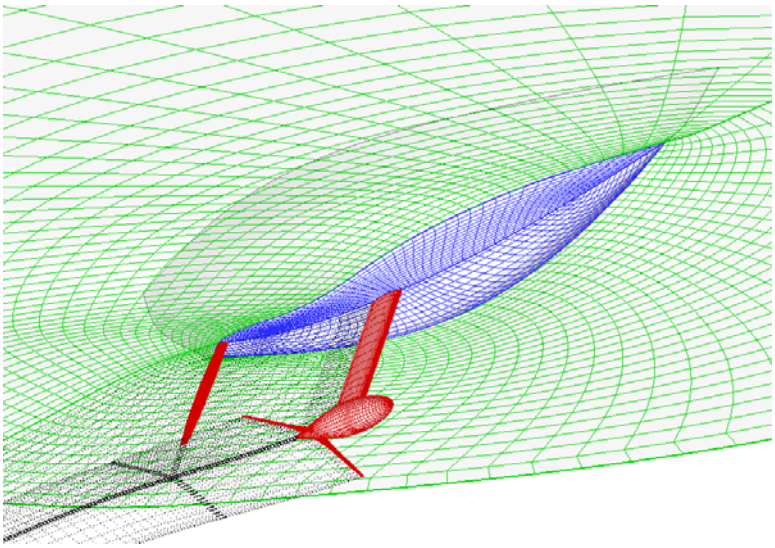
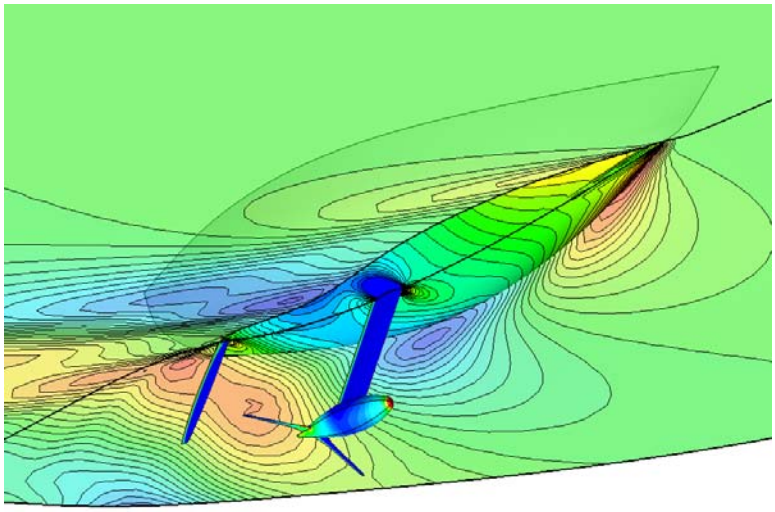
ACCPAN and SPLASH

Installation and User Guide

June 20, 2009





South Bay Simulations, Inc. • 44 Sumpwams Ave. • Babylon, NY 11702
(631) 587-3770 www.panix.com/~brosen brosen@panix.com

WARRANTY AND DISCLAIMER

South Bay Simulations, Inc. (SBS) makes no warranty as to the quality and performance of **SPLASH**. While every effort is made to provide a bug-free and workable product, it is provided with no warranty that it is free from errors or that it will always work. This is the sole and exclusive warranty offered by SBS. There are no other warranties, express or implied, including but not limited to the implied warranties of design, merchantability and fitness for a particular purpose, or arising from a course of dealing, usage, or trade practice. No agent of SBS is authorized to alter or exceed the warranty obligations of SBS as set forth herein.

LIMITATION OF LIABILITY

In no event will SBS be liable for any lost revenues or profits, goodwill, or other special, indirect, consequential or punitive damages however caused by use of **SPLASH** and regardless of theory of liability, even if SBS has been advised of the possibility of such damages. No SBS liability for damages resulting from use of **SPLASH** shall exist.

COPYRIGHT

TRADEMARKS

Tecplot: Amtec Engineering, Inc.

ACCPAN and SPLASH
Installation and User Guide

Contents

ACCPAN and SPLASH
Installation and User Guide

Overview

This guide covers the use of ACCPAN and SPLASH for numerical towing tank tests, primarily for grand prix racing yachts. A sample model test provides further guidance.

The particular versions of the software on which this guide is based are:

ACCPAN Version 5.1.0.05

SPLASH Version 5.1.0.04.

Operating System Administration

ACCPAN and SPLASH numerical tests are meant to be conducted on UNIX or Linux operating systems. A number of system administration tasks must also be completed in order to successfully complete the tests. For the purposes of this guide, a Linux operating system running on an Intel-based computer is assumed. Operation on most popular flavors of UNIX should be similar if not identical. Use on Apple MAC OS or Microsoft Windows operating systems is currently not supported.

Filesystem Layout

For simplicity, it is assumed that supplied files have been installed into subdirectories immediately below the user's home directory. Supplied tar archive files will generally result in the proper filesystem layout, if the tar extraction commands are initiated from the user's home directory.

Default User Shell Environment

Many of the tasks to be completed are controlled by command line user input or by shell script files. Unless noted otherwise herein, supplied script files are written to run in a C-shell environment, and command line inputs likewise employs C-shell syntax. Likewise, the user's default shell upon login is assumed to be C-shell (/bin/csh, or other compatible shell, such as /bin/tcsh).

Under Linux, the user is assigned a home directory, for example /home/user. This location is normally stored in the user shell environment variable $HOME. Hereafter, the user's home directory is referred to as $HOME.

Commands and shell script files often have to locate other shell script and binary executable files. This is accomplished by locating frequently used script and executable files in a common directory, usually $HOME/bin. This directory is then included in the user shell environment variable $PATH, for example by including the following commands in the user's shell initialization file $HOME/.cshrc:

```
echo $PATH | grep $HOME\/bin >& /dev/null
if ($status != 0) then
  set path = ($HOME/bin $path)
endif
```

Shells other than the C-shell may use a different file or command syntax for this. For Korn shell or Bourne shell, the file might be $HOME/.profile, $HOME/.bash_profile, or similar, and the following commands could be included:

```
PATH=$PATH:$HOME/bin
export PATH
```

A list of executable binary and shell script files available in the user's path is constructed upon entry to each new shell instance. However the list is not updated automatically thereafter. It may be user-updated, manually, by issuing the C-shell command "rehash".

Batch Job Queue System Setup

Automated and nonlinear ACCPAN and SPLASH calculations depend upon a batch job queue system for proper operation. A single script file, $HOME/bin/qsub_csh, is supplied to support use of the Linux built-in batch job queue system. Other batch queue software may be available, as part of the operating system, or from other open or commercial sources.

The supplied script file qsub_csh relies upon the Linux built-in batch job queue system "atd" which is accessed by the Linux built-in commands "at" and "batch". The atd system daemon must be activated, and the user must have permission to use the at and batch commands, for this approach to work properly. Access is normally controlled by entries in operating system files such as /etc/at.allow and /etc/at.deny. See "man atd", "man at" and "man batch" on your system for more complete information.

Commands or script files submitted to the batch job queue system using qsub_csh will have additional environment variables defined to assist batch job execution. Some important ones include $QSUB_WORKDIR, set equal to the full pathname to the directory from which the job is submitted (using the qsub_csh command), and $TMPDIR, set equal to the full pathname to a temporary/scratch directory uniquely created by qsub_csh for each separate job submittal.

Installing the Supplied Binary Executable Files

A number of files are supplied in binary executable file format only. These include, in addition to ACCPAN and SPLASH, the programs "sbsinfo" (which outputs system identification information required for software license key generation), and "sbsmesh" (for database mesh refinement, and described in the section devoted to auxiliary tools).

