Cart3D Command Summary:

Quick reference guide to useful codes in the Cart3D distribution

Release v1.5.9
November, 2022

Contents:

$CART3D/bin/$CART3D_ARCH/adapt

Mesh adaptation module for Cart3D. ***adapt*** is used to refine an existing mesh either using feature detection or adjoint-based error estimates. It takes as input an existing mesh and checkpoint file and returns an adapted mesh and an accompanying checkpoint file with the prolonged solution. This mesh & checkpoint file can then be run by ***flowCart*** to advance the solution further. Technical description in *AIAA Paper 2003-0862, & AIAA 2008-6593 available on-line at:*
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/aiaa2002-0863.pdf
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/aiaa2008-6593.pdf

Usage: adapt [argument list]

e.g.: % adapt -adjoint -i ../adapt01/Mesh.mg.c3d -v -growth 1.4 -b 2

Options:

```
-i %s          Input  file name, must be specified. (e.g. Mesh.mg.c3d)
-o %s          Output file name, default:<adaptedMesh.R.c3d>
-in  %s        Input checkpoint file name,  default:<preAdapt.ckpt>
-out %s        Output checkpoint file name, default:<postAdapt.ckpt>
-v             Verbose mode ON
-no_reorder    Don't reorder adapted mesh, outfile='adaptedMesh.c3d'
-sfc %c        H = Peano-Hilbert ordering (default), M = use Morton
-y_is_spanwise Default assumes z_is_spanwise direction
-mesh2d        2D mesh: XY dir refinement only
-seq           Write two-level (input and adapted) mesh sequence
-prolongMap    Prolong error estimates from input to adapted mesh
                  This allows you to call adapt on its own output
-no_ckpt      Do not write output checkpoint file
-Dtagged      Dump tecplottable file <taggedHexes.dat>
-DerrorMetric       Dump text file for histograms <cellwiseErrorMetric.dat>
-Xcut %d      Num of X=const cut planes <aPlanes.dat>
-Ycut %d      Num of Y=const cut planes <aPlanes.dat>
-Zcut %d      Num of Z=const cut planes <aPlanes.dat>
-pre %s       Pre-specified adapt region filename (default = NONE)

     ----- Adaptation Criteria (select one or more) -----
-t %F         Adaptation threshold <default = 2.2>
-b %d         Number of buffering sweeps <default = 2>
-maxRef %d    Max allowed refinement level in mesh <default = none>
-stats        Output error statistics and exit
-useSplitTags Allow split-cell error estimates (fragile, not yet recommended)
-adjoint      use adjoint-based error ests. for refinement
-Rtau         Use Tau LTE estimates for refinement
-Rtrack       Use ref params computed with "flowCart -track"
-Rvmag        Use first diff of velocity for refinement
-RsqrtM       Use first diff of square root of Mach number for refinement
-Rrho         Use first diff of density  for refinement
-RrhoAtm      Use first diff of density variation from atmosphere for refinement
-Rshock       Use shock detection
-RcutCells    Tag all cut-cells
-RallCells    Create embedded mesh (refine all cells)
-growth %F    Target mesh growth factor (Adjoint, Rvmag, Rrho: min 1.01 -- max 8)
-maxErr %F    Specifiy max error value for error normalization (use w/ “-Rtrack")
-tagBelowHeight %F  Tag cells below alt. (> 0). Use only w/ Froude No. & gravity
```

o Required files:

- *Mesh.mg.c3d* => Cart3D mesh in SFC order
- *Mesh.c3d.Info* => Mesh info file for input mesh
- *preAdapt.ckpt* => Cart3D checkpoint file on input mesh

$CART3D/bin/$CART3D_ARCH/adjointCart

Parallel discrete adjoint solver for Cart3D. This code is usually invoked by ***aero.csh***. See technical description in *NASA/TP-2016-219422* and *AIAA Paper 2005-0877*, available on-line at:

https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20160013550.pdf and

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.472&rep=rep1&type=pdf

Usage: adjointCart [argument list]

e.g.: % adjointCart -fine -T -binaryIO -N 150 -mg 3 -limiter 2 -tm 1 -cfl 1.1

Options:

```
          -- Runtime Options--
-N %d               Max number of MultiGrid cycles to advance
-fine               Do -N cycles on finest mesh (incremental if restart) <FALSE>
-choice %d          1=do_mg_new, 2=jameson, 3=orig_do_mg_cycle
-mg %d              Number of multigrid levels
-gs %d              Use grid sequencing on # levels for startup (no mg)
-cfl   %f           CFL number
-rampUp %f          ramp up factor from .01*CFL, default: off
-tm  %f             cut-cell Grad mod (1stOrd=0. -> 1.0=2ndOrd) def: 1.0
-limiter %d         0=None, 1=BJ, 2=VanLeer, 3=SinLim, 4=VanAlbada, 5=MinLim
-buffLim             Buffer vol hex limiters default: <false>
-buffLimCC          Buffer cut-cell limiters, default: <false>
-flux  %d           Flux function (0=VanLeer, 1=Colella, 2=HLLC)
-no_fmg             no full multigrid (start MG at finestLevel)
-subcell            Use subcell resolution on finest mesh
-gr                 Use grads in MGrestrict (auto ON if gradEval all stages)
-nPart %d           Number of Sub Domains for partitioning
-order %d           SubDomain ordering, 0=No Reordering, 1=RCM, 2=MLD
-gradhis            Evaluate optimization gradient in forcesADJ.dat
-y_is_spanwise      Default assumes z_is_spanwise direction

          -- I/O Options --
-v                  verbose mode ON
-mem                Report memory usage (auto on with -v)
-i %s               Input  file name, default:<input.cntl>
-T                  Dump surf triangulation in Tecplot format <surfName.dat>
-clic               Dump surf triangulation in Clic format <surfName.triq>
-no_his             Turn off writing history file <historyADJ.dat>
-binaryIO           Write post-processing data in binary (plotfiles etc.) <FALSE>
-Xcut %d            Num of X=const cut planes <disjointCutPlanes.dat>
-Ycut %d            Num of Y=const cut planes <disjointCutPlanes.dat>
-Zcut %d            Num of Z=const cut planes <disjointCutPlanes.dat>
-version            Dump version info and exit
-Dmatrix            Dump i,j formatted connectivity Matrices
-Dcut               Dump tecplottable file <cutcells.dat>

-no_ckpt            Suppress checkPointing
-restart            Restart into any # of partitions <Restart.file>
```

o Required files:

- *Flow.file* => Converged flow solution check-point file
- *dObjdQ.q* => Right Hand Side (RHS) of linear system
- *dResdX.q* => Residual sensitivity (optional)

$CART3D/bin/$CART3D_ARCH/adjointErrorEst_quad

Parallel adjoint-based error estimation module for Cart3D. ***adjointErrorEst_quad*** takes the coarse mesh, flow solution and adjoint and estimates error on a uniformly refined embedded mesh to produce a cell-wise error estimate on the coarse input mesh. This code is usually invoked by ***aero.csh***. See technical description in *NASA/TP-2016-219422* and *AIAA Paper 2007-4187*, available on-line at:

https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20160013550.pdf and

https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/aiaa2007-4187.pdf

Usage: adjointErrorEst_quad [argument list]

