TECHNICAL DOCUMENT 3293
August 2016

FANS-3D User's Guide (ESTEP Project ER-201031)

Pei-Fang Wang
SSC Pacific

Hamn-Ching Chen
Texas A&M University

SSC Pacific
San Diego, CA 92152-5001

SSC Pacific
San Diego, California 92152-5001

K. J. Rothenhaus, CAPT, USN
Commanding Officer

C. A. Keeney
Executive Director

ADMINISTRATIVE INFORMATION

This report was was developed in support of Environmental Security Technology Certification Program (ESTEP) Project ER-201-031 by the Environmental Sciences Branch (Code 71750), of the Advanced Systems and Applied Sciences Division (Code 71700), Space and Naval Warfare Systems Center Pacific (SSC Pacific), San Diego, CA; and the Ocean Engineering Program, Zachry Department of Civil Engineering, Texas A&M University, College Station, TX,

Released by
P. J. Earley, Head
Environmental Sciences Branch

Under authority of
A. J. Ramirez, Head
Advanced Systems & Applied Sciences Division

SB

CONTENTS

Figures

1. INTRODUCTION

This user's guide details the FANS-3D code model and the procedure of execution of the model. This guide was developed in support of Environmental Security Technology Certification Program (ESTEP) Project ER-201-031.

Dr. Hamn-Ching Chen and his students and collaborators developed the FANS-3D code over the past 25 years. Programmers use this general-purpose computational fluid dynamics (CFD) code for solving the Navier–Stokes equations governing laminar and turbulent flows in body-fitted curvilinear grids. The code employs multi-block overset (chimera) grids, including fully matched, arbitrarily embedded, and/or overlapping grids to facilitate detailed resolution of unsteady laminar and turbulent flows around complex geometries involving arbitrary body motions as well as fluid-structure interactions. Communication between grid components is achieved by Lagrange interpolation at the fringes. The code is fully coupled with the hole-making and donor-finding algorithm, allowing for the relative movement of the grid blocks at each time step for time-domain simulation of fluid-structure interaction problems, including violent free surface motions.

The underlying theory of the local-analytic-based discretization (also known as finite analytic based discretization) is briefly presented in Section 1.1. A complete description of the formulation, including the numerical solution of well-established two-dimensional and three-dimensional benchmarks, is documented in Pontaza, Chen, and Reddy (2005). Additional published work on the theory of the discretization method is from Chen and Chen (1984), Chen, Patel, and Ju (1990), and Chen, Bravo, Chen and Xu (1995).

1.1 THEORY AND NUMERICAL ALGORITHM OF FANS CODE

The authors mentioned above developed the finite analytic method for accurate numerical simulation of the time-dependent incompressible Navier–Stokes equations. To briefly describe the formulation, consider a two-dimensional domain partitioned into equal-sized non-overlapping elements, Ω^e. We linearized the Navier–Stokes equations in each element and write:

$$(\vec{U}_0 \bullet \nabla)\vec{U} - \frac{1}{\mathrm{Re}}\nabla^2\vec{U} = \vec{F} - \frac{\partial^h \vec{U}}{\partial t} - \nabla_h P = L(\vec{U}, P) \quad \text{in } \Omega^e, \tag{1}$$

where $\partial^h/\partial t$ is a discrete representation of the temporal operator (e.g., a backward Euler representation) and ∇_h is a discrete gradient operator in space. Momentarily treating $L(\boldsymbol{U},P)$ as known and constant over the element, we see that the linearized momentum equations are non-homogeneous advection-diffusion equations.

Treating each of the momentum equations as a transport equation for the associated velocity component, we use the natural solution of the linearized equation as boundary conditions along the edges of the square element and solve the associated equations by the method of separation of variables to obtain local analytic interpolants in terms of unknown neighboring nodal values of the velocity components. The interpolant may be written as

$$\vec{U} = \sum_{n=1}^{8} \alpha_n \vec{U}^n + \alpha_f L(\vec{U}, P) \text{ in } \Omega^e. \tag{2}$$

The local analytic interpolants $\{\alpha_n\}_{n=1}^{8}$, α_f are functions of the local velocity field and respond analytically to local flow conditions. In addition, the interpolants satisfy zeroth and first-order consistency requirements, and are always positive. These properties ensure that spurious energy

modes are non-existent in the scheme, and render it stable at high Reynolds numbers. Plots of one of the coefficients for different flow conditions in a single element are shown in Figure 1. A more detailed description of the finite analytic functions is given in Pontaza, Chen, and Reddy (2005) and Chen and Chen (1984).

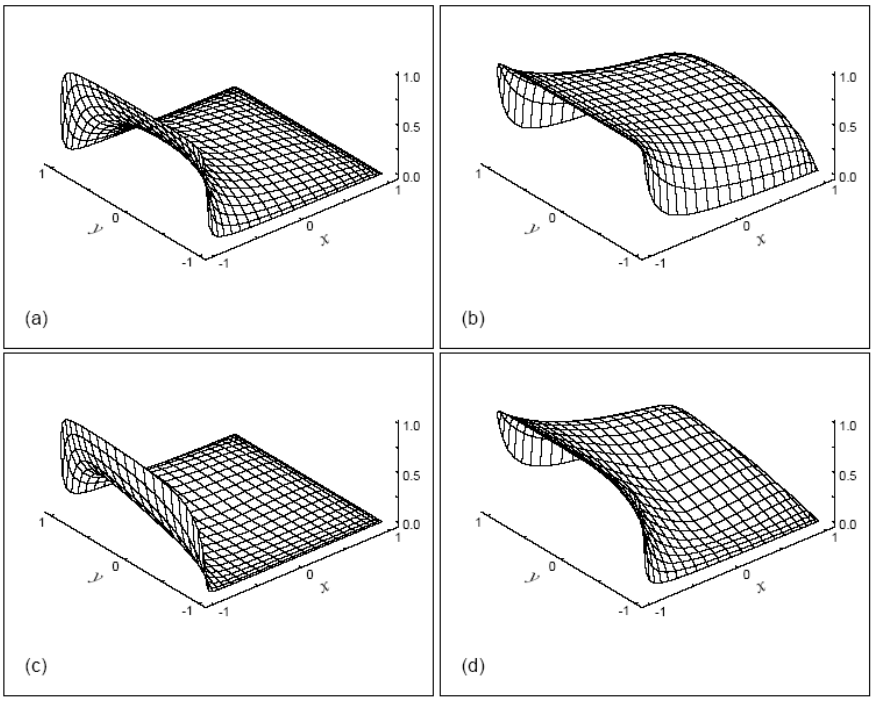

Figure 1. Finite analytic function associated with a node placed a (x,y) = (-1,0), for an element in (x,y) $\in$ [-1,1]× [-1,1], for different flow conditions.

The interpolants satisfy (locally) the linearized momentum equations and a collocation scheme is adopted to form the discrete equations. In other words, the local analytic functions are only evaluated at the center of the element to yield coefficients that make up the stencil relating the center value to its neighbors.

If the pressure field is known a priori, the pressure gradient may be evaluated and a set of discrete equations for each interior nodded can be written using Equation (2). These equations can be assembled to yield a banded, unsymmetrical, definite matrix system. When augmented with suitable boundary conditions, the system can be solved (in an iterative manner for the linearization) to yield the nodal velocity values in a time-marching procedure.

In general, however, the pressure field is not known a priori and must be computed such that the velocity field is divergence-free. This task is achieved by projecting the velocity field onto a divergence-free space through a discrete Poisson equation for the pressure. The discrete representation of the divergence operator is constructed such that a strong velocity-pressure coupling is achieved, effectively avoiding spurious pressure solutions for the co-located node arrangement, where nodal degrees of freedom for velocities and pressure share the same locations. The projection is directly applied to boundaries as well, so that no artificial boundary conditions for the pressure are necessary. Thus, pressure is consistently computed at the boundaries.

The momentum and discrete pressure Poisson equation are solved sequentially in an iterative manner. Pontaza, Chen, and Reddy (2005) showed the method to be second-order accurate in velocities and pressure. Convergence properties of the method are illustrated in Figure 2. When Equation (2) is used as an interpolant, the interpolation is fourth-order accurate, as shown in Figure 2(a). When Equation (2) is used as a collocation discretization procedure, the error decays at a second-order rate, as shown in Figures 2(b) and 2(c) for linear and nonlinear equations. Figure 2(d) shows second-order accuracy in velocities and pressures, indicating good velocity pressure coupling by implementing the segregated solution approach.

In practical implementations, we seldom encounter square domains. The general procedure consists of constructing the local analytic interpolants in a mapped space. Using this approach, we can handle skewed or curvilinear elements with a unified approach. The method has proven robust in the presence of severe mesh skews and high aspect ratio cells (Pontaza, Chen, and Reddy, 2005).

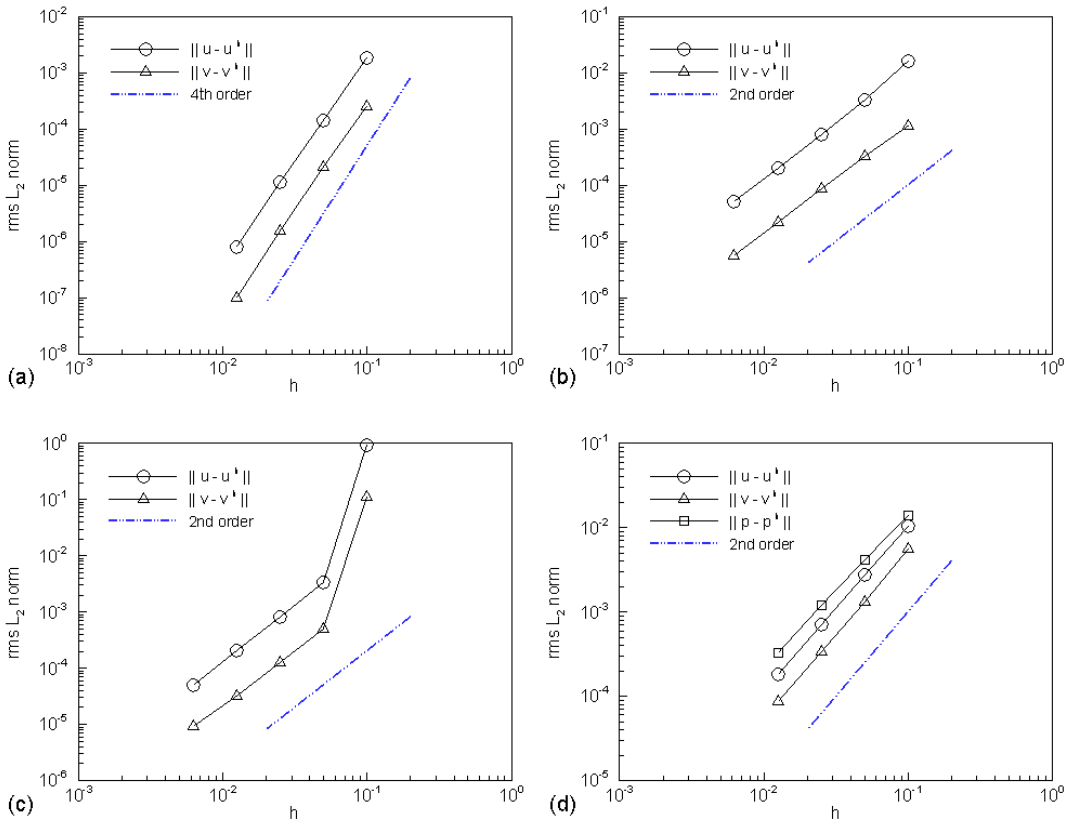

Figure 2. Convergence curves for verification studies of the finite analytic functions as (a) interpolants, and as (b), (c), (d) a collocation discretization procedure.

For time-accurate solutions, the time derivative is represented here by second-order accurate truncated expansions in time domain. Specifically, the time integration scheme corresponds to the generalized α-method family of time integrators. The family is generated by varying a single free-integrator parameter, ρ, for high-frequency damping. Unresolved high frequencies (due to the choice of the time step size) are damped out according to the value of ρ. The choice $\rho = 1.0$, corresponds to the trapezoidal rule, which is known to have no damping for high-frequency modes that may excite odd–even mode oscillations. High-frequency damping is allowed by decreasing the value of ρ.

Additional documentation on this particular family of time integrators is given by Chung and Hulbert (1993) and by Dettmer and Peric (2003). The discrete pressure gradient operator is represented using standard second-order accurate finite differences in each spatial direction.

Extension to the three-dimensional case is straightforward and achieved by superimposing two-dimensional local analytic solutions, such that the three-dimensional equations are satisfied locally. Details of the derivation were first presented by Chen, Patel and Ju (1990) and Chen, Bravo, Chen, and Xu (1995), and are also outlined by Pontaza, Chen, and Reddy (2005). The resulting stencil relates one nodal unknown to its 19 neighbors, and is thus a 19-point finite analytic stencil.

For turbulent flows modeled through the numerical solution of the Reynolds-averaged Navier–Stokes (RANS) equations, see Equations (5) through (11) provided by Wang and Chen (2016) in the technical report published by Space and Naval Warfare Systems Center (SSC Pacific). In the two-layer k-ε model, the k-ε model is patched together with a k-l model used in the near-wall region. Thus, the near-wall region is computed directly and adequate grid resolution must be used there. Additional details can be found in Chen and Patel (1988).

The discretization procedure for the turbulent transport equations is exactly the same used for the momentum equations described earlier, as these equations can always be written in the standard form given by Equation (13). This standard is certainly a major advantage of the formulation, as no special treatment is needed for the turbulence transport equations.

2. FANS-3D SOFTWARE DOCUMENTATION AND EXECUTION

In this current study, the FANS-3D code was employed for the propeller wash simulations of both DDG 51 ship and tugboat cases as described in previous sections. The computer code executables, numerical grids, input files, simulation results, and animation movies for all seven propeller wash scenarios were delivered to Dr. Pei-Fang Wang from SSC Pacific. The deliverables are organized in seven tar (tape archive) files as follows:

1. **ddg51_5kt_33ft.tar.gz**: DDG 51 ship at 5 kts and 33 ft water depth
2. **ddg51_5kt_38ft.tar.gz**: DDG 51 ship at 5 kts and 38 ft water depth
3. **ddg51_10kt_33ft.tar.gz**: DDG 51 ship at 10 kts and 33 ft water depth
4. **ddg51_10kt_38ft.tar.gz**: DDG 51 ship at 10 kts and 33 ft water depth
5. **tugboat_case1.tar.gz**: tugboat scenario 1 with propeller blowing to open water
6. **tugboat_case2.tar.gz**: tugboat scenario 2 with propeller blowing to pier wall
7. **tugboat_case3.tar.gz**: tugboat scenario 3 with propeller blowing parallel to pier wall

Each folder contains the following set of files that must be written by the users:

1. `gridgen0.dat` (or `plot3d0.dat`), this file contains the multi-block numerical grids in either GRIDGEN or PLOT3D format. The file format is given later.
2. `inputblk.dat`, this file assigns a name to each of the computational grid blocks and contains information regarding their size (both active and phantom grids are listed).
3. `inputmpd.dat`, this file contains the multi-processor distribution information.
4. `input.dat`, this is the control program file, where the user may specify, for example, the Reynolds number, the time step size, relaxation factors, etc.
5. `overset.in`, this is the control file for the hole-cutting and donor-searching program
6. `*.bcs`, files containing the boundary condition input for each block in each process, a total of "number of processes" files must be present.