Installing the Software

Extraction from supplied tar archive files will install one or more of the following files on your system (the multiple instances of each software package are intended to provide a selection of executable files appropriate for most user operating systems):

For accpan:

```
$HOME/bin.lnx.pgi611.k8-32-77/accpan_5.1.0.05_lnx.pgi611.k8-32-77
$HOME/bin.lnx.pgi611.p7-77/accpan_5.1.0.05_lnx.pgi611.p7-77
$HOME/bin.lnx.pgi611.x64-77/accpan_5.1.0.05_lnx.pgi611.x64-77
```

For splash:

```
$HOME/bin.lnx.pgi611.k8-32-90/splash_5.1.0.02_lnx.pgi611.k8-32-90
$HOME/bin.lnx.pgi611.p7-90/splash_5.1.0.02_lnx.pgi611.p7-90
$HOME/bin.lnx.pgi611.x64-90/splash_5.1.0.02_lnx.pgi611.x64-90
```

For sbsinfo:

```
$HOME/bin.lnx.pgi611.k8-32-77/sbsinfo_lnx.pgi611.k8-32-77
$HOME/bin.lnx.pgi611.k8-32-90/sbsinfo_lnx.pgi611.k8-32-90
$HOME/bin.lnx.pgi611.p7-77/sbsinfo_lnx.pgi611.p7-77
$HOME/bin.lnx.pgi611.p7-90/sbsinfo_lnx.pgi611.p7-90
$HOME/bin.lnx.pgi611.x64-77/sbsinfo_lnx.pgi611.x64-77
$HOME/bin.lnx.pgi611.x64-90/sbsinfo_lnx.pgi611.x64-90
```

For sbsmesh:

```
$HOME/bin.lnx.pgi611.k8-32-90/sbsmesh_lnx.pgi611.k8-32-90
$HOME/bin.lnx.pgi611.p7-90/sbsmesh_lnx.pgi611.p7-90
$HOME/bin.lnx.pgi611.x64-90/sbsmesh_lnx.pgi611.x64-90
```

Final links which, when invoked, point to a specific version of the software:

```
$HOME/bin/accpan
$HOME/bin/splash_cw
$HOME/bin/sbsinfo
$HOME/bin/sbsmesh
```

With respect to the files listed above, the designation "lnx.pgi611" indicates they were compiled using the Portland Group compiler version 6.1.1 for execution on Linux

systems. The designation "k8-32" indicates compilation for 32-bit execution on an AMD processor, whereas "p7" is for 32-bit execution on an Intel processor (Pentium IV or later), and "x64" provides for execution on both AMD and Intel based 64-bit systems. The designations "77" and "90" refer to Fortran-77 and Fortran-90 compilations, respectively.

The files in $HOME/bin can be copies of the appropriate files from the other listed directories, or they can be links to them. For example, to create links to the 64-bit versions of the files, issue the following commands:

```
cd $HOME/bin

/bin/rm accpan
ln -s $HOME/bin.lnx.pgi611.x64-77/accpan_5.1.0.05_lnx.pgi611.x64-77 accpan

/bin/rm splash_cw
ln -s $HOME/bin.lnx.pgi611.x64-90/splash_5.1.0.04_lnx.pgi611.x64-90 splash_cw

/bin/rm sbsinfo
ln –s $HOME/bin.lnx.pgi611.x64-90/sbsinfo_lnx.pgi611.x64-90 sbsinfo

/bin/rm sbsmesh
ln –s $HOME/bin.lnx.pgi611.x64-90/sbsmesh_lnx.pgi611.x64-90 sbsmesh
```

Software License Keys

After the sbsinfo program has been installed, it may be executed at the command line by entering the command "sbsinfo". System identification information will be output which can be conveyed to South Bay Simulations for software license key generation.

The generated license keys must be stored in a key file for access by the other binary file programs (accpan, splash and sbsmesh). This license key file must contain valid license keys in order for these other programs to execute correctly. The default license key file is "$HOME/sbs/license.sbs". An alternate file or location may be specified by assigning it (in $HOME/.cshrc) to a shell environment variable $SBS_FILE, for example:

```
setenv SBS_FILE /MyLocation/MyFile
```

Installing the Supplied Auxiliary Tools

A wide range of auxiliary tools are supplied in the form of shell script files and Fortran source code packages.

Shell script files should end up located in the appropriate directories during installation of supplied files.

For packages supplied as Fortran source code, the user will need to compile the tools and to move the final executable files to appropriate directories. The supplied "makefile" files controlling compilations assume that the Linux GNU g77 compiler is available. Alternate versions of the makefiles are also supplied, covering a range of different Linux

and UNIX operating systems and compilers. For some systems or compilers the user may have to further customize the makefile.

Access to BLAS and LAPACK Libraries

BLAS and LAPACK are standard basic linear algebra and matrix manipulation packages. During compilation of some of the auxiliary tools, precompiled binary libraries for BLAS and LAPACK may need to be available for linking into the final binary executable files.

Fortran-77 source code versions of the BLAS and LAPACK libraries are supplied from which the user may generate the required libraries. Precompiled binary library files may also be supplied for use with some of the more standard operating systems and compilers.

The BLAS and LAPACK libraries are also freely available at www.netlib.org, or precompiled versions may already be available as part of your operating system or compiler package.

The supplied Fortran-77 source code packages may be compiled and installed as follows:

```
cd $HOME/netlib/lapack-with-blas

./MAKE.lnx.pgi611.x64-77    # libraries for linking into f77 programs

./MAKE.lnx.pgi611.x64-90    # libraries for linking into f90 programs
```

The particular version of the MAKE file to use will depend on the user's target operating system and compiler package with which the libraries are intended for use. Different versions of the MAKE file, and of the make.inc file referenced therein, are supplied for a range of systems and compilers. For some systems or compilers the user may have to customize the MAKE file and/or the make.inc file.

Compiling and Installing the Auxiliary Tools

Each tool is compiled and installed separately. For tools where makefiles are supplied to control the compilation process, it is assumed that a makefile appropriate to the intended operating system and compiler package and compiler options has been arranged for by the user. Compilation and installation may otherwise be completed as follows:

```
# splash network to Gridgen file converter

cd $HOME/fortran
make nettogrda
mv nettogrda $HOME/bin
/bin/rm nettogrda.o
```

```
# splash network to Tecplot file converter

cd $HOME/fortran
make nettotec
mv nettotec $HOME/bin
/bin/rm nettotec.o

# IGES nurbs surface extractor

cd $HOME/igeswork
make –f makefile igeswork
mv igeswork $HOME/bin
/bin/rm *.o

# SPLASH network manipulator

cd $HOME/networkx
make –f makefile networkx
mv networkx $HOME/bin
/bin/rm *.o

# initial fin/bulb intersection line

cd $HOME/intersect/finint
make –f makefile finint
mv finint $HOME/bin
/bin/rm *.o

# initial bulb/wing intersection line

cd $HOME/intersect/wngint
make –f makefile wngint
mv wngint $HOME/bin
/bin/rm *.o

# final fin/bulb/wing intersector

cd $HOME/intersect/fin_blb_wng
make –f makefile fin_blb_wng
mv fin_blb_wng $HOME/bin
/bin/rm *.o

# test matrix converter (text file to fortran subroutine)

cd $HOME/txt-to-setmat-in-out
make –f makefile txt-to-setmat-in-out
mv txt-to-setmat-in-out $HOME/bin
/bin/rm *.o

# final results collector

cd $HOME/collect/collect_03.01
make –f makefile collect
mv collect $HOME/bin/collect_03.01
/bin/rm *.o
```

```
# final results post-processor

cd $HOME/tabtec/tabtec.7.1.1.2
make –f makefile tabtec
mv tabtec $HOME/bin/tabtec.7.1.1.2
/bin/rm *.o
```

Procedures for Running the Sample Model Test

PACT model MX12 was designed as a generic International America's Cup Class (IACC) yacht. The configuration included a hull, keel, bulb and rudder. Numerical towing tank tests for PACT model MX12 (with wings added) using ACCPAN and SPLASH are described.

Typical start-to-finish numerical test procedures are reviewed. These include:

- pre-processing of model databases
- control and input data file setup
- model test details and test matrix definition
- test execution
- post-processing of final results into table and plot file format.

The test matrix test points are split into groups. To facilitate pre- and post-processing, groups should consist of a series of points at similar lifting conditions and at increasing forward speeds. If desired, yaw, tab and rudder setting may vary with speed. Adopting a standardized test group naming syntax will facilitate using the supplied pre- and post-processing tools.

group name	**r00a00**	**r10a00**	**r10r01**	**r10r02**	**r10r03**	**r20a00**	**r20r01**	**r20r02**	**r20r03**
heel	0	10	10	10	10	20	20	20	20
yaw	0	0	0	2	4	0	0	2	4
tab	0	0	5	5	5	0	5	5	5
rudder	0	0	0	2	4	0	0	2	4
speeds (kts)	0	0	0	0	0	0	0	0	0
	4	4				4			
	6	6				6			
	8	8	8	8	8	8	8	8	8
	9	9	9	9	9	9	9	9	9
	10	10	10	10	10	10	10	10	10
	11								
	12	12	12	12	12	12	12	12	12
	14								
	16								

Table I Test Groups for PACT Model MX12

group	w00a00	w10a00	w20a00
heel	0	10	20
yaw	0	0	0
tab	0	0	0
rudder	0	0	0
speeds (kts)	0	0	0

Table II Correction Case (Wakes-Off) Test Groups for PACT Model MX12

Pre-Processing of Model Databases

Model databases must be pre-processed in order to provide ACCPAN with surface mesh networks in the required format for each model component (hull, keel, bulb, rudders and wings). The required format, in general, and for each specific component, is described in detail in the ACCPAN Reference Manual.