Options:

```
          -- Runtime Options--
-etol  %f         Absolute error tolerance on functional
-tm  %f           cut-cell Grad mod (1stOrd=0. -> 1.0=2ndOrd) def: 1.0
-limiter %d       0=None, 1=BJ, 2=VanLeer, 3=SinLim, 4=VanAlbada, 5=MinLim
-buffLim          Buffer vol hex limiters default: <false>
-buffLimCC        Buffer cut-cell limiters, default: <false>
-flux  %d         Flux function (0=VanLeer, 1=Colella, 2=HLLC)
-subcell          Use subcell resolution on finest mesh
-gr               Use grads in MGrestrict (auto ON if gradEval all stages)
-nPart %d         Number of Sub Domains for partitioning
-order %d         SubDomain ordering, 0=No Reordering, 1=RCM, 2=MLD
-y_is_spanwise    Default assumes z_is_spanwise direction

          -- I/O Options --
-v                verbose mode ON
-mem              Report memory usage (auto on with -v)
-i %s             Input  file name, default:<input.cntl>
-binaryIO         Write post-processing data in binary (plotfiles etc.) <FALSE>
-Xcut %d          Num of X=const cut planes of error estimates <disjointCutPlanes.dat>
-Ycut %d          Num of Y=const cut planes of error estimates <disjointCutPlanes.dat>
-Zcut %d          Num of Z=const cut planes of error estimates <disjointCutPlanes.dat>
-version          Dump version info and exit
-histo            Dump cell-wise errors (useful for histograms)
-Dcut             Dump tecplottable file <cutcells.dat>
```

o Required files:

• *Adjoint.file*	=> converged coarse-mesh adjoint check-point file
• *Flow.file*	=> converged coarse-mesh flow check-point file
• *Mesh.mg.c3d*	=> Cart3D mesh in SFC order
• *Mesh.c3d.Info*	=> Mesh info file for input mesh

$CART3D/bin/aero.csh

Top-level driver for adjoint-based mesh adaptation with Cart3D. Copy to working directory and edit user configurable params. See samples & *README.txt* in ***$CART3D/cases/samples_adapt.***

- Use '*restart*' or '-*restart*' to run more adaptation cycles. Handles abnormal exits due to machine crashes, etc. To restart from a specific '***adapt??***' directory, delete all adapt directories that follow it. For example, to restart from ***adapt06*** in a run that finished at ***adapt08***, delete ***adapt07*** and ***adapt08*** and restart. This flag is ignored if there are no '***adapt??***' directories and an error is reported if the file *AERO_-FILE_ARCHIVE.txt* is detected.
- Use '*jumpstart*' or '-*jumpstart*' to start an adaptive from a given input mesh. To do this, put a *Mesh.c3d.Info* file and a mesh file (*Mesh.mg.c3d, Mesh.R.c3d,* or *Mesh.c3d*) in the same directory as ***aero.csh*** (along with the other usual input files) and the run will start from this mesh.
- Use '*skipfinest*' or '-*skipfinest*' to skip solving on the finest mesh. This is used when wishing to run a different solver on the finest mesh, e.g. the MPI version of flowCart. The final adapt directory contains the final mesh, all the input files and a *FLOWCART.txt* file that contains the command line. Note that the ***BEST*** link points to the previous '***adapt??***' directory.
- Use '*archive*' or '-*archive*' to generate a run archive. This option deep cleans the run directory tree, keeping only the essential output files. Once archived, restarts are not possible. This option simply calls the ***aero_archive.csh*** script.
- Set the environment variable “debug_verbose” (% setenv debug_verbose) for more verbose output.

Script returns 0 on success and 1 on error. Read tips, hints and documentation in ***$CART3D/doc/adjoint***. Use '% *touch STOP*' to force a stop after the next flow solve.

Usage: ./aero.csh [help] [restart or jumpstart] [skipfinest] [archive]

Options:

```
-help         Print help message
-restart      Run additional adaptation cycles
-jumpstart    Start from pre-existing mesh <Mesh.{mg,R}.c3d>
-skipfinest   Don’t run flow solve on finest mesh
-archive      Keep only essential output
```

o Required files:

• *Components.i.tri*	=> Triangulation of input geometry
• *input.cntl*	=> input file for flow solver
• *input.c3d*	=> input file for ***cubes***

$CART3D/bin/aero_archive.csh

Remove non-essential files after an ***aero.csh*** run. Execute this script in the run directory. This can be run directly using the “-*archive*” ***aero.csh*** option. The script cleans up the directory, archives *loadsCC.dat* from all the ***adapt??*** directories and keeps only minimum for post-run forensics.

Returns 0 on success and 1 on error.

Usage: aero_archive.csh [-q] [-a]

Options:

```
-q      “quiet mode”, no verbose output.
-a      Keep adjoint solutions in BEST.
```

$CART3D/bin/aero_getResults.pl

Script to harvest adjoint adaptation results. Handles multiple functionals. This code is usually invoked by ***aero.csh***, but may be called manually from within a top-level ***aero.csh*** directory.

Usage: aero_getResults.pl [-v] [-fos] [-update]

Options:

```
-v          verbose
-fos %f     factor of safety limiting error rise between two consecutive cycles [8]
-update     update results to include Richardson extrapolation
```

$CART3D/bin/aero_redo.csh NEW!

Run a different ***flowCart*** (or command line options) on a sequence of grids generated by ***aero.csh***. Creates a REDO directory. Can be executed after an aero.csh run or simultaneously with an ***aero.csh*** run. Copy to your working directory and edit the top few lines of the script to specify your heart's desire.

Usage: ./aero_redo.csh

$CART3D/bin/aero_upgrade.py

Upgrade any older ***aero.csh*** script to the current version. This is very handy if you have a legacy ***aero.csh*** run that you want to re-do with a newer release of Cart3D. ***aero_upgrade.py*** will preserve your run-directory aero.csh customizations and report anything out of the ordinary that it encounters. ***aero_upgrade.py*** always makes a backup of your original aero.csh so that you can refer to it if necessary. The default location for the template is ***$CART3D/bin/aero.csh*** which is included in the standard distribution.

Usage: aero_upgrade.py [-h] [-i I] [-o O] [-template TEMPLATE]

Options:

```
-h, --help           show this help message and exit
-i I                 aero.csh file to upgrade <aero.csh>
-o O                 Output aero.csh filename <aero.csh> (makes
                     backup of original file if run in-place)
-template TEMPLATE   Template aero.csh file. <$CART3D/bin/aero.csh>
```

$CART3D/bin/aerograce.csh

Mesh convergence plotter for aero.csh runs. Invoke from within a top-level ***aero.csh*** directory. This script uses templates in $CART3D/lib/ and requires xmgr/xmgrace in your path. Plots generated include functional convergence with error estimation, as well as plots of error and functional convergence. If there are multiple fun_con_ID.dat files, you can specify the ID to plot a particular output. Data from results_ID.dat will be automatically included. If you omit ID, then func_con.dat and results.dat are used as defaults.

Usage: aerograce.csh [-help] [-nw] [ID]

Options:

```
-nw         "no window" – make plots in batch mode, just create files and quit
-h, --help  Show this help message and exit
```

$CART3D/bin/$CART3D_ARCH/autoInputs

Automatically generates both required and optional input files for ***cubes***, the volume mesh generator. The ***autoInputs*** tool automatically nudges the mesh in the three Cartesian directions to avoid most common degeneracies. In general, the user has the capability to control the overall domain extent as well as the initial background mesh density. The tool also provides the capability to create meshes on half of an assumed symmetric geometry to cut flow solution time The default values for the tool are generally quite good for most cases and the use of this tool is highly recommended.