In all FANS-3D simulations, it is necessary to construct first the numerical grid for each test case. The name of the grid file is specified in `inputblk.dat`. The grid file may be written in either GRIDGEN or PLOT3D format as follows:

(A) GRIDGEN format (`iformat` = 1)

```
! read the volume grid from gridgen0.dat file (specified in inputblk.dat)
! each block has size nxi_GL, net_GL, nzt_GL

do nbk_GL=1,nblocks_GL + nphantoms_GL
ijkst_GL=ijkpos_GL(nbk_GL) + 1
ijknd_GL=ijkpos_GL(nbk_GL)+nxi_GL(nbk_GL)*net_GL(nbk_GL)*nzt_GL(nbk_GL)
read(10,*) nbk_dum,nxi_GL(nbk_GL),net_GL(nbk_GL),nzt_GL(nbk,GL)
read(10,*) (xref_GL(ijk_GL),ijk_GL=ijkst_GL,ijknd_GL), &
          (yref_GL(ijk_GL),ijk_GL=ijkst_GL,ijknd_GL), &
      (zref_GL(ijk_GL),ijk_GL=ijkst_GL,ijknd_GL)
end do
```

(B) PLOT3D format (`iformat` = 2)

```
! read the volume grid from plot3d0.dat file (specified in inputblk.dat)
! each block has size nxi_GL, net_GL, nzt_GL
```

```
read(10,*) ndum
do nbk_GL=1,nblocks_GL + nphantoms_GL
read(10,*) nxi_GL(nbk_GL),net_GL(nbk_GL),nzt_GL(nbk_GL)
end do
do nbk_GL=1,nblocks_GL + nphantoms_GL
ijkst_GL=ijkpos_GL(nbk_GL) + 1
ijknd_GL=ijkpos_GL(nbk_GL)+nxi_GL(nbk_GL)*net_GL(nbk_GL)*nzt_GL(nbk_GL)
read(10,*) (xref_GL(ijk_GL),ijk_GL=ijkst_GL,ijknd_GL), &
          (yref_GL(ijk_GL),ijk_GL=ijkst_GL,ijknd_GL), &
      (zref_GL(ijk_GL),ijk_GL=ijkst_GL,ijknd_GL)
end do
```

As the simulation progresses and the grids move and rotate with respect to one another, the grid motions (e.g., ship motion and propeller rotation) are updated based on the reference configuration in `gridgen0.dat` (or `plot3d0.dat`). In the above pseudo-code statements `nblocks_GL` and `nphantoms_GL` are the number of active (computational) blocks and the number of phantom blocks, respectively; which were already read from `inputblk.dat`. More details of the input files and their contents will be given in the following sections, in the context of the example problems.

3. FANS-3D CODE PARALLELIZATION

The FANS-3D code is a general-purpose CFD code allowing for the numerical solution of the Navier–Stokes equations governing incompressible flow in body-fitted grids. The code allows for multi-block overset (chimera) grids, which can be fully matched, arbitrarily embedded, and/or overlapping with each other. Communication between grid components is achieved by Lagrange interpolation at the fringes. The code is fully coupled with the hole-making and donor-finding algorithm, allowing for the relative movement of the grid blocks at each time step for time-domain simulation of fluid-structure interaction problems including violent free surface motions.

The FANS-3D code is written in Fortran 90/95 standard with dynamic memory allocation and is fully parallelized using Message-Passing-Interface (MPI) bindings. It employs a general data management strategy that allows single or arbitrarily large groups of consecutive or non-consecutive blocks to be assigned to different processors. This strategy enables us to achieve optimal load balancing when dealing with multi-block structured grids with vastly different dimensions among different grid blocks as shown below.

Given a multiple block structured grid with N blocks of different sizes, we would like to distribute the workload amongst P processes. For example, consider the case N = 7, as shown in Figure 3.

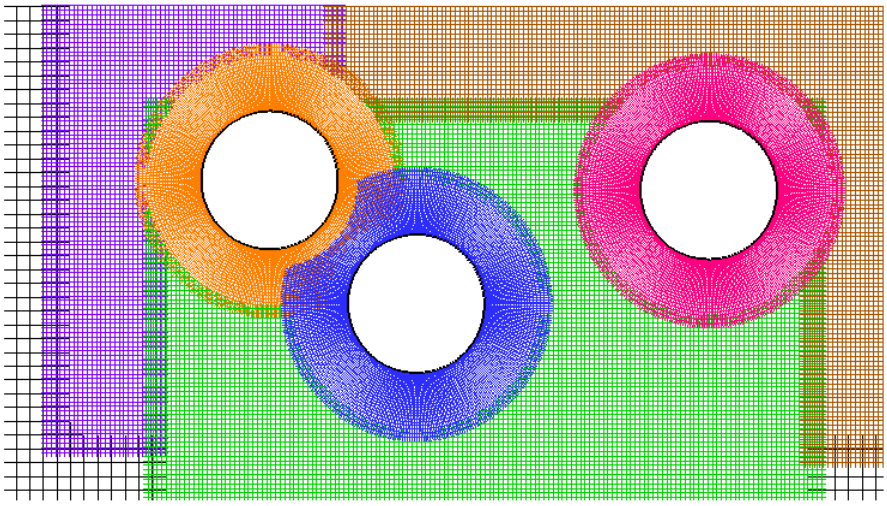

Figure 3. Multiple block structured grid showing N = 7 blocks, which are to be distributed among P ≤ 7 processes.

The minimum number of processes allowed in the parallelized code is P = 2, and the maximum for this case would be P = 7, which would imply that each block is assigned to a single process. Having observed the above constraints, the code allows the user to distribute the load in any other manner. Below are some examples (by no means exhaustive) of valid load distributions, where we fix the number of available processors:

Example #1:
P=2
P1:{1,2,3} and P2:{4,5,6,7}

In this example, process 1 is assigned blocks {1,2,3} and process 2 is assigned blocks {4,5,6,7}.

Example #2:
P=3
P1:{1,4}, P2:{2,5}, and P3:{3,6,7}

In this example, non-consecutive numbered blocks are assigned to different processors. This is particularly advantageous, as the user need not order the blocks in any particular manner during and after the grid generation process.

The load distributions should be such that the load is almost the same amongst all processes. This is not a requirement in the code, but is recommended to make efficient use of the computational resources.

The information on load distribution is read in through the file `inputmpd.dat`, and is as follows for example #1 and #2, respectively.

Example #1:

```
3 4          % blocks per process for each process

1 2 3        % global block numbers for each process
4 5 6 7
```

Example #2:

```
2 2 3        % blocks per process for each process

1 4          % global block numbers for each process
2 5
3 6 7
```

The above input is all that is needed by the code for it to understand and schedule the loads among the different processes. In addition, each processes expects one boundary condition file, containing boundary condition information for all the blocks it was assigned. The format of the boundary condition file is discussed in Appendix A.

4. COMPUTER PLATFORMS, COMPILATION, AND EXECUTION

The FANS-3D code was tested on platforms with Linux® as the operating system, with Intel® Fortran 90/95 compilers and Message Passing Interface Chameleon (MPICH) implementations. Specifically, in the Dell® clusters at Texas A&M Civil Engineering Department, IBM® clusters at Texas A&M Supercomputing Facility, Linux® clusters at U.S. Army Research Laboratory (ARL) High Performance Computing cluster, and the Cray XE6m (Copper) cluster at Department of Defense High Performance Computing Modernization Program (DoD HPCMP). The FANS code and executable can be installed on a wide variety of Unix and Linux® clusters with Message-Passing-Interface (MPI) libraries for parallel computations using multiple processors. For simplicity, we will summarize only the procedures to compile and execute the code on the Copper cluster at DoD Open Research Systems in the following sections.

The FANS-3D code consists of 18 Fortran 90 files, each with a specific function. A list of the files accompanied with a brief description is as follows:

- `main.f90` is the master control file from which all other subroutines are called. The program follows a modular-style programming by making use of Fortran 90 modules, which are invoked and used in this file.
- `global.f90` is where all global variables are defined.
- `sflow.f90` defines flow parameter variables such as the turbulence model coefficients.
- `sinput.f90` reads-in all the program control inputs, allocates memory, and distributes the load among processors.
- `geocoeff.f90` computes and stores the geometric coefficients associated with a well-defined transformation.
- `facoeff.f90` computes the 19-point stencil finite-analytic coefficients.
- `moment.f90` solves the moment equations for the velocity components.
- `pressub.f90` computes the 19-point stencil for the pressure Poisson equation, assembles and solves the associated system of equations.
- `turbsub.f90` solves the turbulence model equations.
- `sources.f90` computes the source functions for the governing equations.
- `boundary.f90` computes and assigns boundary conditions
- `snorms.f90` computes various metrics, such as residual norms, outer iteration norms, time stepping norms, to establish convergence of the iterative solution procedure and time marching procedure.
- `gmotions.f90` grid motions file to control and impose how the grids move relative to each other and compute the grid velocities.
- `datamgmt.f90` contains the subroutines for the multi-block data management.
- `graphics.f90` generates output files for visualization.
- `sclean.f90` deallocates memory.
- `overset.f90` grid interpolation program for overset grids.
- `dwssub.f90` computes directional short-crested waves.

The code is to be compiled by linking the Fortran 90/95 compiler with a MPI library or by using a Fortran 90/95 MPI wrapper (e.g., `mpiifort, mpif90 or ftn`). When using MPI as a library, the following is used to compile the code on DoD HPCMP Copper cluster:

```
prompt%> module swap PrgEnv-pgi PrgEnv-intel
prompt%> ftn –openmp –O2 –o fans3d.exe {list of Fortran files}
```

The code is simply run by typing the following at the prompt or giving the following command in the batch-job file (e.g., for PBS or LSF queue managers):

```
prompt%> aprun –n {number of processors} ./fans3d.exe > fans3d.out
```

Note that the simulation results for all seven propeller wash scenarios described earlier can be reproduced by uploading the corresponding tar files to DoD HPCMP Copper cluster and executing the following four commands (using **`tugboat_case3.tar.gz`** as an example):

1.0 Unzip **`*.tar.gz`** file. The code executable and input data files will be saved in a newly created folder `tugboat_case3`

```
Prompt%> tar xzf tugboat_case3.tar.gz
```

2.0 Change programming environment from the default 'pgi' to 'intel' Fortran

```
Prompt%> module swap PrgEnv-pgi PrgEnv-intel
```

3.0 Switch to working directory

```
Prompt%> cd tugboat_case3
```

4.0 Submit job to the batch queue (with appropriate project number in the job control file)

```
Prompt%> qsub submit_pbs
```

5. FANS-3D DATA EXPORT

On FANS-3D output, the following files are written out to visualize the solution using the commercial flow visualization software such as FieldView, Tecplot®, or MATLAB®:

1. `force.dat`, x, y, and z forces exerted on the propeller blades, ship hull surface, and/or other solid surfaces.
2. `motion.dat`, time history of six-degree-of-freedom ship motion
3. `overset.out`, output file containing grid interpolation information.
4. `fans3d.out`, output file for monitoring of convergence history.
5. `restart_xyz.dat`, instantaneous grid restart file for continuation run
6. `restart_q{number}.dat`, instantaneous flow field restart file for continuation run.
7. `movie_x{number}.dat`, three-dimensional output to visualize the entire grid at time step `{number}`.
8. `movie_q(number}.dat`, three-dimensional output to visualize instantaneous velocity and pressure fields at time step `{number}`.

The `force.dat` contains ASCII data files in column format. It can be read directly into Tecplot® or MATLAB® or other compatible software for 2D line plots of the (x, y, z) forces and moments (with respect to the gravity or center of rotation). For a problem involving six degrees-of-freedom (heave, sway, surge, pitch, yaw, and roll) motions under hydrodynamic loadings, such as wave and current, the code will also output the motion histories in motion.dat file, which is also in ASCII column data format.

The `overset.out` is an ASCII file containing grid interpolation information such as interpolation stencils and interpolation coefficients for the multi-block overset grid system. The `fans3d.out` is also in ASCII format. It is used to monitor the convergence histories of all flow variables. These files are useful for debugging of the input data files.

The restart files `restart_xyz.dat` and `restart_q*.dat` are unformatted files which are used internally by the FANS-3D code for continuation runs. The code will automatically read in the restart files if the users wish to continue a previous simulation for a longer duration.

The `movie_x*.dat` and `movie_q*.dat` output files were written in standard PLOT3D format as follows:

```
! PLOT3D grid output (movie_x{number}.dat) for flow visualization
write(54) nblocks_GL
write(54)
((nxi_GL(nbk_GL),net_GL(nbk_GL),nzt_GL(nbk_GL)),nbk_GL=1,nblocks_GL)

do nbk_GL=1,nblocks_GL
  ijkst=ijkpos_GL(nbk_GL)+1
  ijknd=ijkpos_GL(nbk_GL)+nxi_GL(nbk_GL)*net_GL(nbk_GL)*nzt_GL(nbk_GL)
  write(54)(xp(ijk),ijk=ijkst,ijknd), &
      (yp(ijk),ijk=ijkst,ijknd), &
      (zp(ijk),ijk=ijkst,ijknd), &
      (iblank(ijk),ijk=ijkst,ijknd)
end do
```

```
! PLOT3D flow output (movie_q{number}.dat) for flow visualization
write(55) nblocks_GL
write(55)
((nxi_GL(nbk_GL),net_GL(nbk_GL),nzt_GL(nbk_GL)),nbk_GL=1,nblocks_GL)

do nbk_GL=1,nblocks_GL
  ijkst=ijkpos_GL(nbk_GL)+1
  ijknd=ijkpos_GL(nbk_GL)+nxi_GL(nbk_GL)*net_GL(nbk_GL)*nzt_GL(nbk_GL)
  write(55) alpha,fsmach,reynolds,time
  write(55)(rho(ijk),ijk=ijkst,ijknd), &
      (rho(ijk)*u(ijk),ijk=ijkst,ijknd), &
      (rho(ijk)*v(ijk),ijk=ijkst,ijknd), &
      (rho(ijk)*w(ijk),ijk=ijkst,ijknd), &
      (pr(ijk),ijk=ijkst,ijknd)
end do
```

The PLOT3D grid output files (`movie_x*.dat`) contain the coordinates (x, y, z) and blanking information (iblank) for every grid point in the multi-block overset grid system. The corresponding flow variables, including density, momentum, and pressure (ρ, ρu, ρv, ρw, p) are stored in PLOT3D output files (`movie_q*.dat`). These data files can be imported directly into the commercial software FieldView for flow visualization and saved in animation video files (in avi format). The movie data files can also be imported into the commercial Tecplot® software using the 'PLOT3D Loader' option. Typical results include the velocity contours, velocity vector plots, and pressure contours. Other quantities such as shear stresses and vorticities can also be calculated using the user-defined functions in FieldView and Tecplot®. The users may consult the FieldView and Tec)Plot® manuals for additional information on the post-processing of PLOT3D data.

6. EXAMPLE CASE 1
DDG 51 SHIP AND P4876 PROPELLER WASH STUDY

In this section, we present an example test case for the DDG 51 propeller wash study. The problem demonstrates the many capabilities of the FANS-3D formulation and implementation, which include: embedded and non-matching grids, relative motion between grid components, load distribution among different processes, high Reynolds number flows, and robustness in the presence of high aspect ratio skewed meshes.

Chen and Wang (2016) show the computational domain and multi-block overset grids for this case in Figure 2 of the SSC Pacific technical report. The length of the DDG 51 ship is 142.04 m (466 ft) and the designed draft is 9.4488 m. The diameter of the twin-screw P4876 propellers is 5.4864 m (18 ft), and the center of propeller axis is located at 5.7912 m below the mean water level. We performed a calculation for a shallow water case with water depth H = 10.0584 m (33 ft). Under this condition, the underkeel clearance is only 0.6096 m (2 ft) beneath the sonar dome and the minimum gap between the propeller tip and the sea bottom is 1.524 m (5 ft). The twin-screw propellers are rotating at 51 rpm when the ship speed is 10 kts.

A commercial grid generation software Gridgen® was used to generate the overset grid system for the DDG 51 ship and the five-blade P4876 propeller. As noted earlier, the composite grid consists of 15 computational blocks and 7 phantom grid blocks with 2,369,549 grid points covering half of the solution domain. There are five blocks for five propeller blades, three blocks for propeller shaft and near-wake regions, one block for the ship, and six blocks for the far field. The 15 blocks are shown in different colors in Figure 2 of the SSC Pacific technical report. In addition, seven phantom grids (not shown) are needed to perform the hole-cutting adequately. The end-user does not need to be concerned with phantom grids, as they do not enter into the actual computations, and hence do not need to be listed in the multi-processor input file or the boundary condition input files.

In this particular run, the five propeller blades, the shaft block, are assigned to three processes, the ship is assigned to the fourth process, the propeller near-wake region is divided into two blocks and assigned to two separate processes, and the far-field grids are decomposed into six blocks and assigned to six different processes. For this example, the file `inputblk.dat` contains the following data:

```
! Geometry input file (second line, no more than 40 characters)
gridgen0.dat
 1        ! 1: Gridgen format, 2: Plot3d format
 15  7      ! nblocks + nphantom (including phantom grid)

 62 41 41
propeller01

 62 41 41
propeller02

 62 41 41
propeller03

 62 41 41
propeller04

 62 41 41
```

```
propeller05

 38 21 122
shaft01

 28 32 122
shaft02

 65 21 122
shaft03

121 35 41
ship01

 34 81 77
basin01

 34 81 77
basin02

 34 81 77
basin03

152 65 21
ocean01

 77 65 42
ocean02

 77 65 42
ocean03

 3 41 61
phantom01

 3 41 61
phantom02

 3 41 61
phantom03

 3 41 61
phantom04

 3 41 61
phantom05

 2  2  2
phantom06

 2  2  2
phantom07
```

This input specifies that the name of the composite grid file is `gridgen0.dat`, and it is in Gridgen® format. There are 15 computational blocks and 7 phantom blocks (22 blocks total). Then, for each of the 15 computational blocks, we must specify their (i, j, k) sizes and assign to a name to

them that must be consistent with the names used in the `overset.in` input for the hole-cutting and donor-search algorithm.