A sample database directory and set of pre-processing files are supplied for PACT model MX12. From these, the user can run through the pre-processing from start to finish. First, make a copy of the supplied pre-processing directory template. This directory contains the designer-supplied IGES NURBS surface (type 128, untrimmed NURBS) database files for the model, along with shell scripts and tool input files to control and automate the process.

```
cd $HOME/mx12-sample-090418/nets
cp –Rdp mx12-template mx12
```

Then, to pre-process the databases:

```
cd $HOME/mx12-sample-090418/nets/mx12/IGES
./generate.exec
```

The script file “generate.exec” runs all of the program commands, with appropriate input files, necessary to pre-process this particular set of MX12 databases, as follows:

igeswork simple-minded extraction of crude surface meshes from IGES type 128 untrimmed NURBS surface file(s); visualization of these meshes provides guidance for the final surface mesh extractions to follow

igeswork final extraction of higher quality surface meshes from IGES NURBS surface file(s)

networkx relocate model components, from the as-supplied SWL orientation, to the ACCPAN-preferred DWL orientation.

networkx trim hull mesh at x=constant transom plane

sbsmesh elliptic-equation-based refinement of hull mesh

networkx truncate keel tip to just inside top of bulb

networkx truncate bulb trailing edge at % L, from sharp point, to finite base area

networkx combine separate wing upper and lower surface meshes into one mesh

finint determine intersection of fin and bulb

wngint determine intersection of bulb and wing

fin_blb_wng generate intersected fin/bulb surface meshes

fin_blb_wng generate intersected fin/bulb/wing surface meshes

networkx convert half-model meshes to full-model meshes

This completes the pre-processing of the MX12 databases. The completed results are included in the supplied directory $HOME/mx12-sample-090418/nets/mx12.done.

For other databases, the necessary pre-processing steps may not be known in advance. In such cases the user will need to customize the process, essentially creating a new "generate.exec" script, and new auxiliary tool input files, as required for the user's own databases.

For a new model, with model shape similar to, but slightly different than, MX12, it may also be processed as described above, merely by substituting the new IGES files for the MX12 IGES files. To do this in a straightforward fashion requires that the IGES files for the new model are in the same format as the ones supplied for MX12, containing the same model component(s), in the same order, with the same type-128 NURBS topology (surface layouts and knot orientations), etc. Script and input files may still be modified, for example to customize filenames or identifying character strings.

If a new model requires a different approach to pre-processing the databases, or if the format of the IGES NURBS files is changed, then the supplied shell scripts and tool input files may need to be customized as appropriate for the new model files. This is a very user-driven process, but the supplied tools should be able to accommodate most needs. Once everything has been customized for the new model, pre-processing of other similar models may again be accomplished, merely by substituting the IGES files for other similar models into the process.

Overall Test Control and Input Data File Setup

Automated procedures for model test setup are supplied for PACT model MX12. From these, the user can run the complete sample model test from start to finish.

First, make a copy of the supplied model test directory template. This directory contains a series of shell scripts and tool input files to control and automate the model test setup process.

```
cd $HOME/mx12-sample-090418/runs
cp –Rdp mx12-5.1.0.05-5.1.0.04-template mx12-5.1.0.05-5.1.0.04
```

Next, run the shell script file "setup_files_r.exec" (NOTE: the user must have first compiled the automated test setup program, "setup_files_r", a process described in the following subsection):

```
cd $HOME/mx12-sample-090418/runs/mx12-5.1.0.05-5.1.0.04/files
./setup_files_r.exec
```

This invokes the automated test setup program, "setup_files_r", which in turn creates all of the test-matrix-specific script control files and program input data files required to run the iterative ACCPAN/SPLASH numerical calculations for the sample model test. The script also moves the generated test-matrix-specific post-processing control files to the results subdirectory, and it also creates the subdirectories used to organize the SPLASH input and output files for each test group. Datafile and subdirectory names are determined according to the test group naming syntax described previously.

This completes the setup of the sample test for PACT model MX12. The setup files so generated are included with the supplied sample model test files, as directory $HOME/mx12-sample-090418/runs/mx12-5.1.0.05-5.1.0.04.setup.

This automated setup produces files that are customized to the sample model test matrix and to the supplied MX12 model. It may also be used to set up identical tests for similar, systematic or parametric variations of MX12. The input file for the setup program, setup_files_r.inp, allows the user to change a variety of test particulars that commonly vary from one model to the next in a series of similar tests. For more significant changes to the test procedures, the user will need to modify the input data file setup program "setup_files_r" itself. The setup program is supplied as a separate package in Fortran source code format, thus allowing complete user customization of the test input files that it produces. Customization of the setup program, and its compilation, is described in the following section.

Once "setup_files_r.exec", "setup_files_r.inp", and the "setup_files_r" program files have been customized by the user as desired, they provide a new template that can easily be used to conduct tests for similar models at the same test conditions.

Detailed Model Test Inputs and Test Matrix Setup

The input file "setup_files_r.inp" for the setup program "setup_files_r" allows the user to change a variety of test particulars that commonly vary from one model to the next in a series of similar tests. For more significant changes to the test procedures, or to change

the test matrix, the user will likely need to modify the "setup_files_r" program source code.

For test conditions or test points not covered by the automatically generated model test setup tools, a user always has the option to generate by other means (e.g., copy and edit) new ACCPAN input files and qjob control files (the *.dat and *.qjob files in directory $HOME/mx12-sample-090418/runs/mx12-5.1.0.05-5.1.0.04/files).

But, often, the user will want the automatically generated model test setup to reflect the desired test conditions and test matrix. In such cases the Fortran source code in directory $HOME/mx12-sample-090418/setup_files_r/mx12-5.1.0.05-5.1.0.04 should be modified prior to compilation.

Most parameter values appearing in the automatically generated test-matrix-specific script control files and program input data files required to run the iterative ACCPAN/ SPLASH numerical calculations for the sample model test are first assigned in the "setup_files_r" program subroutines defdat.f and outdat.f. Since most parameter assignments are hardwired into the Fortran source code, changing the parameter values requires changing the "setup_files_r" source code, and recompilation.

The format of the test matrix is reflected in the overall source code structure for the "setup_files_r" program, so changing the test matrix format requires a greater familiarity with the coding than does just changing some physical or numerical constants.

The format of the test matrix includes additional test points, some of which are intended to be used for test post-processing corrections. The MX12 sample test case includes zero boat speed cases at the start of each group of test points, and it also includes nonlifting zero boat speed cases at each heel angle in the test matrix, for leakage corrections.

The source code for "setup_files_r" also contains inactive remnants, for example to generate high-panel-density upright half model cases used for panel density corrections, and to generate rudder-on/rudder-off nonlifting cases at each heel angle and boat speed combination in the test matrix used for surfing rudder corrections. Both corrections are no longer recommended, so the corresponding source code has been deactivated.

If the user is happy with the test matrix format, but wishes to change the specific number or value of heel angles, yaw/tab/rudder combinations, or boat speeds, this information is specified in the Fortran source coding in routine setmat.f. Supplied tools can convert the user's test matrix, from an Excel spreadsheet or a text file listing, to the required Fortran source code routine setmat.f.

To do so, the test matrix should be placed in a text file in the format illustrated by the supplied sample model test file "setmat.prn" (this file is located in the supplied directory $HOME/mx12-sample-090418/setup_files_r/mx12-5.1.0.05-5.1.0.04). The text file is merely the result of saving the original Excel spreadsheet representation of the test matrix, "setmat.xls", to a formatted text file. The supplied text-to-Fortran test matrix

converter tool "txt-to-setmat-in-out" is then invoked (here, from within the setmat.exec script) as follows:

```
cd $HOME/mx12-sample-090418/setup_files_r/mx12-5.1.0.05-5.1.0.04
./setmat.exec
```

Once all of the Fortran source code files for the setup_files_r program have been modified as desired, the setup program must then be compiled, as follows:

```
cd $HOME/mx12-sample-090418/setup_files_r/mx12-5.1.0.05-5.1.0.04
make –f makefile setup_files_r
```

Since there may be many different versions of program "setup_file_r", for example for different model test scenarios, or for different types of boats, it is best to keep the final executable file in its corresponding source code directory, and to point to that particular directory during actual model test setup (as in the supplied sample test setup script $HOME/mx12-sample-090418/runs/mx12-5.1.0.05-5.1.0.04/files/setup_files_r.exec).

