

Appendix A. Updates to the USURP User's Manual

(Last revision: November 2, 2006 for version 2.36)

This document serves as an appendix to "User's Manual for USURP: Unique Surfaces Using Ranked Polygons, Version 2.32," and contains documentation written subsequent to June 30, 2006 and for versions of the source code subsequent to version 2.32.

Contents:

1. Command Line Functionality Added to USURP version 2.33 (OVERFLOW users)
2. Input File Formats for "Generic" Input
3. Additional Compiler Notes Concerning Linux and the Portland Group Compiler
4. Additional Compiler Notes
5. Using USURP with the DPLR Flow Solver

Section A1. Command Line Functionality Added to USURP version 2.33 (OVERFLOW users)

The command line option `--use-map` can be used to generate a `grid.i.triq` given a `grid.i.tri` file and a matching `usurp.map` file without having to repeat the polygon clipping or triangulation steps in USURP. The procedure to use this capability is as follows:

- a. Run `"usurp --full-surface --ignore-solution < mixsur.i"` in order to generate the `grid.i.tri` file and matching `usurp.map`.
- b. Given a new `q.save` file, run `"usurp --use-map < mixsur.i"` in order to generate the `grid.i.triq` file.

Alternatively, `--watertight` can be used in step "a" in place of `--full-surface`.

Section A2. Input File Formats for "Generic" Input

Users of other codes in which all surface points may be considered active or in which the `i-blank` information is stored at cell centers can convert their data to a generic format for input into USURP. In these cases, the following files are required to be in the working directory: `generic.bc`, `generic.grd`, and in some cases, `generic.ib`. These files are described below.

- a. The `generic.bc` file is an ASCII file containing a list of the solid surfaces. Each line takes the following form:

```
block, i1, i2, j1, j2, k1, k2
```

where "block" is the block number, "i1,i2" is the grid point range of the surface in the i-direction, "j1,j2" is the grid point range in the j-direction, and "k1-k2" is the grid point range in the k-direction. Grid point ranges are provided using a vertex-based numbering system. For example, a j-min surface in the 4th block of a grid system might be specified as

```
4, 1,33, 1,1, 1,17
```

Currently, there is no provision to label components (i.e. there is no provision to group surfaces) under the generic input file format, which effects only the use of the --disjoin command line option.

b. The generic.grd file must contain the grid in ASCII, PLOT3D, multiblock format, which is specified in detail below. As the grid is in ASCII format, the grid coordinates may be expressed using single or double precision. The file should be written using commands equivalent to these:

```
open(unit=3,file="generic.grd",form="formatted")
write(3,*)nb
write(3,*)(ni(n),nj(n),nk(n),n=1,nb)
do n = 1,nb
  write(3,*)((x(i,j,k,n),i=1,ni(n)),j=1,nj(n)),k=1,nk(n)), &
    ((y(i,j,k,n),i=1,ni(n)),j=1,nj(n)),k=1,nk(n)), &
    ((z(i,j,k,n),i=1,ni(n)),j=1,nj(n)),k=1,nk(n))
end do
close(3)
```

c. The generic.ib file provides the cell-centered i-blank values for each computational cell. If the generic.ib file exists, USURP interprets the i-blank values to have the following meanings:

```
iblack = 1:    active or field cells
iblack < 0:   fringe or recipient cells
iblack = 101: orphan cells
iblack = 0:   hole or "out" cells
```

Note that all i-blank values less than zero are considered to be equivalent. If generic.ib does not exist, the i-blank value is assumed to be equal to 1 for all cells.

The generic.ib file should be written using commands equivalent to these:

```
open(unit=3,file="generic.ib",form="formatted")
do n = 1,nb
  write(3,*)((ib(i,j,k,n),i=2,ni(n)),j=2,nj(n)),k=2,nk(n))
end do
```

close(3)

Note that the “ib” values are read as integers and that the index range (e.g. 2:ni(n)) includes only the interior cells (one less than the number of vertices in each direction).

Section A3. Additional Compiler Notes Concerning Linux and the Portland Group Compiler

The compilation of “Triangle” (version 1.6) on linux machines (when the `-DLINUX` CPP flag is required) requires the use of the `fpu_control.h` library. That library uses in-line assembly commands. When Portland Group compilers are used, the in-line assembly feature requires version 6.1 of `pgcc` or later to compile. If this version is not available, `gcc` can be used instead. On non-linux machines, the `-DLINUX` flag should not be used.

Section A4. Additional Compiler Notes

`makemake.pl` is a Perl script that can be used to regenerate the Makefile dependencies. Therefore, changes should be made to `makemake.pl` or `Make.sys`, not to Makefile itself.

The Makefile (via `Make.sys`) has been set-up to handle compilation using the Portland Group compilers (`pgf90` and `pgcc` under Linux), XL Fortran for AIX (`xlf90` and `xlC` under AIX), Sun `f90`, Intel `ifort`, SGI and Lahey/Fujitsu Fortran 95 (`lf95`), and `g95`. Other possibilities exist in `Make.sys` but are commented out because they have not been tested with USURP.

Section A5. Using USURP with the DPLR Flow Solver

Beginning with version 2.36, USURP is capable of reading DPLR input files. The basic usage in this case is simply “`usurp < filename`”, where “`filename`” is the name of the DPLR flow solver input file.

USURP has been written to handle several (currently 8) different CFD working environments (e.g. OVERFLOW, UNCLE-M, CFD-SHIP, etc.). Generally, each working environment is characterized by the existence of a particular file (e.g. “`grid.in`” for OVERFLOW, “`cfd_ship.nml`” for CFD-SHIP, etc.). Because DPLR has no such mandatory file names, USURP defaults to reading DPLR input when it fails to identify any of the other supported environments.

Required input to USURP for DPLR users consists of the DPLR (flow solver) input file and the associated grid file (i.e. the grid file identified in the flow solver input file). USURP supports all three formats for the parallel archive grid file that DPLR supports, namely native unformatted (type 1), XDR binary (type 11), and ASCII formatted (type 21).

In order to use the XDR binary format, USURP must be linked to the FXDR library during compilation. The FXDR library was written by David W. Pierce and can be obtained from http://meteora.ucsd.edu/~pierce/fxdr_home_page.html. The user is responsible for downloading and compiling the FXDR library. USURP will automatically link the library if the FXDR_HOME environment variable has been set to point to the FXDR archive (.a) file.

Most USURP command line options are available to DPLR users. Note, however, that USURP serves only as a pre-processor to DPLR. That is, USURP will generate the panel weights (panel_weights.dat) for use in postflow, but no stand-alone integration of the dependent variables is conducted within USURP.