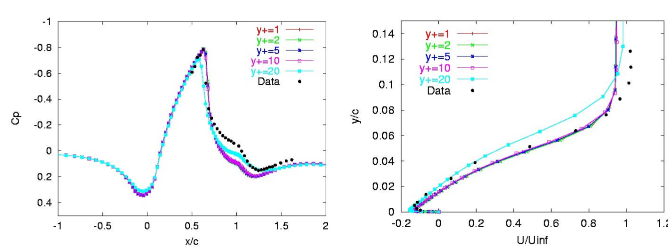
Production — Dissipation

$$P(\tilde{\nu}) = C_{b1}\left(S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}\right)$$

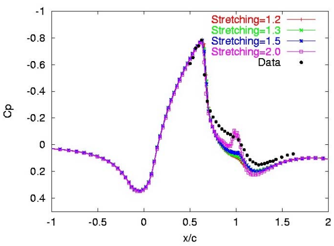
$$D(\tilde{\nu}) = C_{w1} f_w \left(\frac{\tilde{\nu}}{d}\right)^2$$

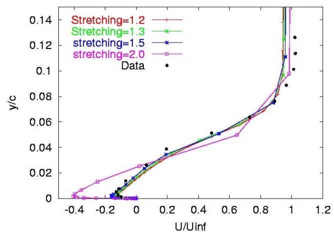
$$\nu_t = \tilde{\nu} f_{v1}$$

Effect of Initial Wall Spacing
Axi Bump - SA Model

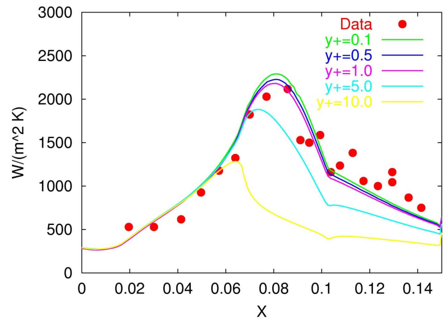


Effect of Grid Stretching Ratio
Axi Bump - SA Model

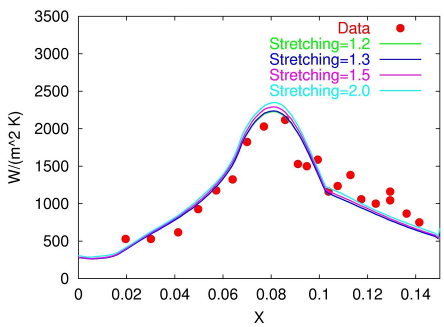




Effect of Initial Wall Spacing on
Nozzle Heat Transfer - SA



Effect of Grid Stretching on
Nozzle Heat Transfer - SA



Spalart-Allmaras Application Hints

1. **The first point off the wall should be located about y^+=1 to obtain reasonable skin friction values and about y^+=0.5 for heat transfer.**
2. **The grid stretching normal to the wall should not exceed 1.3.**
3. **The eddy viscosity should be limited so that it will not run away in some complex applications. Generally a limit of ν_t/ν=200,000 is acceptable.**
4. **Care should be taken not to divide viscous regions such as boundary layers when dividing the computational domain for blocked or overset applications since the model requires the distance from the nearest wall.**
5. **This model tends to smear out three-dimensional vortical flows (rotation and curvature corrections can help).**
6. **The model can overdamp some unsteady flows.**
7. **The model contains no corrections for compressibility and will overpredict the growth rate of high speed shear layers.**

k-ε Transport Equations

$$\frac{\partial k}{\partial t}+U_i\frac{\partial k}{\partial x_i}=\frac{\partial}{\partial x_i}\left[\left(\nu+\frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_i}\right]+P_k-\varepsilon$$

Advection Diffusion Production Dissipation

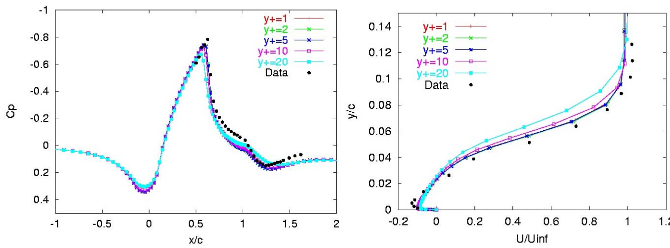
$$\frac{\partial \varepsilon}{\partial t}+U_i\frac{\partial \varepsilon}{\partial x_i}=\frac{\partial}{\partial x_i}\left[\left(\nu+\frac{\nu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_i}\right]+C_{\varepsilon 1}\frac{\varepsilon}{k}P_k-C_{\varepsilon 2}\frac{\varepsilon^2}{k}$$

$$P_k=\nu_t\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}\right)\frac{\partial U_i}{\partial x_j}-\frac{2}{3}k\frac{\partial U_i}{\partial x_i} \qquad \nu_t=C_\mu\frac{k^2}{\varepsilon}$$

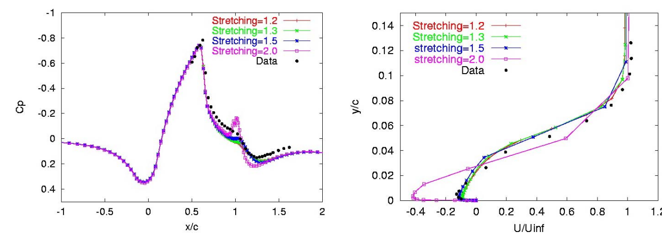
Two-Equation Model Variants

- *k*-*ε*
  - Require wall-damping terms
  - Good in shear layers
  - Lots of corrections available (compressibility, roughness, etc.)
  - Not as good near walls
- Wilcox *k*-*ω* $\left(\omega=\frac{\varepsilon}{C_\mu k}\right)$
  - Additional cross-diffusion term $\left(\nu+\frac{\nu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_i}\frac{\partial k}{\partial x_i}$
  - Good near walls
  - Pretty good in adverse pressure gradients
  - Sensitive to far field value of *ω*
- Mentor's Shear Stress Transport (SST)
  - Blended model with Bradshaw's shear stress relationship in boundary layer (improves performance in adverse pressure gradients)
  - *k*-*ω* near wall
  - *k*-*ε* away from wall

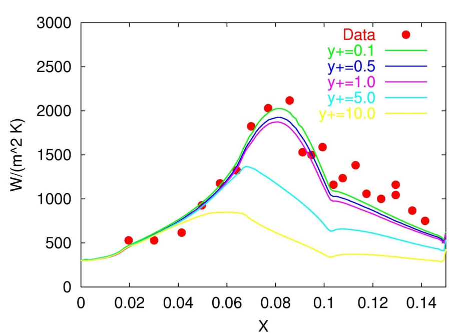
Effect of Initial Wall Spacing
Axi Bump - SST Model



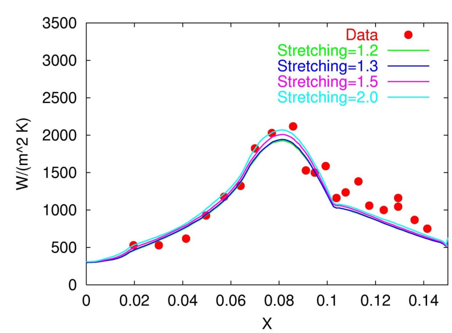
Effect of Grid Stretching Ratio
Axi Bump - SST Model



Effect of Initial Wall Spacing on
Nozzle Heat Transfer - SST



Effect of Grid Stretching on
Nozzle Heat Transfer - SST



Sarkar Compressibility Correction
SST Model

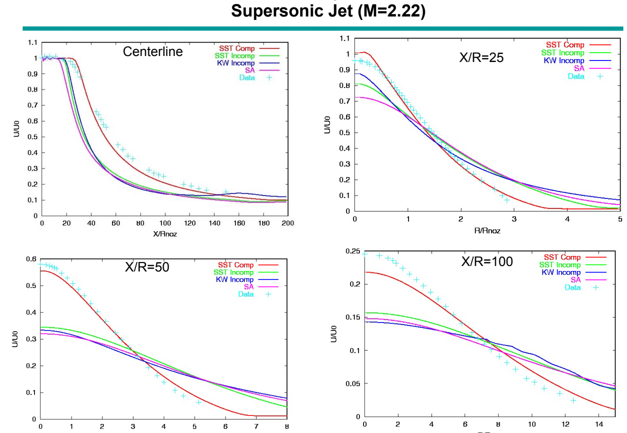
$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho U_i k}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_i}\right] + \rho P_k - \rho\left(\varepsilon + \varepsilon_c\right) + \overline{p''d''}$$

$$\varepsilon_c = \alpha_1 M_t^2 \varepsilon$$

$$\overline{p''d''} = -\alpha_2 \rho P_k M_t^2 + \alpha_3 \rho \varepsilon M_t^2$$

$$\alpha_1 = 1.0, \alpha_2 = 0.4, \alpha_3 = 0.2 \qquad M_t = \sqrt{\frac{2k}{\gamma RT}}$$

Compressibility Corrections
Supersonic Jet (M=2.22)



2-Equation Transport Model Application Hints

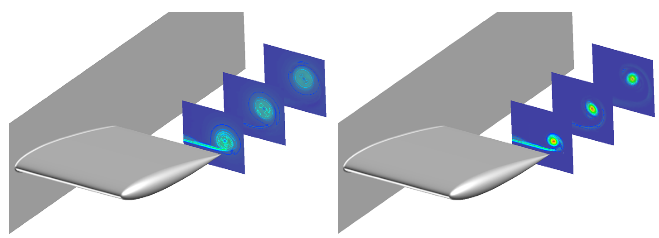
1. The first point off the wall should be located about y+=1 to obtain reasonable skin friction values and about y+=0.5 for heat transfer.
2. The grid stretching normal to the wall should not exceed 1.3. A constant spacing should be used for the first three cells off the wall for heat transfer calculations.
3. The eddy viscosity should be limited so that it will not run away in some complex applications. Generally a limit of ν_t/ν=200,000 is acceptable.
4. Care should be taken not to divide viscous regions such as boundary layers when dividing the computational domain for blocked or overset applications since the model requires the distance from the nearest wall.
5. This model can overdamp some unsteady flows.
6. Compressibility corrections should be included for high-speed shear layer flows.

Rotation and Curvature Corrections

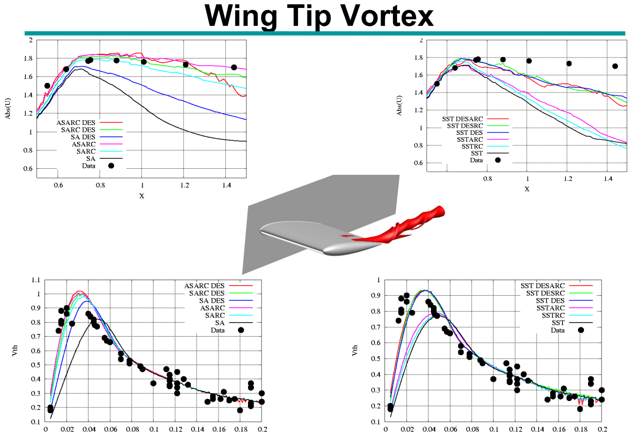
Why Do We Need R&C Corrections?

- SA and SST turbulence models are based on isotropic turbulence assumption
  - Curvature is a non-isotropic effect
- SA model uses vorticity in the production term
  - Vorticity reaches a local maximum in vortex core
  - Eddy viscosity also reaches a local maximum in the vortex core and overdamps the vortex core
- SST uses strain in the production term
  - Strain reaches a local minimum in a vortex core

Vorticity and Strain Magnitude



Wing Tip Vortex



Wall Functions

What are wall functions?

Solid wall boundary conditions based on curve fits from some point inside the boundary layer to the wall requiring functional expressions for:

- Velocity, pressure, and temperature
- Turbulence transport variables
- Wall shear stress and heat transfer
- Species

Why use wall functions?

- Improve solution turnaround by reducing the number of points in a solution
- Simplify grid generation
- Improve numerical stability
- Improve wall approximation with multigrid or grid sequencing methods

Profiles for determining τ_w and q_w

Spalding's equation with the outer velocity profile of White and Christoph

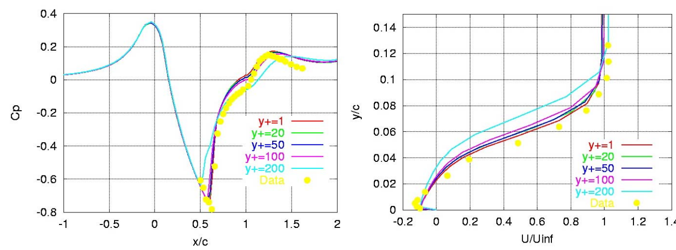
$$y^+ = u^+ + y^+_{White} - e^{-\kappa B}\left[1 + \kappa u^+ + \frac{(\kappa u^+)^2}{2} + \frac{(\kappa u^+)^3}{6}\right]$$

$$y^+_{White} = exp\left\{\frac{\kappa}{\sqrt{\Gamma}}\left[sin^{-1}\left(\frac{2\Gamma u^+ - \beta}{Q}\right) - \varphi\right]\right\} exp(-\kappa B)$$

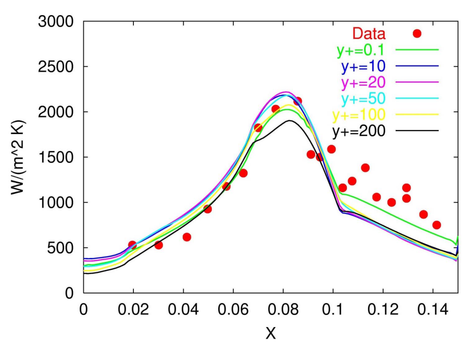
Crocco-Busemann temperature profile

$$T = T_w\left[1 + \beta u^+ - \Gamma(u^+)^2\right]$$

Effect of Initial Spacing on Ames Axi Bump – SST Model with Wall Functions



Effect of Initial Grid Spacing on Nozzle Heat Transfer – SST with Wall Functions



Wall Function Application Hints

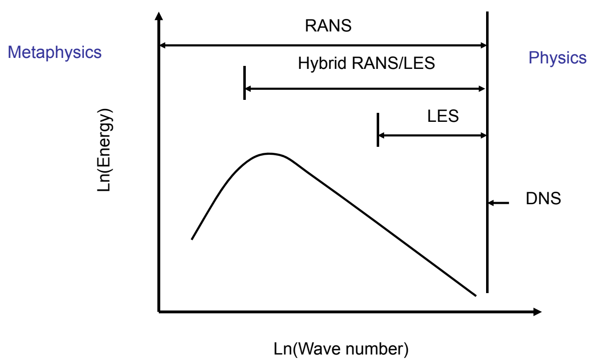
1. **Wall functions are available for all the transport turbulence models.**
2. **The initial wall spacing should be about y+=50.**
3. **The automatic feature allows you to use wall functions in selected areas of a grid since it is controlled by wall spacing.**
4. **The force and moment coefficients generated by OVERFLOW 2 include the wall functions for calculating skin friction.**
5. **The wall functions must be included to accurately post-process skin friction or heat transfer.**

Hybrid RANS/LES Unsteady Turbulence Models

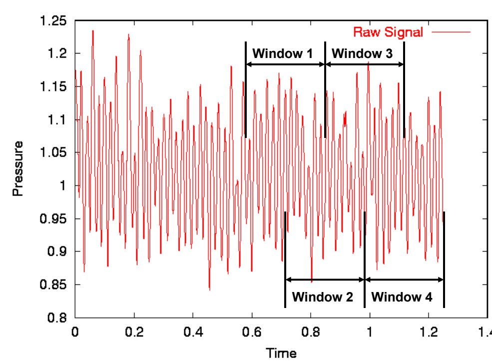
What are Hybrid RANS/LES Models?