Usage: autoInputs [argument list]

e.g: *% autoInputs -r 10 -t Components.i.tri*

Options:

```
-t %s                Input geometry file name <Components.i.tri>
-r %F                Distance to farfield normalized by max geometry size <15>
-nDiv %d             Nominal # of divisions in background mesh <4>
-maxR %d             Max # of cell refinements to perform <11>
-symmX               Make mesh with x-symmetry
-symmY               Make mesh with y-symmetry
-symmZ               Make mesh with z-symmetry
-mesh2d              Make 2D mesh in x-y plane
-halfBody            Input geometry is a half-body
-i %s                File name for mesh input control file <input.c3d>
-pre %s              File name for preSpec control file <preSpec.c3d.cntl>
```

$CART3D/bin/$CART3D_ARCH/breakTris

Add component labels to existing watertight surface triangulations. Labels can be added based on logical entities (individual watertight components) or based on geometric features (“creases” in the geometry). See the *HOWTO* on labeling triangulations for examples and a complete discussion of component labeling. Labels can be applied either to separate watertight components or to entities in the GMP hierarchy of the configuration.

https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/howto/faceLabels/

Usage: breakTris [argument list]

e.g.: % breakTris -seed 14435 13839 -i engine.tri -o engine_tags.i.tri

Options:

```
-i %s          Input  tri file name, default=<surf.tri>
-o %s          Output  tri file name, default=<Components.i.tri>
-v             verbose mode ON
-ascii         make ascii-formatted output
-s             make separate output file starting
               with <component001.a.tri> for each component.
               default single file=<components.a.tri>

        --- Labeling Mode Options (automatic with seed option) --
-t %F                   threshold for dot prod of angle <default=0.8>
-seed ...               seed triangle # (1 or more) to mark inlets etc.
-startNewNumbersAt %d   output # for new components, default = max found+1
-useCompNum             separate based on compIDs, rather than logical closed
                 entities.
```

$CART3D/bin/c3d_checkInputs.py

Check Cart3D input files. Run in aero-shell directory or optimization directory. Add search terms to command line to search error strings ("all" prints all). Output details what settings have been migrated and identifies any unknown or additional additions to legacy ***aero.csh*** scripts. This script uses python 2.6 or 2.7. If that is not default on your system, then you'll want to run using "*% /path/to/python2 c3d_checkInputs.py*"

Usage: c3d_checkInputs.py [options]

Options:

```
-h, --help  show this help message and exit
-q          (Quiet) Print errors, but not warnings and notes
```

$CART3D/bin/$CART3D_ARCH/c3dvis

A post-processing tool for visualizing the solution on the volume mesh. For those cases that the Cartesian cut-plane functionality of flowCart is insufficient, such as arbitrary slices through the grid or three-dimensional streamlines, ***c3dvis*** can be used to create a Tecplot or an Ensight data file. The inputs required for ***c3dvis*** are the surface triangulation, mesh files and input control file previously used as inputs to flowCart and the checkpoint restart file created by flowCart.

Usage: c3dvis [argument list]

Options:

```
-i %s                Mesh file name: <Mesh.mg.c3d>
-ckpt %s             Checkpoint file name <Restart.file>
-v                   Verbose mode <FALSE>
-tecplot %s          Output Tecplot format file <tec.dat>
-ensight %s          Output Ensight gold format file
-ascii               Output Ensight file in ASCII format <FALSE>
```

o Required files:

• *Components.i.tri*	=> Watertight surface triangulation of the configuration
• *input.cntl*	=> flowCart input file
• *Mesh.mg.c3d*	=> Cart3D mesh in SFC order
• *Mesh.c3d.Info*	=> Mesh info file for input mesh
• *check.#####*	=> flowCart checkpoint file with solution

$CART3D/bin/$CART3D_ARCH/clic

Standalone force and moment package that computes forces, moments, and slice extractions from clic triangulations ("*-clic*" option to ***flowCart***). This package has been largely replaced through the use of internal calls within ***flowCart*** & ***mpix_flowCart*** to the clic library (which is more accurate) its still very useful for doing slice extractions on the surface (e.g. *Cp* cuts on wings and bodies). An example showing its use is in the ***$CART3D/cases/samples/oneraM6*** self-running example in the distribution

Usage: clic [argument list]

Options:

```
-i %s                control file name, def:<clic.cntl>
-outDir %s           output directory, def:<./>
-v                   verbose more ON
-mem                 Report memory usage (auto on with -v)
-parse               verbose parsing ON
-T                   Create tecplot file
```

$CART3D/bin/c3d_parallel_runner.pl & domany NEW!

A very convenient job manager for multi-core CPUs. Just setup a list of subdirectories (e.g. *tmp1, tmp2, … tmpN*) and run jobs in all of them. The script will '% cd' into the root-directory (optionally specified by -r) Once there, it searchs for subdirectories that contain the string in '-d', and also contain the string in '-s' (if specified). For example, if you want to run a simple mach alpha sweep and you have run directories called "cases/mach_0.{2,4,6,8}/alpha_{0,2,4,6,8}/", You'd simply say:

*% c3d_parallel_runner.pl -c aero.csh -r cases -d alpha -s mach_*

And it will run all 20 cases simultaneously on your machine. If you want to run these jobs 4 at a time, then just add "-j 4" to the command line.

If the script detects a file called "DONE" in any run directory, that directory gets skipped. Use '-f' option to ignore DONE.

Since ***c3d_parallel_runner.pl*** is equally helpful for heavy lifting outside of Cart3D, it's frequently soft-linked to "***domany***".

% cd $CART3D/bin; ln -s c3d_parallel_runner.pl domany; rehash; cd -

Usage: c3d_parallel_runner.pl [argument list]

```
e.g.:  % c3d_parallel_runner.pl -c pwd -d tmp  -v          # …run "pwd" command in "tmp*" subdirs
       % c3d_parallel_runner.pl -j 4 -c aero.csh -d Mach_  -v  # …run "aero.csh"  in "Mach_*" subdirs
                                                               #    with no more than 4 active jobs at a time
       % domany -c aero_archive.csh -d Mach_  -f            # … aero_archive all the "Mach_*" subdirs
```

Options:

```
 -j   %d max. number of concurrent parallel jobs
 -c   %s command to run <"sboom -e -C 1.e-6">
 -f   force, rerun command again
 -t   test, show job-list and exit

-- Options to identify the location of case directories --
 -r   %s root directory that contains all run directories
 -d   %s string in directory name where case runs, may include regex
 -s   %s optional string to help find run directories, may include regex
 -x   exclude matches on optional '-s string'

-- Options to control IO and other runtime behavior --
 -w   %d IO format extra whitespace <0>
 -z   %d sleep time to check job status in seconds <1>
 -v   be verbose
 -q   be quiet
```

$CART3D/bin/$CART3D_ARCH/closestPair

Efficiently determine the closest component to any particular target component. ***closestPair*** uses particularly efficient search techniques so that it can be called with low overhead.