The file `overset.in` contains the input necessary for the hole-cutting and donor-search program. The format of this file is not discussed here, and the interested reader may consult the Chimera Overset Structured Mesh-Interpolation Code (COSMIC) Users' Manual (Chen, 2009). The input file used for this case is shown in Appendix B.

The file `inputmpd.dat` contains the information necessary for the code to distribute the load among the different processes, as described in the previous section. For this particular case, the file has the following information:

```
% number of blocks per process, for each process
2 2 2 1 1 1 1 1 1 1 1

 1 2     % global block number per process, for each process
 3 4
 5 6
 7
 8
 9
10
11
12
13
14
15
```

Note that only active (computational) blocks are listed in this input, i.e., phantom blocks do not need to be distributed as they do not represent any computational load. In this particular case, we assign propeller blades 1 and 2 (`propeller01, propeller02`) to first process, blades 3 and 4 (`propeller03, propeller04`) to the second process, and blade 5 and the first shaft block (`propeller05, shaft01`) to the third process. The remaining nine computational blocks (two shaft blocks, one ship block, three basin blocks, and three ocean blocks) are assigned to processes #4–#12 with only one single block in each process.

The `input.dat` file is the main control input file and is as follows:

```
1      % MTURB   flag for laminar (0) or turbulent (1) flow
1      % INCOMP   flag for incompressible (1) or compressible (0) flow
0      % IFSURF   flag for (1) free surface flow (0) no free surface
2.1868E7  % RE    Reynolds number
0.04    % TAU    time step size
0.0     % AMP_RHO  frequency damping parameter: 0.0 <= AMP_RHO <= 1.0
1.0E-08   % TOL1    L2 vel tol to stop time stepping
1.0E-03   % TOL2    L1 res tol to stop outer iterations
1      % ITIMEST  starting time step to compute
12500    % ITIMENDending time step to compute
1      % MAXIT_LS  max allowable ADI sweeps for level-set function
3      % MAXITER  max allowable outer iterations
2      % MAXSWP_U  max allowable number of momentum eqns ADI sweeps
2      % MAXSWP_PR max allowable number of pressure eqn ADI/SIP sweeps
2      % MAXSWP_KE max allowable number of k-epsilon eqns ADI sweeps
6      % MAXIT_DIVU max projections of velocity field onto div-free space
0.60     % RFU   relaxation factor for velocities (due to nonlinearity)
0.30     % RFP   relaxation factor for pressure (due to u-p decoupling)
0.010    % RFKE  relaxation factor for turbulent k.e. and dissipation
0.5     % RFPHI  relaxation factor for level-set function
0      % ITIME_BCS flag to indicate (1) time dependent bcs

fans.grd   % GEOFILE geometry input file (HCC: not used in this version)
ddg.bcs   % BCSFILE  boundary conditions input file

1      % IACT_PLOT flag to activate (1) visualization output
100     % ISKP_PLOT time intervals for vis and restart

1      % IACT_ANIME flag to activate (1) animation output
0      % IBGN_ANIME time step number at which animation begins
4      % ISKP_ANIME multiples at which sol is written out for animation

0      % ISOL_PR  (0)TDMA-ADI, (1)SIP-7pt solver for pressure eqn

-5.0     % UMIN
 5.0     % UMAX
-10.     % PMIN
 10.     % PMAX
 0.050    % TKEMAX
 0.02    % TVISMAX
-0.5     % PHILSMIN
 0.5     % PHILSMAX

1.0     % FROUDE  Froude number (gravity acts in negative z-direction)
0.0020    % EWIDE   representative grid size

0.0     % RFG    geometry distortion relaxation parameter

0.0 0.0 0.0 % UINF, VINF, WINF  inflow velocities
-1.103143 0. 0. % UBODY, VBODY, WBODY  body velocities (ship speed)

1      % IMOVE   (0) fixed grid, (1) moving grid

0 0 0   % NBODY (#bodies), NFBODY (#surfaces), MBLK (#blocks for 6-dof)
```

```
1  12  % NPROP (#propellers), NMVPROP (max #blocks moving with propeller)

1  1  % IPROP (propeller ID), IROT (1: clockwise, -1: counter-clockwise)
12  6  % IMVPROP (no. of moving propeller blocks), ISHAFT (shaft ID)
1 2 3 4 5 6 7 16 17 18 19 20 % moving propeller block ID (propeller #1)
```

Most of the entries of the input file are self-descriptive, but we further elaborate on each of them in the following:

- `MTURB` is a flag to specify whether to numerically solve the Navier–Stokes equations directly (laminar flow or DNS) or to solve the Reynolds-averaged Navier–Stokes equations with the near-wall two-layer k-epsilon model.
- `INCOMP` is a flag to specify whether flow is incompressible or compressible.
- `IFSURF` is a flag to specify whether it is necessary to update free surface. For this case, the free surface effect is ignored.
- `RE` is the Reynolds number. For this case, it is based on the propeller diameter and the propeller rotating speed.
- `TAU` is the value for the time step size. In this case the angular velocity of the propeller is such that one revolution is completed in one unit of time.
- `AMP_RHO` is the high-frequency damping parameter for the second-order accurate family of time integrators, as described in the formulation section.
- `TOL1 is` tolerance of L2 velocity-norm to stop time stepping when the steady state is reached.
- `TOL2` is tolerance of L1 velocity-residuals to stop outer iterations within each time step.
- `ITEMEST` is the starting time step of the computation. A value of 1 is specified for new runs. If the value is greater than 1, the code will read-in restart files from previous runs and continue the computation to the new ending time step.
- `ITEMEND` is the ending time step the user wishes to compute, for the previously specified value of the time step size.
- `MAXIT_LS` is the maximum allowable Alternating-Directional-Implicit (ADI) sweeps for the level-set function.
- `MAXITER` is the maximum allowable number of outer iterations on a given time step. For time accurate solutions this value must be greater than one, to allow for good velocity-pressure coupling and hence time accuracy of the flow field.
- `MAXSWP_U, MAXSWP_P, MAXSWP_KE` is the maximum allowable number of inner iterations on a given outer iteration, to iteratively solve the momentum, pressure, and turbulence transport equations, respectively.
- `MAXIT_DIVU` is the maximum allowable number of projections of the velocity field onto a divergence-free space on a given outer iteration.
- `RFU` is the relaxation factor for the velocity field. The optimal values lie in the range [0.4, 1.0], although lower values may be needed for complex problems.
- `RFP` is the relaxation factor for the pressure field. Also, in accordance with well-established practices, we find that optimal values lie in the range [0.1, 0.8], although higher values may also be used and lower values may also be needed.
- `RFKE` is the relaxation factors for the turbulent transport variables. We find that typically optimal values lie in the range [0.01, 0.5]. Although lower values may be needed.

- RFPHI is the relaxation factors for the level-set function. We find that typically optimal values lie in the range [0.2, 1.0].
- BCSFILE is a string specifying the name of the family of boundary conditions files. The family must have "number of processors" members. In this particular case, since there are 12 processes involved, we must have 12 files (ddg00.bcs ~ ddg11.bcs) ready.
- GEOFILE is a string specifying the name of the grid file. It is not needed in this version since the grid name has already been specified in inputmpd.dat.
- IACT_PLOT, ISKP_PLOT are control flags to activate the output and to control how frequently the output files are updated. The output is in PLOT3D format for visualization using commercial software such as FieldView, Tecplot® or other compatible flow visualization tools.
- IACT_ANIME, IBGN_ANIME, ISKP_ANIME are control flags to write out a movie, which is to be processed by the software FieldView. For this case, the movie corresponds to grid coordinates and grid blanking values, density, velocity vectors, and pressure on every point in the flow field.
- ISOL_PR is a flag for pressure solvers. The pressure can be solved using either tridiagonal matrix algorithm (TDMA-ADI) or strongly-implicit method (SIP-7pt).
- UMIN, UMAX, PMIN, PMAX, TKEMAX, TVISMAX are limiters on the velocity, pressure, and turbulent transport variables. These are set to high values, and are just a safeguard against a poor initial guess, which may cause the fields to oscillate violently in the initial stages of the iterations.
- PHILSMIN, PHILSMAX are limiters for the level-set function. They are set to high values, and are just a safeguard against a poor initial guess, which may cause the fields to oscillate violently in the initial stages of the iterations.
- EWIDE is a representative grid size used specifying the transitional zone thickness adjacent to the air-water interface.
- RFG is a geometry distortion parameter. The default value is 1 for orthogonal or nearly-orthogonal grids, but may be reduced to improve convergence for highly-skewed grids. The relaxation parameter does not affect accuracy for orthogonal grids and has negligible effects for nearly orthogonal grids.
- UINF, VINF, WINF are the values of (x,y,z) components of the free-stream velocity. For this case, the ambient current velocity is zero.
- UBODY, VBODY, WBODY are the values of (x,y,z) components of the body velocity (i.e., ship speed) normalized by the characteristic velocity nD, when n is the propeller rotating speed and D is the propeller diameter. For this case, the ship is traveling in negative-x direction with a normalized speed equals to the propeller advance coefficient J=V/nD.
- IMOVE is a flag for grid motion. A value of 0 is specified for fixed grid system. The value is set to 1 for moving grid in this case since the ship is moving at constant forward speed and the propeller is also turning.
- NBODY, NFBODY, MBLK are the number of body for force/moment integration, maximum number of surfaces for force/integration, and the maximum number of blocks with six-degree-of-freedom motions. These parameters are not needed for the propeller wash study considered in the present study.
- NPROP is the number of propellers. A value of 1 is specified for single-screw propeller. The value is set to 2 for twin-screw propellers. In this case, we set NPROP = 1 since the computation was performed for only one-half of the solution domain. A value of 2 should be specified for fully domain calculations involving twin-screw propellers.

- NMVPROP is the maximum number of grid blocks rotating with any propellers.
- IPROP is the propeller ID. For twin-screw propellers, the propeller rotating directions and the computational grid blocks associated with each propeller can be defined separately.
- IROT is a flag specifying the propeller rotating direction. A value of 1 indicates that the propeller is rotating in clockwise direction. For counter-rotating twin-screw propellers, it is convenient to straightforward to specify IROT = −1 for the second propeller rotating in counter-clockwise direction. Also, it is convenient to change the signs of IROT if the same propellers are under the crash-astern condition.
- IMVPROP is the total number of grid blocks (including phantom grids) rotating with a given propeller. In this case, there are 5 propeller blade blocks, 2 shaft/hub blocks, and 5 phantom blocks (one for each propeller blade) rotating with each propeller. The identification numbers of the rotating grid blocks are specified in the next line.
- ISHAFT is the block identification number of the shaft grid block. This allows the users to specify the center or rotation for each propeller.

Since the workload is distributed to 12 processors for parallel execution, it is necessary to write 12 separate boundary condition files (ddg00.bcs~ddg11.bcs) which are included in the **ddg51_10kt_33ft.tar.gz** for the present case. The boundary condition files follow the format outlined in Appendix A. Below we discuss, as an example, the boundary conditions specified for one of the propeller blades.

```
propeller01                    % global block #1
  1  6  62  41  41  0          % mb,nfabcs,ni,nj,nk

  2                       % No. of two-layer regions
  1  1  3  1  62  1  21  1  41 % nreg,iedy,idist,(i,j,k)
  2  2  0  1  62  20  41  1  41 % nreg,iedy,idist,(i,j,k)

  1                       % No. of free surface regions
  1  3  1  62  1  41  1  41   % nLSreg, nLS, (i,j,k)

  1  1                    % Face #1
  4  4  4  4  4  4  4       % (u,v,w,p,k,epsilon,phiLS)
  1  41  1  41             % (i,j,k) range
  2  1                    % Face #2
  4  4  4  4  4  4  4
  1  41  1  41
  3  1                    % Face #3
  9  9  9  2  1  3  3
  1  62  1  41
  4  1                    % Face #4
  4  4  4  4  4  4  4
  1  62  1  41
  5  1                    % Face #5
  9  9  9  2  1  3  3
  1  62  1  41
  6  1                    % Face #6
  11  11  11  11  11  11  11
  1  62  1  41
  14  2  64  41             % nbk_GL,i,j,k for pressure datum
```

The first line lists the `blockname` for block identification. The second line specifies that this is block #1 for the given process, it has six faces with boundary conditions, and the (i, j, k) dimensions of the block are 62 × 41 × 41. The fourth line indicates that two regions need to be identified to apply the near-wall two-layer k-epsilon turbulence model. The following two lines specify the two-layer model types (`iedy`, near-wall or outer), the identification (`idist`) of each wall boundary, and the (i, j, k) range of the specific region. The eighth line indicates that only one region exists for the level-set function (phiLS) specification. The free surface solver option (`nLS`) and the (i, j, k) range for that region are specified in the next line. These free surface boundary conditions are not used in this case since the free surface effect is neglected with the flag `IFSURF = 0`.

Then, for each face, we read the face number and the number of sections in the face. For each section on a given face, we read 7 boundary conditions associated with each of the 7 field variables: (u, v, w, p, k, ε, phiLS), and the surface limits on that face.

Faces #1 and #2 (i-min and i-max, respectively) of the blade block receive interpolation information, and thus all the field variables have boundary condition #4. Face #3 (j-min) is the solid-surface of the blade, for which (u, v, w) are assigned the grid velocity due to the rotation of the blade, p is linearly extrapolated, turbulent kinetic energy is zero on the wall, and the Neumann boundary conditions are used for turbulent energy dissipation and level-set function. Face #4 (j-max) receives interpolation information and all field variables have boundary condition #4. Face #5 (k-min) is part of the shaft's solid-surface and its boundary conditions are identical to those on face #3 for a solid-wall. Face #6 (k-max) is a branch cut around the blade tip where the flow variables are updated by averaging the adjacent nodal values on either side of the branch cut plane.

For this example run, the flow field is initialized with calm water condition and the propeller is allowed to rotate for 100 revolutions. The ship travels at a constant forward speed of 10 kts and the propeller rotating speed is 51 rpm (0.85 rps). The flow conditions correspond to an advance coefficient J=1.103 and a Reynolds number of 2.1868×10^7 based on the propeller diameter. This corresponds to a Reynolds number of 6.245×10^8 based on the ship length and ship speed. The RANS equations are solved with the near-wall two-layer k-ε turbulence model.

As noted earlier, the PLOT3D grid output files (`movie_x*.dat`) contain (x, y, z, iblank) for the multi-block overset grids, while the PLOT3D flow output files (`movie_q*.dat`) contain the flow variables (ρ, ρu, ρv, ρw, p). These data files can be imported directly into FieldView for flow visualization and post-processing. Typical results include the velocity contours, velocity vector plots, and pressure contours as shown in Figures 2 through 7. Other derived quantities such as shear stresses and vorticities can also be calculated using the user-defined functions in FieldView. Figure 8 shows the shear stresses on the sea bed which can be readily obtained by evaluating the velocity gradients adjacent to the bottom boundary using the following formula:

$$\tau = \mu \frac{\partial q}{\partial n} = \mu \left. \frac{\Delta q}{\Delta n} \right|_{wall}, \quad q = \sqrt{u^2 + v^2 + w^2}, \tag{14}$$

where q is the velocity magnitude, μ is the dynamic viscosity of the seawater, and Δn is the normal distance from the wall. The same shear stress data can also be plotted using another commercial code MATLAB®, as shown in Figure 9.

7. EXAMPLE CASE 2
TUGBOAT AND DUCTED PROPELLER WASH STUDY

In the second test case, we consider a tugboat boat with two ducted propellers under bollard-pull condition (i.e., zero forward speed), as shown in Figure 11 of the SSC Pacific technical report and Figures 1 and 2 in this document. The composite grid was generated by the commercial grid generation software Gridgen®. It consists of 47 computational blocks and 9 phantom grid blocks, with 7,070,832 total grid points. There are four blades for each propeller, and each blade is divided into two overlapping computational blocks. Each ducted propeller assembly is surrounded by five computational blocks covering the upstream, downstream, inner, and outer regions between the propeller shaft and the shroud. In addition, two near-wake cylindrical grid blocks (one for each propeller) are added to provide accurate resolution of the propeller wake flows. The tugboat is surrounded by a single boundary-fitted grid block, and the far field is covered by 18 overlapping rectangular grid points. A near-wall spacing of 10^{-6} ft was used near the sea bottom to provide accurate resolution of the turbulent boundary layer flow. This allows us to calculate the shear stresses on the seabed directly without relying on the wall-function approximations.