- Similar to LES
  - Use a filtered RANS model as an LES subgrid model
  - Require low numerical dissipation in NS flux scheme
  - Only for unsteady applications
- Goal is to get RANS in boundary layer, LES everywhere else
- Applicable to flows with large-scale turbulent structures away from the walls
  - Vortex shedding
  - Weapons bays
  - Shear layers
- Solutions are grid and time step dependent
  - Must use statistical parameters to judge convergence
  - Work best for nearly isotropic grids

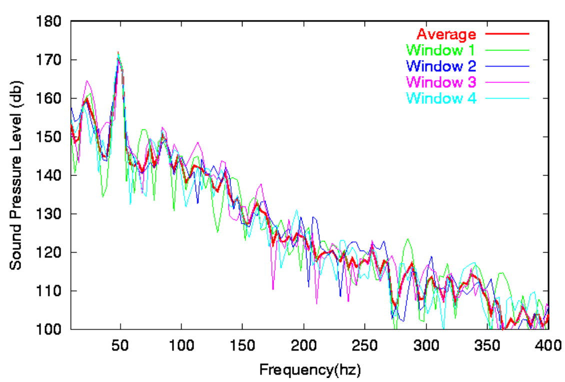
Turbulence Model Partitioning of the Energy Spectrum



Data Sample Windows



Spectral Variation



Statistics Summary

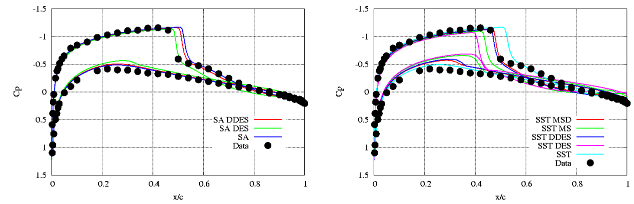
	Average Pressure	OASPL (db)	Peak SPL (db)	Frequency of Peak SPL (hz)
Window 1	1.02447	171	171	48.577
Window 2	1.02412	171	171	48.577
Window 3	1.01888	171	170	48.577
Window 4	1.02000	171	171	48.577
Average	1.02187	171	171	48.577
Max Error	0.29%	0.22%	0.747%	0.0%

Detached Eddy Simulation (DES) And Multiscale (MS)

- SA – Modify destruction term by replacing distance from wall with local grid scale
  - Increases turbulent destruction as grid is refined
  - Becomes a Smagorinsky model in LES limit
  - Does not include a turbulent length scale
- SST – Modify destruction term in k equation to be a function of the ratio of the turbulent length scale ($k^{3/2}/\varepsilon$) to local grid scale
  - Adds additional turbulent destruction to k equation as grid is refined
- MS – Filters eddy viscosity as a function of the ratio of the turbulent length scale ($k^{3/2}/\varepsilon$) to local grid scale

Delayed Detached Eddy Simulation (DDES)

- DES and MS models tended to transition prematurely to LES in the boundary layer when grid becomes refined
- Can produce solutions that are neither RANS or LES
- DDES slows transition to LES in boundary layer using functions of turbulent length scale to distance from the wall



WICS Bay Experiment

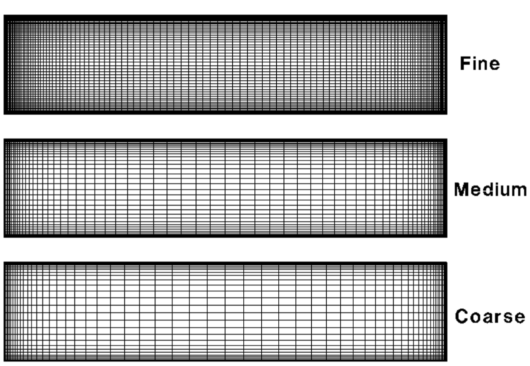
•Geometry
–Rectangular bay 18”x4”x4”
–15” flat plate in front bay
•Test Conditions
–M=0.95
–Re=2.5x10^6/ft
•Instrument Locations
–K16-bay ceiling centerline 0.275” from back wall
–K18-bay back wall centerline 0.725” from opening

WICS Bay Grid Systems

Grid	Total Points	Bay Grid Dimensions	Bay Grid Δx_{max}	Bay Grid Δy_{max}	Bay Grid Δz_{max}
Fine	1.8x10^6	121x61x61	0.3 in.	0.1 in.	0.1 in.
Medium	1.1x10^6	71x41x41	0.6 in.	0.2 in.	0.2 in.
Coarse	7.9x10^5	61x31x31	0.75 in.	0.3 in.	0.3 in.

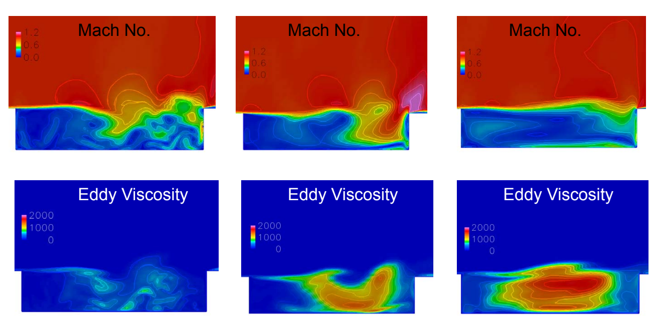
Wall Spacing of y^+=50 for all grids

WICS Bay Centerline Grids



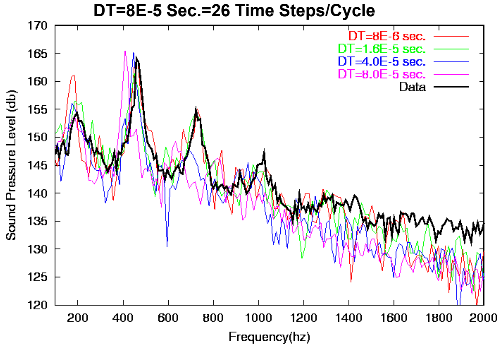
WICS Bay Centerline

FINE Medium Coarse

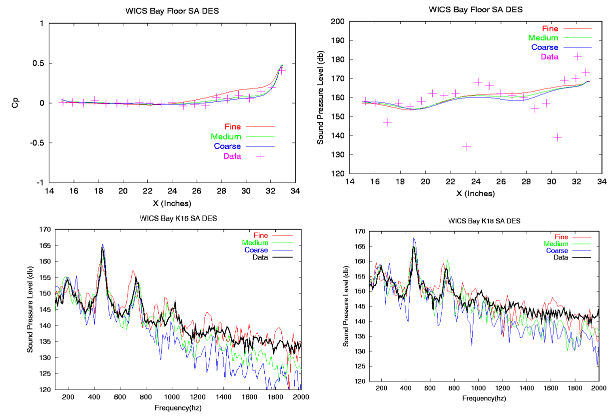


Time Step Study

DT=8.0E-6 Sec.=260 Time Steps/Cycle
DT=1.6E-5 Sec.=130 Time Steps/Cycle
DT=4.0E-5 Sec.=53 Time Steps/Cycle
DT=8E-5 Sec.=26 Time Steps/Cycle



Time Averaged and Spectral Results



Hybrid RANS/LES Application Hints

1. **These models are for unsteady applications, and should not be used with local time-stepping or with other non-time accurate algorithms.**
2. **Turbulent flows are three dimensional, and hence these models should be used only in 3D.**
3. **Because of the unsteady nature of these models, they may require a large number of time-steps to obtain a statistically stationary solution for analysis.**
4. **These models may be sensitive to the computational mesh because the filter function is inversely proportional to the grid spacing. A rule-of-thumb is that the ratio of the turbulent scale to grid length scale should be greater than two in the region of interest when using hybrid models.**
5. **As with all unsteady applications, care should be taken to be sure the time step is small enough to temporally capture the unsteady phenomena of interest.**

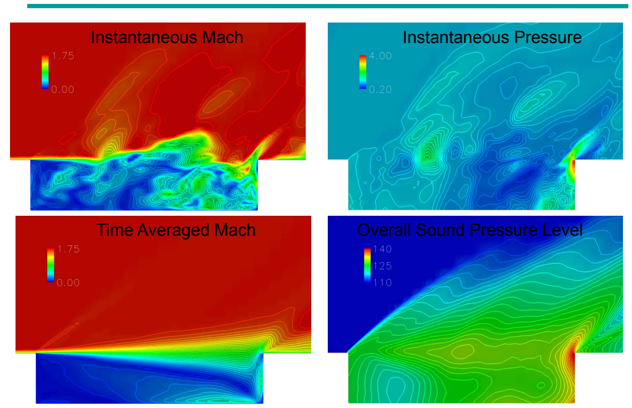
Unsteady Flow Outputs

Time Averaged Flow Output for Entire Grid System

- Code will compute a running average of the q variables and also ρ'^2, u'^2, v'^2, w'^2, and p'^2
- Averaging begins when solution reaches time step ISTART_QAVG
- Results are written to file q.avg
- q.avg file is overwritten each restart

```
$GLOBAL
    NSTEPS = 12000, ISTART_QAVG = 2000,
$END
```


WICS Bay M=1.75

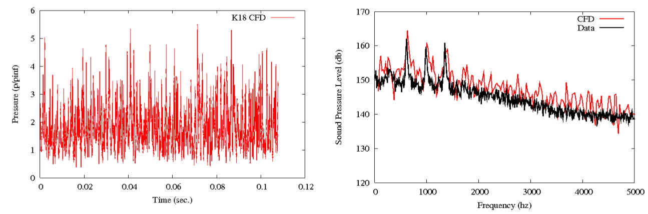


Unsteady Flow Output for Region

- IBTYPE = 201 will write the q values at each iteration for a given region of the flow to a file
- File name can be input using BCFILE
- Default name is BC_201.mesh.bc#

```
$BCINP
     IBTYP = 201,  201,  201,
     IDIR =       1,     1,      1,
     JBCS =      1,     1,      1,
     JBCE =      1,    -1,     -1,
     KBCS =      1,     1,      1,
     KBCE =      1,     1,     -1,
     LBCS =      1,     1,      1,
     LBCE =      1,     1,     -1,
$END
```


WICS Bay M=1.75

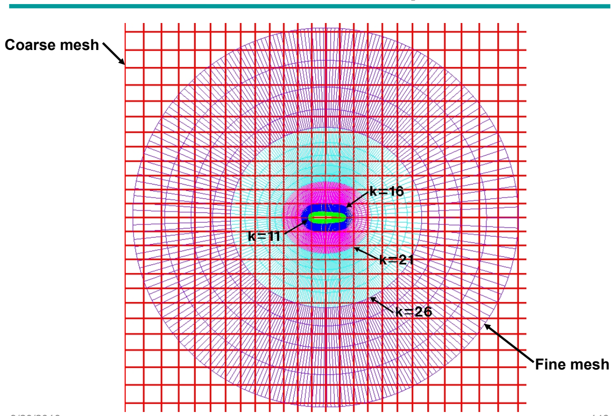


Output Information for Moving Body Simulations

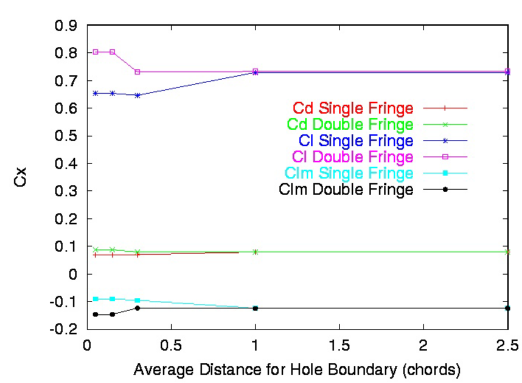
- Input parameters in **$GLOBAL**
  - **NSAVE** – grid system, flow solution, and 6-DOF restart information save every **NSAVE** steps as **x.step#**, **q.step#**, **sixdof.step#**
  - **NFOMO** – force and moment coefficients are written to fomoco.out every **NFOMO** steps (automatically set to 1 for 6-DOF simulations)
- NAMELIST **$SPLITM**: write subsets of grid and solution every nsteps (similar to CGT utilities SPLITMX, SPLITMQ)
  - **XFILE**, **QFILE**, **QAVGFILE** – specify base names for grid, solution, and/or Q-average data (if blank, don't write); step# appended to base name
  - **NSTART**, **NSTOP** – start/stop step numbers for writing output files (use -1 for last)
  - **IPRECIS** – output file precision (0 – default, 1 – single, 2 – double)
  - **IG(subset#)** – subset grid number; use **IG()=-1** for cut of all off body grids
  - **JS,JE,JI,KS,KE,KI,LS,LE,LI(subset#)** – subset ranges and increments
  - CUT(sunset#), VALUE(subset#) – off-body grid cut type ("x", "y", or "z") and corresponding x, y, or z value
  - Can have multiple $SPLITM namelists for multiple files

Overset Considerations for RANS Turbulence Models

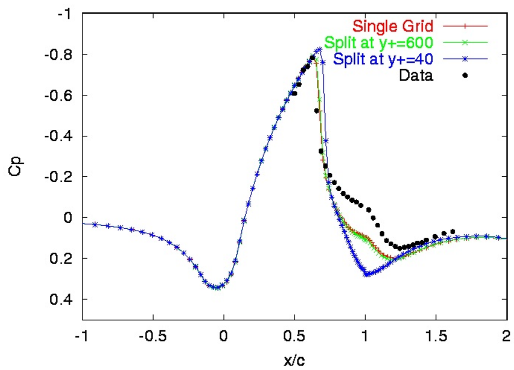
Chimera Domain Decomposition NACA 0012 Example



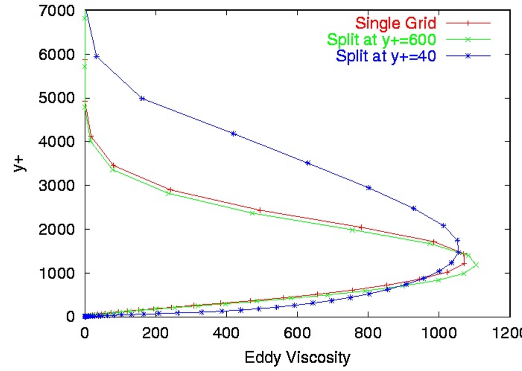
Chimera Domain Decomposition NACA 0012 Example



Chimera Domain Decomposition
Ames Axi Bump Example
WALLDIST = 1



Chimera Domain Decomposition
Ames Axi Bump Example
WALLDIST = 1



OVERSET Application Hints

1. **Holes should be cut as far from a body as possible. It is highly desirable to match the cell sizes in the overlap region.**
2. **Point injected boundaries are preferable if possible since they allow a conservative exchange of information between the computational domains.**
3. **Double fringe stencils are preferable.**
4. **Care should be taken to ensure that the turbulence model has all the information it needs within its own domain. Wall distances are required by many turbulence models and the walls that affect a domain should be included in the domain. For algebraic models care should be taken not to split the grid such that the profile from which the eddy viscosity is derived is split.**