Usage: closestPair [argument list]

Options:

```
-t %s       Input triangulation file name, default:<Components.i.tri>
-c %d       Component ID of target, default = 1
-v          Turn on verbose messages
-T          dump Tecplot file 'closestPair.dat'
```

$CART3D/bin/$CART3D_ARCH/comp2tri

Combine a set of individual components or combinations of components into a Cart3D un-intersected configuration triangulation with multiple components. See the HOWTO on labeling triangulations for examples and a complete discussion with many examples. ***comp2tri*** can also be used to modify component IDs or GMP tags on-the-fly during the assembly process. See useful examples online:

https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/howto/faceLabels/

Usage: comp2tri [OPTIONS] file0.tri file1.tri file2.tri ...

e.g.: % comp2tri -trix -v wing_body_tail_gmp.i.tri engine_gmp.i.tri engine_mirror.i.tri

Options:

```
-inflate            Inflate geometry to break degeneracies
-o %s               Output filename <Components.tri>
-trix               Output file will be eXtended-TRI (trix) format
-ascii              Output file will be ASCII (if not trix)
-keepComps          Preserve 'intersect' component tags
-makeGMPtags        Create GMPtags from volume indexes
-gmpTagOffset %d    Renumber GMPtags by adding '1 x offset' to tags
                      in file1.tri, '2 x offset' to tags in file2.tri, etc.
                      First file, file0.tri, gets no offset,
                      i.e. tags are unchanged <0>
-gmp2comp           Copy GMPtags to IntersectComponents
-config             Write Config.xml using component tags
-dp                 Use double precision vert-coordinates <FALSE>
-v                  Verbose more ON
```

$CART3D/bin/$CART3D_ARCH/config_space

Utility to manipulate and position individual components within a Cart3D configuration with a hierarchical configuration description (Config.xml) according to a pre-set schedule established by *ConfigSpace.xml.*

Usage: config_space [argument list]

Options:

```
-in %s            Input triangulation file name
-out %s           Output triangulation file name
-ascii            ascii output default: <FALSE>
-tri              Output Cart3D tri file instead of trix (VTU) default:<FALSE>
-param_list ...   parameter list, e.g. Flap=2, ...
-v                verbose mode ON
```

$CART3D/bin/$CART3D_ARCH/cubes

Cart3D's multi-level Cartesian mesh generator with cut-cell boundaries. The Cart3D website includes a large section on underlying concepts and usage. For specific usage info, the easiest place to get started are the examples in ***$CART3D/cases/samples/***. Background and theory with other important details can be found in (1) *AIAA Jol.* **36**(6):952-960, 1998, (2) the *CRC handbook of Mesh Generation*, or (3) *VKI Lecture Series 1998-02*. Online documentation is at: https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/meshGeneration.html

Additional background and theory references are available on-line at:
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/AIAA-97-0196.pdf
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/crcChapterDraft.ps.gz
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/aftosmis_97vkiNotes.pdf

Usage: cubes [argument list]

e.g.: % cubes -v -maxR 12 -Xcut 3 -Zcut 5 -verify -b 4

Options:

```
     -- Meshing Options--
-maxR %d            Max number of refinements (overrides input file)
-a %f               Angle threshold (deg) for geom refinement <25 deg>
-b %d               Number of layers of buffer cells <3>
-pre %s             Prespecified adapt region filename <None>
-d %f               Angle (deg) for directional refinement (aniso only) <10 deg>
-Internal           Mesh the *interior* of the geometry
-mesh2d             Make 2D mesh in X-Y, no refinement in Z
-I                  Restrict cells to isotropic division only
-remesh             Set cell density from "refMesh.{mg.c3d,c3d.Info}"
-reorder            Reorder output mesh in space-filling curve order
-sfc %c             Mesh order: H=Peano-Hilbert, M=Morton <H>
-verify             Verify that all cut cells close

     -- Advanced Options--
-sf %d              Number of additional levels at sharp edges
-weight             Area-weight triangles in divide criteria
-TPC %d             Adapt based on # of triangles-per-cutCell
-vtest              Special vtest for buffering
-lin %s             Linearize cut-cells for specified design variable
-try_exact          (if -N specified) Use Newton solve to hit -N exactly

     -- I/O Options--
-i %s               Input file name <input.c3d>
-o %s               Output file name <Mesh.c3d>
-no_file            Suppress output file
-v                  Verbose mode ON
-quiet              Don't make excessive noise
-mem                Report memory usage (auto on with -v)
-h                  Print history of # of cells with refinement (auto on with -v)
-STARS              Record intermediate refinement history
-Dunset             Dump Tecplot file <unset.dat>
-Dcut               Dump Tecplot file <cutcells.dat>
-Dflow              Dump Tecplot file <flowcells.dat>
-Dsolid             Dump Tecplot file <solidcells.dat>
-Dsplit             Dump Tecplot file <splitcells.dat>
-Daniso             Dump Tecplot file <anisocells.dat>
-Xcut %d            Number of X=const cut planes <cutPlanes.dat>
-Ycut %d            Number of Y=const cut planes <cutPlanes.dat>
-Zcut %d            Number of Z=const cut planes <cutPlanes.dat>
```

(***cubes*** *options continued...*)

```
          -- Memory Options--
      -memLim %f          Target memory usage (in MB) of final mesh
      -N %d               Target final number of Cartesian cells
      -no_est             Don't estimate memory requirements (slower, less compact)

          -- (DEPRECATED)--
      -P
      -T
      -ascii
```

o Required files:

• *Components.i.tri*	=> Watertight surface triangulation of the configuration
• i*nput.c3d*	=> cubes control file
• *preSpec.cntl.c3d*	=> (optional) pre-specified adaptation regions and XLevs

$CART3D/bin/$CART3D_ARCH/cutter

Cut-cell engine that optionally produces sensitivities of cut-cells to perturbations in the geometry. This code is typically called internally within the Cart3D package and design framework and is not generally called directly by users from the command line.

Usage: cutter [OPTIONS]

Options:

```
      -m %s          Mesh file <Mesh.c3d>
      -lin %s        Design variable name
      -verify        Check cut-mesh and linearization
      -Dsplit        Dump split-cells (tecplot)
      -v             Verbose mode ON
```

$CART3D/bin/diagnoseGeom.pl

This script runs intersect on all the pairwise combinations of components of a geometry to find any offending component combinations. In Cart3D release v1.4.9 and later the "-ascii" flag is no longer needed since ascii/binary filetypes automatically detected.

Usage: diagnoseGeom.pl [-ascii -i|base=basename -s|split -v|erbose -h|elp]

e.g.: % diagnoseGeom.pl -ascii -i <myTri.a.tri> -s -v | grep -e === -e Err

Options:

```
         -- Meshing Options--
    -i, -base=%s        Base name of the triangulation
    -s, -split          Split up all components of the triangulation
    -v, -verbose        Make a lot of noise about whats going on
    -ascii              (depricated 2016/03) Input file is ascii formatted
```

$CART3D/bin/$CART3D_ARCH/diffTriq

Take the point-wise difference between *Cp* and flow values in 2 different **.triq* files on the same triangulation. Useful for finding point-wise changes in surface loads at various unsteady time steps or multigrid cycles.

Usage: diffTriq [argument list]

e.g.: % diffTriq -t0 mySurf_at_step_1000.i.tri -t1 mySurf_at_step_1100.i.tri -nScalars 6

Options:

```
-t0 %s          Base *triq filename
-t1 %s          Other *triq filename
-q_inf %F       Freestream dynamic pressure <default = 1.0>
-o %s           Output Tecplot File Name <pdiff.dat>
-f              flip sign so its delta = t0-t1
-nScalars %d    Number of scalars in triq file <default=6>
-v              Verbose mode ON
```

$CART3D/bin/do_compAvgs.csh

Run this script from the top of the windspace directories created with ***ws_builder.csh*** to iteratively average loads outputs for all components for which loads histories exist. If the integer *averaging_window* is unspecified, the script will try to detect it from ***flowCart's*** stdout or from the setting in ***aero.csh***. Since internal averaging was added to flowCart in 2016 (v1.5 and later), this script has been largely replaced by the average loads data found in *"**loadsCC.avg.dat**"* (which is done internally by ***flowCart*** and is generally more accurate).