The results of all the steps outlined in this section are included in the supplied directory $HOME/mx12-sample-090418/setup_files_r/mx12-5.1.0.05-5.1.0.04 (except for final compilation, which is left to the user).

Running the Tests

At this point, all of the files required to run the sample model test for PACT model MX12 with wings should be in place. Operating system administration tasks have been completed. The supplied binary executable files and license keys have been installed. The supplied auxiliary tools have been compiled and installed. The model databases for the sample test have been pre-processed for input to ACCPAN. And the overall test control and input data file setup has been completed. The tests are ready to run.

The test setup process will have produced a master control job file, sample.qjob, located in the master control subdirectory "files". To run the tests, the job file is submitted to the batch job queue system as follows:

```
cd $HOME/mx12-sample-090418/runs/mx12-5.1.0.05-5.1.0.04/files
qsub_csh sample.qjob
```

The master control job file, sample.qjob, invokes the main script for overseeing iterative ACCPAN and SPLASH calculations, accpan-and-splash.exec. The job file invokes the main script for each ACCPAN standard input file *.dat, one such input file corresponding to each of the test point groups discussed previously.

The main script for overseeing iterative ACCPAN and SPLASH calculations, accpan-and-splash.exec, is normally located in $HOME/bin. It implements all the steps required to run the programs in an interactive fashion, thereby enabling the nonlinear free to sink and trim and nonlinear free-surface capabilities. These steps include:

- first iteration: initialize iteration count and set first iteration options
- subsequent iterations: increment iteration count and reset iteration options
- run ACCPAN for the specified test group:
    - process ACCPAN standard input file, *.dat
    - process model database files
    - maintain a listing file of unconverged test points in the test group, accpan.data (located in $TMPDIR)
    - for each unconverged test point in the test group:
        - update sink, trim, and free-surface elevations (initialize, restart, or update from preceding iteration) using latest *.fm and *.p00 and *.t00 files
        - update splash panel model via standard input file, *.g00
        - update the developing free-surface elevation file, *.p00
        - update the force/moment/sink/trim history file, *.fm
    - accumulate commands to run SPLASH for the unconverged test points into a temporary script file, splash.exec (located in $TMPDIR)
- execute the temporary script file, splash.exec, which in turn runs SPLASH for each of the unconverged test points in the test group:
    - for each test point, the SPLASH run updates the force/moment/sink/trim history file, *.fm, and the flow solution file, *.t00
- iterate (repeat) the above steps until all test points in the test group input data file have converged in sink, trim and free-surface elevation, or upon reaching the maximum number of iterations specified in the ACCPAN standard input file.

Upon submitting sample.qjob to qsub_csh, the sample model test can then take several hours to complete (it takes about 3 hours on an AMD 2.6GHz Opteron 252). Once the main script is invoked for a particular test group input data file, all test points in the group should converge, in sink, trim and free-surface elevation, in approximately 4 to 20 ACCPAN/SPLASH iterations. The master job control file then invokes the main script for the next test group input data file. Depending on the system, submitted jobs may send user email upon completion.

The user may also monitor test progress by listing and examination of various input and output files being created and updated during the test, in the model test subdirectories, as well as in the temporary/scratch directory $TMPDIR unique to each batch job submittal.

Completed iterative ACCPAN and SPLASH calculations for the sample test are supplied in the directory $HOME/mx12-sample-090418/runs/mx12-5.1.0.05-5.1.0.04.done.

ACCPAN and SPLASH are not multiple processor capable. If multiple processors are available, then the calls in sample.qjob to accpan-and-splash.exec for the complete sequence of test groups can usually be split amongst several *.qjob files. The supplied files temp1.qjob and temp2.qjob illustrate splitting the calls between two qjob files which are run as follows:

```
qsub_csh temp1.qjob
qsub_csh temp2.qjob
```

The two jobs, each with their own set of test groups, will then run simultaneously on separate processors. This approach can accommodate as many processors as there are test groups. For more processors, the user might split test groups themselves, but this requires modified ACCPAN input files (*.dat) and corresponding job control files (*.qjob) to run. Another approach is to run tests for more than one model at the same time, as a different processor can be used for each test.

Correction cases are normally "restarted" from final results for corresponding standard cases and should always be run only after successful completion of the corresponding standard cases.

Convergence of Nonlinear Calculations: Sink, Trim, and Free-Surface Elevation

Two shell scripts are supplied to assist with convergence checking, $HOME/bin/check-runs.exec, and $HOME/mx12-sample-090418/check.exec, which are invoked as follows:

```
cd $HOME/mx12-sample-090418/runs
./check.exec mx12-5.1.0.05-5.1.0.04
```

This will create a file "$HOME/mx12-sample-090418/runs/mx12-5.1.0.05-5.1.0.04/mx12-5.1.0.05-5.1.0.04.check" containing a summary of convergence and error condition results. The shell scripts obtain this information by examination of all the ACCPAN standard output files, *.accout.

Convergence of calculations for nonlinear sink, trim and free-surface elevation is not guaranteed. Results can depend on any of the input data, notably: hull form, appendage configuration, heel angle and forward speed, and lift settings (yaw, tab and rudder angles).

Calculations which fail to converge may show oscillation or other cyclical change of the results from iteration to iteration, at a magnitude sufficient to preclude convergence. Or calculations may diverge, due to a catastrophic failure, either by ACCPAN while constructing a SPLASH panel model input file, or by SPLASH while calculating the corresponding flow solution.

Whatever the cause, inspection of all the various files involved in the complete test process may be required in order to ascertain the nature of the problem. For fairly small oscillations, the results of the last iteration(s) might be acceptable, or additional iterations with smaller underrelaxation factors might lead to convergence. This may also work to bring larger oscillations under control. Rerunning a failed test point using different initial or restart conditions, and/or using smaller underrelaxation factors, can also be attempted.

If unacceptably large oscillations persist, or the calculations continue to fail, there may be no practical approach to achieve a converged result for a particular model using the specified test conditions and other input parameters. Changing other test input parameters on a point-by-point basis (aside from heel, speed, yaw, tab and rudder settings) is not recommended, especially when conducting boat-to-boat comparisons.

Catastrophic failure of ACCPAN or SPLASH can leave various output files for the corresponding test point in a partially completed state. In rare circumstances, a failure in ACCPAN may disrupt the test process for all test points which appear after the failed test point in the test group input data file. Corrupted files may need to be repaired before any attempts are made to rerun or restart particular test points or before post-processing of the final results.

Dealing with difficulties due to oscillation or divergence of iterative ACCPAN/SPLASH calculations can be quite tedious and therefore somewhat frustrating. From time to time, the user might reassess the model test matrix, and minimize or eliminate further testing at conditions where problems occur.

Post-Processing of Final Results

Once all ACCPAN/SPLASH model test calculations have completed, the final results may be collected and post-processed using the auxiliary tools "collect" and "tabtec". Post-processing is normally carried out starting in the model test subdirectory "results". The "collect" program gathers the final results for each test point, including correction cases, into final *.fm files. The "tabtec" program combines the hydrostatic and inviscid and viscous hydrodynamic force and moment data, applies requested corrections, and outputs the post-processed results in several tabular and plotting file formats.

The sample model test is post-processed as follows:

```
cd $HOME/m x12-sample-090418/runs/mx12-5.1.0.05-5.1.0.04/results
./generate.exec
```

The shell script file merely runs the "collect" and "tabtec" programs. It also parses some of the generated final results files into additional results files, merely to organize the results somewhat differently (for example, with the primary groupings containing results at different yaw angles, rather than at different boat speeds).

Completed post-processed results are also included in the supplied directory $HOME/mx12-sample-090418/runs/mx12-5.1.0.05-5.1.0.04.done/results.

Each test group will have a corresponding *.clt file in the results directory. These files, which were automatically generated by the "setup_files_r" program during the model test setup process, become inputs to the "collect" program, telling it which groups to post-process, and which test points in each group.

Entries in the *.clt files for specific test points for which the results have not converged to the user's satisfaction must be deleted from the *.clt files prior to post-processing the results. Test points without entries in the *.clt files will not be included in the post-processing for final results.

The "tabtec" program can apply various user-selected corrections to generate the final results. Only a brief summary of the corrections is presented here. A more complete description of tabtec, and the available corrections, is available in the section devoted to the auxiliary tools themselves.