References

1. Murphy, K.J., Buning, P.G., Pamadi, B.N., Scallion, W.I., and Jones, K.M., "Overview of Transonic to Hypersonic Stage Separation Tool Development for Multi-Stage-to-Orbit Concepts," AIAA-2004-2595, June 2004.
2. Buning, P.G., Gomez, R.J., and Scallion, W.I., "CFD Approaches for Simulation of Wing-Body Stage Separation," AIAA-2004-4838, Aug. 2004.
3. Buning, P.G., Chiu, I.T., Obayashi, S., Rizk, Y.M., and Steger, J.L., "Numerical Simulation of the Integrated Space Shuttle Vehicle in Ascent," AIAA-88-4359, Aug. 1988.
4. Renze, K.J., Buning, P.G., and Rajagopalan, R.G., "A Comparative Study of Turbulence Models for Overset Grids," AIAA-92-0437, Jan. 1992.
5. Meakin, R.L., "Object X-Rays for Cutting Holes in Composite Overset Structured Grids," AIAA-2001-2537, June 2001.
6. Meakin, R.L., "Automatic Off-Body Grid Generation for Domains of Arbitrary Size," AIAA-2001-2536, June 2001.
7. Chan, W.M., Meakin, R.L., and Potsdam, M.A., "CHSSI Software for Geometrically Complex Unsteady Aerodynamic Applications," AIAA-2001-0539, Jan. 2001.
8. Meakin, R.L., "An Efficient Means of Adaptive Refinement within Systems of Overset Grids," AIAA-95-1722, June 1995.
9. Meakin, R.L., "On Adaptive Refinement and Overset Structured Grids," AIAA-97-1858, June 1997.
10. Benek, J.A., Buning, P.G., and Steger, J.L., "A 3-D CHIMERA Grid Embedding Technique," AIAA-85-1523, July 1985.
11. Pulliam, T.H., and Chaussee, D.S., "A Diagonalized Form of an Implicit Approximate Factorization Algorithm," *J. Comp. Physics*, Vol. 39, 1981.
12. Kandula, M., and Buning, P.G., "Implementation of LU-SGS Algorithm and Roe Upwinding Scheme in Overflow Thin-Layer Navier-Stokes Code," AIAA-94-2357, June 1994.
13. Jespersen, D.C., Pulliam, T.H., and Buning, P.G., "Recent Enhancements to OVERFLOW (Navier-Stokes Code)," AIAA-97-0644, Jan. 1997.
14. Jespersen, D.C., "Parallelism and OVERFLOW," NAS Technical Report NAS-98-013, NASA Ames Research Center, Moffett Field, CA, Oct. 1998.
15. Liou, M.S., and Buning, P.G., "Contribution of the Recent AUSM Schemes to the OVERFLOW Code: Implementation and Validation," AIAA-2000-4404, June 2000.
16. Pandya, S.A., Venkateswaran, S., and Pulliam, T.H., "Implementation of Preconditioned Dual-Time Procedures in OVERFLOW", AIAA-2003-0072, Jan. 2003.
17. Nichols, R., Tramel, R., and Buning, P., "Solver and Turbulence Model Upgrades to OVERFLOW 2 for unsteady and High-Speed Applications," AIAA-2006-2824, June 2006.
18. Beam, R., and Warming, R.F., "An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form," *J. Comp. Physics*, Vol. 22, Sep. 1976, pp 87-110.
19. Klopfer, G.H., Hung, C.M., Van der Wijngaart, R.F., and Onufer, J.T., "A Diagonalized Diagonal Dominant Alternating Direction Implicit (D3ADI) Scheme and Subiteration Correction," AIAA-98-2824, June 1998.
20. Steger, J.L., and Warming, R.F., "Flux-Vector Splitting of the Inviscid Gasdynamics Equations with Application to Finite Difference Methods," *J. Comp. Physics*, Vol. 40, No. 2, Apr. 1981, pp 263-293.
21. Nichols, R., Tramel, R., and Buning, P., "Evaluation of Two High-Order Weighted Essentially Nonoscillatory Schemes," *AIAA Journal*, Vol. 46, No. 12, pp 3090-3102.
22. Nichols, R. and Heikkinen, B., "Validation of Implicit Algorithms for Unsteady Flows Including Moving and Deforming Grids," *Journal of Aircraft*, Vol. 43, No. 5, pp 1341-1351.
23. Nichols, R., "Comparison of Hybrid RANS/LES Turbulence Models for a Circular Cylinder and a Cavity," *AIAA Journal*, Vol. 44, No. 6, pp 1207-1219.
24. Nichols, R. and Nelson, C., "Wall Function Boundary Conditions Including Heat Transfer and Compressibility," *AIAA Journal*, Vol. 42, No. 6, pp 1107-1114.
25. Tramel, R., Nichols, R., and Buning, P., "Addition of Improved Shock-Capturing Schemes to OVERFLOW 2.1,", AIAA-2009-3988., Jul. 2009.
26. Nichols,R., "Algorithm and Turbulence Model Requirements for Simulating Vortical Flows," AIAA-2008-0337, Jan. 2008.

OVERFLOW 2 Training Class
Afternoon Session

Pieter G. Buning
NASA Langley Research Center
pieter.g.buning@nasa.gov

Robert H. Nichols
University of Alabama at Birmingham
CREATE-AV API Principal Developer
robert.nichols@arnold.af.mil

10th Symposium on Overset Composite Grids & Solution Technology
NASA Ames Research Center
September 20-23, 2010

Class Outline - Afternoon

- **Introduction/review**
- **OVERFLOW-D mode without grid motion**
- **OVERFLOW-D mode with grid motion**
- **Solution adaption for off-body grids**
- **Compiling and running OVERFLOW**
- **Utilities and test cases**
- **Future directions**

Introduction/Review

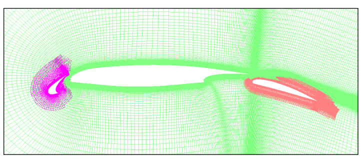
- Overset grid process
  - Compare OVERFLOW mode vs. OVERFLOW-D mode
  - Input files
  - Moving body simulation

Overset Grid Approaches

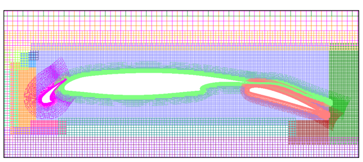
- OVERFLOW 2: two modes of operation
  - OVERFLOW mode with grid joining input from Pegasus 5 or SUGGAR
  - OVERFLOW-D mode using DCF
- OVERFLOW mode
  - All grids are created external to the flow solver (**grid.in**)
  - Pegasus 5 (or SUGGAR) used to cut holes and establish interpolation stencils (**XINTOUT**)
  - No moving body capability
  - OVERFLOW namelist input
- OVERFLOW-D mode using DCF (Domain Connectivity Function)
  - Near-body grids are created external to the flow solver (**grid.in**)
  - X-rays of body surfaces used for cutting holes (**xrays.in**)
  - Additional namelist inputs
  - Optional Geometry Manipulation Protocol (GMP) files to describe bodies and motion (**Config.xml, Scenario.xml**)
- OVERFLOW 2 can do either approach
  - Decision is based on whether additional namelists are present in input file

Example: Multi-Element Airfoil

- OVERFLOW mode
  - All grids supplied
  - Grid system from Peg 5
  - Holes are cut automatically, based on comparable cell sizes
  - Better quality grid system



- OVERFLOW-D mode
  - Only near-body grids supplied
  - Distance from surfaces specified for hole cutting
  - Holes cut by DCF inside OVERFLOW
  - Hole cutting is fast enough for moving body problems



Input Files for OVERFLOW 2

What do you need to be able to run?

- OVERFLOW mode
  - **grid.in** (all grids)
  - **mixsur.inp** (input to force and moment preprocessor)
  - **XINTOUT** (Pegasus 5 hole cutting and interpolation stencils)
  - OVERFLOW namelist input
- OVERFLOW-D mode
  - **grid.in** (near-body grids)
  - **mixsur.inp** (input to force and moment preprocessor)
  - **xrays.in** (x-rays for hole cutting)
  - OVERFLOW namelist input
  - **Config.xml, Scenario.xml** (body properties and positioning)

Moving Body Simulation Process

- Pre-processing:
  - Near-body grid generation
  - Definition of force and moment integration surfaces
  - Creating X-rays for hole cutting
- OVERFLOW grid processing:
  - Off-body grid generation
  - Hole cutting and boundary interpolation stencils
- Moving body simulation
  - Body motion (GMP interface)
  - Time-advance scheme
  - Saving motion, forces, flow solution
- Post-processing
  - Non-trivial!

Flow Simulation Process

- Starting and for grid adaption:
  - Read near-body grids, move to dynamic position(s)
  - Make off-body grids
  - Interpolate flow solution onto new off-body grids
  - Run DCF (cut holes, find interpolation stencils)
  - Advance flow solution one step
  - Compute forces and moments
- Every step:
  - Update near-body grid positions
  - Run DCF
  - Advance flow solution one step
  - Compute forces and moments

OVERFLOW-D Mode Without Grid Motion

- **NAMELIST inputs**
- **Near-body grid generation**
- **Force and moment integration**
- **Generating X-rays**
- **Off-body grid generation**
- **Grid assembly with DCF**
- **Data surface grids**

OVERFLOW Namelist Input

- **$GLOBAL** (global inputs for OVERFLOW)
- **$OMIGLB** (global inputs for OVERFLOW-D)
- **$GBRICK** (off-body grid generation)
- **$BRKINP** (user-specified proximity regions)
- **$GROUPS** (grid splitting and load-balancing)
- **$XRINFO** (hole cutting)
- **$DCFGLB** (stencil quality and repair)
- **$FLOINP** (flow parameters)
- **$VARGAM** (variable gamma/multiple species)
- Other namelists per grid…
- (Optional) **$SIXINP** (moving body properties and initial conditions, if not using GMP interface)

*Unique to OVERFLOW-D mode

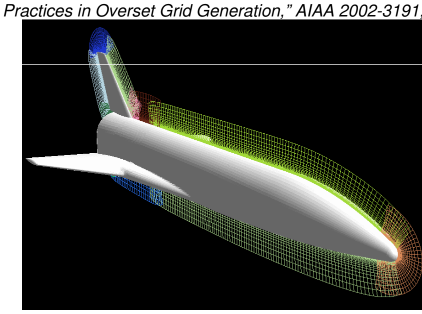
OVERFLOW Namelists per Grid

- **$GRDNAM** (grid name)
- **$NITERS** (subiterations per grid)
- **$METPRM** (numerical method selection)
- **$TIMACU** (time accuracy)
- **$SMOACU** (smoothing parameters)
- **$VISINP** (viscous and turbulence modeling)
- **$BCINP** (boundary conditions)
- **$SCEINP** (species convection equations)

See **over2.2x/doc/namelist.pdf** for a detailed list of all input parameters and definitions

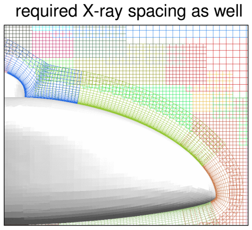
Near-Body Grid Generation

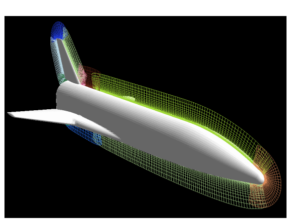
- Volume grids are generated from overset surface grids
- Use Chimera Grid Tools (CGT) or commercial package
- All near-body volume grids concatenated into **grid.in**
- Reference: *W.M. Chan, R.J. Gomez III, S.E. Rogers, and P.G. Buning, "Best Practices in Overset Grid Generation," AIAA 2002-3191, June 2002*



Near-Body Grid Generation

- Distance off wall (**S**) and outer grid spacing (**ΔS**) contribute to the size of the off-body grids
  - We will refer to **S** and **ΔS**, and how they affect the grid generation process, in following sections
- One philosophy:
  - Grow volume grids out until grid cells are roughly square
  - Grow out a total distance **S** which is about 10 times the outer cell size **ΔS**
  - This will determine the off-body grid spacing, and will contribute to the required X-ray spacing as well





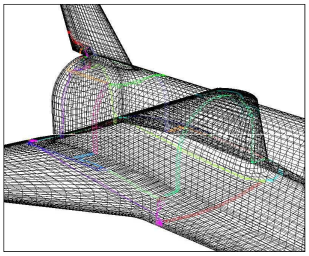
Force and Moment Integration

- Use the **mixsur** utility from CGT to generate combination of surface grids and triangular "zipper grids" for force and moment integration
  - Input file is commonly called **mixsur.inp**
  - Documentation and examples provided with CGT
- For 6-DOF simulations, aerodynamic forces will be used in OVERFLOW to update body position
  - Component (body) names in **mixsur.inp** must match GMP file component names
- Reference conditions (length, surface area, moment reference center)
  - Moment reference center will move with first body using this reference condition
  - Define separate reference conditions for each moving body
  - Example: 2 airfoils (one moves, the other is fixed)

```
0, 500, 500, -1, 0, 0 FSMACH,ALPHA,BETA,REY,GAMINF,TINF
2           NREF
1., 1., 0.25, 0., 0.    REFL,REFA,XMC,YMC,ZMC
1., 1., 0.25, 0., 0.    REFL,REFA,XMC,YMC,ZMC
```

Force and Moment Integration

- For **mixsur**, be sure to visually check resulting integration surfaces!
  - PLOT3D command files generated automatically
  - Look for missing triangles, tangled zipper grids
- **USURP** (Unique Surfaces Using Ranked Polygons) by David Boger (Penn State) is an open source alternative to **mixsur**
  - Same input file; output also recognized by OVERFLOW
  - Designed to overcome zipper grid problems



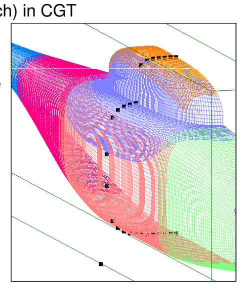
Creating X-Rays

- Creating X-rays
- Picking X-ray spacing
- Using OVERGRID to create X-rays
- X-ray number and Body ID
- Using **gen_x** to create X-rays
- Examples
- Notes and comments

Creating X-Rays

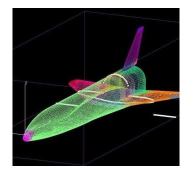
- An X-ray is an (x,y) array of z-value pierce-points of a body
  - These are used inside OVERFLOW for faster hole-cutting for grid connectivity
- Process relies (entirely) on Chimera Grid Tools (CGT)
- Create the **xrays.in** file before running OVERFLOW
- Use OVERGRID (interactive) or use **gen_x** (batch) in CGT

- Before you start, you will need to:
  - Generate a PLOT3D grid file of each body surface
  - Pick the X-ray spacing