The composite grid load is now distributed among 35 processes and we consider the bollard-pull (zero tugboat speed) condition with the ducted propellers blowing parallel to a pier wall. The file `inputblk.dat` contains the following data:

```
! Geometry input file (second line, no more than 40 characters)
gridgen0.dat
1         ! 1: Gridgen format, 2: Plot3d format
 47  9      ! nblock + nphantom0 (including phantom0 grid)

 62 35 42
propeller01a

 62 35 42
propeller02a

 62 35 42
propeller03a

 62 35 42
propeller04a

 29  4  5
tip01a

 29  4  5
tip02a

 29  4  5
tip03a

 29  4  5
tip04a

 21 57 122
duct01a

 66 24 122
```

```
duct02a

 53 35 122
duct03a

 53 23 122
duct04a

 21 79 122
duct05a

 40 41 122
wake01a

 62 35 42
propeller01b

 62 35 42
propeller02b

 62 35 42
propeller03b

 62 35 42
propeller04b

 29  4  5
tip01b

 29  4  5
tip02b

 29  4  5
tip03b

 29  4  5
tip04b

 21 57 122
duct01b

 66 24 122
duct02b

 53 35 122
duct03b

 53 23 122
duct04b

 21 79 122
duct05b

 40 41 122
wake01b

 20 116 95
```

```
ocean01

 20 116 95
ocean02

 20 116 95
ocean03

 20 116 95
ocean04

 20 116 95
ocean05

 20 116 95
ocean06

 20 116 95
ocean07

 20 116 95
ocean08

 20 116 95
ocean09

 20 116 95
ocean10

 20 116 95
ocean11

 20 116 95
ocean12

 20 116 95
ocean13

 20 116 95
ocean14

 20 116 95
ocean15

 16 116 95
ocean16

151 37 33
ocean17

151 37 33
ocean18

107 34 61
barge01

 3 31 41
```

```
phantom01a

 3 31 41
phantom02a

 3 31 41
phantom03a

 3 31 41
phantom04a

 3 31 41
phantom01b

 3 31 41
phantom02b

 3 31 41
phantom03b

 3 31 41
phantom04b

 2  3  2
phantom05
```

The file `inputmpd.dat` contains the information necessary for the code to distribute the load among the 35 different processes. For this particular case, the file has the following information:

```
% number of blocks per process, for each process (excluding phantom grids)
4 4 1 1 1 1 1 1 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 1 5 2 6   % global block number per process, for each process
 3 7 4 8
 9
10
11
12
13
14
15 19 16 20
17 21 18 22
23
24
25
26
27
28
29
30
31
32
33
34
35
```

```
36
37
38
39
40
41
42
43
44
45
46
47
```

Once again, the only active (computational) blocks are listed in this input, i.e., phantom blocks do not need to be distributed as they do not represent any computational load. In this particular case, there are 47 actives blocks, including 14 blocks for each ducted propeller assembly, 2 blocks for propeller near-wakes, 1 block for the tugboat, and 18 blocks for the far field. To balance the workload for each process, we assign two blade and two tip blocks to a single processor. More specifically, the first eight computational blocks (#1 though #8) consist of four blade and four tip grids for the first ducted propeller are assigned to processes #0 (master process) and #1, while the other eight blade/tip blocks (#15 through #22) for the second ducted propeller are assigned to processes #8 and #9 as shown in the `inputmpd.dat` file. The remaining 31 computational blocks containing the ship, shroud, near-wake, and far-field grids are assigned to 31 different processes. As noted earlier, the user need not order the blocks in any particular manner during and after the grid generation process.

The `input.dat` file (the main control input file) requires only minor modifications, relative to the DDG 51 ship case. The most notable difference is that there are two co-rotating propellers in the present full-domain simulation. For each ducted propeller, there are 12 rotating grid blocks (4 blade surface blocks, 4 blade tip blocks, and 4 phantom grids). It is straightforward to specify the rotating direction, shaft block ID, and the IDs of rotating grid blocks associated with each propeller in the following `input.dat` file.

```
1     % MTURB   turbulence model: (0)laminar (1)k-epsilon (2)LES
1     % INCOMP   flag for incompressible (1) or compressible (0) flow
0     % IFSURF   flag for free surface flow (1) or no free surface (0)
2.6468E5  % RE     Reynolds number (L=1ft, U=1ft/s, T=L/U=1s)
0.04    % TAU    time step size
0.0    % AMP_RHO  frequency damping parameter: 0.0 <= AMP_RHO <= 1.0
1.0E-08% TOL1     L2 vel tol to stop time stepping
1.0E-03  % TOL2    L1 res tol to stop outer iterations
1     % ITIMEST  starting time step to compute
12500   % ITIMENDending time step to compute
1     % MAXITER_LS max allowable outer equation for level-set eqn
6     % MAXITER  max allowable outer iterations
2     % MAXSWP_U  max allowable number of momentum eqns ADI sweeps
2     % MAXSWP_PR max allowable number of pressure eqn ADI/SIP sweeps
2     % MAXSWP_KE max allowable number of k-epsilon eqns ADI sweeps
2     % MAXIT_DIVU max projections of velocity field onto div-free space
0.4    % RFU    relaxation factor for velocities (due to nonlinearity)
0.2    % RFP    relaxation factor for pressure (due to u-p decoupling)
0.001    % RFKE    relaxation factor for turbulent k.e. and dissipation
0.5     % RFPHI   relaxation factor for level set funtion
```

```
0      % ITIME_BCS flag to indicate (1) time dependent bcs

fans.grd  % GEOFILE  geometry input file (HCC: not used in this version)
prop.bcs  % BCSFILE  boundary conditions input file

1      % IACT_PLOT flag to activate (1) visualization output
100     % ISKP_PLOT time intervals for vis and restart

1      % IACT_ANIME flag to activate (1) animation output
0      % IBGN_ANIME time step number at which animation begins
20     % ISKP_ANIME multiples at which sol is written out for animation

0      % ISOL_PR  (0)TDMA-ADI, (1)SIP-7pt solver for pressure eqn

-30.    % UMIN
 30.    % UMAX
-200.    % PMIN
 200.    % PMAX
 1.0    % TKEMAX
 0.01    % TVISMAX
-30.    % PHILSMIN
 30.    % PHILSMAX

1.0     % FROUDE  Froude number (gravity acts in negative z-direction)
0.003    % EWIDE   representative grid size

0.0     % RFG    geometry distortion relaxation parameter

0.0 0.0 0.0 % UINF, VINF, WINF  inflow velocities
0.0 0.0 0.0 % UBODY, VBODY, WBODY  body velocities (ship speed)
1      % IMOVE (0) fixed grid, (1) moving grid

0 0 0   % NBODY (#bodies), NFBODY (#surfaces), MBLK (#blocks for 6-dof)

2  12  % NPROP (#propellers), NMVPROP (max #blocks moving with a propeller)

1  1   % IPROP (propeller #1), IROT (1: clockwise, -1: counter-clockwise)
12 10   % IMVPROP (no. of moving propeller blocks), ISHAFT (shaft ID)
1 2 3 4 5 6 7 8 48 49 50 51  % moving propeller block ID (propeller #1)

2  1   % IPROP (propeller #2), IROT (1: clockwise, -1: counter-clockwise)
12 24   % NMVPROP (no. of moving propeller blocks), ISHAFT (shaft ID) 15
16 17 18 19 20 21 22 52 53 54 55 % moving propeller ID (propeller #2)
```

The file `overset.in` needs to be suitably modified for the hole-cutting and donor-search algorithm, and is provided in Appendix C. New boundary condition files need to be created for the new tugboat and ducted propeller geometries. In addition, the boundary condition files for the far field need to be modified slightly to enforce the no-slip boundary conditions on pier walls. All 35 boundary condition files are included in **`tugboat_case3.tar.gz`** zipped folder.

For this example run, the flow field is initialized with a calm water condition and the propeller is allowed to rotate for 500 revolutions under the bollard-pull condition with zero forward speed. The simulation was performed for 12,500 time steps with a time increment of 0.04 T_o, where T_o is a characteristic time for the propeller to turn one revolution. For simplicity, the characteristic length L_o was chosen to be 1 ft so the full scale tugboat and propeller grids (in ft) can be used directly without

rescaling. This gives a Reynolds number of 2.647×10^5 based on the characteristic length $L_o = 1$ ft when the propeller is rotating at 200 rpm. The corresponding Reynolds number based on the propeller diameter is 1.488×10^7 based on the propeller diameter. The RANS equations are solved with the near-wall two-layer k-ε turbulence model.

The `movie_x*.dat` and `movie_q*.dat` output files were post-processed using the FieldView flow visualization software. Typical results include the velocity contours and velocity vector plots at selected coordinate surfaces, and pressure contours on the propeller blade and shroud surfaces. The velocity and pressure fields induced by the twin propellers were shown earlier in Figures 19 through 23 for this case. These velocity contours and velocity-vector plots clearly illustrate that the right propeller wake is strongly affected by the parallel pier wall. Furthermore, there is a strong interaction between the left and right propellers with the two ducted propellers rotating in the same rotation. For the co-rotating propellers considered here, there is a partial suppression of the swirling flow momentums in the overlap region between two propeller wakes. This resulted in a deflection of the weaker left propeller wake (away from the pier wall) toward the sea bottom, as shown in Figure 23 provided in the SSC Pacific technical report.

REFERENCES

Chen, C.-J, R. H. Bravo, H.-C. Chen, and Z. Xu. 1995. "Accurate Discretization of Incompressible Three-Dimensional Navier–Stokes Equations," *Numerical Heat Transfer, Part B: Fundamentals*, vol. 27, no. 4, pp. 371–392.

Chen, C.-J., and H.-C. Chen. 1984. "Finite Analytic Numerical Method for Unsteady Two-dimensional Navier–Stokes Equations," *Journal of Computational Physics*, vol. 53, pp. 209–226.

Chen, H.-C. and V. C. Patel. 1988. "Near-Wall Turbulence Models for Complex Flows Including Separation," *AIAA Journal*, vol. 26, no. 6, pp. 641–648.

Chen, H.-C., V.C. Patel, and S. Ju. 1990. "Solutions of Reynolds-Averaged Navier–Stokes Equations for Three-Dimensional Incompressible Flows," *Journal of Computational Physics*, vol. 88, no. 2, pp. 305–336.

Chung, J. and G. M. Hulbert. 1993. "A Time Integration Algorithm for Structural Dynamics with Improved Numerical Dissipation: The Generalized-α Method," *Journal of Applied Mechanics*, vol. 60, pp. 371–375.

Dettmer, W. and D. Peric. 2003. "An Analysis of the Time Integration Algorithms for the Finite Element Solutions Of Incompressible Navier–Stokes Equations Based on a Stabilised Formulation," *Computer Methods in Applied Mechanics and Engineering*, vol. 192. pp. 1177–1226.

Pontaza J. P., H.-C. Chen, and J. N. Reddy. 2005, "A Local-analytic-based Discretization Procedure for the Numerical Solution of Incompressible Flows." *International Journal for Numerical Methods in Fluids*, vol. 49, no. 6, pp. 657–699.

Wang, Pei-Fang and H.-C Chen. 2016. "FANS Simulation of Propeller Wash at Navy Harbors." Technical Report 2085. Space and Naval Warfare Systems Center Pacific (SSC Pacific), San Diego, CA.

APPENDIX A
STRUCTURE OF THE BOUNDARY CONDITION INPUT

The structure of the boundary conditions input is best explained by showing the pseudo-code used in the program to read the data:

```
do nbk=1,nblocks

read(LB,'(a40)') blockname0
         ! dummy read, blockname specified in inputblk.dat
read(LB,*) mb,nfabcs(mb),ni_dum,nj_dum,nk_dum,mlamp(mb)

read(LB,*)
read(LB,*) nregions(mb)

nregs=nregions(mb)
do nr=1,nregs
      read(LB,*)nreg,iedy(mb,nr),idist(mb,nr), &
      read(LB,*)mst1(mb,nr),mnd1(mb,nr), &
        mst2(mb,nr),mnd2(mb,nr), &
        mst3(mb,nr),mnd3(mb,nr)
end do

read(LB,*)
read(LB,*) nLSregions(mb)

nLSregs=nLSregions(mb)
do nLSr=1,nLSregs
   read(LB,*)nLSreg,nLS(mb,nLSr),
        mst1_LS(mb,nLSr),mnd1_LS(mb,nLSr), &
        mst2_LS(mb,nLSr),mnd2_LS(mb,nLSr), &
        mst3_LS(mb,nLSr),mnd3_LS(mb,nLSr)
end do

read(LB,*) mb,nfabcs(mb),nregions(mb)
nfbcs=nfabcs(mb)
do nf=1,nfbcs
read(LB,*)nfa(mb,nf),nsec(mb,nf)
nsect=nsec(mb,nf)
do ns=1,nsect
read(LB,*)nu(mb,nf,ns),nv(mb,nf,ns),nw(mb,nf,ns),npr(mb,nf,ns), &
             ntke(mb,nf,ns),ntds(mb,nf,ns),nphiLS(mb,nf,ns)
read(LB,*)nst1(mb,nf,ns),nnd1(mb,nf,ns), &
     nst2(mb,nf,ns),nnd2(mb,nf,ns)
end do
end do

end do
read(LB,*) nbk_prd,i_prd,j_prd,k_prd
close(LB)
```

Each process expects a boundary condition input file and executes the above given pseudo-code. The first read statement is a dummy read of the `blockname` to provide clarity, while the `blockname` specified earlier in the `inputblk.dat` file will be used for block identification.

The second read statement reads-in the local block number, the number of faces with boundary conditions for that block, and the (i, j, k) dimensions of the block.

The third read statement reads-in the number of regions needed to identify near-wall and outer regions for the two-layer k-ε model. For each region, we read the `iedy` flag, indicating whether the region is a near-wall region (a value of 1) or an outer region (a value of 2). If the region is a near-wall region then the value of `idist` is the block face number on which the wall lies. We then read the (i, j, k) size of the region.

The fourth read statement reads-in the number of regions needed for various treatments of level-set function for free surface flows. For each region, we read the `nLS` flag, indicating whether the free surface should be solved directly using the advection equation for level-set function (a value of 1), updated using zero-gradient condition (a value of 2, typically used for the near-wall region), or skipped (a value of 3, for single-phase regions without air-water interface). The (i, j, k) size was then specified for each region. For the propeller wash study considered here, the free surface wave effects were negligible and the initial level-set function for calm free surface was used throughout the entire simulation by specifying `nLS` = 3.

After defining various regions for the two-layer k-ε turbulence model and the level-set function, we then specify boundary conditions for all boundary surfaces in the following order: i = imin (Face #1), i = imax (Face #2), j = jmin (Face #3), j = jmax (Face #4), k = kmin (Face #5), and k = kmax (Face #6). For each face, we first read the face number and the number of sections in the face. For each section on a given face, we then read 7 boundary conditions associated with each of the 7 field variables: (u, v, w, p, k, ε, phiLS), and the surface limits on that face. Once this is done for all faces, we proceed to read the block, region, and surface data for the next local block on the same processor (if more than one blocks are assigned to the same CPU). Finally, we must specify where the global pressure datum is located. The information is stored in the variables, `nbk_prd, i_prd, j_prd`, and `k_prd`.

A list of available boundary conditions for the velocity components and turbulence field variables is given below:

- #1: Dirichlet boundary condition, which is set by the initial guess or in the initial input
- #2: linear-extrapolation boundary condition
- #3: homogeneous Neumann or zero gradient boundary condition
- #4: interior boundary condition for overset grids, interpolation using donor data
- #5: prescribed boundary condition, which is specified by initial input or updated in flow solver
- #6: moving surface boundary condition, assign grid velocities
- #7, #8: free (not used at the moment)
- #9: moving surface boundary condition, assign grid velocities
- #10: free (not used at the moment)
- #11: branch cut in lower index, average across branch cut
- #12: branch cut in higher index, average across branch cut
- #13: collapse-to-axis in lower index, average in circumferential direction
- #14: collapse-to-axis in higher index, average in circumferential direction

The following is a list of available boundary conditions for the pressure:

- #1: free (not used at the moment)
- #2: linear-extrapolation boundary condition

- #3: homogeneous Neumann or zero gradient boundary condition
- #4: interior boundary condition for overset grids, interpolation using donor data
- #5: prescribed boundary condition, which is specified by initial input or updated in flow solver
- #6: free (not used at the moment)
- #7: compute pressure consistently, using conservation of mass
- #8, #9, #10: free (not used at the moment)
- #11: branch cut in lower index, average across branch cut
- #12: branch cut in higher index, average across branch cut
- #13: collapse-to-axis in lower index, average in circumferential direction
- #14: collapse-to-axis in higher index, average in circumferential direction

Typical boundary conditions for a stationary wall are either of the following:

```
1 1 1 2 1 1 3  or
1 1 1 3 1 1 3
```

In the first one, pressure is linearly extrapolated at the wall, and in the second it is computed consistently at the wall using conservation of mass at the boundary itself. For highly skewed meshes in the near wall region, linear extrapolation is more stable.