The recommended tabtec post-processing options include:

- Addition of ACCPAN-calculated hydrostatic forces and moments acting on the wetted database surfaces to the SPLASH-calculated hydrodynamic forces and moments (SPLASH-calculated hydrostatic forces and moments acting on the wetted panel model surfaces are less accurate and are not used by tabtec). To avoid division by zero in final force and moment coefficients and in final force areas and moment volumes, for zero boat speed test points the hydrostatic forces and moments are never added to the SPLASH hydrodynamic results.
- Addition of viscous stripping estimates to the SPLASH-calculated inviscid forces and moments (except, of course, for any zero boat speed test points). Two different types of viscous stripping results are available:
    - The ACCPAN-based stripping is recommended, and should be activated during the iterative ACCPAN/SPLASH model test calculations if it is to be available later during the post-processing stage (the ACCPAN-based stripping should also be activated to properly include the contribution of viscous forces and moments to equilibrium in sink and trim). The ACCPAN-based stripping yields forces and moments due to viscous drag acting in a distributed fashion over the wetted database surfaces.
    - A tabtec-based stripping is also available. It is calculated after the iterative ACCPAN/SPLASH model test calculations, and is therefore easily changed (for example, to investigate different viscous drag approximations). However, it yields only the drag due to the viscous forces, and not any moments. Because it is based on total component wetted area only, and not on the distribution (with respect to the moment reference point) of that wetted area and the associated viscous forces.
- An aerodynamic leakage correction.

NOTE: The current version of tabtec requires that keel and rudder database mesh files contain half-model representations of those surfaces. However, the latest versions of ACCPAN require that keel and rudder database mesh files contain full-model representations of those surfaces. The post-process tabtec viscous stripping therefore may not work correctly for keel and rudder databases that have been pre-processed for use with the latest versions of ACCPAN. Instead, the viscous stripping built into ACCPAN can be activated, during the model tests, by setting ivisc=-1 in the ACCPAN input files, and the ACCPAN-generated viscous forces and

moments can be included in the final post-processed results, by setting ivisc=-1 in the tabtec input file.

The post-processing options and corrections must be thoroughly understood and applied with caution. Test points that fail to give converged results may also be required in order to determine a correction for other, converged test points. If a desired correction for a converged test point is not available, then its final post-processed results are not correct. Results for such test points should therefore be omitting during the post-processing of corrected model test results. For example, by deleting their corresponding entries in the *.clt files.

This is not too severe a restriction since the only currently recommended correction is for "aerodynamic leakage" which depends only on the correction cases calculated at nonlifting, zero boat speed conditions which are among the least likely of all the test points to fail.

The aerodynamic leakage correction accounts for the fact that, for inviscid and irrotational flow past a closed nonlifting body in an unbounded domain, the lift (side force) and drag are identically zero. These assumptions are not valid for a non-closed body, which is the case in the event that any transom immersion is present for any of the nonlifting, zero boat speed correction case test points. Under such circumstances, application of the aerodynamic leakage correction in not appropriate. Nor should it be applied during the post-processing of tests for other models with which boat-to-boat comparisons will be made.

The aerodynamic leakage correction is applied to side and drag forces, but not to any moments. Because not all the moments are necessarily zero, even for a closed nonlifting body in an unbounded domain (of course, for upright panel model calculations with complete port/starboard hull and appendage symmetry, roll and yaw moment are zero, but this is due merely to symmetry).

Applying the leakage correction to the side and drag forces but not to the moments can alter the calculated vertical and longitudinal locations of the side force center of effort, particularly if the centers of effort of the "correction forces" are far from the uncorrected force locations. Although this does not appear to be a problem for convention monohull configurations, in other circumstances it may become as issue. To avoid any inconsistencies, the user is free to generate uncorrected results to use for addressing load and balance issues, and corrected results to use for addressing performance issues. The user might also modify the tabtec source coding, for example to also apply the leakage correction to the roll and yaw moments, although, as discussed, the theoretical argument for correcting the forces does not in general extend to the moments.

Auxiliary Tool Descriptions

igeswork

The primary function of igeswork is to extract model database surface meshes (specifically, surface mesh corner point x/y/z coordinate data) from IGES NURBS files. The IGES NURBS files can be generated using many CAD-type naval architecture packages commonly used by designers.

The tool can be run directly at the command line. The user is prompted for additional input. User responses are typically stored in files from which igeswork is to read input. This simplifies running and automating the pre-processing of model database geometry.

The primary main options are "(1) input new IGES file" and "(4) output specified 128 entities to new mesh file". To input a new IGES file, the user is prompted for the filename. To generate a mesh file, the user is prompted for the following input data:

iyzflip if not 0, swaps y and z coordinates

xscale,yscale,zscale:
geometric scaling factors applied to the geometry in each of the coordinate directions (use 1.,1.,1. for no scaling)

The user is then prompted for one of two options: "(0) all meshes" or "(1) specified meshes and mesh parameters". With the "all meshes" option, the user is prompted only for the number of points to extract in the u and in the v directions. The same number of points is used for all meshes, with points evenly spaced in u and v.

With the "specified meshes and mesh parameters" option, the user is prompted for the relative IGES 128 entity number (1 for the first such entity in the file, 2 for the second, and so on), an identifying mesh title (a character string), and the following input data:

iflip if not 0, reverses the direction of the I-coordinate (the u-coordinate of the IGES surface)

jflip if not 0, reverses the direction of the J-coordinate (the v-coordinate of the IGES surface)

ijflip if not 0, swaps the I- and J-coordinates (the u- and v-coordinates of the IGES surface)

inum number of mesh points in the I-direction

ndegi degree of polynomial used to distribute points in the I-direction:
1 even spacing
2 quadratic, start or end cluster factor given (input a negative cluster factor at the unspecified end)
3 cubic, start and end cluster factors given
4 quartic, start and end cluster factors given, start or end curvature (rate of change of cluster) factor given (input a negative curvature factor at the unspecified end).

5 quintic, start and end cluster factors given, and start and end curvature (rate of change of cluster) factors given

d1i1 relative cluster factor at the first point in the I-direction

d1i2 relative cluster factor at the last point in the I-direction

d2i1 curvature factor at the first point in the I-direction

d2i2 curvature factor at the last point in the I-direction

jnum number of mesh points in the J-direction

ndegj degree of polynomial to distribute points in the J-direction:

1 even spacing
2 quadratic
3 cubic
4 quartic
5 quintic

d1j1 relative cluster factor at the first point in the J-direction

d1j2 relative cluster factor at the last point in the J-direction

d2j1 curvature factor at the first point in the J-direction

d2j2 curvature factor at the last point in the J-direction

irefit after initial extraction of the requested point distribution in the u-coordinate, the number of re-extractions used by igeswork to obtain the requested point distribution the I-direction in physical space

jrefit after initial extraction of the requested point distribution in the v-coordinate, the number of re-extractions used by igeswork to obtain the requested point distribution the J-direction in physical space

i1clos if not 0, set the z-coordinate of all I=1 mesh points to zero, after the initial extraction and each refit re-extraction

i2clos if not 0, set the z-coordinate of all I=IMAX mesh points to zero, after the initial extraction and each refit re-extraction

j1clos if not 0, set the z-coordinate of all J=1 mesh points to zero, after the initial extraction and each refit re-extraction

j2clos if not 0, set the z-coordinate of all J=JMAX mesh points to zero, after the initial extraction and each refit re-extraction

These inputs are repeated for each mesh to be generated, and are terminated by an input of 0 for the relative IGES 128 entity number

With either of the two meshing options, the user is then prompted for an output file format and filename. After output of the mesh file, the user is returned to the main menu options.

networkx

The networkx tool incorporates various surface mesh manipulation capabilities. All input surface mesh files must be in ACCPAN/SPLASH surface mesh network file format. The same format is also used for all output surface mesh files.

The tool can be run directly at the command line. The user is prompted for additional input. User responses are typically stored in files from which networkx is to read input. This simplifies running and automating the pre-processing of model database geometry.