Picking X-Ray Spacing

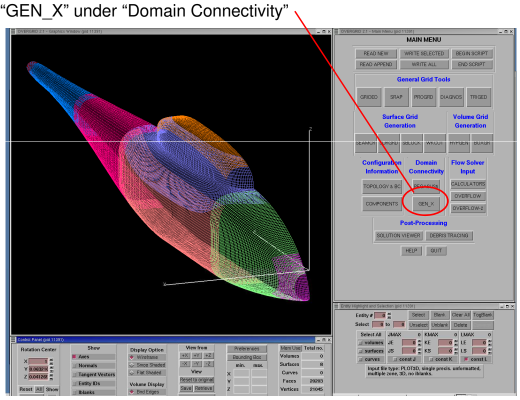
- This is the "resolution" of the body surface for the hole-cutting operation
  - The X-rays need to represent the body geometry sufficiently well to cut holes in other grids
- For single-body applications, use ½ to 1 times the outer cell size of the near-body grids (**ΔS**)
  - Too-fine X-ray spacing slows down hole-cutting (very important for moving-body problems)
  - X-rays take memory in the flow solver (proportional to spacing squared)



- For bodies in close proximity, use 0.1 to 0.2 times the distance between bodies
  - Can use different x-rays (with different spacing) for different regions
- X-rays used for collision detection also need higher resolution
  - More on this later…

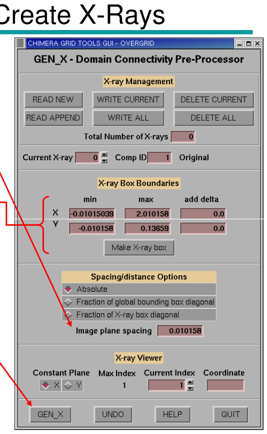
Using OVERGRID to Create X-Rays

- Start OVERGRID with the surface grid file
- Click "GEN_X" under "Domain Connectivity"



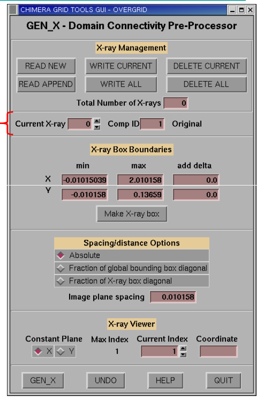
Using OVERGRID to Create X-Rays

- Enter X-ray spacing as "Image plane spacing"
  - Type <ENTER> to automatically adjust box boundaries
- Adjust box boundaries if needed
  - Ignore "add delta"
- Click "Make X-ray box"
- Click "GEN_X" to generate the X-rays
- Click "WRITE CURRENT" or "WRITE ALL" to save the X-rays to a file



X-Ray Number and Body ID

- X-rays are numbered sequentially and will be referred to by number in the OVERFLOW input
- Each X-ray is tied to a body, identified by "Comp(onent) ID" number (so when the body moves, the hole-cutting moves with it)
  - Body ID (Component ID) can be set here
  - Body ID=**n** refers to the **n**[th] component defined in the **Config.xml** file (discussed later)
- A text-input utility **xrayed** (part of CGT) allows manipulation of X-ray files
  - Combining X-ray files
  - Splitting files
  - Duplicating X-rays
  - Changing body IDs



Using **gen_x** to Create X-rays

- **gen_x** is a text-input utility in Chimera Grid Tools (CGT)
- Documentation is included with CGT (excerpted here)
  - Input files:
    - PLOT3D surface grid file
    - Input parameter file:

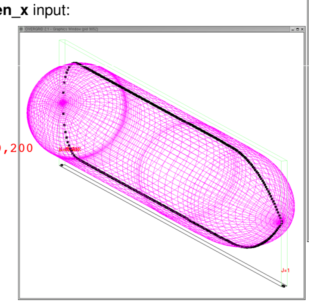
```
surface_grid_filename
1                    ISOPT(1/2/3)
0.01                 DS
0                    DELTA        (Ignore DELTA parameter)
1                    NCROPS
1                    IDBODY
-0.1, 1.1, 0, 1.5, -1, 1        XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX
```

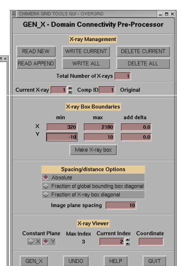
  - Output files:
    - X-ray file **gen_x.xry**
    - Output messages
  - Execution:
    - gen_x < input_parameter_file > output_messages_file
- X-rays can be read into OVERGRID for viewing

Example: Axisymmetric External Tank

- For 2D or axisymmetric geometries, X-rays only need to bound the center (y=0) grid plane
  - Create the surface grid to represent the geometry within ± the X-ray spacing of y=0
  - Set the X-ray bounding box y limits to ± the X-ray spacing
  - Comparable **gen_x** input:

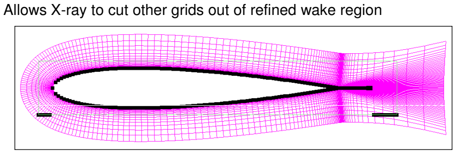
```
et.srf
1  ISOPT(1/2/3)
10 DS
0  DELTA
1  NCROPS
1  IDBODY
320,2180,-10,10,-200,200
```



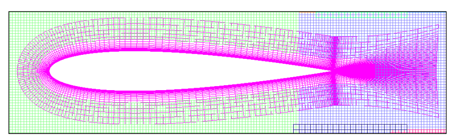


Example: 2D Airfoil

- For airfoils and wings, include a thin section of the C-grid wake with the surface grid
  - Use L=2 (or K=2) surface for finite thickness wake
  - Allows X-ray to cut other grids out of refined wake region



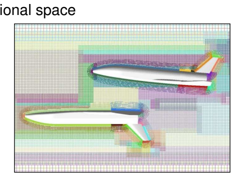
Airfoil grid and X-ray, showing extension into wake



Resulting hole in off-body grids

Notes and Comments

- "Duplicated" X-rays are useful in some cases
  - For example when multiple bodies are different only in position
  - Special format in X-ray file does not take additional space
  - X-rays can be duplicated using **xrayed** utility

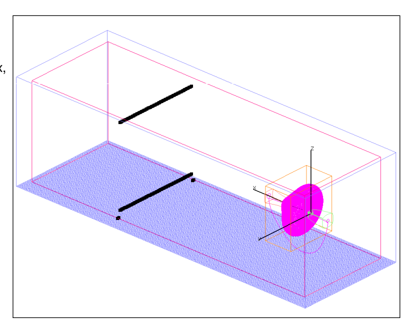


- Remember that when creating X-rays, surface grids for different bodies have to be in different files. Resulting X-ray files then have to be merged (again, using **xrayed**).

Notes and Comments

- If a user-generated box grid is added, an X-ray must be generated to cut off-body grids from the inside of the box
  - A surface grid file must be created using interior surfaces of the box grid, for example constant planes of J,K,L=8 and -8

Sample capsule plus wake box, with X-rays for the wake box. Capsule will cut a hole in the wake box; wake box will cut a hole in off-body grids.

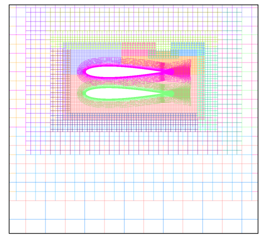
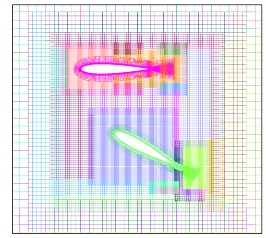


Automatic Off-Body Grid Generation

- Function of off-body grids
- Basic controls
- Specifying additional refined regions
- Controlling the rate of grid coarsening
- Specifying symmetry planes, ground planes, etc.
- Far-field boundary conditions
- Matching near-body and off-body grid spacing
- Examples
- Notes and comments

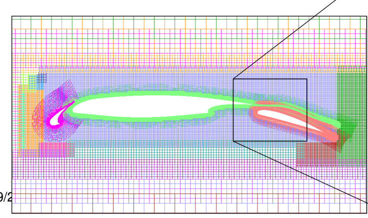
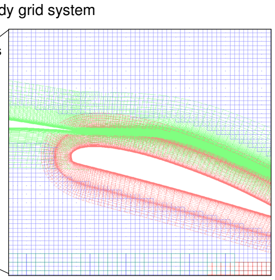
Function of Off-Body Grids

- Level-1 (finest) off-body grids:
  - Surround (all) near-body grids
  - Fill user-specified regions
  - Solution adaption (if used)
- Level-2 and coarser grids fill in to the far-field boundary

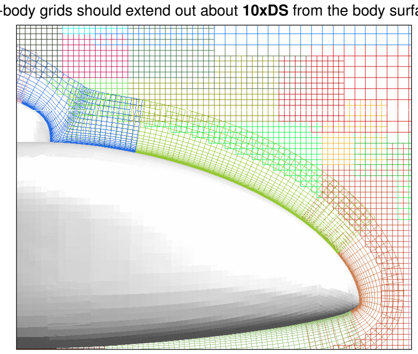
Basic Controls

- Basic controls (input parameters in **$GBRICK**):
  - **DS** – spacing for level-1 (finest) off-body grids
    - This parameter is critical for (a) proper communication with near-body grids, (b) resolving off-body flow gradients, and (c) controlling overall number of grid points.
  - **DFAR** – distance to (all) outer boundaries
  - **CHRLEN** – characteristic body length (*no longer used*)
    - Default is 1, use (major) dimension of body
  - **XNCEN,YNCEN,ZNCEN** – center of off-body grid system
    - Default is center of near-body grids
    - Must be specified for moving body problems

Matching Near-Body and Off-Body Grid Spacing

- How to pick **DS** (or, how far to grow near-body grids)?
  - **DS** should match **ΔS** (outer boundary spacing of near-body grids)
  - **DS** (and **ΔS**) should be sized to resolve off-body flow gradients
  - Near-body grids should extend out about **10xDS** from the body surface



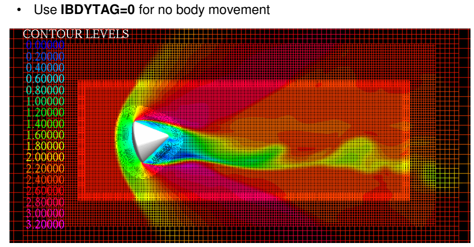
Off-Body Process Output

- Output indicates number of off-body grids generated for each level of coarseness (level-1 is finest)

```
................. START BRICK .....................

    Off-body grids generated with     2 fringe points.

    LEVEL  1:  GENERATED    35 GRIDS
    LEVEL  2:  GENERATED    44 GRIDS
    LEVEL  3:  GENERATED     5 GRIDS
    LEVEL  4:  GENERATED     5 GRIDS
    LEVEL  5:  GENERATED     5 GRIDS
    LEVEL  6:  GENERATED     5 GRIDS
    LEVEL  7:  GENERATED     5 GRIDS

................... END BRICK .....................
```


Specifying Additional Refined Regions

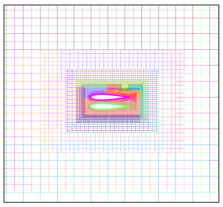
- Additional regions of level-1 grids (input parameters in **$BRKINP**):
  - **NBRICK** – number of user-specified proximity regions
    - If **NBRICK** is positive, these regions add to geometry regions
    - If **NBRICK** is negative, these regions *replace* geometry regions
  - **(XBRKMIN,XBRKMAX, YBRKMIN,YBRKMAX, ZBRKMIN,ZBRKMAX)** – min/max of proximity region
  - **IBDYTAG** – proximity region will move with motion of the indicated body
    - Example where you don't want this, e.g., wakes
    - Use **IBDYTAG=0** for no body movement



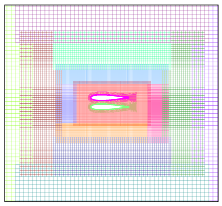
Controlling the Rate of Grid Coarsening

- Effect of **MINBUF** (in **$GBRICK**):
  - Default **MINBUF=4** gives minimum overlap between successively coarser off-body grids
  - Larger values give more gradual coarsening, but use more grid points
  - 2-airfoil example:
    - **MINBUF=4** (if geometry were 3D, off-body grids would have 2 million points)
    - **MINBUF=8** (3D off-body grids would have 3 million points)

MINBUF=4

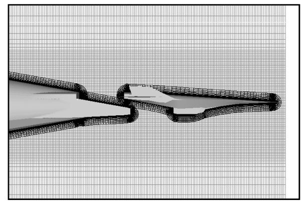


MINBUF=8

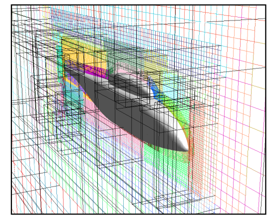


Specifying Symmetry Planes, Ground Planes, etc.

- Special planes (input parameters in **$GBRICK**):
  - Used to set a ground plane, symmetry plane, inflow plane, etc.
  - **I_XMIN=1** – use value of **P_XMIN** as off-body grid X(minimum)
  - **I_XMIN=0** – default is to use **DFAR** to set X (minimum)
  - Same for **I_XMAX, I_YMIN,I_YMAX, I_ZMIN,I_ZMAX,** and **P_XMAX, P_YMIN,P_YMAX, P_ZMIN,P_ZMAX**
  - Can only set one out of each (x,y,z) pair of values



Hyper-X supersonic inflow plane:
I_XMIN=1, P_XMIN=-20



Symmetry plane for helicopter fuselage:
I_YMIN=1, P_YMIN=0

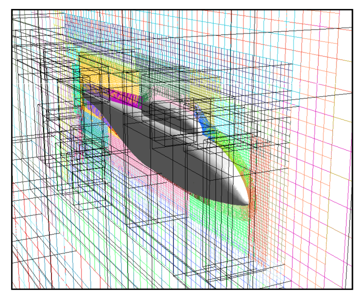
Far-Field Boundary Conditions

- Far-field boundary conditions (input parameters in **$OMIGLB**):
  - **IBXMIN** – boundary condition type for off-body grid system X (minimum) boundary
  - Same for **IBXMAX, IBYMIN,IBYMAX, IBZMIN,IBZMAX**
  - A limited number of boundary conditions are implemented:
    - Inflow/outflow conditions: BC types 30,35,37,40,41,47,49
    - 2D or axisymmetric condition (y only): BC types 21,22
    - Axis condition (z only, and combined with axisymmetric in y): BC type 16
    - Symmetry conditions: BC types 11,12,13,17
    - Inviscid wall: BC type 1
  - Default is free-stream/characteristic condition (BC type 47)

Example 1: Helicopter Fuselage

- 100-inch long body, symmetry at y=0, far-field at 1000 inches
  - Set off-body symmetry boundary condition with **IBYMIN=17**
  - Make off-body grids start at y=0 by using **I_YMIN, P_YMIN**

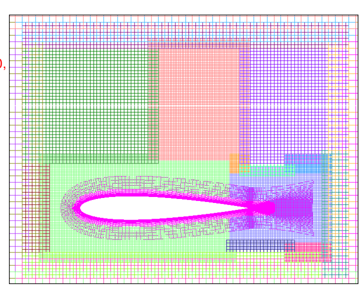
```
$OMIGLB  IBYMIN=17, …  $END
$GBRICK
    DS=1, DFAR=1000, CHRLEN=100,
    XNCEN=50, YNCEN=0, ZNCEN=0,
    I_YMIN=1, P_YMIN=0,
    $END
$BRKINP $END
```