At a free-stream inflow the following are valid options:

```
5 5 5 2 1 1  or
5 5 5 3 1 1
```

In the first one, a zero pressure gradient is enforced, and in the second pressure is computed consistently at the boundary using conservation of mass. The second option, where pressure is computed consistently, is also valid at the inflow of a channel – where a pressure drop is present. The user can appreciate the versatility of the consistent pressure boundary condition, as it applies to virtually any situation where velocities are prescribed.

Similarly, at an outflow, the following are valid options:

```
2 2 2 2 2 2  or
2 2 2 7 2 2
```

For problems involving free surface, the available boundary conditions for the level-set function are listed in the following:

- #1: free (not used at the moment)
- #2: linear-extrapolation boundary condition
- #3: homogeneous Neumann or zero gradient boundary condition
- #4: interior boundary condition for overset grids, interpolation using donor data
- #5: prescribed boundary condition, which is specified by initial input or updated in flow solver
- #6, #7, #8, #9, #10: free (not used at the moment)
- #11: branch cut in lower index, average across branch cut
- #12: branch cut in higher index, average across branch cut

- #13: collapse-to-axis in lower index, average in circumferential direction
- #14: collapse-to-axis in higher index, average in circumferential direction

APPENDIX B
COSMIC INPUT FILE FOR DDG-51 AND P4876 PROPELLER WASH STUDY

```
! example input for DDG-51 Ship and P4876 propeller wash study

! global parameters

<global>
   fringe = 1,
   quality = 0.01,
   nquality = 4,
   eps = 0.001,
</global>


! grid block definition

<block name = "propeller01">
    linking_grid_list = <"propeller01","shaft01","shaft02","basin01",
              "basin02","propeller05","propeller02","ocean03">
</block>
<block name = "propeller02">
    linking_grid_list = <"propeller02","shaft01","shaft02","basin01",
              "basin02","propeller01","propeller03","ocean03">
</block>
<block name = "propeller03">
    linking_grid_list = <"propeller03","shaft01","shaft02","basin01",
              "basin02","propeller02","propeller04","ocean03">
</block>
<block name = "propeller04">
    linking_grid_list = <"propeller04","shaft01","shaft02","basin01",
              "basin02","propeller03","propeller05","ocean03">
</block>
<block name = "propeller05">
    linking_grid_list = <"propeller05","shaft01","shaft02","basin01",
              "basin02","propeller04","propeller01","ocean03">
</block>
<block name = "shaft01">
    linking_grid_list = <"shaft01","shaft02","shaft03","basin01",
              "basin02","propeller01","propeller02",
              "propeller03","propeller04","propeller05",
              "ocean02","ocean03">
</block>
<block name = "shaft02">
    linking_grid_list = <"shaft02","shaft01","shaft03","basin01",
              "basin02","propeller01","propeller02",
              "propeller03","propeller04","propeller05",
              "ocean02","ocean03">
</block>
<block name = "shaft03">
    linking_grid_list = <"shaft03","shaft01","shaft02","basin01",
              "ship01","ocean02","ocean03">
</block>
<block name = "ship01">
    linking_grid_list = <"basin01","basin02","basin03","ocean01",
```

```
              "ocean02","ocean03","shaft01","shaft02",
              "shaft03">
</block>
<block name = "basin01">
    linking_grid_list = <"basin02","shaft01","shaft02","shaft03",
              "ocean01","ocean02","ocean03",
              "propeller01","propeller02","propeller03",
              "propeller04","propeller05","ship01">
</block>
<block name = "basin02">
    linking_grid_list = <"basin01","basin03","shaft01","shaft02",
              "shaft03","ocean01","ocean02","ocean03",
              "propeller01","propeller02","propeller03",
              "propeller04","propeller05","ship01">
</block>
<block name = "basin03">
    linking_grid_list = <"basin02","ocean01","ocean03">
</block>
<block name = "ocean01">
    linking_grid_list = <"basin01","basin02","basin03",
              "ocean02","ocean03">
</block>
<block name = "ocean02">
    linking_grid_list = <"ocean01","ocean03","ship01","basin01",
              "shaft01","shaft02","shaft03">
</block>
<block name = "ocean03">
    linking_grid_list = <"ocean01","ocean02","basin01","basin02",
              "basin03","ship01","shaft03","shaft02",
              "shaft01">
</block>
<block name = "phantom01">
    linking_grid_list = <"phantom01">
</block>
<block name = "phantom02">
    linking_grid_list = <"phantom02">
</block>
<block name = "phantom03">
    linking_grid_list = <"phantom03">
</block>
<block name = "phantom04">
    linking_grid_list = <"phantom04">
</block>
<block name = "phantom05">
    linking_grid_list = <"phantom05">
</block>
<block name = "phantom06">
    linking_grid_list = <"phantom06">
</block>
<block name = "phantom07">
    linking_grid_list = <"phantom07">
</block>


! hole boundary definition

<boundary name = "phantom01 hole boundary">
```

```
     parent_grid = "phantom01",
     hole_cutting_list = <"shaft01","shaft02","basin01","basin02">
</boundary>

<boundary name = "phantom02 hole boundary">
     parent_grid = "phantom02",
     hole_cutting_list = <"shaft01","shaft02","basin01","basin02">
</boundary>

<boundary name = "phantom03 hole boundary">
     parent_grid = "phantom03",
     hole_cutting_list = <"shaft01","shaft02","basin01","basin02">
</boundary>

<boundary name = "phantom04 hole boundary">
     parent_grid = "phantom04",
     hole_cutting_list = <"shaft01","shaft02","basin01","basin02">
</boundary>

<boundary name = "phantom05 hole boundary">
     parent_grid = "phantom05",
     hole_cutting_list = <"shaft01","shaft02","basin01","basin02">
</boundary>

<boundary name = "phantom06 hole boundary">
      parent_grid = "phantom06",
      hole_cutting_list = <"ship01">
</boundary>

<boundary name = "phantom07 hole boundary">
      parent_grid = "phantom07",
      hole_cutting_list = <"ship01","ocean03">
</boundary>

<boundary name = "shaft01 hole boundary">
      parent_grid = "shaft01"
      hole_cutting_list = <"basin01","basin02">
</boundary>

<boundary name = "shaft03 hole boundary">
      parent_grid = "shaft03"
      hole_cutting_list = <"basin01">
</boundary>

<boundary name = "ship01 hole boundary">
      parent_grid = "ship01",
      hole_cutting_list = <"shaft03","basin01","basin02",
                "ocean02","ocean03">
</boundary>


! hole surface definitions

<surface name = "phantom01 hole boundary">
     ijk_range =   1,  1,  1, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "-ijk",
```

```
</surface>
<surface name = "phantom01 hole boundary">
     ijk_range =   3,  3,  1, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom01 hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 61,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom01 hole boundary">
     ijk_range =   1,  3, 41, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom01 hole boundary">
     ijk_range =   1,  3,  1, 41, 61, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "phantom02 hole boundary">
     ijk_range =   1,  1,  1, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom02 hole boundary">
     ijk_range =   3,  3,  1, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom02 hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 61,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom02 hole boundary">
     ijk_range =   1,  3, 41, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom02 hole boundary">
     ijk_range =   1,  3,  1, 41, 61, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "phantom03 hole boundary">
     ijk_range =   1,  1,  1, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom03 hole boundary">
     ijk_range =   3,  3,  1, 41,  1, 61,
```

```
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom03 hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 61,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom03 hole boundary">
     ijk_range =   1,  3, 41, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom03 hole boundary">
     ijk_range =   1,  3,  1, 41, 61, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "phantom04 hole boundary">
     ijk_range =   1,  1,  1, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom04 hole boundary">
     ijk_range =   3,  3,  1, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom04 hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 61,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom04 hole boundary">
     ijk_range =   1,  3, 41, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom04 hole boundary">
     ijk_range =   1,  3,  1, 41, 61, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "phantom05 hole boundary">
     ijk_range =   1,  1,  1, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom05 hole boundary">
     ijk_range =   3,  3,  1, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
```

```
<surface name = "phantom05 hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 61,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom05 hole boundary">
     ijk_range =   1,  3, 41, 41,  1, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom05 hole boundary">
     ijk_range =   1,  3,  1, 41, 61, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "phantom06 hole boundary">
     ijk_range =   1,  1,  1,  2,  1,  2,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom06 hole boundary">
     ijk_range =   2,  2,  1,  2,  1,  2,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom06 hole boundary">
     ijk_range =   1,  2,  1,  1,  1,  2,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom06 hole boundary">
     ijk_range =   1,  2,  2,  2,  1,  2,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom06 hole boundary">
     ijk_range =   1,  2,  1,  2,  1,  1,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom06 hole boundary">
     ijk_range =   1,  2,  1,  2,  2,  2,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "phantom07 hole boundary">
     ijk_range =   1,  1,  1,  2,  1,  2,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom07 hole boundary">
     ijk_range =   2,  2,  1,  2,  1,  2,
     boundary_condition ="cut",
```

```
     surface_normal = "+ijk",
</surface>
<surface name = "phantom07 hole boundary">
     ijk_range =   1,  2,  1,  1,  1,  2,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom07 hole boundary">
     ijk_range =   1,  2,  2,  2,  1,  2,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom07 hole boundary">
     ijk_range =   1,  2,  1,  2,  1,  1,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom07 hole boundary">
     ijk_range =   1,  2,  1,  2,  2,  2,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "shaft01 hole boundary">
     ijk_range =   1, 38,  3,  3,  1, 121,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "shaft03 hole boundary">
     ijk_range =   6, 65,  3,  3,  1, 121,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "ship01 hole boundary">
     ijk_range =   1, 121, 21, 21,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>



!   outer boundary definition

<boundary name = "propeller01 outer boundary">
      parent_grid = "propeller01",
</boundary>
<boundary name = "propeller02 outer boundary">
      parent_grid = "propeller02",
</boundary>
<boundary name = "propeller03 outer boundary">
      parent_grid = "propeller03",
</boundary>
```

```
<boundary name = "propeller04 outer boundary">
      parent_grid = "propeller04",
</boundary>
<boundary name = "propeller05 outer boundary">
      parent_grid = "propeller05",
</boundary>
<boundary name = "shaft01 outer boundary">
      parent_grid = "shaft01",
</boundary>
<boundary name = "shaft02 outer boundary">
      parent_grid = "shaft02",
</boundary>
<boundary name = "shaft03 outer boundary">
      parent_grid = "shaft03",
</boundary>
<boundary name = "ship01 outer boundary">
      parent_grid = "ship01",
</boundary>
<boundary name = "basin01 outer boundary">
      parent_grid = "basin01",
</boundary>
<boundary name = "basin02 outer boundary">
      parent_grid = "basin02",
</boundary>
<boundary name = "basin03 outer boundary">
      parent_grid = "basin03",
</boundary>
<boundary name = "ocean01 outer boundary">
      parent_grid = "ocean01",
</boundary>
<boundary name = "ocean02 outer boundary">
      parent_grid = "ocean02",
</boundary>
<boundary name = "ocean03 outer boundary">
      parent_grid = "ocean03",
</boundary>


! outer boundary surface definition

<surface name = "propeller01 outer boundary">
      ijk_range =  1,  1,  1, 41,  1, 41,
      boundary_condition = "periodic",
      donor_grid = "propeller01"
      donor_ijk_range = 61, 61,  1, 41,  1, 41,
</surface>
<surface name = "propeller01 outer boundary">
      ijk_range = 62, 62,1, 41,    1, 41,
      boundary_condition = "periodic",
      donor_grid = "propeller01"
      donor_ijk_range =  2,2,1, 41,1, 41,
</surface>
<surface name = "propeller01 outer boundary">
      ijk_range =  1, 62, 41, 41,  2, 40,
</surface>
```

```
<surface name = "propeller02 outer boundary">
      ijk_range =  1,  1,  1, 41,  1, 41,
      boundary_condition = "periodic",
      donor_grid = "propeller02"
      donor_ijk_range = 61, 61,  1, 41,  1, 41,
</surface>
<surface name = "propeller02 outer boundary">
      ijk_range = 62, 62,1, 41,    1, 41,
      boundary_condition = "periodic",
      donor_grid = "propeller02"
      donor_ijk_range =  2,2,1, 41,1, 41,
</surface>
<surface name = "propeller02 outer boundary">
      ijk_range =  1, 62, 41, 41,  2, 40,
</surface>


<surface name = "propeller03 outer boundary">
      ijk_range =  1,  1,  1, 41,  1, 41,
      boundary_condition = "periodic",
      donor_grid = "propeller03"
      donor_ijk_range = 61, 61,  1, 41,  1, 41,
</surface>
<surface name = "propeller03 outer boundary">
      ijk_range = 62, 62,1, 41,    1, 41,
      boundary_condition = "periodic",
      donor_grid = "propeller03"
      donor_ijk_range =  2,2,1, 41,1, 41,
</surface>
<surface name = "propeller03 outer boundary">
      ijk_range =  1, 62, 41, 41,  2, 40,
</surface>


<surface name = "propeller04 outer boundary">
      ijk_range =  1,  1,  1, 41,  1, 41,
      boundary_condition = "periodic",
      donor_grid = "propeller04"
      donor_ijk_range = 61, 61,  1, 41,  1, 41,
</surface>
<surface name = "propeller04 outer boundary">
      ijk_range = 62, 62,1, 41,    1, 41,
      boundary_condition = "periodic",
      donor_grid = "propeller04"
      donor_ijk_range =  2,2,1, 41,1, 41,
</surface>
<surface name = "propeller04 outer boundary">
      ijk_range =  1, 62, 41, 41,  2, 40,
</surface>


<surface name = "propeller05 outer boundary">
      ijk_range =  1,  1,  1, 41,  1, 41,
      boundary_condition = "periodic",
      donor_grid = "propeller05"
      donor_ijk_range = 61, 61,  1, 41,  1, 41,
</surface>
```

```
<surface name = "propeller05 outer boundary">
      ijk_range = 62, 62,1, 41,    1, 41,
      boundary_condition = "periodic",
      donor_grid = "propeller05"
      donor_ijk_range =  2,2,1, 41,1, 41,
</surface>
<surface name = "propeller05 outer boundary">
      ijk_range =  1, 62, 41, 41,  2, 40,
</surface>


<surface name = "shaft01 outer boundary">
      ijk_range =  1,  1,  1, 21,  1, 122,
</surface>
<surface name = "shaft01 outer boundary">
      ijk_range =  1, 38, 21, 21,  1, 122,
</surface>
<surface name = "shaft01 outer boundary">
     ijk_range =  1, 38,  1,  21,  1,  1,
     boundary_condition = "periodic",
     donor_grid = "shaft01"
     donor_ijk_range =  1, 38,  1, 21, 121, 121,
</surface>
<surface name = "shaft01 outer boundary">
     ijk_range =  1, 38,  1, 21, 122, 122,
     boundary_condition = "periodic",
     donor_grid = "shaft01"
     donor_ijk_range =  1,  38,  1, 21,  2,  2,
</surface>


<surface name = "shaft02 outer boundary">
      ijk_range =  1,  1,  1, 31,  1, 122,
</surface>
<surface name = "shaft02 outer boundary">
      ijk_range = 28, 28,  1, 31,  1, 122,
</surface>
<surface name = "shaft02 outer boundary">
      ijk_range =  1, 28,  1,  1,  1, 122,
</surface>
<surface name = "shaft02 outer boundary">
      ijk_range =  1, 28, 31, 31,  1, 122,
</surface>
<surface name = "shaft02 outer boundary">
     ijk_range =  1, 28,  1,  31,  1,  1,
     boundary_condition = "periodic",
     donor_grid = "shaft02"
     donor_ijk_range =  1, 28,  1, 31, 121, 121,
</surface>
<surface name = "shaft02 outer boundary">
     ijk_range =  1, 28,  1, 31, 122, 122,
     boundary_condition = "periodic",
     donor_grid = "shaft02"
     donor_ijk_range =  1,  28,  1, 31,  2,  2,
</surface>
```

```
<surface name = "shaft03 outer boundary">
      ijk_range = 65, 65,  1, 21,  1, 122,
</surface>
<surface name = "shaft03 outer boundary">
      ijk_range =  2, 65, 21, 21,  1, 122,
</surface>
<surface name = "shaft03 outer boundary">
     ijk_range =  1, 65,  1,  21,  1,  1,
     boundary_condition = "periodic",
     donor_grid = "shaft03"
     donor_ijk_range =  1, 65,  1, 21, 121, 121,
</surface>
<surface name = "shaft03 outer boundary">
     ijk_range =  1, 65,  1, 21, 122, 122,
     boundary_condition = "periodic",
     donor_grid = "shaft03"
     donor_ijk_range =  1, 65,  1, 21,  2,  2,
</surface>
<surface name = "shaft03 outer boundary">
      ijk_range =  1, 65,  1,  1,  1, 122,
      boundary_condition = "body",
</surface>


<surface name = "ship01 outer boundary">
      ijk_range =  1, 121, 35, 35,  1, 41,
</surface>


<surface name = "basin01 outer boundary">
      ijk_range =  1,  1,  1, 81,  1, 77,
</surface>
<surface name = "basin01 outer boundary">
      ijk_range = 34, 34,  1, 81,  1, 77,
</surface>
<surface name = "basin01 outer boundary">
      ijk_range =  1, 34, 81, 81,  1, 77,
</surface>
<surface name = "basin01 outer boundary">
      ijk_range =  1, 34,  1, 81,  1,  1,
</surface>
<surface name = "basin01 outer boundary">
      ijk_range =  1, 34,  1, 81, 77, 77,
</surface>


<surface name = "basin02 outer boundary">
      ijk_range =  1,  1,  1, 81,  1, 77,
</surface>
<surface name = "basin02 outer boundary">
      ijk_range = 34, 34,  1, 81,  1, 77,
</surface>
<surface name = "basin02 outer boundary">
      ijk_range =  1, 34, 81, 81,  1, 77,
</surface>
<surface name = "basin02 outer boundary">
      ijk_range =  1, 34,  1, 81,  1,  1,
```

```
</surface>
<surface name = "basin02 outer boundary">
      ijk_range =  1, 34,  1, 81, 77, 77,
</surface>


<surface name = "basin03 outer boundary">
      ijk_range =  1,  1,  1, 81,  1, 77,
</surface>
<surface name = "basin03 outer boundary">
      ijk_range = 34, 34,  1, 81,  1, 77,
</surface>
<surface name = "basin03 outer boundary">
      ijk_range =  1, 34, 81, 81,  1, 77,
</surface>
<surface name = "basin03 outer boundary">
      ijk_range =  1, 34,  1, 81,  1,  1,
</surface>
<surface name = "basin03 outer boundary">
      ijk_range =  1, 34,  1, 81, 77, 77,
</surface>


<surface name = "ocean01 outer boundary">
      ijk_range =  1, 152,  1, 65, 21, 21,
</surface>


<surface name = "ocean02 outer boundary">
      ijk_range = 77, 77,  1, 65,  1, 42,
</surface>
<surface name = "ocean02 outer boundary">
      ijk_range =  1, 77,  1, 65,  1,  1,
</surface>


<surface name = "ocean03 outer boundary">
      ijk_range =  1,  1,  1, 65,  1, 42,
</surface>
<surface name = "ocean03 outer boundary">
      ijk_range =  1, 77,  1, 65,  1,  1,
</surface>
```