The available capabilities include:

input	read in a specified mesh file
transform	swap y and z coordinates, then a sign change in the y-coordinate
reverse	reverse the points in the I or J direction
transpose	swap the I and J directions
respline	change the number of points, or their distribution, in the I or J direction, according to user specified inputs for start and end cluster parameters
scale	applies input scale factors to the surface x,y,z coordinates
translate	applies input translation factors to the surface x,y,z coordinates
output	output resulting mesh surface to a specified mesh file
quit	exit the program
ijmax	extend the IMAX and/or JMAX dimension(s) of the surface mesh array, to the specified input values. Does not change the current number of points or their distribution, but defines additional array space into which an adjacent surface mesh can then be joined to the current surface mesh using the **setblock** and **input** options
truncate	truncate a mesh at an x-coordinate which is specified as a percentage of overall body length (the overall extent of the surface in the x-direction), while maintaining the same overall nondimensional arclength distribution of surface mesh points
splitbulb	split a half-model representation of a bulb-type surface at its maximum half breadth
setblock	allows the user to specify the indices of a sub-block, a subset of the current mesh array, to which subsequent operations will apply. The **setblock** option remains in effect until deactivated by again executing the **setblock** option and setting all sub-block indices to 0. Other option behavior or availability may be affected when **setblock** is active. For example, changing the number of points in the I-direction in a sub-block using the **respline** option is not permitted, unless the sub-block is specified to extend from J=1 to J=JMAX
settitle	change the character string used to identify a surface mesh
rotate	rotate the mesh geometry a specified angle about an axis parallel to the x, y or z coordinate axis and passing through a specified point
extend	extend the current mesh beyond one of its four edges, to a specified x, y or z value, using the specified number of points and the specified degree of extrapolation to perform the extension
dextend	extend the current mesh beyond one of its four edges, by a specified distance in the x, y or z coordinate, using the specified number of points and the specified degree of extrapolation to perform the extension
tranatx	truncate a hull-type mesh at a specified x-coordinate, maintaining the same overall nondimensional arclength distribution of surface mesh points

tranxlin truncate a hull-type mesh at a specified x-coordinate, maintaining the same overall nondimensional arclength distribution of surface mesh points. Uses a lower order interpolation to redistribute, compared to **tranatx**

sbsmesh

The sbsmesh tool provides surface mesh refinement using elliptic-based grid generation equations. The underlying surface geometry is taken from the input mesh and remains unchanged thereafter. The elliptic refinement uses fixed-grid background control functions and Sorenson-type GRAPE foreground control functions to improve orthogonality and spacing near the edges of the mesh. Only compiled, binary executable file versions of sbsmesh are supplied (Fortran source code is not provided).

The tool can be run directly at the command line. The user is prompted for the filenames of the original (input) surface mesh file and the refined (output) surface mesh file. User responses are typically stored in files from which sbsmesh is to read input. This simplifies running and automating the pre-processing of model database geometry.

finint

The finint tool calculates the line of intersection between a keel-type component mesh (the source network) and a bulb-type component mesh (the target network). Half-model meshes are used for both input and output.

The tool can be run directly at the command line. The user is prompted for additional input (unless prompting is deactivated in the Fortran source code). User responses are typically stored in files from which finint is to read input. This simplifies running and automating the pre-processing of model database geometry. Required user inputs are:

fis name of file containing input source (keel) network

dxs 2-character string specifying intersecting (xsi) direction on source (I or J varying) and in what sense (I or J increasing or decreasing):

- `+m` source xsi increases as I increases
- `-m` source xsi decreases as I increases
- `+n` source xsi increases as J increases
- `-n` source xsi decreases as J increases

des 2-character string specifying nonintersecting (eta) direction on source:

- `+m` source eta increases as I increases
- `-m` source eta decreases as I increases
- `+n` source eta increases as J increases
- `-n` source eta decreases as J increases

imins if not 0, the source network may be extended at I=1 in the –I direction (using linear extrapolation) to produce an intersection

imaxs if not 0, the source network may be extended at I=IMAX in the +I direction (using linear extrapolation) to produce an intersection

fit name of file containing input target (bulb) network

dxt 2-character string specifying intersecting (xsi) direction on target (I or J varying) and in what sense (I or J increasing or decreasing):

- `+m` target xsi increases as I increases
- `-m` target xsi decreases as I increases
- `+n` target xsi increases as J increases
- `-n` target xsi decreases as J increases

det 2-character string specifying nonintersecting (eta) direction on target:

- `+m` target eta increases as I increases
- `-m` target eta decreases as I increases
- `+n` target eta increases as J increases
- `-n` target eta decreases as J increases

imint if not 0, the target network may be extended at I=1 in the –I direction (using linear extrapolation) to produce an intersection

imaxt if not 0, the target network may be extended at I=IMAX in the +I direction (using linear extrapolation) to produce an intersection

jmint if not 0, the target network may be extended at J=1 in the –J direction (using linear extrapolation) to produce an intersection

jmaxt if not 0, the target network may be extended at J=JMAX in the +J direction (using linear extrapolation) to produce an intersection

Any remaining lines of input are obsolete and not read in by the program.

wngint

The wngint tool calculates the line of intersection between a bulb-mounted-winglet-type component mesh (the source network) and a bulb-type component mesh (the target network). Half-model meshes are used for both input and output.

The tool can be run directly at the command line. The user is prompted for additional input (unless prompting is deactivated in the Fortran source code). User responses are typically stored in files from which wngint is to read input. This simplifies running and automating the pre-processing of model database geometry. Required user inputs are:

fis name of file containing input source (winglet) network

dxs 2-character string specifying intersecting (xsi) direction on source (I or J varying) and in what sense (I or J increasing or decreasing):

- `+m` source xsi increases as I increases
- `-m` source xsi decreases as I increases
- `+n` source xsi increases as J increases
- `-n` source xsi decreases as J increases

des 2-character string specifying nonintersecting (eta) direction on source:

- `+m` source eta increases as I increases
- `-m` source eta decreases as I increases
- `+n` source eta increases as J increases

`-n` source eta decreases as J increases

imins if not 0, the source network may be extended at I=1 in the –I direction (using linear extrapolation) to produce an intersection

imaxs if not 0, the source network may be extended at I=IMAX in the +I direction (using linear extrapolation) to produce an intersection

fit name of file containing input target (bulb) network

dxt 2-character string specifying intersecting (xsi) direction on target (I or J varying) and in what sense (I or J increasing or decreasing):

`+m` target xsi increases as I increases
`-m` target xsi decreases as I increases
`+n` target xsi increases as J increases
`-n` target xsi decreases as J increases

det 2-character string specifying nonintersecting (eta) direction on target:

`+m` target eta increases as I increases
`-m` target eta decreases as I increases
`+n` target eta increases as J increases
`-n` target eta decreases as J increases

imint if not 0, the target network may be extended at I=1 in the –I direction (using linear extrapolation) to produce an intersection

imaxt if not 0, the target network may be extended at I=IMAX in the +I direction (using linear extrapolation) to produce an intersection

jmint if not 0, the target network may be extended at J=1 in the –J direction (using linear extrapolation) to produce an intersection

jmaxt if not 0, the target network may be extended at J=JMAX in the +J direction (using linear extrapolation) to produce an intersection

Any remaining lines of input are obsolete and not read in by the program.

fin_blb_wng

The fin_blb_wng tool generates the final intersected database mesh files for the keel, bulb and bulb-mounted-winglet. The finint and wngint programs must be run just prior to running fin_blb_wng. Half-model meshes are used for both input and output.

The tool can be run directly at the command line. The user is prompted for additional input, but a comment line must precede each actual line of input. User responses are typically stored in files from which fin_blb_wng is to read input. This simplifies running and automating the pre-processing of model database geometry. Required user inputs are as follows:

fileik name of file containing input keel network (or ‘none’)

fileib name of file containing input bulb network

fileiw name of file containing input winglet network (or ‘none’)

fileok name of file containing output intersected keel network (omit this input if fileik=’none’)

fileob name of file containing output intersected bulb network

fileow name of file containing output intersected winglet network (omit this input if fileiw='none')

imo number of points in I direction for final intersected bulb network

jmo nominal number of points in J direction for final intersected bulb network (not counting the doubled-up line due to a bulb/wing intersection)

itertm number of iterations for initial bulb elliptic grid generation equation solution (background control functions only)

omegtm underrelaxation factor for initial bulb elliptic grid generation equation solution (background control functions only)

itergr number of iterations for final bulb elliptic grid generation equation solution (foreground and background control functions)

omeggr underrelaxation factor for final bulb elliptic grid generation equation solution (foreground and background control functions)

wp foreground control function underrelaxation factor

plim maximum allowable change in control function during one iteration

ib if greater than 0, use foreground C.F's along bottom edge (at I=1)

it if greater than 0, use foreground C.F's along top edge (at I=IMAX)

il if greater than 0, use foreground C.F's along left edge (at J=1)

ir if greater than 0, use foreground C.F's along right edge (at J=JMAX)

ibl,ibr,itl,itr,ilb,ilt,irb,irt not used

gb fraction of distance from bottom edge (toward top edge) at which bottom edge control functions decay to 10% of their bottom edge values