Example 2: Airfoil With Refined Shock Grid

- 2D airfoil, chord=1, far-field at 100 chords
  - Use **$BRKINP** to add a refined level-1 region for shock
  - Since **IBDYTAG=1**, this region is tied to motion of the airfoil

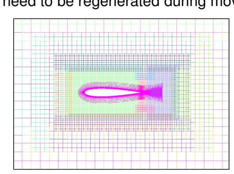
```
$OMIGLB  IBYMIN=21, …  $END
$GBRICK
    DS=0.01, DFAR=100, CHRLEN=1,
    XNCEN=0.5, YNCEN=0, ZNCEN=0,
    $END
$BRKINP
    NBRICK=1,
    XBRKMIN=0.5, XBRKMAX=0.9,
    YBRKMIN=0, YBRKMAX=0,
    ZBRKMIN=0, ZBRKMAX=1,
    IBDYTAG=1,
    $END
```



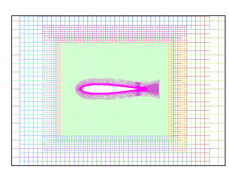
Example 3: Oscillating Airfoil

- Airfoil forced oscillation problem
  - Use **$BRKINP** to make level-1 region big enough to capture expected body motion, so that off-body grids will not need to be regenerated during moving-body run

```
$OMIGLB  IBYMIN=21, …  $END
$GBRICK
    DS=0.01, DFAR=100, CHRLEN=1,
    XNCEN=0.5, YNCEN=0, ZNCEN=0,
    $END
$BRKINP
    NBRICK= -1,
    XBRKMIN= -0.3, XBRKMAX= 1.5,
    YBRKMIN= 0, YBRKMAX= 0,
    ZBRKMIN= -0.8, ZBRKMAX= 0.8,
    IBDYTAG= 0,
    $END
```



Without **$BRKINP**

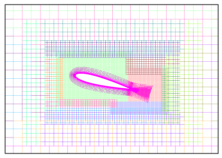


With **$BRKINP**

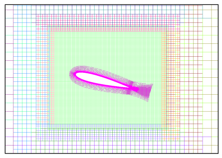
Example 3: Oscillating Airfoil

- Airfoil forced oscillation problem
  - Use **$BRKINP** to make level-1 region big enough to capture expected body motion, so that off-body grids will not need to be regenerated during moving-body run

```
$OMIGLB  IBYMIN=21, ...  $END
$GBRICK
    DS=0.01, DFAR=100, CHRLEN=1,
    XNCEN=0.5, YNCEN=0, ZNCEN=0,
    $END
$BRKINP
    NBRICK= -1,
    XBRKMIN= -0.3, XBRKMAX= 1.5,
    YBRKMIN= 0, YBRKMAX= 0,
    ZBRKMIN= -0.8, ZBRKMAX= 0.8,
    IBDYTAG= 0,
    $END
```



Without **$BRKINP**



With **$BRKINP**

Notes and Comments

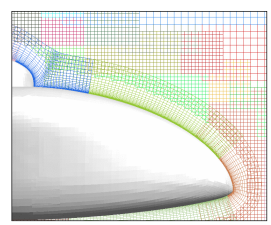
- **OBGRIDS=FALSE** – no off-body grids created
  - For single-grid problems, or where background grids are already supplied
- Files created: **brkset.save, brkset.restart**
  - Needed for moving body restarts to generate consistent off-body grids
  - Delete these files to force OVERFLOW to generate new off-body grids (for example, if you change the input parameters)
- Residual history for off-body grids is grouped by level in **resid.out**, **turb.out**, **species.out**
  - Instead of one entry per level-n grid, there is one entry representing *all* level-n grids
  - Entry contains L_2 and L_∞-norms of RHS and ΔQ
  - Entry lists (x,y,z) instead of (j,k,l) location of L_∞-norm (off-body grids only)
  - Especially appropriate for moving body or solution adaption problems, where the number of off-body grids changes every adapt cycle

DCF: Hole Cutting and Grid Assembly

- Using X-rays to cut holes
- Choosing **XDELTA**
- Orphan points and donor quality
- Double fringe interpolation
- Viscous stencil repair
- Examples
- Notes and comments

Using X-Rays to Cut Holes

- Specifying X-ray cutters (input parameters in multiple **$XRINFO**):
  - **IDXRAY** – X-ray number
  - **IGXLIST** – list of grids to be cut
    - Special grid number "-1" refers to (all) off-body grids
  - Or use **IGXBEG,IGXEND** – starting/ending grids to be cut
  - **XDELTA** – offset of hole from body surface
- Example: $XRINFO IDXRAY=1, IGXLIST=-1, XDELTA=0.05 $END
  - Use the first X-ray in **xrays.in** file, cut a hole in the off-body grids, 0.05 grid units off the X-ray surface:



Using X-Rays to Cut Holes

- Example: multi-element airfoil

```
$XRINFO  IDXRAY=1, IGXLIST=-1, XDELTA=0.02  $END
$XRINFO  IDXRAY=2, IGXLIST=-1, XDELTA=0.02  $END
$XRINFO  IDXRAY=3, IGXLIST=-1, XDELTA=0.02  $END
```

  - Slat, main, and flap X-rays (X-rays 1,2,3) cut holes in off-body grids

```
$XRINFO  IDXRAY=1, IGXLIST=2, XDELTA=0.005  $END
```

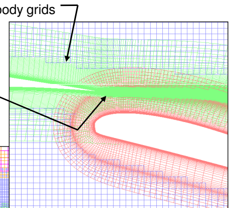
  - Slat X-ray cuts hole in main grid (grid 2)

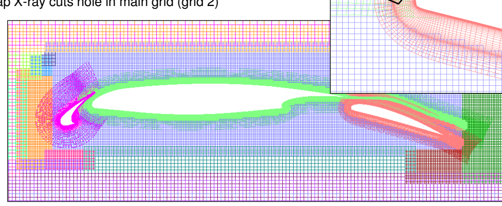
```
$XRINFO  IDXRAY=2, IGXLIST=1,3, XDELTA=0.005  $END
```

  - Main X-ray cuts hole in slat and flap grids (grids 1,3)

```
$XRINFO  IDXRAY=3, IGXLIST=2, XDELTA=0.005  $END
```

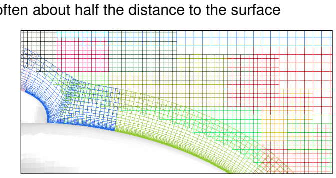
  - Flap X-ray cuts hole in main grid (grid 2)





Choosing **XDELTA**

- Holes should be cut to keep coarser grids out of high-gradient regions (such as boundary layers)
- Holes should be cut so that grids have similar resolution in overlap regions, and have sufficient overlap for interpolation of boundary data
- When cutting holes in off-body grids, choose **XDELTA** to be 5 times **DS**, *in from the outer boundary* of the near-body grids, or **XDELTA = S – 5xDS**
  - This is often about half the distance to the surface



- When cutting holes in nearby bodies, **XDELTA** must be less than half the expected minimum distance between the bodies to avoid orphan points
  - Can use different values for different cutters

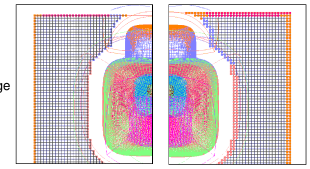
Orphan Points and Donor Quality

- Some overset grid definitions (thanks to Ralph Noack):
  - **Blanked-out points** – points inside bodies or holes, where the solution is not computed or is ignored
  - **Fringe points** – inter-grid boundary points where solution values are obtained via interpolation from another grid
  - **Donor points** – points contributing to interpolation stencils
  - **Orphan points** – fringe points without valid donors; resulting from hole cutting failure (no possible donor) or only poor quality donors are available (insufficient overlap)
- Donor stencil quality (input parameter in **$DCFGLB**):
  - "Quality" of the donor stencil refers to how much of the interpolated information has to come from donor points that are interior to the flow solution, i.e., not fringe points themselves
  - **DQUAL=1** – donor stencils must consist of only field points (default)
  - **DQUAL=0** – stencils which include all fringe points may be accepted
    - This is *not* a good idea—the simulation may simply pass boundary data back and forth between grids
  - **DQUAL=0.1** is generally acceptable

Double Fringe Interpolation

- Single vs. double fringe refers to the number of layers of interpolated points at overlapped grid boundaries. These are analogous to ghost points.
  - For single fringe interpolation, the flow solver must revert to lower-order differencing at the boundaries
  - Double fringe interpolation supports 5-point stencil differencing (up to **FSO=3** for central or upwind)
- **Double fringe is *always* highly recommended**
- Input parameter in **$OMIGLB**:
  - **LFRINGE=1** – single fringe
  - **LFRINGE=2** – double fringe; any double fringe points that are orphan points are "repaired," i.e., changed back to field points (default)
  - **LFRINGE=-2** – double fringe; do not revert orphan points to field points

Single fringe



Double fringe

Triple (and Higher) Fringe Interpolation

- Higher-order schemes in OVERFLOW need more than double fringe
  - 4th-order central, 5th-order WENO schemes need **LFRINGE=3**
  - 6th-order central needs **LFRINGE=4**
  - Default **LFRINGE** is based on numerical scheme
  - **LFRINGE** can be changed whenever grid connectivity is recomputed (DCF process)
- Off-body grids need more overlap as well
  - Use **OFRINGE** in **$GBRICK** to specify number of fringe points for off-body grids
  - Default **OFRINGE** is based on numerical scheme
  - But, **OFRINGE** cannot be changed without regenerating off-body grids
  - If you plan to use a higher-order scheme later, set **OFRINGE** now
- Orphan points cause fringe layers to degrade gradually
  - Triple fringe will locally change to double fringe, then to single fringe until orphans are resolved or converted to field points

Viscous Stencil Repair

- Viscous stencil repair (input parameters in **$DCFGLB**):
  - **MORFAN** – enable/disable viscous stencil repair (1/0)
  - **NORFAN** – number of points above a viscous wall subject to viscous stencil repair
  - Viscous stencil repair is needed to handle bad interpolations when overlapping surface grids lie on the same curved surface. If not corrected, this can result in orphan points (convex surfaces) or interpolations too high in the boundary layer (concave surfaces).
  - WARNING: Interpolation stencils for boundary points within NORFAN points of a viscous surface will be modified, using the assumption that all viscous walls have the same grid distribution in the normal direction. QUALITY OF REPAIRED STENCILS IS NOT CHECKED.
  - A better scheme is needed!

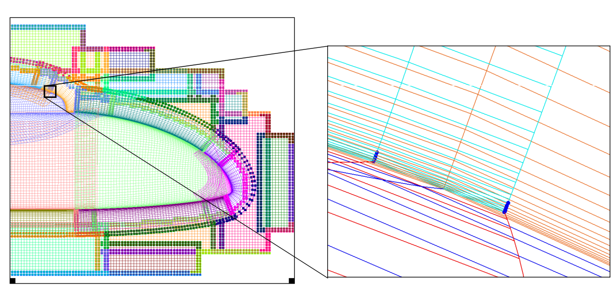
DCF Output

- Output from DCF process indicates the number of double fringe and viscous stencils repaired, and the final number of orphans

```
...............     START DCFCRT    ..................

  WARNING:  USING VISCOUS STENCIL REPAIR WITHIN      6 POINTS OF A WALL.
    Interpolation stencils for boundary points within NORFAN points of a
    viscous surface will modified, using the assumption that all viscous
    walls have the same grid distribution in the normal direction.
  WARNING:  QUALITY OF REPAIRED STENCILS IS NOT CHECKED.

  WARNING:        278 viscous stencils/orphans repaired in DCFCRT
  NO orphan points found in DCFCRT

   ORPHAN POINT SUMMARY:
     *Numbers are approximate due to grid splitting.
      Points in overlap region may be counted twice.

            Initial  Visc Stencils   Visc Orphans  Double Fringe     Final
    Grid    Orphans    Repaired        Repaired   Orphs Repaired    Orphans
    ----------------------------------------------------------------------
       1*        0        214              0              0             0
       2*       36         28             36              0             0

...............      END DCFCRT     ..................
```


Example 1

- Helicopter fuselage
  - $OMIGLB LFRINGE=2, … $END
  - $DCFGLB DQUAL=0.3, MORFAN=1, NORFAN=6, $END
  - $XRINFO IDXRAY=1, IGXLIST=-1, XDELTA=0.035, $END

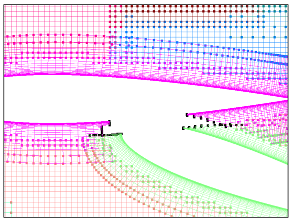


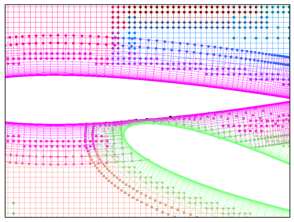
Example 2

- Airfoil drop
  - For bodies that are very close to each other, very small values of **XDELTA** may be needed

```
$OMIGLB  LFRINGE=2, ...  $END
$DCFGLB  DQUAL=0.3,  $END
$XRINFO  IDXRAY=1, IGXLIST=2,-1, XDELTA=0.04,  $END
$XRINFO  IDXRAY=2, IGXLIST=1,-1, XDELTA=0.04,  $END
```

```
$OMIGLB  LFRINGE=2, ...  $END
$DCFGLB  DQUAL=0.3,  $END
$XRINFO  IDXRAY=1, IGXLIST=-1, XDELTA=0.04,  $END
$XRINFO  IDXRAY=2, IGXLIST=-1, XDELTA=0.04,  $END
$XRINFO  IDXRAY=1, IGXLIST=2, XDELTA=0.0,  $END
$XRINFO  IDXRAY=2, IGXLIST=1, XDELTA=0.0,  $END
```





Notes and Comments

- It is OK to have "some" orphan points
  - But you should understand why, and where they are in the grid system
  - Be careful of compromising grid quality because you don't want to refine the off-body grids, or don't want to fix the near-body grids
- Orphan points become much harder to control in moving body problems
  - Have to anticipate grid movement
- OVERFLOW "fills" orphan points (and *all* hole points) with average of neighboring point values
- Input parameter **IRUN** in **$OMIGLB** allows test run of DCF:
  - **IRUN=1** – just do off-body grid generation (write **x.save** file)
  - **IRUN=2** – do off-body grid generation *and* DCF (write **x.save**)
  - **IRUN=0** – do a complete run, including flow solver
  - When changing inputs, be sure to delete **brkset.restart** and **INTOUT**, or OVERFLOW will not rerun these steps

Data Surface Grids

- Can be used to extract acoustic data surfaces, velocity profiles, pressure tap locations, 2D slices, etc.
- Any "1D" or "2D" (*m*x1x1 or *m*x*n*x1) grid in the **grid.in** file will be treated by DCF as a "data surface grid"
  - Flow solution at all points will be interpolated from other grids
  - Grid and solution will be saved in usual files (**x.save**, **q.save**)