APPENDIX C
COSMIC INPUT FILE FOR TUGBOAT AND DUCTED-PROPELLER WASH STUDY

```
! example input for tugboat and ducted propellers

! global parameters

<global>
   fringe = 1,
   quality = 0.01,
   nquality = 4,
   eps = 0.001,
</global>



! grid block definition

<block name = "propeller01a">
    linking_grid_list = <"propeller01a","tip01a","propeller04a",
                "propeller02a","duct02a","duct03a">
</block>
<block name = "propeller02a">
    linking_grid_list = <"propeller02a","tip02a","propeller01a",
                "propeller03a","duct02a","duct03a">
</block>
<block name = "propeller03a">
    linking_grid_list = <"propeller03a","tip03a","propeller02a",
                "propeller04a","duct02a","duct03a">
</block>
<block name = "propeller04a">
    linking_grid_list = <"propeller04a","tip04a","propeller03a",
                "propeller01a","duct02a","duct03a">
</block>
<block name = "tip01a">
    linking_grid_list = <"propeller01a","duct03a">
</block>
<block name = "tip02a">
    linking_grid_list = <"propeller02a","duct03a">
</block>
<block name = "tip03a">
    linking_grid_list = <"propeller03a","duct03a">
</block>
<block name = "tip04a">
    linking_grid_list = <"propeller04a","duct03a">
</block>
<block name = "duct01a">
    linking_grid_list = <"duct01a","duct02a","duct03a","duct04a",
                "barge01","ocean03","ocean04">
</block>
<block name = "duct02a">
    linking_grid_list = <"duct02a","duct01a","duct03a","duct05a",
                "wake01a","propeller01a","propeller02a",
```

```
              "propeller03a","propeller04a">
</block>
<block name = "duct03a">
    linking_grid_list = <"duct03a","duct01a","duct02a","duct05a",
              "propeller01a","propeller02a",
              "propeller03a","propeller04a","tip01a",
              "tip02a","tip03a","tip04a","barge01">
</block>
<block name = "duct04a">
    linking_grid_list = <"duct04a","duct01a","duct05a","wake01a",
              "barge01","ocean04">
</block>
<block name = "duct05a">
    linking_grid_list = <"duct05a","duct02a","duct03a","duct04a",
              "wake01a","barge01","ocean04">
</block>
<block name = "wake01a">
    linking_grid_list = <"wake01a","duct02a","duct03a","duct04a",
              "duct05a","barge01","ocean04","ocean05">
</block>
<block name = "propeller01b">
    linking_grid_list = <"propeller01b","tip01b","propeller04b",
              "propeller02b","duct02b","duct03b">
</block>
<block name = "propeller02b">
    linking_grid_list = <"propeller02b","tip02b","propeller01b",
              "propeller03b","duct02b","duct03b">
</block>
<block name = "propeller03b">
    linking_grid_list = <"propeller03b","tip03b","propeller02b",
              "propeller04b","duct02b","duct03b">
</block>
<block name = "propeller04b">
    linking_grid_list = <"propeller04b","tip04b","propeller03b",
              "propeller01b","duct02b","duct03b">
</block>
<block name = "tip01b">
    linking_grid_list = <"propeller01b","duct03b">
</block>
<block name = "tip02b">
    linking_grid_list = <"propeller02b","duct03b">
</block>
<block name = "tip03b">
    linking_grid_list = <"propeller03b","duct03b">
</block>
<block name = "tip04b">
    linking_grid_list = <"propeller04b","duct03b">
</block>
<block name = "duct01b">
    linking_grid_list = <"duct01b","duct02b","duct03b","duct04b",
              "barge01","ocean03","ocean04">
</block>
<block name = "duct02b">
    linking_grid_list = <"duct02b","duct01b","duct03b","duct05b",
              "wake01b","propeller01b","propeller02b",
              "propeller03b","propeller04b">
</block>
```

```
<block name = "duct03b">
    linking_grid_list = <"duct03b","duct01b","duct02b","duct05b",
                "propeller01b","propeller02b",
                "propeller03b","propeller04b","tip01b",
                "tip02b","tip03b","tip04b","barge01">
</block>
<block name = "duct04b">
    linking_grid_list = <"duct04b","duct01b","duct05b","wake01b",
                "barge01","ocean04">
</block>
<block name = "duct05b">
    linking_grid_list = <"duct05b","duct02b","duct03b","duct04b",
                "wake01b","barge01","ocean04">
</block>
<block name = "wake01b">
    linking_grid_list = <"wake01b","duct02b","duct03b","duct04b",
                "duct05b","barge01","ocean04","ocean05">
</block>
<block name = "ocean01">
    linking_grid_list = <"ocean02","ocean17","ocean18","barge01">
</block>
<block name = "ocean02">
    linking_grid_list = <"ocean01","ocean03","ocean17","ocean18",
                "barge01">
</block>
<block name = "ocean03">
    linking_grid_list = <"ocean02","ocean04","ocean17","ocean18",
                "duct01a","duct01b","duct04a","duct04b",
                "barge01">
</block>
<block name = "ocean04">
    linking_grid_list = <"ocean03","ocean05","ocean17","ocean18",
                "wake01a","wake01b","duct01a","duct01b",
                "duct04a","duct04b","duct05a","duct05b",
                "barge01">
</block>
<block name = "ocean05">
    linking_grid_list = <"ocean04","ocean06","ocean17","ocean18",
                "wake01a","wake01b","barge01">
</block>
<block name = "ocean06">
    linking_grid_list = <"ocean05","ocean07","ocean17","ocean18",
                "wake01a","wake01b","barge01">
</block>
<block name = "ocean07">
    linking_grid_list = <"ocean06","ocean08","ocean17","ocean18",
                "barge01">
</block>
<block name = "ocean08">
    linking_grid_list = <"ocean07","ocean09","ocean17","ocean18",
                "barge01">
</block>
<block name = "ocean09">
    linking_grid_list = <"ocean08","ocean10","ocean17","ocean18",
                "barge01">
</block>
<block name = "ocean10">
```

```
    linking_grid_list = <"ocean09","ocean11","ocean17","ocean18",
                "barge01">
</block>
<block name = "ocean11">
    linking_grid_list = <"ocean10","ocean12","ocean17","ocean18">
</block>
<block name = "ocean12">
    linking_grid_list = <"ocean11","ocean13","ocean17","ocean18">
</block>
<block name = "ocean13">
    linking_grid_list = <"ocean12","ocean14","ocean17","ocean18">
</block>
<block name = "ocean14">
    linking_grid_list = <"ocean13","ocean15","ocean17","ocean18">
</block>
<block name = "ocean15">
    linking_grid_list = <"ocean14","ocean16","ocean17","ocean18">
</block>
<block name = "ocean16">
    linking_grid_list = <"ocean15","ocean17","ocean18">
</block>
<block name = "ocean17">
    linking_grid_list = <"ocean01","ocean02","ocean03","ocean04",
                "ocean05","ocean06","ocean07","ocean08",
                "ocean09","ocean10","ocean11","ocean12",
                "ocean13","ocean14","ocean15","ocean16",
                "ocean18","barge01">
</block>
<block name = "ocean18">
    linking_grid_list = <"ocean01","ocean02","ocean03","ocean04",
                "ocean05","ocean06","ocean07","ocean08",
                "ocean09","ocean10","ocean11","ocean12",
                "ocean13","ocean14","ocean15","ocean16",
                "ocean17","barge01">
</block>
<block name = "barge01">
    linking_grid_list = <"duct01a","duct01b","duct04a","duct04b",
                "duct05a","duct05b","wake01a","wake01b",
                "ocean01","ocean02","ocean03","ocean04",
                "ocean05","ocean06","ocean07","ocean08",
                "ocean09","ocean10","ocean17","ocean18">
</block>
<block name = "phantom01a">
    linking_grid_list = <"phantom01a">
</block>
<block name = "phantom02a">
    linking_grid_list = <"phantom02a">
</block>
<block name = "phantom03a">
    linking_grid_list = <"phantom03a">
</block>
<block name = "phantom04a">
    linking_grid_list = <"phantom04a">
</block>
<block name = "phantom01b">
    linking_grid_list = <"phantom01b">
</block>
```

```
<block name = "phantom02b">
    linking_grid_list = <"phantom02b">
</block>
<block name = "phantom03b">
    linking_grid_list = <"phantom03b">
</block>
<block name = "phantom04b">
    linking_grid_list = <"phantom04b">
</block>
<block name = "phantom05">
    linking_grid_list = <"phantom05">
</block>


! hole boundary definition

<boundary name = "phantom01a hole boundary">
     parent_grid = "phantom01a",
     hole_cutting_list = <"duct02a","duct03a">
</boundary>

<boundary name = "phantom02a hole boundary">
     parent_grid = "phantom02a",
     hole_cutting_list = <"duct02a","duct03a">
</boundary>

<boundary name = "phantom03a hole boundary">
     parent_grid = "phantom03a",
     hole_cutting_list = <"duct02a","duct03a">
</boundary>

<boundary name = "phantom04a hole boundary">
     parent_grid = "phantom04a",
     hole_cutting_list = <"duct02a","duct03a">
</boundary>

<boundary name = "phantom01b hole boundary">
     parent_grid = "phantom01b",
     hole_cutting_list = <"duct02b","duct03b">
</boundary>

<boundary name = "phantom02b hole boundary">
     parent_grid = "phantom02b",
     hole_cutting_list = <"duct02b","duct03b">
</boundary>

<boundary name = "phantom03b hole boundary">
     parent_grid = "phantom03b",
     hole_cutting_list = <"duct02b","duct03b">
</boundary>

<boundary name = "phantom04b hole boundary">
     parent_grid = "phantom04b",
     hole_cutting_list = <"duct02b","duct03b">
</boundary>

<boundary name = "phantom05 hole boundary">
```

```
     parent_grid = "phantom05",
     hole_cutting_list = <"ocean01","ocean02","ocean03",
                "ocean04","ocean05","ocean06",
                "ocean07","ocean08","ocean09",
                "ocean10">
</boundary>

<boundary name = "duct01a hole boundary">
     parent_grid = "duct01a",
     hole_cutting_list = <"ocean03","ocean04","barge01">
</boundary>


<boundary name = "duct04a hole boundary">
     parent_grid = "duct04a",
     hole_cutting_list = <"wake01a","ocean04","ocean03",
                "barge01">
</boundary>

<boundary name = "duct05a hole boundary">
     parent_grid = "duct05a",
     hole_cutting_list = <"wake01a","ocean04","barge01">
</boundary>

<boundary name = "wake01a hole boundary">
     parent_grid = "wake01a",
     hole_cutting_list = <"ocean04","ocean05","barge01">
</boundary>

<boundary name = "duct01b hole boundary">
     parent_grid = "duct01b",
     hole_cutting_list = <"ocean03","ocean04","barge01">
</boundary>


<boundary name = "duct04b hole boundary">
     parent_grid = "duct04b",
     hole_cutting_list = <"wake01b","ocean04","ocean03",
                "barge01">
</boundary>

<boundary name = "duct05b hole boundary">
     parent_grid = "duct05b",
     hole_cutting_list = <"wake01b","ocean04","barge01">
</boundary>

<boundary name = "wake01b hole boundary">
     parent_grid = "wake01b",
     hole_cutting_list = <"ocean04","ocean05","barge01">
</boundary>

<boundary name = "barge01 hole boundary">
     parent_grid = "barge01",
     hole_cutting_list = <"ocean01","ocean02","ocean03",
                "ocean04","ocean05","ocean06",
                "ocean07","ocean08","ocean09",
                "ocean10","wake01a","wake01b",
```

```
                "duct01a","duct01b","duct04a",
                "duct04b","duct05a","duct05b">
</boundary>


! hole surface definitions

<surface name = "phantom01a hole boundary">
     ijk_range =   1,  1,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom01a hole boundary">
     ijk_range =   3,  3,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom01a hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom01a hole boundary">
     ijk_range =   1,  3, 31, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom01a hole boundary">
     ijk_range =   1,  3,  1, 31, 41, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "phantom02a hole boundary">
     ijk_range =   1,  1,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom02a hole boundary">
     ijk_range =   3,  3,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom02a hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom02a hole boundary">
     ijk_range =   1,  3, 31, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom02a hole boundary">
     ijk_range =   1,  3,  1, 31, 41, 41,
     boundary_condition ="cut",
```

```
     surface_normal = "+ijk",
</surface>


<surface name = "phantom03a hole boundary">
     ijk_range =   1,  1,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom03a hole boundary">
     ijk_range =   3,  3,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom03a hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom03a hole boundary">
     ijk_range =   1,  3, 31, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom03a hole boundary">
     ijk_range =   1,  3,  1, 31, 41, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "phantom04a hole boundary">
     ijk_range =   1,  1,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom04a hole boundary">
     ijk_range =   3,  3,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom04a hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom04a hole boundary">
     ijk_range =   1,  3, 31, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom04a hole boundary">
     ijk_range =   1,  3,  1, 31, 41, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
```

```
<surface name = "phantom01b hole boundary">
     ijk_range =   1,  1,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom01b hole boundary">
     ijk_range =   3,  3,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom01b hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom01b hole boundary">
     ijk_range =   1,  3, 31, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom01b hole boundary">
     ijk_range =   1,  3,  1, 31, 41, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "phantom02b hole boundary">
     ijk_range =   1,  1,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom02b hole boundary">
     ijk_range =   3,  3,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom02b hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom02b hole boundary">
     ijk_range =   1,  3, 31, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom02b hole boundary">
     ijk_range =   1,  3,  1, 31, 41, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "phantom03b hole boundary">
     ijk_range =   1,  1,  1, 31,  1, 41,
```

```
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom03b hole boundary">
     ijk_range =   3,  3,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom03b hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom03b hole boundary">
     ijk_range =   1,  3, 31, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom03b hole boundary">
     ijk_range =   1,  3,  1, 31, 41, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "phantom04b hole boundary">
     ijk_range =   1,  1,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom04b hole boundary">
     ijk_range =   3,  3,  1, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom04b hole boundary">
     ijk_range =   1,  3,  1,  1,  1, 41,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom04b hole boundary">
     ijk_range =   1,  3, 31, 31,  1, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom04b hole boundary">
     ijk_range =   1,  3,  1, 31, 41, 41,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "phantom05 hole boundary">
     ijk_range =   1,  1,  1,  3,  1,  2,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
```

```
<surface name = "phantom05 hole boundary">
     ijk_range =   2,  2,  1,  3,  1,  2,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom05 hole boundary">