Usage: do_compAvgs.csh [averaging_window]

$CART3D/bin/dxf2tri.pl

An input interface that converts a basic *"*.dxf"* format triangulation file to a Cart3D triangulation format. Additional documentation available on-line at:
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/surfaceModeling.html

Usage: dxf2tri.pl infile.dxf outfile.tri

$CART3D/bin/$CART3D_ARCH/flowCart & mpix_flowCart

flowCart / mpix_flowCart is the basic inviscid flow solver in the Cart3D package. It is a highly efficient, scalable, multilevel, linearly-exact upwind solver which uses domain-decomposition to achieve excellent parallel scalability for both steady and time-dependent flows. Full documentation and quick-start information is available on-line at the Cart3D website.
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/flowSolvers.html
For a technical discussion of what's under the hood see *AIAA Paper 2000-0808*, *AIAA Paper 2005-0490, and AIAA Paper 2018-0334 all available on the pubs page of the Cart3D website:*
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/

Usage: **flowCart** [argument list]
or mpiexec -n <NumThreads> **mpix_flowCart** [argument list]
e.g.: % flowCart -T -binaryIO -mg 4 -N 200
% mpiexec -n 8 mpi_flowCart -T -binaryIO -mg 4 -N 200

Options:

```
       -- Runtime Options --
-N %d                Max # of multigrid cycles to advance (# inners if TD)
-fine                Do -N cycles on finest mesh (incremental if restart) <FALSE>
-mg %d               Number of multigrid levels
-gs %d               Number of grid sequencing levels (disables MG)
-cfl %f              CFL number
-tm %f               Cutcell gradient mod (1stOrder = 0.-->1.0 = 2nd Order) <1.0>
-limiter %d          0=None, 1=BJ, 2=VanLeer, 3=SinLim, 4=VanAlbada, 5=MinLim <0>
-buffLim             Buffer limiters to neighbors <FALSE>
-flux %d             Flux function: 0=VanLeer,1=VanLeer-Hanel,2=Colella,3=HLLC <0>
-subcell             Use subcell resolution on finest mesh
-gr                  Use grads in MGrestrict <auto ON if gradEval all stages>
-choice %d           1=do_mg_new, 2=Jameson, 3=orig_do_mg_cycle <3>
-y_is_spanwise       Y direction is spanwise <def: Z is spanwise>
-stats %d            Compute running avgs over this many cycles or timesteps
       -- Steady Run Options --
-no_fmg              Disable full multigrid startup (begin MG at finest level)
       -- Unsteady Run Options --
-nSteps %d           Max # of unsteady time steps (also signals time-dependent)
-dt %f               Non-dimensional size of physical timesteps Lref/ainf
-checkptTD %d        TD checkpoint every "checkptTD" timesteps <end of run>
-checkptGrads        Write out checkpoint files for gradients at end of run
-vizTD %d            Output viz files every "vizTD" timesteps <same as checkptTD>
-autoDT              Auto adjust physical time step for roughly constant CFLphys <FALSE>
-refWaveSpeed %F     Reference max wave speed (with -autoDT option) <UNSET>
-track %d            Track error for adapt every "track" time steps <don't track err>
-clean               'Relax' solution before time-stepping and reset nSteps  <FALSE>
       -- I/O Options --
-no_ckpt             Suppress checkpointing
-restart             Restart from checkpoint "Restart.file" (into any # of cores)
-v                   Verbose mode ON
-T                   Dump surface triangulation in Tecplot format [surfName.dat]
-clic                Dump surface triangulation in Clic format [surfName.triq]
-his                 (DEPRECATED) Write [history.dat,forces.dat] <auto on>
-no_his              Don't write history files [history.dat,forces.dat]
-binaryIO            Write Tecplot plotfiles in binary <FALSE (ASCII)>
-i %s                Input file name <input.cntl>
-Xcut %d             Number of X=const cut planes [disjointCutPlanes.dat]
-Ycut %d             Number of Y=const cut planes [disjointCutPlanes.dat]
-Zcut %d             Number of Z=const cut planes [disjointCutPlanes.dat]
-Dcut                Dump Tecplot file of cutcells [cutcells.dat]
-version             Dump version info and exit
```

$CART3D/bin/$CART3D_ARCH/inspectMesh

inspectMesh is a quick way to print the vital stats of an existing *Mesh.mg.c3d, Mesh.c3d,* or *Mesh.R.c3d* file. It tells you how many meshes are in the file (multigrid levels), if the mesh is 2D or 3D, and how many cells and faces of each type there are. Its also handy for locating a particular cell within the mesh, the "*-cell %d"* option reports the cell's coordinates and refinement levels in each direction.

Usage: inspectMesh [OPTIONS] Mesh1.mg.c3d Mesh2.R.c3d ...

Options:
```
-v        Verbose mode
-cell %d  Print info for this cell (flowCart single domain “global” index)
 ...      [list of meshes]
```

$CART3D/bin/$CART3D_ARCH/intersect

Extracts the wetted surface of a configuration. "Configurations" are collections of components output either by ***triangulate***, ***comp2tri***, or made by some other method. intersect is extensively documented in *AIAA Paper 97-0197*. The wetted surface extracted by intersect is in the form of a Cart3D wetted surface triangulation and is watertight. Component information is retained. By convention, output files are generally named **.i.tri* to indicate that they are post-intersection and do not contain any internal geometry. If "intersect" ever fails, it drops an "*Error.dat*" file which is a tecplotable file containing geometry local to the problem which caused it to fail. Intersect is quite robust, and it begins and ends with a geometry verification phase. If intersect stops during the initial geometry verification it will suggest possible problems in the input geometry (e.g. "Component N" is not closed, non-manifold, etc.) These checks are topological in nature and do not depend on floating point math. They are therefore robust. Intersect is based on boolean intersection predicates and uses adaptive precision floating point math with automatic tie-breaking to resolve degeneracies. See the ***$CAR3D/cases/samples/*** and on-line documentation at:
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/surfaceModeling.html

Additional background and theory references are available on-line at:
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/AIAA-97-0196.pdf
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/aftosmis_97vkiNotes.pdf

Usage: intersect [-i infile -o outfile -T -v -intersections -ascii -mem -cutout %d -overlap %d]

Options:
```
-i              Input  triangulation filename <Components.tri>
-o              Output triangulation filename <Components.i.tri>
-ascii          Input geometry file is ASCII
-T              Also output Tecplot file "Components.i.plt"
-fast           Also output unformatted FAST file "Components.i.fast"
-v              Verbose Mode
-mem            Report memory usage (auto on with "-v")
-cutout %d      Perform boolean subtraction (A-B) for component <%d>
-overlap %d     Perform boolean intersection of comp <%d> with others
-intersections  Write line segments of all intersections "intersect.dat"
```

$CART3D/bin/LDM.pl

A simple output formatter for the ***clic*** force and moment utility. The LDM.pl script takes in the output from a clic run and reformats the output info in a tabular format (for shipping to MS Excel etc..)

Usage: LDM.pl clic.output

e.g.: % clic -i clic.cntl -ascii -v | LDM.pl

$CART3D/bin/livePlot.pl

livePlot.pl is a "live" convergence history and force summary plotter for ***{mpix_}flowCart*** and ***adjointCart***. Invoke from within a directory where ***flowCart*** is currently running (or has already run), and it will create a plot using ***xmgrace***. If the case is currently running, the plot will update every few seconds. ***livePlot.pl*** can also be run from within a top level ***aero.csh*** directory and will automatically update as the simulation evolves through multiple adaptation cycles. This is an extremely handy utility. ***livePlot.pl*** depends on ***xmgrace/xmgr*** which needs to be in your path.