OVERFLOW-D Mode With Grid Motion

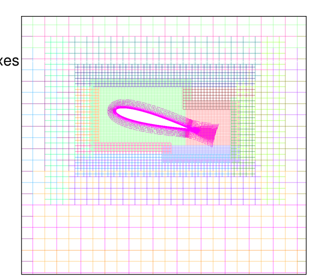
- General moving body process
- Off-body grid adaption
- GMP files **Config.xml** and **Scenario.xml**
- Non-dimensionalization of dynamics quantities
- Time step specification
- Simulating collisions
- Output information for moving body problems
- Visualizing body motion in OVERGRID
- Some references

General Moving Body Process

- Current recommendation: **run OVERFLOW in double precision**
  - Quaternion variables need to be stored as 64-bit (most, but not all complete)
- General process (input parameters in **$OMIGLB**):
  - **DYNMCS=.TRUE.** – enable body dynamics (default is FALSE)
  - **I6DOF=2** – Prescribed and/or 6-DOF motion for different components. Specified via the GMP interface (**Config.xml** and **Scenario.xml** files) (**$SIXINP** is ignored). **This is the recommended (and supported) option for moving body problems.**
    - **I6DOF=1** – 6-DOF body motion, specified via **$SIXINP** namelist input.
    - **I6DOF=0** – User-specified motion, controlled by user-supplied USER6 subroutine.
  - **NADAPT** – number of steps between adaption (regeneration) of the off-body grid system
    - **NADAPT=-n** – off-body grids adapt to geometry only
    - **NADAPT=0** – off-body grids will not be regenerated during solution process
    - **NADAPT=n** – off-body grids adapt to geometry and flow solution (see next section)

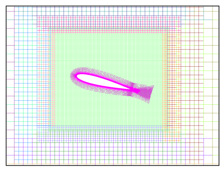
General Moving Body Process

- DCF (hole-cutting and interpolation stencil-finding) is done every step
  - Want interpolation stencils to change less than one cell per step for time accuracy
  - Estimate maximum velocity of fringe points and compare to donor grid cell size
  - This sets maximum desired physical time step
- Look at a simple example:
  - We have level-1 boxes
  - We have a near-body grid inside the boxes
  - The body is moving, the boxes are not
- What happens in OVERFLOW?
  - Body motion is computed
  - Body is moved
  - DCF is performed
  - Flow solution is advanced



Off-Body Grid Adaption

- As body moves, near-body grid gets close to the edge of the level-1 boxes
- Off-body grids must be regenerated, and the flow solution transferred (interpolated) to the new off-body grids
- **NADAPT=-n** gives the number of time steps between off-body grid adaption
  - Usually every 20-50 steps (based on time step **DTPHYS** and **MINBUF**)
    - Check this by running sample cases
  - In some cases we can avoid this by creating a larger level-1 grid (e.g., for pitching airfoil problem)



Geometry Manipulation Protocol Files: Config.xml

- Defines body (component) names and the associated grids
- Specifies any initial body transforms to assemble components into their starting positions
- Components and their transforms can be defined hierarchically
- Example:

```
<?xml version='1.0' encoding='utf-8'?>
<Configuration AngleUnit="degree">
   <Component Name="wing" Type="struc">
      <Data> Grid List=1-5 </Data>
   </Component>
   <Component Name="aileron" Parent="wing" Type="struc">
      <Data> Grid List=6,7 </Data>
      <Transform>
         <Rotate Center="0.7,0,0" Axis="0,-1,0" Angle="10"/>
      </Transform>
   </Component>
</Configuration>
```


GMP Files: **Scenario.xml**

- Prescribed motion: specify
  - Start time and duration
  - Translation and rotation rates
- Example (prescribed motion):

```
<?xml version='1.0' encoding='utf-8'?>
<Scenario Name="Forced Oscillation" AngleUnit="degree">
   <Prescribed Component="aileron" Start="0", Duration="0">
      <Rotate Center="0.7,0,0" Axis="0,-1,0"
       Speed="20.*2.*pi/100.*cos(2.*pi/100.*t+pi/2.)"/>
   </Prescribed>
</Scenario>
```


GMP Files: **Scenario.xml**

- 6-DOF motion: specify
  - Start time and duration
  - Component inertial properties
  - Applied forces
  - Motion constraints
- Example (constrained 6-DOF motion):

```
<?xml version='1.0' encoding='utf-8'?>
<Scenario Name="Constrained Motion" AngleUnit="degree">
   <Aero6dof Component="aileron" Start="0", Duration="0">
      <InertialProperties Mass="1.0" CenterOfMass="0.7,0,0"
       PrincipalMomentsOfInertia="0,2,0"/>
      <Constraint Rotate="1,0,1" Frame="body" Start="0"/>
      <Constraint Translate="1,1,1" Frame="body" Start="0"/>
   </Aero6dof>
</Scenario>
```


GMP Files: **Config.xml** and **Scenario.xml**

- For 6-DOF problems, GMP component names *must* match component names in **mixsur.inp** (force and moment calculation)
- GMP files can be created in a text editor or using OVERGRID
  - Be careful that the motion illustrated in OVERGRID is the same as that in OVERFLOW (should be OK with CGT 2.1)
- Some GMP capabilities do not work in OVERFLOW:
  - Principal axes not aligned with the original (x,y,z) axes in **grid.in**
  - Moments of inertia do not change with moving parts

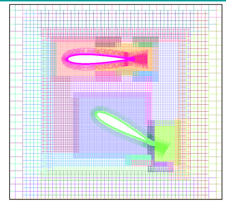
Non-Dimensionalization of Dynamics Quantities

- This is critical!
- Non-dimensionalizations in the flow solver are easy (free-stream density $\rho^*_\infty=1$, free-stream speed-of-sound $c^*_\infty=1$)
- Non-dimensionalizations for all dynamics and time-accurate information are based on V_{ref} rather than c_∞
  - V_{ref} is defined as V_{ref}=**REFMACH***c_∞
  - V_{ref} is the same as V_∞ if **REFMACH** is not explicitly specified in **$FLOINP**
    - **REFMACH** defaults to **FSMACH**
    - **REFMACH** may be different from **FSMACH**, for example for hover problems (**FSMACH=0**)
  - This includes **DTPHYS**; all quantities in GMP files (or **$SIXINP**); and output forces and moments, velocities and angular rates
  - Some quantities can be very large (or small)

Non-Dimensionalization of Dynamics Quantities

- Non-dimensionalizations of dynamic quanities are thus based on
  - Length: L=1 grid unit
  - Time: L/V_{ref}
  - Mass: $\rho_\infty L^3$
- Indicating non-dimensional quantities with a *:
  - Length: len* = len / L
  - Mass: m* = $m / (\rho_\infty L^3)$
  - Velocity: V* = V / V_{ref}
  - Time: t* = $t (V_{ref}/L)$
  - Acceleration: a* = $a (L/V_{ref}^2)$
  - Force: F* = $F / (\rho_\infty V_{ref}^2 L^2)$
  - Moment of inertia: I* = $I / (\rho_\infty L^5)$
  - Angular velocity: ω* = $\omega (L/V_{ref})$
  - Moment: M* = $M / (\rho_\infty V_{ref}^2 L^3)$

Non-Dimensionalization Example: Airfoil Drop



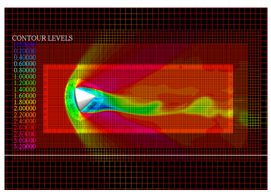
- Assume standard sea-level conditions:
  - ρ_∞ = 0.002378 slug/ft^3
  - c_∞ = 1117 ft/sec
  - Gravity = 32.2 ft/sec^2
- Pick airfoil properties:
  - chord = 1 ft
  - weight = 30 lb (heavy!)
- Flow conditions:
  - Mach = 0.2
  - Re/chord = 1 million
- From these we have:

dimensional		non-dimensional		
L	= 1 ft	L*	= 1	(grid is in chords)
Vref	= 223.4 ft/sec	Vref*	= 1	
g	= 32.2 ft/sec^2	g*	= 645x10^{-6}	
Wt	= 30 lb	Wt*	= 0.2528	
mass	= 0.9317 slug	mass*	= 392	
Iyy	= 0.05054 slug-ft^2	Iyy*	= 21.25	

- And pick (so that 400 steps is 0.1 sec):

Δt	= 0.00025 sec	Δt*	= 0.05585

Non-Dimensionalization Example: Apollo Ballistic Range Model



- Ballistic range model properties:
  - diameter = 63 mm
  - mass = 575.9 g
  - (Ixx, Iyy, Izz) = (0.1833, 0.1761, 0.1761)x10^6 g-mm^2
- Assume standard sea-level conditions:
  - ρ_∞ = 1.226 g/mm^3
  - c_∞ = 0.3405x10^6 mm/sec
  - Gravity = 9807 mm/sec^2
  - μ_∞ = 1.781 g/mm-sec
- Flow conditions:
  - Mach = 2.5
  - Re/mm = 58,610/mm
- From these we have:

dimensional		non-dimensional		
L	= 1 mm	L*	= 1	(grid is in mm)
Vref	= 0.8512x10^6 mm/sec	Vref*	= 1	
g	= 9807 mm/sec^2	g*	= 13.53x10^{-9}	
mass	= 575.9 g	mass*	= 469.7x10^6	
Ixx	= 0.1833x10^6 g-mm^2	Ixx*	= 149.5x10^9	
Iyy = Izz	= 0. 1761x10^6 g-mm^2	Iyy* = Izz*	= 143.6x10^9	

GMP Example: Capsule Forced Oscillation

- **Config.xml** file is common to forced oscillation, free oscillation, and ballistic range (6-DOF) simulation
  - Grids 1-3 are the capsule, grid 4 is the wake box
  - 4 deg initial angle
  - Capsule CG is at (21,0,0)

```
<?xml version='1.0' encoding='utf-8'?>
<Configuration AngleUnit="degree">
   <Component Name="Capsule" Type="struc">
      <Data> Grid List=1-3 </Data>
      <Transform>
         <Rotate Center="21.0,0,0" Axis="0,1,0" Angle="4.0"/>
      </Transform>
   </Component>
   <Component Name="Box" Type="struc">
      <Data> Grid List=4 </Data>
      <Transform>
         <Rotate Center="21.0,0,0" Axis="0,1,0" Angle="0.0"/>
      </Transform>
   </Component>
</Configuration>
```


GMP Example: Capsule Forced Oscillation

- **Scenario.xml** file for forced oscillation
  - Time period for 1 oscillation is 20100 (non-dimensionalized)

```
<?xml version='1.0' encoding='utf-8'?>
<Scenario Name="Forced Oscillation" AngleUnit="degree">
   <Prescribed Component="Capsule" Start="0" >
      <Rotate Center="21.0,0,0" Axis="0,1,0"
       Speed="4.*2.*pi/20100.*cos(2.*pi/20100.*t+pi/2.)"
       Frame="parent" />
   </Prescribed>
</Scenario>
```


GMP Example: Capsule Free Oscillation

- **Scenario.xml** file for free oscillation
  - Capsule inertial properties
  - Constraints: only allow rotation about y

```
<?xml version='1.0' encoding='utf-8'?>
<Scenario Name="Free Oscillation" Gravity="0,0,-13.53e-9"
 AngleUnit="degree">
   <Aero6dof Component="Capsule" Start="0">
      <InertialProperties Mass="469.7e6" CenterOfMass="21.0,0,0"
       PrincipalMomentsOfInertia="149.5e9, 143.6e9, 143.6e9"
       <PrincipalAxesOrientation Axis="1,0,0" Angle="0"/>
      </InertialProperties>
      <Constraint Start="0" Translate="1,1,1" Rotate="1,0,1"/>
   </Aero6dof>
</Scenario>
```

GMP Example: Capsule Ballistic Range Shot

- **Scenario.xml** file for ballistic range simulation
  - Capsule inertial properties
  - Box "flies" with capsule, but no rotations allowed
  - In **mixsur.inp**, "Box" component is defined to be the same as "Capsule"

```
<?xml version='1.0' encoding='utf-8'?>
<Scenario Name="Ballistic Range" Gravity="0,0,-13.53e-9"
 AngleUnit="degree">
   <Aero6dof Component="Capsule" Start="0">
      <InertialProperties Mass="469.7e6" CenterOfMass="21.0,0,0"
       PrincipalMomentsOfInertia="149.5e9, 143.6e9, 143.6e9">
       <PrincipalAxesOrientation Axis="1,0,0" Angle="0"/>
      </InertialProperties>
</Aero6dof>
<Aero6dof Component="Box" Start="0">
      <InertialProperties Mass="469.7e6" CenterOfMass="21.0,0,0"
       PrincipalMomentsOfInertia="149.5e9, 143.6e9, 143.6e9">
       <PrincipalAxesOrientation Axis="1,0,0" Angle="0"/>
      </InertialProperties>
      <Constraint Rotate="1,1,1" Start="0" Frame="body" />
   </Aero6dof>
</Scenario>
```


Time Step Specification for Time-Accurate Simulations

- Dual time-stepping (input parameters in **$GLOBAL**):
  - **DTPHYS** – physical time-step (non-dimensionalized by V_{ref})
  - **TPHYS** – to reset simulation time, e.g., **TPHYS=0** when starting dynamics
  - **FSONWT** – 1^{st}- or 2^{nd}-order time advance (always use default 2^{nd}-order)
  - **NITNWT** – (maximum) number of Newton/dual subiterations
    - 3 for simplest problems ("rough" time-accuracy)
    - 10-20 for general problems
    - 40+ for difficult problems
    - Affected by overset grid boundaries (explicit boundaries slow information transfer)
    - Affected by boundary conditions (like C-grid wakes)
  - **ORDNWT** – order of convergence for early cutoff of subiterations
    - >0 – use L_∞-norm(RHS)
    - <0 – use L_∞-norm(ΔQ)
    - =0 – do not limit subiterations

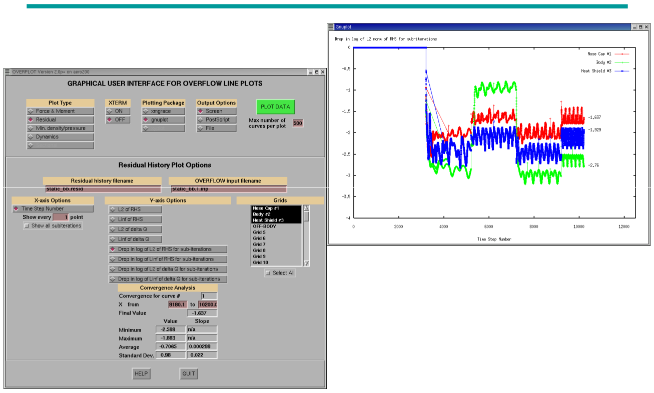
Newton/Dual Subiteration

- Subiteration is used for several functions:
  - Drives out left-hand side factorization error
  - Converges explicit boundary conditions
  - Converges unsteady simulation to the next physical time step
- Choose physical time step **DTPHYS**
  - For dual time-stepping, also choose local time-stepping (**ITIME, DT, CFLMIN, CFLMAX**) and multigrid
  - For Newton subiteration, local time-step is the same as **DTPHYS** (set **ITIME=0**)
  - Goal is to converge subiterations at least 2 orders of magnitude (ad hoc rule)
- Subiteration convergence can be improved by increasing **NITNWT** and/or decreasing **DTPHYS**
  - Reducing **DTPHYS** also improves physical time-accuracy
- Generally, choose **DTPHYS**:
  - To resolve physical scales (<u>at least</u> 100-200 steps per cycle)
  - To have sufficiently resolved body motion for accurate DCF
  - To have adequate drop in subiteration residual