gt fraction of distance from top edge (toward bottom edge) at which top edge control functions decay to 10% of their top edge values

gl fraction of distance from left edge (toward right edge) at which left edge control functions decay to 10% of their left edge values

gr fraction of distance from right edge (toward left edge) at which right edge control functions decay to 10% of their right edge values

gbl fraction of distance along bottom edge (from left edge toward right edge) at which bottom-left corner angle effect decays to 10%

gbr fraction of distance along bottom edge (from right edge toward left edge) at which bottom-right corner angle effect decays to 10%

gtl fraction of distance along top edge (from left edge toward right edge) at which top-left corner angle effect decays to 10%

gtr fraction of distance along top edge (from right edge toward left edge) at which top-right corner angle effect decays to 10%

glb fraction of distance along left edge (from bottom edge toward top edge) at which bottom-left corner angle effect decays to 10%

glt fraction of distance along left edge (from top edge toward bottom edge) at which top-left corner angle effect decays to 10%

grb fraction of distance along right edge (from bottom edge toward top edge) at which bottom-right corner angle effect decays to 10%

grt fraction of distance along right edge (from top edge toward bottom edge) at which top-right corner angle effect decays to 10%

ibigb exponent amplification factor, applied when first spacing at bottom edge remains larger than requested (nominally, ibigb=0)

ismlb exponent amplification factor, applied when first spacing at bottom edge remains smaller than requested (nominally, ismlb=0)

ibigt exponent amplification factor, applied when first spacing at top edge remains larger than requested (nominally, ibigt=0)

ismlt exponent amplification factor, applied when first spacing at top edge remains smaller than requested (nominally, ismlt=0)

ibigl exponent amplification factor, applied when first spacing at left edge remains larger than requested (nominally, ibigl=0)

ismll exponent amplification factor, applied when first spacing at left edge remains smaller than requested (nominally, ismll=0)

ibigr exponent amplification factor, applied when first spacing at right edge remains larger than requested (nominally, ibigr=0)

ismlr exponent amplification factor, applied when first spacing at right edge remains smaller than requested (nominally, ismlr=0)

ikleb final intersected bulb I-value at keel leading edge

ikteb final intersected bulb I-value at keel trailing edge

iwleb final intersected bulb I-value at wing leading edge

iwteb final intersected bulb I-value at wing trailing edge

nettogrda

The nettogrda tool converts surface mesh files from ACCPAN/SPLASH network format to Gridgen format.

The tool can be run directly at the command line. The user is prompted for the filenames of the input surface mesh file, in ACCPAN/SPLASH network format, and the output surface mesh file, in Gridgen format. User responses may be stored in files from which nettogrda is to read input. This simplifies running and automating the pre-processing of model database geometry.

nettotec

The nettotec tool converts surface mesh files from ACCPAN/SPLASH network format to Tecplot format.

The tool can be run directly at the command line. The user is prompted for the filenames of the input surface mesh file, in ACCPAN/SPLASH network format, the output surface mesh file, in Tecplot format, and an identifying character string description to be included in the output Tecplot file. User responses may be stored in files from which nettotec is to read input. This simplifies running and automating the pre-processing of model database geometry.

setup_files_r

The setup_files_r tool helps to simplify and automate the setup of files for numerical model tests, especially when the same modeling parameters and test matrix are desired for tests for a series of model parametric design variations. As mentioned previously in the discussion of automated model test setup procedures, user modification of the Fortran source code for setup_files_r is normally required in order to change or customize most numerical modeling parameters or the numerical test matrix. The user may need to become familiar with Fortran, and with ACCPAN and SPLASH input data, output files, and other operational considerations, in order to perform the required customizations.

Once the tool is customized and compiled, it can be run directly at the command line. The small input file contains various ACCPAN and SPLASH inputs which typically may vary from one model design variation to the next. Required user inputs are:

title	a string of up to 60 characters identifying the model test details
isavf	ACCPAN input parameter "isavf" and SPLASH input parameter "ifmst", both of which specify the iteration history file format
xdatum	DWL x-coordinate of model-fixed datum point for moments and sinkage
ydatum	DWL y-coordinate of model-fixed datum point for moments and sinkage
zdatum	DWL z-coordinate of model-fixed datum point for moments and sinkage
sref	reference area for force and moment coefficients
cref	reference length for moment coefficients, for geometric tests and tolerances, and for numerous Froude-number-type free-surface terms
xfrnt	distance in x-direction from model center to upstream boundary of paneled free-surface (nondimensionalized by cref)
xback	distance in x-direction from model center to downstream boundary of paneled free-surface (nondimensionalized by cref)
zside	distance in z-direction from model center to transverse boundary of paneled free-surface (nondimensionalized by cref)
grav	acceleration due to gravity
dens	fluid (water) density
visc	fluid (water) viscosity
xftab1	x-coordinate of first point defining the forward rudder hinge line
yftab1	y-coordinate of first point defining the forward rudder hinge line
xftab2	x-coordinate of second point defining the forward rudder hinge line
yftab2	y-coordinate of second point defining the forward rudder hinge line
xktab1	x-coordinate of first point defining the keel tab hinge line
yktab1	y-coordinate of first point defining the keel tab hinge line
xktab2	x-coordinate of second point defining the keel tab hinge line
yktab2	y-coordinate of second point defining the keel tab hinge line
xrtab1	x-coordinate of first point defining the aft rudder hinge line
yrtab1	y-coordinate of first point defining the aft rudder hinge line
xrtab2	x-coordinate of second point defining the aft rudder hinge line
yrtab2	y-coordinate of second point defining the aft rudder hinge line
ispnhl	spanload analysis option flag for hull component

ispnkl	spanload analysis option flag for keel component
ispnbl	spanload analysis option flag for bulb component
ispnwg	spanload analysis option flag for wing component
ispnfr	spanload analysis option flag for forward rudder component
ispnrd	spanload analysis option flag for aft rudder component
vcgmwl	final VCG (model-fixed y-coordinate of c.g.) at MWL conditions
wcrew	crew weight at MWL conditions
lcgcrew	crew weight LCG (model-fixed x-coordinate of c.g.) at MWL conditions
vcgcrew	crew weight VCG (model-fixed y-coordinate of c.g.) at MWL conditions
wsail	sail weight at MWL conditions
lcgsail	sail weight LCG (model-fixed x-coordinate of c.g.) at MWL conditions
vcgsail	sail weight VCG (model-fixed y-coordinate of c.g.) at MWL conditions
wball	ballast weight at MWL conditions
lcgball	ballast weight LCG (model-fixed x-coordinate of c.g.) at MWL conditions
vcgball	ballast weight VCG (model-fixed y-coordinate of c.g.) at MWL conditions
ycsail	rig height RCE (model-fixed y-coordinate of sail center of effort)

The final two inputs are: the name of the directory containing the pre-processed database surface mesh files; and the name of the directory containing the subdirectories and files for a previous model test, from which the sink, trim and free-surface elevations for the current test will be restarted (initialized). Specifying "none" for the latter will result in the tests starting from scratch.

txt-to-setmat-in-out

The txt-to-setmat-in-out tool converts a user's test matrix details (speed, heel, yaw, tab and rudder) from a tabulated text file format into a Fortran source code subroutine file. The subroutine file is intended to be substituted for "setmat.f" prior to compilation of the automated model test control and input data file setup program "setup_files_r".

The tool can be run directly at the command line. The user is prompted for the filename of the input test matrix, in tabulated text format, and the filename of the output test matrix, in Fortran subroutine format. In the input text file, test points should be in group order (as discussed previously with respect to the sample model tests) or the program may not work correctly.

The input text-format file can be generated by any available means, such as by saving an Excel spreadsheet as an appropriately formatted ASCII text file.

collect

The collect tool gathers the raw results from the iterative ACCPAN and SPLASH tests into more compact raw results files which are required for final post-processing using the tabtec program.

The tool can be run directly at the command line using input files (the *.clt files) that specify which specific test point results are to be collected. User inputs are:

- name of file (excluding the final ".fm" extension) where the collected results will be stored
- name of directory where the raw results files from the iterative ACCPAN and SPLASH tests reside
- iteration of interest (use 0 to obtain results from the final iteration)
- the list of filenames (excluding the final ".fm" extension), residing in the specified raw results directory, from which the raw results are to be collected

tabtec

The tabtec tool performs the final post-processing of collected raw results files into more readily useable final results and into more convenient file formats. The tool also applies various user-selected post-test corrections.