You can download xmgrace from:
http://plasma-gate.weizmann.ac.il/Grace/

Usage: livePlot [-mom -adj -lin -mf -dir <plotdirectory>]

e.g.: % livePlot.pl
% livePlot.pl -dir adapt00/FLOW
% livePlot.pl -adj -dir adapt00/AD_A_J

Options:

```
-adj      Plot adjoint convergence
-mom      Plot moment convergence
-adj      Plot adjoint convergence
-lin      Plot linCart convergence
-dir %s   Search down the specified directory for plotting candidates
```

$CART3D/bin/lsee.pl

This script generates point-wise estimates of discretization error along line sensors extracted from a sequence of meshes. See figs.6, 7 & 8 in *AIAA Paper 2017-3255* for some useful examples.
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/aiaa_2017-3255.pdf

Notes:

1. You must specify at least a medium and a fine mesh signature. If only two are specified, first-order spatial convergence is assumed.
2. Signature file format needs to be one of: two columns of x-y values, Cart3D lineSensor (reads fourth and fifth columns), or Tecplot 'ZONE' file (reads first and fourth columns). Data must be ordered.
3. Output files are prepended with ***ee_***. Check headers for help. Output files named ***ee_jeb_*** can be used to plot error bars. File name strings _l, _m and _h represent low, medium and high confidence error regions. File ***ee_cm_bes***t represents the error region as a continuous poly-line. This is a alternate way to plot the error regions --- fill in or shade the polygon.

Usage: lsee.pl [OPTIONS] -c *coarseLineSensor.dat* -m *medLineSensor.dat* -f *fineLineSensor.dat*

e.g.: % *lsee.pl -c adapt09/line.dat -m adapt10/line.dat -f adapt11/line.dat -d 2*

% *lsee.pl -c coarse.dat -m medium.dat -f fine.dat -n cells.dat -v*

Options:

```
-v      verbose
-d      problem dimension, needed for computing refinement ratios <3>
-c      file name for data from coarse mesh
-m      file name for data from medium mesh
-f      file name for data from fine mesh
-n      file name for the number of control volumes in each mesh,
        list in increasing order, one-per-line
        if not specified attempts to parse fun_con.dat
-id     optional ID when using Tecplot files with multiple zones,
        this is the zone title string, e.g. 'HL=0.85000,PHI=0.00000'
        if not specified, first zone will be parsed
-bxy    %d:%d format that specifies the columns of the input files to use
        as the x and y data sets:
          default for Tecplot files is 1:4
          default for Cart3D lineSensor files is 4:5
          otherwise default is 1:2
-sharp  account for error offset from coarse and medium mesh when
        interpolating error to finest mesh (do not use this if you prefer
        a more conservative error estimate)
-re     write realization error file
-debug  outputs coarse and medium error estimates

 -- Options to scale, translate and trim the output data --
-dx     x-axis offset (shift in x position) <0>
-sx     factor to normalize x axis, e.g. body length <1>
-sy     factor to normalize y axis <1>
-triml  trim low side of signature to define metric section, e.g. avoid
        upstream region, applied after dx and sx, default <-9999999>
-trimh  trim high side of signature to define metric section, e.g. avoid
        trailing wake, applied after dx and sx, default <9999999>
```

$CART3D/bin/$CART3D_ARCH/mesh2mesh

Transfer the solution (and optionally the residual) between two meshes with different geometries for warm-starting nearby cases. ***mesh2mesh*** uses an the space filling curves to do this transfer with linear time complexity so its very fast. This code permits solutions warm starting of solutions for perturbations of the geometry (e.g. control surface deflection) or small modifications to the geometry during design. A worked example is in ***$CART3D/cases/samples/mesh2mesh_ex/***, Other examples, along with a technical description and approach are in:

https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/aiaa2004_1232.pdf

Usage: mesh2mesh [argument list]

e.g.: % mesh2mesh -v -m1 Mesh1.mg.c3d -m2 Mesh.mg.c3d -q1 check.00100 -q2 Restart.file

Options:

```
-m1 %s              Source mesh filename
-m2 %s              Target mesh filename
-q1 %s              Source checkpoint filename
-q2 %s              Output checkpoint filename
-g1 %s              Source mesh gradient checkpoint file for time N (optional)
-g1o %s             Source mesh gradient checkpoint file for time N-1 (optional)
                    (NOTE: providing gradients at N turns on linear prolongation)
-o %s               Output map file name <m2m.map>

-mesh2d             Mesh is 2D
-sfc %c             SFC ordering: H = Peano-Hilbert (default), M = Morton
-Uinf               Initialize newly exposed cells with free stream conditions <FALSE>
-no_fill            (DEBUG) Don't fill new cells <FALSE>

-v                  Verbose mode <FALSE>

     -- Time-dependent and moving geometry options --
-TD                 Checkpoint is from unsteady simulation <FALSE>
-move_geom          DEPRECATED - Moving geometry <FALSE>
-r1 %s              DEPRECATED - Source residual filename (optional)
-r2 %s              DEPRECATED - Output residual filename (optional)
-t1 %s              DEPRECATED - Source surface geometry
-t2 %s              DEPRECATED - Target surface geometry
-timeN %F           DEPRECATED - Time at level N <0.0>
-timeN1 %F          DEPRECATED - Time at level N+1 <0.0>
```

$CART3D/bin/$CART3D_ARCH/mgPrep & mgTree

mgPrep/mgTree is the coarse mesh generator for the multi-level Cartesian meshes that are produced by ***cubes***. It takes reordered meshes (from ***reorder*** or ***cubes -reorder***) usually named *Mesh.R.c3d* and produces hierarchies of meshes (usually named ***Mesh.mg.c3d***). The ordering of the input meshes is preserved and is propagated to coarser meshes in the hierarchy. So if you pass it a mesh that you reordered with Peano-Hilbert, all the coarse meshes will be ordered by Peano-Hilbert too. ***mgPrep*** works best with 3D meshes, for 2D meshes, use ***mgTree***.

On-line documentation:
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/flowCart_mgPrep.html

Technical description and approach:
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/aiaa2004_1232.pdf

Usage: mgPrep [argument list]
or
mgTree [argument list]

Options:

```
-n %d            Number of multigrid levels to prepare (n>0) <5>
-i %s            Input  file name <Mesh.R.c3d>
-o %s            Output file name <Mesh.mg.c3d>
-sfc %c          Input mesh sfc order: H=Peano-Hilbert, M=Morton <H>
-v               Verbose mode ON
-mesh2d          Input mesh is 2D (x-y adaptation only)
-separate        Dump coarser meshes in separate files
-pmg             Write the finest mesh twice (for p-multigrid)
-no_stats        Suppress printing of coarsening statistics
-verifyInput     Verify the input mesh before coarsening
-Xcut %d         Number of X=const cut planes (mgPlanes.dat)
-Ycut %d         Number of Y=const cut planes (mgPlanes.dat)
-Zcut %d         Number of Z=const cut planes (mgPlanes.dat)
-Dcut            Dump Tecplot file (cutcells.dat)
   -Dflow           Dump Tecplot file (flowcells.dat)
-Dall            Dump Tecplot file (allcells.dat)
```

$CART3D/bin/mk_aeroTables.csh

For a specified list of components, traverse the wind-space directory structure and harvest aero performance data run this script from within a particular windspace (where ***M*A*B*R**** subdirectories are visible). You must run ***do_compAvgs.csh*** first. Any component who's force/moment is requested in the "$__-Force_Moment_Processing:" in input.cntl is allowed, default is entire.

Usage: mk_aeroTables.csh entire [gmp_comp1] ...

```
e.g.:  %  mk_aeroTables
       %  mk_aeroTables  wing
       %  mk_aeroTables  fuselage wing htail
```

$CART3D/bin/model2aero.pl

Script to convert from model to body to aero frames using *forces.dat* and *moments.dat* history files output from ***flowCart***.

Usage: model2aero.pl [OPTIONS]

Options:

```
-c %s      component force and moment history file <entire.dat>
-i %s      flowCart input file <input.cntl>
```

$CART3D/bin/$CART3D_ARCH/net2p3d

Convert a Langley Wireframe Grid Standard (LaWGS) mesh network to a multiple grid plot3D structured grid. See the sample in $CART3D/cases/samples/oneraM6/ for a good description and illustration.