Converging Newton/Dual Subiterations

- In residual history files (**resid.out**, etc.) there is one entry (for each grid, or off-body grid level) per subiteration
- First subiteration right-hand side (RHS) residual represents the unsteady forcing function
  - If this is decreasing (converging), the flow is becoming more steady
- The drop in RHS residual from first to last subiteration represents the numerical accuracy of computing the unsteady flow
  - This should be at least 2 orders-of-magnitude (unless the flow is steady)
  - Try using **ORDNWT=2** to do this
- If selected grids are not converging as well as others, try setting **ITER=2** for those grids
- A 2-order-of-magnitude drop indicates that the time advance is *numerically* converged; it does not guarantee that the physical time-step is small enough to resolve physical processes
- Use OVERPLOT from CGT to plot **resid.out** –type files, as well as subiteration convergence

Converging Newton/Dual Subiterations

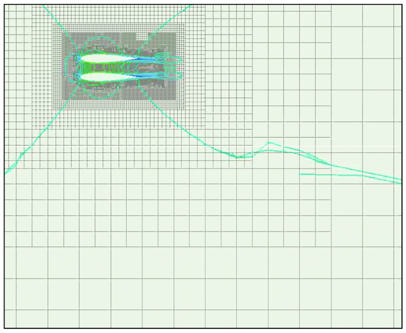


Simulating Collisions

- Contact between bodies is detected by using X-ray hole-cutting applied to surface grids of other bodies
- Contact detection is enabled (per body) by adding grid "0" to **IGXLIST** in X-ray cutter(s)
- Example: airfoil drop
  - $XRINFO IDXRAY=1, IGXLIST=-1, XDELTA=0.04, $END
  - $XRINFO IDXRAY=2, IGXLIST=-1, XDELTA=0.04, $END
  - $XRINFO IDXRAY=1, IGXLIST=2,0, XDELTA=0.0, $END
  - $XRINFO IDXRAY=2, IGXLIST=1,0, XDELTA=0.0, $END
- Accurate geometric representation of collisions may require much finer X-rays than hole-cutting
  - To keep DCF process from becoming very slow, can make collision X-rays separate from DCF X-rays
- **R_COEF** in **$OMIGLB** sets (global) coefficient of restitution
- Time of contact is accurate only to within **DTPHYS**

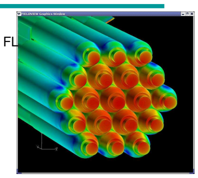
Collision Example: Airfoil Drop

- Two collisions
- Off-body grids follow airfoil
- Airfoil remains inside level-1 grid (**NADAPT** is OK)

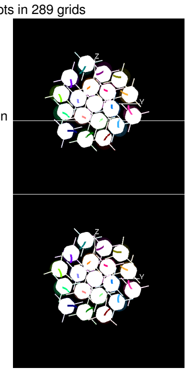


Collision Example: Dart Dispense

- Navy dart dispense problem
  - Problem being worked by Michael Neaves, NSWC, Panama City, FL
  - Current simulation models dispersal of 19-pack of darts
  - 25 Mpts in 289 grids

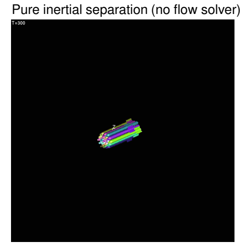


Without collision dynamics



With collision dynamics

Pure inertial separation (no flow solver)



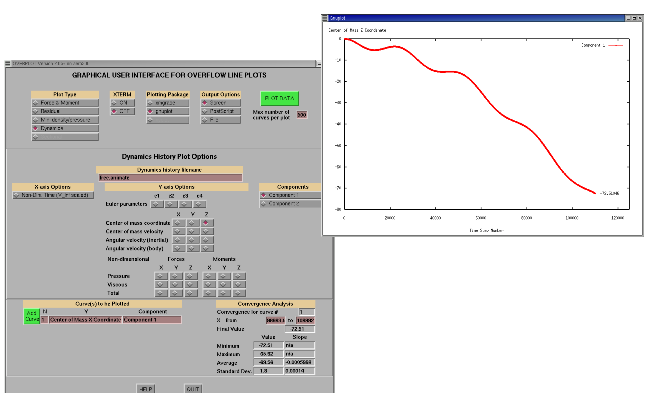
Output Information for Moving Body Simulations

- Input parameters in **$GLOBAL**:
  - **NSAVE** – grid system, flow solution, and 6-DOF restart information is saved every **NSAVE** steps, as **x.step#, q.step#, sixdof.step#**
  - **NFOMO** – force and moment coefficients are written to fomoco.out every **NFOMO** steps (automatically set to 1 for 6-DOF simulations)
- Namelist **$SPLITM**: write subsets of grid and solution every n steps (similar to CGT utilities SPLITMX, SPLITMQ)
  - **XFILE,QFILE,QAVGFILE** – specify base names for grid, solution, and/or Q-average data (if blank, don't write); step# will be appended to base name
  - **NSTART,NSTOP** – start/stop step numbers for writing output files (use -1 for "last")
  - **IPRECIS** – output file precision (0—default,1—single, 2—double)
  - **IG(subset#)** – subset grid number; use **IG()=-1** for cut of all off-body grids
  - **JS,JE,JI,KS,KE,KI,LS,LE,LI(subset#)** – subset ranges and increments
  - **CUT(subset#),VALUE(subset#)** – off-body grid cut type ("x", "y", or "z") and corresponding x, y, or z value
  - Can have multiple **$SPLITM** namelists for multiple files

Output Information for Moving Body Simulations

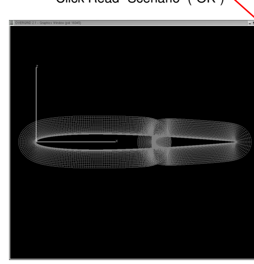
- History files:
  - **fomoco.out** – force and moment coefficients per component, per step (same as for static problems, except moment reference center moves with body)
  - **animate.out** – body ID, physical time, body position and orientation (quaternion notation), velocity and rotation rates, aero forces and moments (not coefficients)
  - **contact.out** – lists step #, body IDs, contact point and normal vector, reaction impulse, and linear and angular velocity changes (this is more for debugging collisions)
  - Note that OVERRUN script concatenates these files into **basename.{fomoco,animate,contact}**
- Use OVERPLOT from CGT to plot **fomoco.out, animate.out** –type files

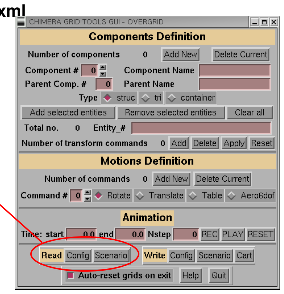
Sample Trajectory Plot in OVERPLOT

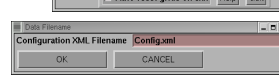


Visualizing Body Motion in OVERGRID

- Prescribed motion can be visualized in OVERGRID by reading in (surface grids or) **grid.in, Config.xml** and **Scenario.xml**
  - Start OVERGRID with surface grids or grid.in
  - Click "COMPONENTS"
  - On COMPONENTS menu,
    - Click Read "Config" ("OK")
    - Click Read "Scenario" ("OK")

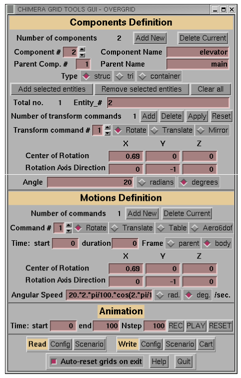


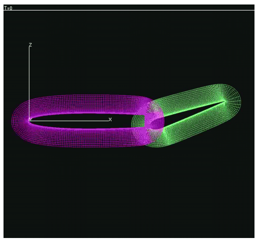




Visualizing Body Motion in OVERGRID

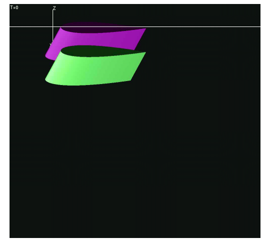
- Menu shows information on each component
  - Component names and hierarchy
  - Initial transforms from **Config.xml**
  - Prescribed motions from **Scenario.xml**
- Enter animation information
  - Start/end time and number of steps
  - Click "PLAY"

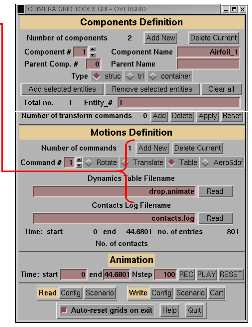




Visualizing Body Motion in OVERGRID

- For visualizing 6-DOF motion (after the OVERFLOW simulation is complete) read in **basename.animate**:
  - Click "Add New" motion command
  - Click "Table"
  - Type in animate filename and click "Read"
  - Click "PLAY"



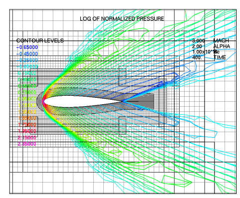


Some References

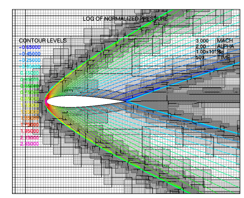
- GMP interface
  - S.M. Murman, W.M. Chan, M.J. Aftosmis, and R.L. Meakin, "An Interface for Specifying Rigid-Body Motions for CFD Applications," AIAA 2003-1237, Jan. 2003
- Solution adaption
  - R.L. Meakin, "An Efficient Means of Adaptive Refinement Within Systems of Overset Grids," AIAA 95-1722, June 1995
  - R.L. Meakin, "On Adaptive Refinement and Overset Structured Grids," AIAA 97-1858, June 1997
- Hole cutting using X-rays
  - R.L. Meakin, "Object X-Rays for Cutting Holes in Composite Overset Structured Meshes," AIAA 2001-2537, June 2001
- Off-body grid generation
  - R.L. Meakin, "Automatic Off-Body Grid Generation for Domains of Arbitrary Size," AIAA 2001-2536, June 2001
- Collision dynamics
  - R.L. Meakin, "Multiple-Body Proximate-Flight Simulation Methods," AIAA 2005-4621, June 2005

Solution Adaption for Off-Body Grids

- Allows off-body refinement grids that are finer than level-1
- Refinement levels are labelled -1, -2, etc., and have grid spacing of **DS**/2, **DS**/4, etc.
- Can be used with or without grid motion, for steady-state or time-accurate simulations



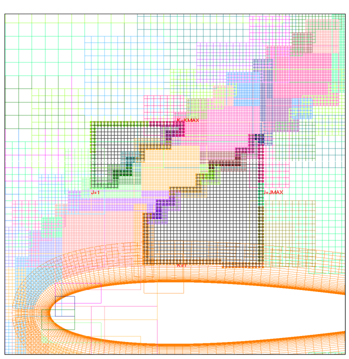
400 steps, no adaption



Additional 100 steps, adapting every 10 steps

Solution Adaption for Off-Body Grids

- Level-1 and finer grids are cut by X-rays
- Refinement grids blank out regions in coarser level grids



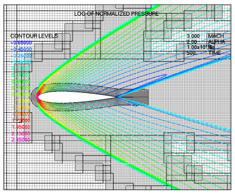
Input Parameters

- Basic control parameters (in **$OMIGLB**)
  - **NADAPT=n** – adapt solution every n steps
    - 0—do not adapt
    - >0—adapt to geometry and sensor function
    - <0—adapt to geometry only
  - **NREFINE=m** – allow up to m levels of refinement
  - **ETYPE** – sensor function for adaption
    - 0—undivided 2nd-difference of Q variables (squared)
    - 1—vorticity magnitude
    - 2—undivided vorticity magnitude
  - **EREFINE**/**ECOARSEN** – refine above/coarsen below these function values
  - **SIGERR** – shortcut method to set **EREFINE** and **ECOARSEN**
    - **EREFINE**=$(1/8)^{\mathbf{SIGERR}}$, **ECOARSEN**=$(1/8)^{\mathbf{SIGERR}+2}$

Airfoil Example: Effect of **NREFINE**

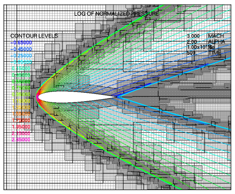
$OMIGLB NADAPT=10, NREFINE=0,
ETYPE=0, SIGERR= 5, $END

- No grids finer than level-1
- Results in 84 grids and 185K points



$OMIGLB NADAPT=10, NREFINE=2,
ETYPE=0, SIGERR= 5, $END

- Two levels of refinement
- Results in 342 grids and 754K points

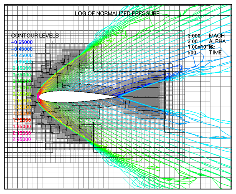


Input Parameters

- Parameters to restrict refinement regions (in **$BRKINP**)
  - **(XREFMIN,XREFMAX, YREFMIN,YREFMAX, ZREFMIN,ZREFMAX)** – min/max of region to limit grid refinement
  - **REFLVL=n** – limit grid refinement level to ≥n for this region
  - **REFINOUT** – grid refinement level is limited “INSIDE” or “OUTSIDE” the specified region
  - **IBDYREF** – limit region min/max box is tied to this Body ID for motion transformations (0 for no motion)

- Example: no adaption outside specified region

$OMIGLB NREFINE=2, ..., $END
$BRKINP
XREFMIN=-0.5, XREFMAX=1.5,
ZREFMIN=-0.5, ZREFMAX=0.5,
REFLVL=2, REFINOUT='OUTSIDE',
$END

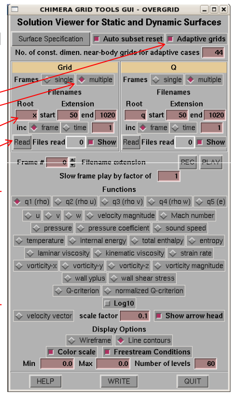


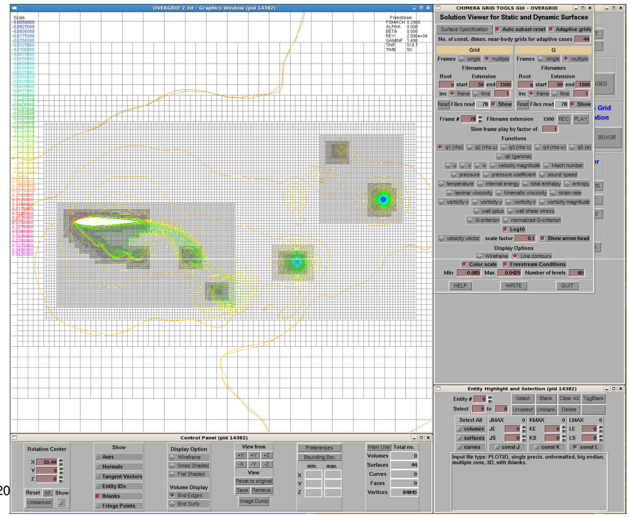
Notes and Comments

- Be careful of too many points!
- For time-accurate simulations, adjust NADAPT to make sure that adapted regions keep up with flow features and geometry
- Because the adaption can generate a large number of small grids, time-accurate simulations may need (more) subiterations to ensure good communication across grid boundaries