     ijk_range =   1,  2,  1,  1,  1,  2,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom05 hole boundary">
     ijk_range =   1,  2,  3,  3,  1,  2,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "phantom05 hole boundary">
     ijk_range =   1,  2,  1,  3,  1,  1,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "phantom05 hole boundary">
     ijk_range =   1,  2,  1,  3,  2,  2,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "duct01a hole boundary">
     ijk_range =   7,  7,  1, 37,  1, 121,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "duct01a hole boundary">
     ijk_range =   7, 21, 37, 37,  1, 121,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "duct04a hole boundary">
     ijk_range =   1, 53,  3,  3,  1, 121,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "duct05a hole boundary">
     ijk_range =   1, 15, 59, 59,  1, 121,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "wake01a hole boundary">
     ijk_range =   1,  1,  1, 29,  1, 121,
     boundary_condition ="cut",
     surface_normal = "-ijk",
```

```
</surface>
<surface name = "wake01a hole boundary">
     ijk_range =  36, 36,  1, 29,  1, 121,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "wake01a hole boundary">
     ijk_range =   1, 36, 29, 29,  1, 121,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "duct01b hole boundary">
     ijk_range =   7,  7,  1, 37,  1, 121,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "duct01b hole boundary">
     ijk_range =   7, 21, 37, 37,  1, 121,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "duct04b hole boundary">
     ijk_range =   1, 53,  3,  3,  1, 121,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "duct05b hole boundary">
     ijk_range =   1, 15, 59, 59,  1, 121,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "wake01b hole boundary">
     ijk_range =   1,  1,  1, 29,  1, 121,
     boundary_condition ="cut",
     surface_normal = "-ijk",
</surface>
<surface name = "wake01b hole boundary">
     ijk_range =  36, 36,  1, 29,  1, 121,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>
<surface name = "wake01b hole boundary">
     ijk_range =   1, 36, 29, 29,  1, 121,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


<surface name = "barge01 hole boundary">
```

```
     ijk_range =   1, 107, 17, 17,  1, 61,
     boundary_condition ="cut",
     surface_normal = "+ijk",
</surface>


!   outer boundary definition


<boundary name = "propeller01a outer boundary">
      parent_grid = "propeller01a",
</boundary>
<boundary name = "propeller02a outer boundary">
      parent_grid = "propeller02a",
</boundary>
<boundary name = "propeller03a outer boundary">
      parent_grid = "propeller03a",
</boundary>
<boundary name = "propeller04a outer boundary">
      parent_grid = "propeller04a",
</boundary>
<boundary name = "tip01a outer boundary">
      parent_grid = "tip01a",
</boundary>
<boundary name = "tip02a outer boundary">
      parent_grid = "tip02a",
</boundary>
<boundary name = "tip03a outer boundary">
      parent_grid = "tip03a",
</boundary>
<boundary name = "tip04a outer boundary">
      parent_grid = "tip04a",
</boundary>
<boundary name = "duct01a outer boundary">
      parent_grid = "duct01a",
</boundary>
<boundary name = "duct02a outer boundary">
      parent_grid = "duct02a",
</boundary>
<boundary name = "duct03a outer boundary">
      parent_grid = "duct03a",
</boundary>
<boundary name = "duct04a outer boundary">
      parent_grid = "duct04a",
</boundary>
<boundary name = "duct05a outer boundary">
      parent_grid = "duct05a",
</boundary>
<boundary name = "wake01a outer boundary">
      parent_grid = "wake01a",
</boundary>
<boundary name = "propeller01b outer boundary">
      parent_grid = "propeller01b",
</boundary>
<boundary name = "propeller02b outer boundary">
      parent_grid = "propeller02b",
```

```
</boundary>
<boundary name = "propeller03b outer boundary">
      parent_grid = "propeller03b",
</boundary>
<boundary name = "propeller04b outer boundary">
      parent_grid = "propeller04b",
</boundary>
<boundary name = "tip01b outer boundary">
      parent_grid = "tip01b",
</boundary>
<boundary name = "tip02b outer boundary">
      parent_grid = "tip02b",
</boundary>
<boundary name = "tip03b outer boundary">
      parent_grid = "tip03b",
</boundary>
<boundary name = "tip04b outer boundary">
      parent_grid = "tip04b",
</boundary>
<boundary name = "duct01b outer boundary">
      parent_grid = "duct01b",
</boundary>
<boundary name = "duct02b outer boundary">
      parent_grid = "duct02b",
</boundary>
<boundary name = "duct03b outer boundary">
      parent_grid = "duct03b",
</boundary>
<boundary name = "duct04b outer boundary">
      parent_grid = "duct04b",
</boundary>
<boundary name = "duct05b outer boundary">
      parent_grid = "duct05b",
</boundary>
<boundary name = "wake01b outer boundary">
      parent_grid = "wake01b",
</boundary>
<boundary name = "ocean01 outer boundary">
      parent_grid = "ocean01",
</boundary>
<boundary name = "ocean02 outer boundary">
      parent_grid = "ocean02",
</boundary>
<boundary name = "ocean03 outer boundary">
      parent_grid = "ocean03",
</boundary>
<boundary name = "ocean04 outer boundary">
      parent_grid = "ocean04",
</boundary>
<boundary name = "ocean05 outer boundary">
      parent_grid = "ocean05",
</boundary>
<boundary name = "ocean06 outer boundary">
      parent_grid = "ocean06",
</boundary>
<boundary name = "ocean07 outer boundary">
      parent_grid = "ocean07",
```

```
</boundary>
<boundary name = "ocean08 outer boundary">
      parent_grid = "ocean08",
</boundary>
<boundary name = "ocean09 outer boundary">
      parent_grid = "ocean09",
</boundary>
<boundary name = "ocean10 outer boundary">
      parent_grid = "ocean10",
</boundary>
<boundary name = "ocean11 outer boundary">
      parent_grid = "ocean11",
</boundary>
<boundary name = "ocean12 outer boundary">
      parent_grid = "ocean12",
</boundary>
<boundary name = "ocean13 outer boundary">
      parent_grid = "ocean13",
</boundary>
<boundary name = "ocean14 outer boundary">
      parent_grid = "ocean14",
</boundary>
<boundary name = "ocean15 outer boundary">
      parent_grid = "ocean15",
</boundary>
<boundary name = "ocean16 outer boundary">
      parent_grid = "ocean16",
</boundary>
<boundary name = "ocean17 outer boundary">
      parent_grid = "ocean17",
</boundary>
<boundary name = "ocean18 outer boundary">
      parent_grid = "ocean18",
</boundary>
<boundary name = "barge01 outer boundary">
      parent_grid = "barge01",
</boundary>

! outer boundary surface definition


<surface name = "propeller01a outer boundary">
      ijk_range =  1,  1,  1, 35,  1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller01a"
      donor_ijk_range = 61, 61,  1, 35,  1, 42,
</surface>
<surface name = "propeller01a outer boundary">
      ijk_range = 62, 62,1, 35,    1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller01a"
      donor_ijk_range =  2,2,1, 35,1, 42,
</surface>
<surface name = "propeller01a outer boundary">
      ijk_range =  1, 62, 35, 35,  2, 41,
</surface>
<surface name = "propeller01a outer boundary">
```

```
      ijk_range =  1, 62,  2, 35, 42, 42,
</surface>


<surface name = "propeller02a outer boundary">
      ijk_range =  1,  1,  1, 35,  1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller02a"
      donor_ijk_range = 61, 61,  1, 35,  1, 42,
</surface>
<surface name = "propeller02a outer boundary">
      ijk_range = 62, 62,1, 35,    1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller02a"
      donor_ijk_range =  2,2,1, 35,1, 42,
</surface>
<surface name = "propeller02a outer boundary">
      ijk_range =  1, 62, 35, 35,  2, 41,
</surface>
<surface name = "propeller02a outer boundary">
      ijk_range =  1, 62,  2, 35, 42, 42,
</surface>


<surface name = "propeller03a outer boundary">
      ijk_range =  1,  1,  1, 35,  1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller03a"
      donor_ijk_range = 61, 61,  1, 35,  1, 42,
</surface>
<surface name = "propeller03a outer boundary">
      ijk_range = 62, 62,1, 35,    1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller03a"
      donor_ijk_range =  2,2,1, 35,1, 42,
</surface>
<surface name = "propeller03a outer boundary">
      ijk_range =  1, 62, 35, 35,  2, 41,
</surface>
<surface name = "propeller03a outer boundary">
      ijk_range =  1, 62,  2, 35, 42, 42,
</surface>


<surface name = "propeller04a outer boundary">
      ijk_range =  1,  1,  1, 35,  1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller04a"
      donor_ijk_range = 61, 61,  1, 35,  1, 42,
</surface>
<surface name = "propeller04a outer boundary">
      ijk_range = 62, 62,1, 35,    1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller04a"
      donor_ijk_range =  2,2,1, 35,1, 42,
</surface>
<surface name = "propeller04a outer boundary">
```

```
      ijk_range =  1, 62, 35, 35,  2, 41,
</surface>
<surface name = "propeller04a outer boundary">
      ijk_range =  1, 62,  2, 35, 42, 42,
</surface>


<surface name = "tip01a outer boundary">
      ijk_range =  1,  1,  1,  4,  2, 5,
</surface>
<surface name = "tip01a outer boundary">
      ijk_range = 29, 29,  1,  4,  2, 5,
</surface>
<surface name = "tip01a outer boundary">
      ijk_range =  1, 29,  1,  1,  1, 5,
</surface>
<surface name = "tip01a outer boundary">
      ijk_range =  1, 29,  4,  4,  1, 5,
</surface>
<surface name = "tip01a outer boundary">
      ijk_range =  1, 29,  1,  4,  5, 5,
</surface>


<surface name = "tip02a outer boundary">
      ijk_range =  1,  1,  1,  4,  2, 5,
</surface>
<surface name = "tip02a outer boundary">
      ijk_range = 29, 29,  1,  4,  2, 5,
</surface>
<surface name = "tip02a outer boundary">
      ijk_range =  1, 29,  1,  1,  1, 5,
</surface>
<surface name = "tip02a outer boundary">
      ijk_range =  1, 29,  4,  4,  1, 5,
</surface>
<surface name = "tip02a outer boundary">
      ijk_range =  1, 29,  1,  4,  5, 5,
</surface>


<surface name = "tip03a outer boundary">
      ijk_range =  1,  1,  1,  4,  2, 5,
</surface>
<surface name = "tip03a outer boundary">
      ijk_range = 29, 29,  1,  4,  2, 5,
</surface>
<surface name = "tip03a outer boundary">
      ijk_range =  1, 29,  1,  1,  1, 5,
</surface>
<surface name = "tip03a outer boundary">
      ijk_range =  1, 29,  4,  4,  1, 5,
</surface>
<surface name = "tip03a outer boundary">
      ijk_range =  1, 29,  1,  4,  5, 5,
</surface>
```

```
<surface name = "tip04a outer boundary">
      ijk_range =  1,  1,  1,  4,  2, 5,
</surface>
<surface name = "tip04a outer boundary">
      ijk_range = 29, 29,  1,  4,  2, 5,
</surface>
<surface name = "tip04a outer boundary">
      ijk_range =  1, 29,  1,  1,  1, 5,
</surface>
<surface name = "tip04a outer boundary">
      ijk_range =  1, 29,  4,  4,  1, 5,
</surface>
<surface name = "tip04a outer boundary">
      ijk_range =  1, 29,  1,  4,  5, 5,
</surface>


<surface name = "duct01a outer boundary">
      ijk_range =  1,  1,  1, 57,  1, 122,
</surface>
<surface name = "duct01a outer boundary">
      ijk_range = 21, 21,  1, 34,  1, 122,
</surface>
<surface name = "duct01a outer boundary">
      ijk_range = 21, 21, 36, 57,  1, 122,
</surface>
<surface name = "duct01a outer boundary">
      ijk_range =  8, 21,  1,  1,  1, 122,
</surface>
<surface name = "duct01a outer boundary">
      ijk_range =  1, 21, 57, 57,  1, 122,
</surface>
<surface name = "duct01a outer boundary">
      ijk_range =  1, 21,  1, 57,  1,  1,
      boundary_condition = "periodic",
      donor_grid = "duct01a"
      donor_ijk_range =  1, 21,  1, 57, 121, 121,
</surface>
<surface name = "duct01a outer boundary">
      ijk_range =  1, 21,  1, 57, 122, 122,
      boundary_condition = "periodic",
      donor_grid = "duct01a"
      donor_ijk_range =  1, 21,  1, 57,  2,  2,
</surface>


<surface name = "duct02a outer boundary">
      ijk_range = 66, 66,  1, 24,  1, 122,
</surface>
<surface name = "duct02a outer boundary">
      ijk_range =  1, 66, 24, 24,  1, 122,
</surface>
<surface name = "duct02a outer boundary">
      ijk_range =  1, 66,  1, 24,  1,  1,
      boundary_condition = "periodic",
      donor_grid = "duct02a"
```

```
      donor_ijk_range =  1, 66,  1, 24, 121, 121,
</surface>
<surface name = "duct02a outer boundary">
      ijk_range =  1, 66,  1, 24, 122, 122,
      boundary_condition = "periodic",
      donor_grid = "duct02a"
      donor_ijk_range =  1, 66,  1, 24,  2,  2,
</surface>


<surface name = "duct03a outer boundary">
      ijk_range =  1,  1,  1, 35,  1, 122,
</surface>
<surface name = "duct03a outer boundary">
      ijk_range = 53, 53,  1, 35,  1, 122,
</surface>
<surface name = "duct03a outer boundary">
      ijk_range =  1, 53,  1,  1,  1, 122,
</surface>
<surface name = "duct03a outer boundary">
      ijk_range =  1, 53,  1, 35,  1,  1,
      boundary_condition = "periodic",
      donor_grid = "duct03a"
      donor_ijk_range =  1, 53,  1, 35, 121, 121,
</surface>
<surface name = "duct03a outer boundary">
      ijk_range =  1, 53,  1, 35, 122, 122,
      boundary_condition = "periodic",
      donor_grid = "duct03a"
      donor_ijk_range =  1, 53,  1, 35,  2,  2,
</surface>


<surface name = "duct04a outer boundary">
      ijk_range =  1,  1,  1, 23,  1, 122,
</surface>
<surface name = "duct04a outer boundary">
      ijk_range = 53, 53,  1, 23,  1, 122,
</surface>
<surface name = "duct04a outer boundary">
      ijk_range =  1, 53, 23, 23,  1, 122,
</surface>
<surface name = "duct04a outer boundary">
      ijk_range =  1, 53,  1, 23,  1,  1,
      boundary_condition = "periodic",
      donor_grid = "duct04a"
      donor_ijk_range =  1, 53,  1, 23, 121, 121,
</surface>
<surface name = "duct04a outer boundary">
      ijk_range =  1, 53,  1, 23, 122, 122,
      boundary_condition = "periodic",
      donor_grid = "duct04a"
      donor_ijk_range =  1, 53,  1, 23,  2,  2,
</surface>


<surface name = "duct05a outer boundary">
```

```
      ijk_range =  1,  1,  1, 56,  1, 122,
</surface>
<surface name = "duct05a outer boundary">
      ijk_range =  1,  1, 58, 79,  1, 122,
</surface>
<surface name = "duct05a outer boundary">
      ijk_range = 21, 21,  1, 79,  1, 122,
</surface>
<surface name = "duct05a outer boundary">
      ijk_range =  1, 21, 79, 79,  1, 122,
</surface>
<surface name = "duct05a outer boundary">
      ijk_range =  1, 21,  1, 79,  1,  1,
      boundary_condition = "periodic",
      donor_grid = "duct05a"
      donor_ijk_range =  1, 21,  1, 79, 121, 121,
</surface>
<surface name = "duct05a outer boundary">
      ijk_range =  1, 21,  1, 79, 122, 122,
      boundary_condition = "periodic",
      donor_grid = "duct05a"
      donor_ijk_range =  1, 21,  1, 79,  2,  2,
</surface>


<surface name = "wake01a outer boundary">
      ijk_range =  1,  1,  1, 41,  1, 122,
</surface>
<surface name = "wake01a outer boundary">
      ijk_range = 40, 40,  1, 41,  1, 122,
</surface>
<surface name = "wake01a outer boundary">
      ijk_range =  1, 40, 41, 41,  1, 122,
</surface>
<surface name = "wake01a outer boundary">
      ijk_range =  1, 40,  1, 41,  1,  1,
      boundary_condition = "periodic",
      donor_grid = "wake01a"
      donor_ijk_range =  1, 40,  1, 41, 121, 121,
</surface>
<surface name = "wake01a outer boundary">
      ijk_range =  1, 40,  1, 41, 122, 122,