Usage: net2p3d [-i infile -o outfile -v -m -C Compfile -no -ascii]

Options:

```
-i ........ Input  file name, def:<LaWGS.net>
-o ........ Output file name, def:<mgrid.unf>
-v ........ Verbose mode
-m ........ "memory verbose" report malloc/free
-C ........ Component list file name,def:<Component.list>
-no ....... Dont write out a Component list
-ascii..... Output file ascii format
```

$CART3D/bin/$CART3D_ARCH/reorder

reorder takes meshes (*Mesh.c3d*) output by ***cubes*** and re-orders them using a space-filling-curve based ordering. Think of it as a backend for ***cubes***. ***flowCart*** takes these re-ordered meshes and can partition them on-the-fly onto any number of processors. Even if you're going to run on a single CPU (unpartitioned domain) its worth reordering since reordered meshes have better locality and will execute faster on cache-based machines. ***reorder*** is the key to ***flowCart's*** domain-decomposition strategy and is required for mesh coarsening. reorder can be invoked from within cubes using the "*-reorder*" flag on the ***cubes*** command line.

On-line documentation:
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/flowCart_reorder.html

Technical description and approach:
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/aiaa2004_1232.pdf

Usage: reorder [argument list]

Options:

```
-i %s        Input  mesh file name, default:<Mesh.c3d>
-o %s        output mesh file name, default:<Mesh.R.c3d>
-m %s        Mesh info file,        default:<Mesh.c3d.Info>
-sfc %c      sfc choice, H=peano-hilbert (default), M=morton
-s           Use perfect sort of face list (default is bin sort)
```

$CART3D/bin/stl2tri.pl

An input interface that converts an ascii stereolithography *"*.stl"* format triangulation file to a Cart3D indexed triangulation format with component information. NOTE: ***admesh*** is a prerequisite and must be in your path. You can download ***admesh*** at
https:/github.com/admesh/admesh

Additional documentation available on-line at:
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/surfaceModeling.html

Usage: stl2tri.pl infile.stl outfile.a.tri

$CART3D/bin/$CART3D_ARCH/tri2stl

The inverse of ***stl2tri.pl***. Convert a Cart3D indexed triangulation to an ascii-formatted stereolithography triangulation. All component information is lost.

Usage: tri2stl [argument list]

Options:

```
-i %s     Input surface triangulation:<comp.tri
-o %s     Output STL file name <Comp.a.stl>
```

$CART3D/bin/$CART3D_ARCH/triangulate

Triangulate a structured grid file. Triangulate takes a multiple-grid plot3d format configuration and triangulates it component-by-component. Points with duplicate geometry (same point in physical space) are removed. (Comparisons for point removal use a default tolerance of 1.e-6 model units.) It is used for converting components specified from structured geometry sources into intersection-ready triangulations. Component information is retained for each triangulation. See the example in the distribution at ***$CART3D/cases/samples/oneraM6/*** Additional documentation available on-line at:
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/surfaceModeling.html

Usage: triangulate [-i infile -o outfile -T -v -r -n -C comp_file -inward -inflate]

Options:

```
-i ........ Input  file name, def:<mgrid.unf>
-o ........ Output file name, def:<Components.tri>
-T ........ Output to tecplot file "tec.dat"
-fast ..... Output to unformatted FAST file "Components.fast"
-C ........ Component list for infile, def:<Component.list>
-v ........ Verbose Mode
-r ........ Remove duplicate nodes in input file (keeps order)
-lex ...... Lex sort the triangle verts (needs -r)
-n ........ Dont perturb identical pts on diff components
-inward ... Norm vectors on input geom face INWARD. def:<outward>
-ascii .... input file is ascii fmt -(output single-precision unformatted)
-dp ....... double precision I/O (output trix)
-zero ..... set (data < ZERO) to 0.0000 - (param ZERO = 1.E-12)
-inflate .. Inflate geometry by 10*Eps to break  degeneracies
```

$CART3D/bin/$CART3D_ARCH/trimCutPlanes NEW!

Uses new polygon types in Tecplot to produce pretty cut-planes that more accurately display the geometry actually being solved by *flowCart*. The standard ***cut planes.{dat,plt}*** do not make any attempt to accurately represent the cut-cells in Cart3D. Instead, they simply show full (un-cut) quadrilaterals which often contain misleading values — especially in cells split into multiple control volumes, or in cut cells in which only a small fraction of the cell is actually in the flow. ***trimCutPlanes*** rectifies this by using a general polygonal representation of the slice through the cut-cell.

Usage: trimCutPlanes [argument list]

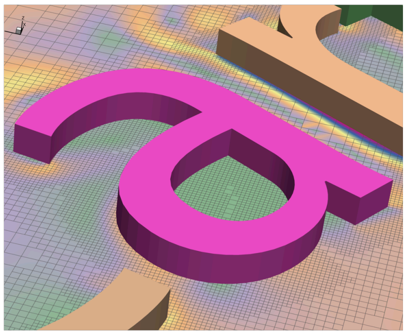
e.g.: *% trimCutPlanes -ckpt Restart.file*

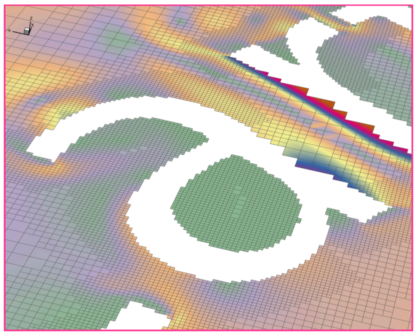
Options:

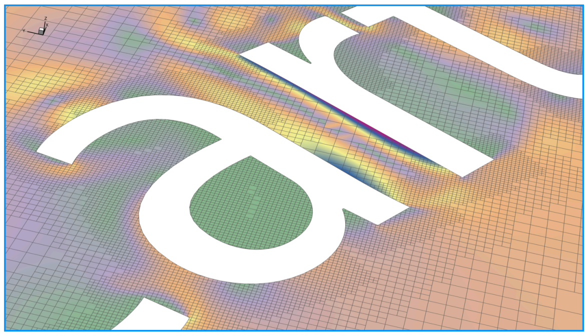
```
-ckpt %s  Checkpoint file name <Restart.file>
-v        Verbose mode <FALSE>
```

o Required files: (these should all exist in any flowCart run directory)

• Components.i.tri	==>	surface triangulation
• Mesh.c3d.Info	==>	Mesh info file from cubes
• Mesh.mg.c3d	==>	Cart3D Mesh file
• *check.#####*	==>	flowCart checkpoint file with solution
• input.cntl	==>	flowCart input file


Flow simulation around Helvetica letters


cutPlanes.{dat,plt} from ***flowCart***


trimCutPlanes.plt from ***trimCutPlanes***

$CART3D/bin/$CART3D_ARCH/trix

The Swiss army knife of triangulations: format converter, translations, and more. The default action is to convert the input files to the Cart3D extended triangulation (VTK) format. Shape sensitivities (if present) are automatically adjusted to reflect any geometry manipulations. Trix can also be used to move/rotate selected disconnected components with respect to others in the configuration.

Usage: trix [OPTIONS] [file(s) ...]