Visualizing Moving and/or Adapting Grids and Solutions using OVERGRID

- Moving or adapting (surface or 2D) grids can be visualized in OVERGRID by using the SOLUTION button (under “Viewers and Special Modules”)
  - Start OVERGRID with x.save (or some grid)
  - Click “SOLUTION”
  - On SOLUTION menu,
    - Click “Adaptive grids”
    - Click “multiple”
    - Adjust Root name, start, end
    - Click “Read”
    - Same for Q
    - Select Function
    - Click “PLAY” or step through frames





Compiling and Running OVERFLOW

- Unpacking and compiling
- Execution scripts
- Parallel processing and MPI load-balancing
- Hints and warnings
- Utility codes
- Test cases included with OVERFLOW

Unpacking and Compiling

- Code comes as a gzipped tar file:

```
tar zxvf over2.2x.tar.gz
```

  - Documentation is in **over2.2x/doc**
  - Test cases are in **over2.2x/test**
  - Tools are in **over2.2x/tools/***
- Flow solver makefiles **Makefile, Makefilempi**
  - Makefiles for tools, other utilities too
  - **Make.sys** contains compiler options for a large variety of machines
    - These are current suggested options only
    - User's responsibility to check and update
    - Set for "big-endian" file I/O
  - Sample **makeall** script to compile and install flow solver and tools: single and double precision, MPI and non-MPI (e.g., `makeall intel`)
  - For MPI compilation and running, **.cshrc** (or equivalent) must define environment variable MPI_ROOT, for example

```
setenv MPI_ROOT /usr/local/mpich
```


Execution Scripts

- Execution scripts **overrun** and **overrunmpi**
  - Moves ***.save** files to ***.restart** before starting
  - Highlights warnings and errors
  - Creates a log file with the time/date, machine name, executable name, and namelist input file name
  - Concatenates output history files upon completion
  - Expected namelist input file of the form **basename.n.inp**

```
overrun basename n
overrunmpi –np <ncpus> –machinefile <hostfile> basename n
```

  - I strongly recommend using these scripts, unless OVERFLOW is built into another process

Parallel Processing Options

- MPI (Message Passing Interface)
  - Useful for PC clusters (how many depends on network)
  - Good efficiency for shared memory machines with special MPI library (SGI Altix)
  - Parallelized on groups of grids (more on load-balancing later)
  - Two general flavors, MPICH and LAM, available on the web
- OpenMP
  - Useful for shared memory machines (8 CPUs or less)
  - Useful for multi-core machines (but memory bandwidth may limit performance)
  - Parallelized on grid planes (doesn't work for 2D or axisymmetric problems)
- Hybrid parallelization (mix of MPI and OpenMP)
  - Depends on system; MPI library must allow multiple threads

MPI Load-Balancing

- Number of groups == number of processes in the MPI run
- Default load-balancing scheme:
  - Based on equal distribution of grid points between processes (target group size)
  - Grids are split in half (with overlap added) until each grid is less than half the target group size
  - Grids are distributed, from largest to smallest, to current smallest group
  - This scheme works quite well for grid systems with large numbers of grids, and reasonably well for smaller systems
  - Some pathological cases:
    - 1 grid, 2 processes (grid is split into 4 instead of 2)
    - 1 grid, 3 processes (grid is split into 8, load-balance is 3/8,3/8,2/8)
  - Note that grid splitting introduces additional explicit boundaries, which affects convergence behavior

MPI Load-Balancing

- Controlling load-balancing (input parameters in **$GROUPS** and **$GLOBAL**):
  - **Use of the following inputs is rarely needed**
  - **USEFLE=.TRUE.** – use previous timing information in **grdwghts.restart** for distributing grids to groups (FALSE – use default load-balancing scheme)
    - Same as **GRDWTS** in **$GLOBAL**
  - **WGHTNB** – weighting factor for near-body grids vs. off-body grids in default load-balancing scheme (for example if viscous terms are turned off in off-body grids)
  - **MAXNB** – control splitting of near-body grids
    - **MAXNB=0** – use automatic splitting algorithm
    - **MAXNB>0** – specified (weighted) size limit
    - **MAXNB<0** – do not split grids
    - Same as **MAX_GRID_SIZE** in **$GLOBAL**
  - **MAXGRD** – control splitting of off-body grids (same options as **MAXNB**)
  - **IGSIZE** – maximum group size during grid adaption (default is 10Mpts)
  - Example: pathological case 1 (single grid (1 million points), 2 processes)
    - $GROUPS MAXNB=600000, $END
    - Grid will be split once, with both halves smaller than 600,000 pts
    - Each process will get one piece

Load-Balancing Diagnostics

- What grid splitting was done for load-balancing?

```
Target (weighted) near-body grid size from grouping:           12862
Checking near-body grids...
Original number of near-body grids:     2
  Splitting grid     1 at K =    20
  Splitting grid     1 at K =    11
  Splitting grid     2 at K =    20
  Splitting grid     2 at K =    11
  Splitting grid     3 at J =   121
  Splitting grid     5 at J =   121
Final    number of near-body grids:     8

Target (weighted)  off-body grid size from grouping:           14752
Checking  off-body grids...
Original number of  off-body grids:    30
  Splitting grid     9 at J =    59
Final    number of  off-body grids:    31
```

- Detailed list of split grids is also given in the output

Load-Balancing Diagnostics

- What is the resulting grouping of grids?

```
Load balance will be based on grid size.

    Summary of work distribution for     4 groups:

    Group     Kpts      %load   Grid list
       1        30       100     4     8    11    17    14    22    21    33    31    34
                                39
       2        29        99     6     7    12    19    13    18    20    32    26    37
       3        30       100     1     9     3    23    24    28    30    35    38
       4        30       100     2    10     5    15    25    29    27    36    16

    Predicted parallel efficiency is   100%,

      based on a maximum of    30K grid points per group
      compared to an average of    30K points (weighted)

    Estimated parallel speedup is     4.0
```


Load-Balancing Diagnostics

- What is the actual load-balance?

```
GROUP TIMING SUMMARY   (Time each group spent in OVERFL)
 (*) STEP loop, (/) Chimera BC, (a) Adapt, (D) DCFCRT, (s) Grid & Q save

            0           25          50          75          100
            |-----------|------------|------------|-----------|
Group:   1  |***********************///aaaaaaaaaaaaaaaaaaaaDDs    97%
Group:   2  |***********************///aaaaaaaaaaaaaaaaaaaDD     98%
Group:   3  |***********************///aaaaaaaaaaaaaaaaaaaDD     97%
Group:   4  |***********************///aaaaaaaaaaaaaaaaaaaaDDs    99%

 Overall Measured Parallel Efficiency:  97.9%
```


Load-Balancing – What to Look For

- Predicted parallel efficiency is low

```
Predicted parallel efficiency is    75%
```

  - Not able to split or group grids effectively
  - Some grids may not be split because of boundary conditions (axis, C-grid wake)
  - Change the number of CPUs or manually split problem grid
- Histogram shows groups are not well balanced

```
Group:   1   |*********************************************///s   97%
Group:   2   |**********************///                           47%
Group:   3   |**********************///                           47%
Group:   4   |**********************///s                          49%
```

  - Group 1 is sharing the CPU with another process
  - Eliminate other process or use a different CPU
- Large amount of time (~50%) spent exchanging Chimera BCs

```
Group:   1   |***********************/////////////////////////s   98%
Group:   2   |**********************//////////////////////////    96%
Group:   3   |**********************//////////////////////////    97%
Group:   4   |**********************//////////////////////////s   99%
```

  - Network is too slow to permit efficient use of this many CPUs
  - Use fewer CPUs

Load-Balancing – What to Look For

- Measured parallel efficiency is less than predicted efficiency (5-10%)

```
Predicted parallel efficiency is    96%
    . . .
            0           25          50          75          100
            |-----------|------------|------------|-----------|
Group:   1  |*************************************//DDD            87%
Group:   2  |***********************************//DDD              85%
Group:   3  |**********************************//DDD               82%
Group:   4  |***********************************///DDD             84%
Group:   5  |***********************************//DDD              84%
    . . .
Group:  26  |*******************************************//DDD      99%
Group:  27  |******************************************//DDD       97%
Group:  28  |******************************************//DDD       98%
Group:  29  |*****************************************//DDD        96%
Group:  30  |*****************************************//DDD        96%
Group:  31  |*****************************************//DDD        97%
Group:  32  |*****************************************//DDD        97%

 Overall Measured Parallel Efficiency:  92.4%
```

  - Set **USEFLE=.TRUE.** to use timing from previous run for load-balancing
  - This may improve the performance SOME

Hints and Warnings

- Unexplained errors while reading **grid.in** or **q.restart** file: check that all input files are the correct precision, correct "endian", and match the executable being run
- Unexplained segmentation violation while running (Intel Linux machines?): available stack memory has been exceeded, add "limit stacksize unlimited" in **.cshrc** file
- "overflow killed" message on console: process ran out of memory, check problem size

Utilities and Test Cases

- Utility codes and more utility codes
- Test cases included with OVERFLOW

Utility Codes

- **Chimera Grid Tools** (CGT version 2.1)
  - Grid generation and manipulation utilities
  - Scripting process for grid generation and assembly
  - Force & moment integration: **mixsur, overint, USURP**
  - Post-processing utilities, OVERPLOT
  - OVERGRID user interface
  - Available from William Chan and Stuart Rogers, NASA Ames *(http://www.nas.nasa.gov/~rogers/home.html)*

Utility Codes

OVERBUG, OVERTIME utility codes replaced by ***DEBUG*** *input parameter in* ***$GLOBAL:***

- **DEBUG=1** – turbulence information quantities
  - Surface quantities: wall spacing, y^+, turbulence index
  - Field quantities: μ_t, vorticity, damping functions, k, ω, etc.
  - Different quantities per model—see OVERFLOW 2.2 manual, Section 6.1
  - Data output in "fake" q file **q.turb**
- **DEBUG=2** – time step information
  - Field quantities: Δt, J,K,L, and overall CFL#
  - Data output in "fake" q file **q.time**
- **DEBUG=3** – flow solver residual information
  - Field quantities: flow solver residuals (right-hand side before time-step scaling)
  - Data output in "fake" q file **q.resid**
- **DEBUG=4** – solution adaption information
  - Field quantities: sensor function, coarsen/refine marker array, $\log_{10}$(sensor fn)
  - Data output in "fake" q file **q.errest**

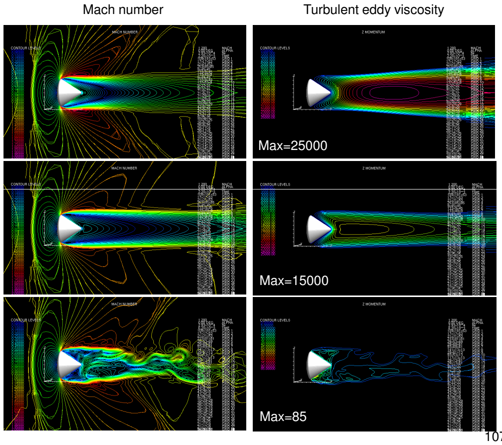
Apollo Static Aero, Mach 1.2

Turbulence model: Mach number / Turbulent eddy viscosity

Spalart-Allmaras

Baldwin-Barth

Laminar outside boundary layer



More Utility Codes

- Converting files between 32- and 64-bit
  - over2.2x/tools/run: **grid32_to_64**, etc.
- Converting files between big- and little-endian
  - Intel compiler environment flag F_UFMTENDIAN (select format per unit number)
  - OVERGRID (for grid files)
  - over2. 2x/tools/endian_convert
- Extracting or setting turbulence field quantities in a q file
  - over2. 2x/tools/turbulence: **addbb, addke, bbplot**
- Estimating viscous wall spacing
  - over2. 2x/tools/turbulence: **find_y, find_y2**
- Plotting flow quantities with variable gamma and/or multiple species
  - PLOT3D assumes a constant gamma=1.4, so thermodynamic quantities are not correct
  - over2. 2x/tools/variable_gamma/**vgplot** writes out fake q files with (pressure, temperature, Mach number, stagnation enthalpy, gamma), and (species mass fractions)
- Chemistry table for generating polynomial coefficients for variable gamma options in OVERFLOW (do not use!)
  - over2. 2x/tools/chemistry: **gaschem.f, fort.4**

Utility Codes From Bobby Nichols

- Scan a **q.save** file for min/max density, pressure, temperature, Mach number
  - over2.2x/tools/run/checkq
- Convert a **q.save** file between 1- and 2-equation turbulence models
  - over2.2x/tools/turbulence/turb_init
- Calculate surface skin friction and heat transfer coefficients
  - over2.2x/tools/unsupported/cfwf
- Create an **fvbnd** file for Fieldview
  - over2.2x/tools/unsupported/fvbnd

Test Cases Included With OVERFLOW

- Simple 2D cases (steady flow, single grid):
  - flat_plate, flat_plate_high_re
  - flat_plate_wf (tests wall function skin friction)
  - shear_layer
  - driven_cavity_2d (low-Mach preconditioning test case)
  - curved_wall_2d (tests turbulence model curvature corrections)
  - 3gas (simple multiple species convection case)
- Transonic 2D or axisymmetric cases (steady flow, single grid):
  - bump (axisymmetric bump, shock-induced separation)
  - naca, naca4412, naca_ogrid
  - et_axi, srb_axi
- Hypersonic 2D cases (steady flow, single grid):
  - cylinder, cyl_holden (2D Mach 8,16 flow)

Test Cases, Continued

- 2D multiple grids:
  - af3_96 (multi-element airfoil)
  - cascade
- 2D moving body cases:
  - airfoil_drop_2d
  - rotating_paddle_2d
  - pitching_airfoil_2d
- Propulsion cases:
  - nozzle (rocket nozzle inflow/outflow boundary conditions)
  - eggers, seiner (axisymmetric plume flows)
  - powered_nacelle (jet engine inflow/outflow boundary conditions)
  - normal_jet_2d (simple jet-in-crossflow)
- Classical time-accurate cases:
  - shock_tube
  - vortex_convection, vortex_convection_HiO, lambVortex_convection
  - stokes_1st_problem (impulsively started plate)
  - oscillating_sphere (acoustic test case)

Test Cases, Continued

- Subsonic/transonic 3D (steady, single grid):
  - m2129_s_duct (S-duct inlet)
  - rotating_disk (infinite rotating plate)
  - onera_m6 (classic transonic wing test cases)
  - inf_swept (infinite swept wing)
  - ogive_cylinder
- Subsonic/transonic 3D (steady, multiple grid):
  - wingbody (AGARD test case)
  - bizjet (assembling and running a wing/body/pylon/nacelle)
  - robin_sym (helicopter fuselage, illustrates some numerical problems)
- Off-body grid adaption cases:
  - airfoil_adapt (new)
  - normal_jet_adapt (simple jet-in-crossflow) (new)

Future Directions

- Near-term:
  - Near-body solution-adaptive gridding
  - Turbulence model with transition (Langtry-Menter SST)
- Longer-term:
  - More robust moving body process
    - adaptive time-step, subiteration control
  - Improve flow solver robustness
    - No dumping core or negative density/pressure
  - Improve cache and multi-core performance
  - Modularize for incorporation into (some form of) scripting framework