The tool can be run directly at the command line. The input file specifies the post-processing parameters and options. Required user inputs are:

general inputs:

title a string of up to 80 characters identifying the post-processed model test
velin scale factor applied to raw velocities prior to post-processing (usually 1.0)
velout scale factor applied to final velocities after post-processing (e.g., for final output in knots, use knots per meter/second or per foot/second)
visc fluid viscosity, used for Reynolds number based skin friction calculations when tabtec's post-process viscous stripping is activated (when ivisc>0)
dens fluid density (usually the same value used for ACCPAN and SPLASH)
sclmod model scale factor used to adjust final dimensional results, for example, sclmod=2.0 to convert half-scale tests to full-scale results (values other than sclmod=1.0 may not be fully supported)

treatment of sail forces and moments:

ssail reference area by which raw sail forces and moments in the *.fm files are nondimensionalized (if these are already dimensional, set **ssail=1.0**)
csail reference chord length by which raw sail yaw and pitch moments in the *.fm files are nondimensionalized (if these are already dimensional, set **csail=1.0**)
bsail reference span length by which raw sail roll moment in the *.fm files is nondimensionalized (if this is already dimensional, set **bsail=1.0**)
qsail input value indicates if raw sail forces and moments in the *.fm files are nondimensionalized by dynamic pressure (set qsail=1.0 if they are; set **qsail=0.0** if they are not)

treatment of hydrostatic forces and moments:

sstat reference area by which raw hydrostatic forces and moments in the *.fm files are nondimensionalized (if these are already dimensional, set **sstat=1.0**)

cstat reference chord length by which raw hydrostatic yaw and pitch moments in the *.fm files are nondimensionalized (if these are already dimensional, set **cstat=1.0**)

bstat reference span length by which raw hydrostatic roll moment in the *.fm files is nondimensionalized (if this is already dimensional, set **bstat=1.0**)

qstat input value indicates if raw hydrostatic forces and moments in the *.fm files are nondimensionalized by dynamic pressure (set qstat=1.0 if they are; set **qstat=0.0** if they are not)

treatment of viscous forces and moments:

svisc reference area by which viscous forces and moments in the *.fm files are nondimensionalized (typically, the same value as the reference area used for the ACCPAN and SPLASH calculations)

cvisc reference chord length by which viscous yaw and pitch moments in the *.fm files are nondimensionalized (typically, the same value as the reference (boat) length used for the ACCPAN and SPLASH calculations)

bvisc reference span length by which viscous roll moment in the *.fm files are nondimensionalized (typically, the same value as the reference (boat) length used for the ACCPAN and SPLASH calculations)

qvisc input value indicates if viscous forces and moments in the *.fm files are nondimensionalized by dynamic pressure (set **qvisc=1.0** if they are; set qvisc=0.0 if they are not)

treatment of hydrodynamic forces and moments:

shdyn reference area by which raw hydrodynamic forces and moments in the *.fm files are nondimensionalized (typically, the same value as the reference area used for the ACCPAN and SPLASH calculations)

chdyn reference chord length by which raw hydrodynamic yaw and pitch moments in the *.fm files are nondimensionalized (typically, the same value as the reference (boat) length used for the ACCPAN and SPLASH calculations)

bhdyn reference span length by which raw hydrodynamic roll moment in the *.fm files are nondimensionalized (typically, the same value as the reference (boat) length used for the ACCPAN and SPLASH calculations)

qhdyn input value indicates if raw hydrodynamic forces and moments in the *.fm files are nondimensionalized by dynamic pressure (set **qhdyn=1.0** if they are; set qhdyn=0.0 if they are not)

treatment of final force and moment results:

sref reference area by which forces and moments in the final results files are to be nondimensionalized (set **sref=1.0** to produce full-dimensional, or force-area and moment-volume, type results)

cref reference chord length by which yaw and pitch moments in the final results files are to be nondimensionalized set **cref=1.0** to produce full-dimensional, or force-area and moment-volume, type results)

bref reference span length by which roll moment in the final results files are to be nondimensionalized (set **bref=1.0** to produce full-dimensional, or force-area and moment-volume, type results)

qref input value indicates if forces and moments in the final results files are to be multiplied by dynamic pressure (set qref=1.0 if they are; set **qref=0.0** if they are not) (set **qref=0.0** for coefficient, or force-area and moment-volume, type results)

adjustments and corrections to the final force and moment results:

istat option flag indicating that the hydrostatic forces and moments are to be included in the final results (set istat=0 if they are not; set **istat=1** if they are). Hydrostatic forces are never included in results for cases at zero boat speed.

ivisc option flag indicating that viscous forces and moments are to be included in final results (set ivisc=0 if they are not; set ivisc=1 if viscous drag is to be computed by tabtec during post-processing; set **ivisc=-1** to include viscous forces and moments computed by ACCPAN's viscous stripping and contained in the raw *.fm files)

ileak option flag indicating that hydrodynamic leakage corrections are to applied to the final results (set ileak=0 if they are not; set ileak=1 if they are). Correction forces are intended to be those computed at zero boat speed at zero yaw, tab and rudder, and wakes-off.

For full-model groups, heeled and yawed, the correction values are taken to be the first of the following values to be identified:

- wakes off, zero boat speed, heeled and yawed
- wakes off, zero boat speed, heeled
- wakes off, zero boat speed, upright
- zero boat speed, heeled
- zero boat speed, upright

For half-model groups, upright, the correction values are taken to be the first of the following values to be identified:

- wakes off, zero boat speed, upright half-model
- zero boat speed, upright half-model

Leakage corrections are applied to the wakes-off groups used for leakage and other corrections, as well as to the standard, wakes-on, groups

NOTE: Leakage corrections must be derived from wakes-off cases. Requesting the correction, but not supplying the appropriate wakes-off cases, can therefore result in improper corrections.

irest option flag indicating that the hydrostatic at-rest corrections are to be applied to the final results (set irest=0 if they are not subtracted; set **irest=1** if they are subtracted). Hydrostatic forces and moments at zero boat speed are subtracted from the final results. Hydrostatic forces are never subtracted from results for cases at zero boat speed.

For full-model groups, heeled and yawed, the subtracted values are taken to be the first of the following values to be identified:

- zero boat speed, heeled and yawed
- zero boat speed, heeled

For half-model groups, upright, the correction values are taken to be the first of the following values to be identified:

- zero boat speed, upright half-model

At-rest hydrostatic corrections are applied to the wakes-off groups used for leakage and other corrections, as well as to the standard, wakes-on, groups.

isurf option flag indicating that the surfing rudder corrections are to be applied to the final results (set **isurf=0** if they are not; set isurf=1 if they are). Correction forces are intended to be the difference between rudder-off and rudder-on values, both computed at zero yaw, tab and rudder, and wakes-off.

For full-model groups, heeled and yawed, rudder-on, the correction values are taken to be the first of the following values to be identified:

- wakes off, heeled and yawed
- wakes off, heeled
- wakes off, upright
- zero boat speed, heeled
- zero boat speed, upright

Surfing rudder corrections are not applied to half-model, upright results.

NOTE: Surfing rudder corrections must be derived from wakes-off cases. Requesting the correction, but not supplying the appropriate wakes-off cases, can therefore result in improper corrections.

The remainder of the input is necessary only if ivisc≠0, and it is used only if ivisc>0, that is, if the post-process tabtec viscous stripping is requested. The remaining inputs are the name of the directory containing the database surface mesh files (as appropriately pre-processed for use with ACCPAN) and, for each of the hull, keel, bulb, wing, aft rudder and forward rudder components (whether or not the models include each such component), a stripping option flag and the name of the component database surface mesh file. The stripping option flag is an integer value of 0 (no stripping) or a value from 1 to 6 indicating the type of stripping, according to following convention:

- 1 Blasius laminar
- 2 Schlichting turbulent
- 3 Prandtl-Schlichting transitional
- 4 simple transitional
- 5 turbulent-airfoil-type transitional
- 6 laminar-airfoil-type transitional

NOTE: The current version of tabtec requires that keel and rudder database mesh files contain half-model representations of those surfaces. However, the latest versions of ACCPAN require that keel and rudder database mesh files contain full-model representations of those surfaces. The post-process tabtec viscous stripping therefore may not work correctly for keel and rudder databases that have been pre-processed for use with the latest versions of ACCPAN. Instead, the viscous stripping built into ACCPAN can be activated, during the model tests, by setting ivisc=-1 in the ACCPAN input files, and the ACCPAN-generated viscous forces and moments can be included in the final post-processed results, by setting ivisc=-1 in the tabtec input file.