Options:

(options are listed in the order in which they are applied)

```
-select ...         Select component(s) to mirror/scale/translate/rotate,
                    e.g. -select 1 2 5, omit to select all
      Scale Geometry:
-mirror %c          Mirror the X, Y or Z coordinate <none>
-sx %F              Scale geometry in X <1.>
-sy %F              Scale geometry in Y <1.>
-sz %F              Scale geometry in Z <1.>
      Translate geometry:
-x %F               Translate geometry in x-direction <0.>
-y %F               Translate geometry in y-direction <0.>
-z %F               Translate geometry in z-direction <0.>
      Rotate geometry (order rx-ry-rz):
-cx %F              X center of rotation <0.>
-cy %F              Y center of rotation <0.>
-cz %F              Z center of rotation <0.>
-rx %F              Rotate geometry around x-axis (deg) <0.>
-ry %F              Rotate geometry around y-axis (deg) <0.>
-rz %F              Rotate geometry around z-axis (deg) <0.>
-rg %F %F %F %F     Rotate geometry around vector with tail <cx, cy, cz>
                    and head < 0., 0., 0. > (deg) <0.>
      IO options:
-v                  Be verbose <FALSE>
-dp                 Use double precision vert-coordinates <FALSE>
-o %s               Output filename prefix <Components> (in VTU format)
-T                  Output a Tecplot file for each component (.dat)
-tri                Output all files as traditional Cart3D tri-files
-noVTK              Do not write an extended triangulation (VTK) file <FALSE>
      Tag manipulations:
-comp2gmp           Overwrite or create GMPtags from component tags
-tagRegion %F %F %F %F %F %F
                    Tag rectangular region inside XMIN XMAX YMIN YMAX ZMIN ZMAX
-add2comp %d        Increment or decrement component tags by this amount <0>
-add2gmp %d         Increment or decrement GMPtags by this amount <0>
     Linearization:
-dx                 Linearize with respect to x translation <FALSE>
-dy                 Linearize with respect to y translation <FALSE>
-dz                 Linearize with respect to z translation <FALSE>
      Input file list:
 ...                file1.tri file2.triq (extensions not important)
```

$CART3D/bin/$CART3D_ARCH/viscousDrag

Simple viscous drag calculator that can be run on a Cart3D ***.triq** file output from ***flowCart***. The method uses a single-pass boundary layer solution against the inviscid solution in the ***.triq** file.

On-line documentation:
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/howto/viscousDrag/

Technical description and approach:
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/AIAA_2006-0652.pdf

Usage: viscousDrag [argument list]

Options:

```
                -- Runtime Options--
    -v                  verbose mode ON
    -flatPlate          Use flat plate boundary layer profile
    -mem                Report memory usage (auto on with -v)
    -i %s               Input  file name, default:<input.cntl>
    -clic %s            Use CLiC file named (e.g. modelName.triq)
    -scale %F           Number of feet per-unit-model length def:<1.0>
    -version            Dump version info and exit
```

$CART3D/bin/wrl2c3d.pl

An input interface that converts a VRML (Virtual Reality Modeling Language) format triangulation files into a Cart3D single component triangulation. VRML's ***.wrl** files typically don't include any component information, so each *wrl file is assumed to contain only one sold. If the input contains many topologically separate objects, they may be separated with ***breakTris***.

See the on-line documentation for info on surface modeling and file formats.
https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/surfaceModeling.html

Usage: wrl2c3d.pl infile.wrl outfile.tri

$CART3D/bin/ws_builder.csh

This automation script builds and populates a 3 dimensional "windspace" for parameter studies over a range of Mach numbers, incidence angles (alpha), and side slip angles (beta) based upon a single template directory. Windspace directories use the naming convention ***M*A*B**** and can then be traversed by ws_runner.sh, and post-processed with ***do_compAvgs.csh*** and ***mk_aeroTables.csh****.* The contents of the template directory are copied into each and the input files are modified for each individual case. ***ws_builder.csh*** does not run the cases, it simply sets up the directories. To use, make a copy of the script in your local directory and edit the top few lines of the script to set your range of Mach, alpha and beta. To run, put all required top-level ***aero.csh*** files in a directory called "***template***" within the launch directory.

```
Usage:    ./ws_builder.csh
```

$CART3D/bin/ws_runner.csh

Unsophisticated automation utility for running an windspace parameter study setup by ***ws_builder.csh***. This script simply runs cases one after the other on the local hardware. Set the desired number of threads at the top of the script. More sophisticated versions of this script are available from the Cart3D development team including versions developed for systems using the Portable Batch System (PBS). This script is also a good starting point for automation of post-processing tasks.

To run, launch this script from the top-level windspace directory with the ***M*A*B**** directories created by ***ws_builder.csh*** all visible at the current level.

```
Usage:    ws_runner.csh
```

$CART3D/bin/$CART3D_ARCH/xsensit

Computes the sensitivity of the objective function (in ***input.cntl*** or Functionals.xml) to changes in the flow state -- frequently referred to as dObj/dQ or dJ/dQ. This code is not usually invoked by hand and is usually run either by ***aero.csh*** (for error estimation) or by the Cart3D design framework. The code runs in parallel on the number of cores set by the $OMP_NUM_THREADS environment variable. Execution requires about the same time as one or two fine-grid iterations on the current mesh. Memory usage is approximately the same as the flow solver.

Usage: xsensit [argument list]

e.g.: % xsensit -dQ -limiter 1

Options:

```
          -- Runtime Options--
-tm  %f             cut-cell Grad mod (1stOrd=0. -> 1.0=2ndOrd) def: 1.0
-limiter %d         0=None, 1=BJ, 2=VanLeer, 3=SinLim, 4=VanAlbada, 5=MinLim
-buffLim             Buffer vol hex limiters default: <false>
-buffLimCC          Buffer cut-cell limiters, default: <false>
-flux  %d           Flux function (0=VanLeer, 1=Colella, 2=HLLC)
-subcell            Use subcell resolution on finest mesh
-nPart %d           Number of Sub Domains for partitioning
-order %d           SubDomain ordering, 0=No Reordering, 1=RCM, 2=MLD
-y_is_spanwise      Default assumes z_is_spanwise direction

          -- I/O Options --
-v                  verbose mode ON
-mem                Report memory usage (auto on with -v)
-i %s               Input  file name, default:<input.cntl>
-T                  Dump surf triangulation in Tecplot format <surfName.dat>
-clic               Dump surf triangulation in Clic format <surfName.triq>
-binaryIO           Write post-processing data in binary (plotfiles etc.) <FALSE>
-Xcut %d            Num of X=const cut planes <disjointCutPlanesLIN.dat>
-Ycut %d            Num of Y=const cut planes <disjointCutPlanesLIN.dat>
-Zcut %d            Num of Z=const cut planes <disjointCutPlanesLIN.dat>
-version            Dump version info and exit
-Dmatrix            Dump i,j formatted connectivity Matrices
-Dcut               Dump tecplottable file <cutcells.dat>
-no_ckpt            Suppress checkPointing

          -- Sensitivities --
-dMach              Compute dR/dMinf and dJ/dMinf
-dAlpha             Compute dR/dAlpha and dJ/dAlpha
-dBeta              Compute dR/dBeta and dJ/dBeta
-dRoll              Compute dR/dRoll and dJ/dRoll
-dRhoinf            Compute dR/dRhoinf and dJ/dRhoinf
-dShape             Compute dR/dShape and dJ/dShape
-dBackPressureBC %d Compute dR/dInletPressureRatioBC and dJ/dInletPressureRatioBC
                    for a specified component
-dVelocityBC %d     Compute dR/dInletVelocityBC and dJ/dInletVelocityBC
                    for a specified component
-dQ                 Compute dJ/dQ
-objGrad            Evaluate objective function gradient
```

o Required files:

• Flow.file	=> converged flow solution check-point file
• dResdX.q	=> R.H.S. of linear system
• dObjdQ.q	=> Objective function sensitivity (optional)