      boundary_condition = "periodic",
      donor_grid = "wake01a"
      donor_ijk_range =  1, 40,  1, 41,  2,  2,
</surface>


<surface name = "propeller01b outer boundary">
      ijk_range =  1,  1,  1, 35,  1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller01b"
      donor_ijk_range = 61, 61,  1, 35,  1, 42,
</surface>
<surface name = "propeller01b outer boundary">
      ijk_range = 62, 62,  1, 35,  1, 42,
      boundary_condition = "periodic",
```

```
      donor_grid = "propeller01b"
      donor_ijk_range =  2,  2,  1, 35,  1, 42,
</surface>
<surface name = "propeller01b outer boundary">
      ijk_range =  1, 62, 35, 35,  2, 41,
</surface>
<surface name = "propeller01b outer boundary">
      ijk_range =  1, 62,  2, 35, 42, 42,
</surface>


<surface name = "propeller02b outer boundary">
      ijk_range =  1,  1,  1, 35,  1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller02b"
      donor_ijk_range = 61, 61,  1, 35,  1, 42,
</surface>
<surface name = "propeller02b outer boundary">
      ijk_range = 62, 62,1, 35,    1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller02b"
      donor_ijk_range =  2,2,1, 35,1, 42,
</surface>
<surface name = "propeller02b outer boundary">
      ijk_range =  1, 62, 35, 35,  2, 41,
</surface>
<surface name = "propeller02b outer boundary">
      ijk_range =  1, 62,  2, 35, 42, 42,
</surface>


<surface name = "propeller03b outer boundary">
      ijk_range =  1,  1,  1, 35,  1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller03b"
      donor_ijk_range = 61, 61,  1, 35,  1, 42,
</surface>
<surface name = "propeller03b outer boundary">
      ijk_range = 62, 62,1, 35,    1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller03b"
      donor_ijk_range =  2,2,1, 35,1, 42,
</surface>
<surface name = "propeller03b outer boundary">
      ijk_range =  1, 62, 35, 35,  2, 41,
</surface>
<surface name = "propeller03b outer boundary">
      ijk_range =  1, 62,  2, 35, 42, 42,
</surface>


<surface name = "propeller04b outer boundary">
      ijk_range =  1,  1,  1, 35,  1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller04b"
      donor_ijk_range = 61, 61,  1, 35,  1, 42,
</surface>
```

```
<surface name = "propeller04b outer boundary">
      ijk_range = 62, 62,1, 35,    1, 42,
      boundary_condition = "periodic",
      donor_grid = "propeller04b"
      donor_ijk_range =  2,2,1, 35,1, 42,
</surface>
<surface name = "propeller04b outer boundary">
      ijk_range =  1, 62, 35, 35,  2, 41,
</surface>
<surface name = "propeller04b outer boundary">
      ijk_range =  1, 62,  2, 35, 42, 42,
</surface>


<surface name = "tip01b outer boundary">
      ijk_range =  1,  1,  1,  4,  2, 5,
</surface>
<surface name = "tip01b outer boundary">
      ijk_range = 29, 29,  1,  4,  2, 5,
</surface>
<surface name = "tip01b outer boundary">
      ijk_range =  1, 29,  1,  1,  1, 5,
</surface>
<surface name = "tip01b outer boundary">
      ijk_range =  1, 29,  4,  4,  1, 5,
</surface>
<surface name = "tip01b outer boundary">
      ijk_range =  1, 29,  1,  4,  5, 5,
</surface>


<surface name = "tip02b outer boundary">
      ijk_range =  1,  1,  1,  4,  2, 5,
</surface>
<surface name = "tip02b outer boundary">
      ijk_range = 29, 29,  1,  4,  2, 5,
</surface>
<surface name = "tip02b outer boundary">
      ijk_range =  1, 29,  1,  1,  1, 5,
</surface>
<surface name = "tip02b outer boundary">
      ijk_range =  1, 29,  4,  4,  1, 5,
</surface>
<surface name = "tip02b outer boundary">
      ijk_range =  1, 29,  1,  4,  5, 5,
</surface>


<surface name = "tip03b outer boundary">
      ijk_range =  1,  1,  1,  4,  2, 5,
</surface>
<surface name = "tip03b outer boundary">
      ijk_range = 29, 29,  1,  4,  2, 5,
</surface>
<surface name = "tip03b outer boundary">
      ijk_range =  1, 29,  1,  1,  1, 5,
</surface>
```

```
<surface name = "tip03b outer boundary">
      ijk_range =  1, 29,  4,  4,  1, 5,
</surface>
<surface name = "tip03b outer boundary">
      ijk_range =  1, 29,  1,  4,  5, 5,
</surface>


<surface name = "tip04b outer boundary">
      ijk_range =  1,  1,  1,  4,  2, 5,
</surface>
<surface name = "tip04b outer boundary">
      ijk_range = 29, 29,  1,  4,  2, 5,
</surface>
<surface name = "tip04b outer boundary">
      ijk_range =  1, 29,  1,  1,  1, 5,
</surface>
<surface name = "tip04b outer boundary">
      ijk_range =  1, 29,  4,  4,  1, 5,
</surface>
<surface name = "tip04b outer boundary">
      ijk_range =  1, 29,  1,  4,  5, 5,
</surface>


<surface name = "duct01b outer boundary">
      ijk_range =  1,  1,  1, 57,  1, 122,
</surface>
<surface name = "duct01b outer boundary">
      ijk_range = 21, 21,  1, 34,  1, 122,
</surface>
<surface name = "duct01b outer boundary">
      ijk_range = 21, 21, 36, 57,  1, 122,
</surface>
<surface name = "duct01b outer boundary">
      ijk_range =  8, 21,  1,  1,  1, 122,
</surface>
<surface name = "duct01b outer boundary">
      ijk_range =  1, 21, 57, 57,  1, 122,
</surface>
<surface name = "duct01b outer boundary">
      ijk_range =  1, 21,  1, 57,  1,  1,
      boundary_condition = "periodic",
      donor_grid = "duct01b"
      donor_ijk_range =  1, 21,  1, 57, 121, 121,
</surface>
<surface name = "duct01b outer boundary">
      ijk_range =  1, 21,  1, 57, 122, 122,
      boundary_condition = "periodic",
      donor_grid = "duct01b"
      donor_ijk_range =  1, 21,  1, 57,  2,  2,
</surface>


<surface name = "duct02b outer boundary">
      ijk_range = 66, 66,  1, 24,  1, 122,
</surface>
```

```
<surface name = "duct02b outer boundary">
      ijk_range =  1, 66, 24, 24,  1, 122,
</surface>
<surface name = "duct02b outer boundary">
      ijk_range =  1, 66,  1, 24,  1,  1,
      boundary_condition = "periodic",
      donor_grid = "duct02b"
      donor_ijk_range =  1, 66,  1, 24, 121, 121,
</surface>
<surface name = "duct02b outer boundary">
      ijk_range =  1, 66,  1, 24, 122, 122,
      boundary_condition = "periodic",
      donor_grid = "duct02b"
      donor_ijk_range =  1, 66,  1, 24,  2,  2,
</surface>


<surface name = "duct03b outer boundary">
      ijk_range =  1,  1,  1, 35,  1, 122,
</surface>
<surface name = "duct03b outer boundary">
      ijk_range = 53, 53,  1, 35,  1, 122,
</surface>
<surface name = "duct03b outer boundary">
      ijk_range =  1, 53,  1,  1,  1, 122,
</surface>
<surface name = "duct03b outer boundary">
      ijk_range =  1, 53,  1, 35,  1,  1,
      boundary_condition = "periodic",
      donor_grid = "duct03b"
      donor_ijk_range =  1, 53,  1, 35, 121, 121,
</surface>
<surface name = "duct03b outer boundary">
      ijk_range =  1, 53,  1, 35, 122, 122,
      boundary_condition = "periodic",
      donor_grid = "duct03b"
      donor_ijk_range =  1, 53,  1, 35,  2,  2,
</surface>


<surface name = "duct04b outer boundary">
      ijk_range =  1,  1,  1, 23,  1, 122,
</surface>
<surface name = "duct04b outer boundary">
      ijk_range = 53, 53,  1, 23,  1, 122,
</surface>
<surface name = "duct04b outer boundary">
      ijk_range =  1, 53, 23, 23,  1, 122,
</surface>
<surface name = "duct04b outer boundary">
      ijk_range =  1, 53,  1, 23,  1,  1,
      boundary_condition = "periodic",
      donor_grid = "duct04b"
      donor_ijk_range =  1, 53,  1, 23, 121, 121,
</surface>
<surface name = "duct04b outer boundary">
      ijk_range =  1, 53,  1, 23, 122, 122,
```

```
      boundary_condition = "periodic",
      donor_grid = "duct04b"
      donor_ijk_range =  1, 53,  1, 23,  2,  2,
</surface>


<surface name = "duct05b outer boundary">
      ijk_range =  1,  1,  1, 56,  1, 122,
</surface>
<surface name = "duct05b outer boundary">
      ijk_range =  1,  1, 58, 79,  1, 122,
</surface>
<surface name = "duct05b outer boundary">
      ijk_range = 21, 21,  1, 79,  1, 122,
</surface>
<surface name = "duct05b outer boundary">
      ijk_range =  1, 21, 79, 79,  1, 122,
</surface>
<surface name = "duct05b outer boundary">
      ijk_range =  1, 21,  1, 79,  1,  1,
      boundary_condition = "periodic",
      donor_grid = "duct05b"
      donor_ijk_range =  1, 21,  1, 79, 121, 121,
</surface>
<surface name = "duct05b outer boundary">
      ijk_range =  1, 21,  1, 79, 122, 122,
      boundary_condition = "periodic",
      donor_grid = "duct05b"
      donor_ijk_range =  1, 21,  1, 79,  2,  2,
</surface>


<surface name = "wake01b outer boundary">
      ijk_range =  1,  1,  1, 41,  1, 122,
</surface>
<surface name = "wake01b outer boundary">
      ijk_range = 40, 40,  1, 41,  1, 122,
</surface>
<surface name = "wake01b outer boundary">
      ijk_range =  1, 40, 41, 41,  1, 122,
</surface>
<surface name = "wake01b outer boundary">
      ijk_range =  1, 40,  1, 41,  1,  1,
      boundary_condition = "periodic",
      donor_grid = "wake01b"
      donor_ijk_range =  1, 40,  1, 41, 121, 121,
</surface>
<surface name = "wake01b outer boundary">
      ijk_range =  1, 40,  1, 41, 122, 122,
      boundary_condition = "periodic",
      donor_grid = "wake01b"
      donor_ijk_range =  1, 40,  1, 41,  2,  2,
</surface>


<surface name = "ocean01 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
```

```
</surface>
<surface name = "ocean01 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean02 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean02 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean02 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean03 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean03 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean03 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean04 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean04 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean04 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean05 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean05 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean05 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean06 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean06 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean06 outer boundary">
```

```
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean07 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean07 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean07 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean08 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean08 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean08 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean09 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean09 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean09 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean10 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean10 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean10 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean11 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean11 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean11 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>
```

```
<surface name = "ocean12 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean12 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean12 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean13 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean13 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean13 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean14 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean14 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean14 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean15 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean15 outer boundary">
      ijk_range = 20, 20,  1, 116,  1, 95,
</surface>
<surface name = "ocean15 outer boundary">
      ijk_range =  1, 20,  1,  1,  1, 95,
</surface>


<surface name = "ocean16 outer boundary">
      ijk_range =  1,  1,  1, 116,  1, 95,
</surface>
<surface name = "ocean16 outer boundary">
      ijk_range =  1, 16,  1,  1,  1, 95,
</surface>


<surface name = "ocean17 outer boundary">
      ijk_range =  1, 151, 37, 37,  1, 33,
</surface>
```

```
<surface name = "ocean17 outer boundary">
      ijk_range =  1, 151,  1, 37, 33, 33,
</surface>


<surface name = "ocean18 outer boundary">
      ijk_range =  1, 151, 37, 37,  1, 33,
</surface>
<surface name = "ocean18 outer boundary">
      ijk_range =  1, 151,  1, 37,  1,  1,
</surface>


<surface name = "barge01 outer boundary">
      ijk_range =  1, 107, 34, 34,  1, 61,
</surface>
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-01-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
August 2016	Final	

4. TITLE AND SUBTITLE

FANS-3D User's Guide (ESTEP Project ER-201031)

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS

Pei-Fang Wang
SSC Pacific
Hamn-Ching Chen
Texas A&M University

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

SSC Pacific
53560 Hull Street
San Diego, CA 92152–5001

Ocean Engineering Program
Zachry Department of Civil Engineering
Texas A&M University
College Station, TX 77843-3136

8. PERFORMING ORGANIZATION REPORT NUMBER

TD 3293

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Environmental Security Technology Certification Program
4800 Mark Center Drive, Suite 17D08
Alexandria, VA 22350-3605

10. SPONSOR/MONITOR'S ACRONYM(S)
ESTEP

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This user's guide details the FANS-3D code model and the procedure of execution of the model. This guide was developed in support of Environmental Security Technology Certification Program (ESTEP) Project ER-201-031.

Dr. Hamn-Ching Chen and his students and collaborators developed the FANS-3D code over the past 25 years. Programmers use this general-purpose computational fluid dynamics (CFD) code for solving the Navier–Stokes equations governing laminar and turbulent flows in body-fitted curvilinear grids. The code employs multi-block overset (chimera) grids, including fully matched, arbitrarily embedded, and/or overlapping grids to facilitate detailed resolution of unsteady laminar and turbulent flows around complex geometries involving arbitrary body motions as well as fluid-structure interactions. Communication between grid components is achieved by Lagrange interpolation at the fringes. The code is fully coupled with the hole-making and donor-finding algorithm, allowing for the relative movement of the grid blocks at each time step for time-domain simulation of fluid-structure interaction problems, including violent free surface motions.

The underlying theory of the local-analytic-based discretization (also known as finite analytic based discretization) is briefly presented in this user's guide.

15. SUBJECT TERMS

Mission Area: Environmental Science
FANS-3D code; Naier–Stokes equations; finite analytic base discretion; Poisson equations;

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Pei-Fang Wang
U	U	U	U	79	19B. TELEPHONE NUMBER *(Include area code)* (619) 553-9192

INITIAL DISTRIBUTION

84300	Library	(1)
85300	Archives	(1)
71750	P.F. Wang	(1)

Defense Technical Information Center
Fort Belvoir, VA 22060–6218 (1)

Hamm-Ching Chen
Ocean Engineering Program
Zachry Department of Civil Engineering
Texas A&M University
College Station, TX 77843-3136 (1)

Approved for public release.

SSC Pacific
San Diego, CA 92152-5001