# SOWER Manual

# Version 1.1

Edition 0.1/25 August 2023

Farhat Research Group (FRG)
Stanford University

This manual was prepared with Texinfo (http://www.gnu.org/software/texinfo).

## Short Contents

## Table of Contents

---

Up: (dir)

# SOWER

---

Next: InputFiles

# 1 Introduction

**SOWER** is a pre/post-processing software which was originally developed at the Center for Aerospace Structures at the University of Colorado to support the parallel execution of the **AERO-S** and **AERO-F3D** codes. This User's Manual documents the usage of **SOWER**.

**SOWER** can be used to perform the following tasks

- Generate the appropriate binary input files for **AERO-F**.
- Generate the binary matcher input files for the parallel execution of **AERO-S** for a coupled **AERO-F/AERO-S** simulation (for example, an aeroelastic computation).
- Assemble the distributed binary output (result) files generated by the

**AERO-S** and **AERO-F** codes and convert them into the ASCII format for visualization with **XPost**.
- Perform other miscellaneous input/output file transformations.

Both the serial and parallel executions of the **AERO-F** code require the generation of appropriate binary input files using **SOWER**. The parallel execution of **AERO-S** for aeroelastic and other coupled **AERO-F**/**AERO-S** simulations requires the conversion of the matcher output files to binary format using **SOWER** (see also **AERO-S**'s User's Manual for other cases where generating appropriate binary input files for this code is required).

---

Next: OutputFiles, Previous: Introduction

# 2 Input Files

This sections describes the input files used by **SOWER** for pre/post-processing the **AERO-S** and **AERO-F** codes. The input file names do not follow any naming convention.

---

Next: Input Files for FEM, Up: InputFiles

## 2.1 Input Files for AERO-F

- `<fluid_mesh_geometry_file>`. This file describes in **XPost** format the unstructured discretization of the fluid domain as well as the surface boundary conditions. The latter are speficied as element sets with the following conventions for naming the element sets: “OutletMoving” for a moving outflow surface boundary, “InletMoving” for a moving inflow surface boundary, “StickMoving” for a moving viscous wall boundary, “SlipMoving” for a moving Euler wall boundary, “Symmetry” for the symmetry boundary, “SlipFixed” for a fixed Euler wall boundary, “StickFixed” for a fixed viscous wall boundary, “InletFixed” for a fixed inflow surface boundary, “OutletFixed” for a fixed outflow surface boundary, “Periodic” for a periodic surface boundary, and “Kirchhoff” for the internal surface suitable for computing the sound field using the Kirchhoff integral method. By default, **AERO-F** computes the aerodynamic forces, moments, lift, and drag only on “SlipMoving” and “StickMoving” boundary surfaces. However, the user can also request that these quantities be computed on any surface identified by a specific tag (for more details see the **CD2TET** and **AERO-F** User's Manuals).
- `<domain_decomposition_file>`. This file is optional. It contains the mesh decomposition of the fluid mesh generated by **Metis** or **XPost** for the sake

of parallel processing. The content and format of this file are described in the **XPost** User's Manual. If not specified, **SOWER** assumes a single subdomain.

- `<Fluid_Matcher_File>`. This optional file contains the nodes on the fluid-structure interface. It is generated by the **MATCHER** software. Its contents and format are described in details in the **MATCHER** User's Manual.

---

Previous: Input Files for AERO-F, Up: InputFiles

## 2.2 Input Files for FEM

- `<FEM_input_file>`. This file is the ASCII input file used by the **AERO-S** code to perform a finite element structural simulation. Its content and format are described in the **AERO-S** User's Manual.
- `<domain_decomposition_file>`. This file is optional. It contains the mesh decomposition of the finite element structural model generated by **AERO-S** or **XPost** for the sake of parallel processing. The content and format of this file are described in the **XPost** User's Manual. If not specified, **SOWER** assumes a single subdomain.
- `<Structure_Matcher_File>`. This optional file contains a list of elements and their local matched points on the fluid-structure interface. It is generated by the **MATCHER** program. Its contents and format are described in details in the **MATCHER** User's Manual.

---

Next: LineCommandInput, Previous: InputFiles

# 3 Output Files

For the purpose of parallel processing, **AERO-F** and **AERO-S** organize their computations and corresponding output around the concepts of subdomains and “clusters”, respectively. A subdomain is a collection cells, and/or dual cells, and/or elements and grid points (among others) that is assigned, possibly with other subdomains, to a single processor. An output cluster pertains to a collection of subdomains that are assigned to a single processor. Hence, the concepts of processors, subdomains, and clusters are related but the number of processors, number of subdomains, and number of clusters can be chosen independently with the constraint that each of the number of clusters and number of processors is less or equal to the number of subdomains. If one or multiple output clusters are assigned per processor, **AERO-F** and **AERO-S** write one or multiple binary output files per result and per processor containing each the trace of the global result on the subdomains defining the cluster. This strategy delivers optimal parallel I/O performance on distributed memory machines. Similarly, if one or multiple output clusters are assigned per “box” of processors, **AERO-F** and **AERO-S** write one or multiple binary output files per result and per box machine containing each the trace of the global result on the subdomains defining the cluster. This strategy delivers optimal parallel I/O performance on hybrid memory machines. Finally, if a single output cluster is assigned to all processors, **AERO-F** and **AERO-S** write a single binary output file per result for the entire mesh. This strategy is convenient for shared memory machines. In pre-processing mode, **SOWER** supports this approach to parallel execution and parallel I/O adopted by **AERO-F** and **AERO-S** by generating for each of them the required subdomain and cluster information based on the standard ASCII input to these codes. In post-processing mode, **SOWER** can be used to assemble all distributed binary files associated with the same result into a single ASCII file.

The following binary output files are generated by **SOWER**. They follow a suffix naming convention for quick identification of their purpose. Similar output files are produced for both the **AERO-F** and **AERO-S** codes.

**Warning: existing files with same names are overwritten!**

---

Next: Post-Processing Data Files, Up: OutputFiles

## 3.1 Pre-Processing Data Files

The following output files are generated by **SOWER** when used to generate the binary input files for the **AERO-S** and **AERO-F** codes.

- `<file_name>.mshxx`. These binary files contain the nodal/element geometry data for the submeshes of the clusters as well as their boundary conditions. Hence, the number of these files is equal to the number of clusters specified by the command line and is indicated by a number at the end of the filename.
- `<file_name>.decxx`. These binary files contain the element-to-subdomain connectivity data for the clusters. Hence, the number of these files is equal to the number of clusters specified by the command line and is indicated by a number at the end of the filename.
- `<file_name>.con`. This single binary file contains the cluster to subdomain connectivity and the subdomain to subdomain connectivity data.
- `<file_name>.matchxx`. These optional binary files contain the matching data. The number of these files will be equal to the number of clusters specified at the command line and is indicated with a two digit number at the end of the filename.
- `<file_name>.xxcpu`. This single text file contains the cpu-to-subdomain

mapping.

Previous: Pre-Processing Data Files, Up: OutputFiles

### 3.2 Post-Processing Data Files

After executing the **AERO-S** or **AERO-F** distributed code, **SOWER** can be used to merge all distributed binary output files associated with the same result into a single ASCII output file readable by **XPost** as explained in the next section.

Previous: OutputFiles

# 4 Line Command Input

The different line-commands for performing the various tasks described in the introduction section of this manual are as follows.

- Pre-Processing Commands
- Post-Processing Commands
- Other Processing Commands

Next: Post-Processing Commands, Up: LineCommandInput

### 4.1 Pre-Processing Commands

This section describes the commands for producing the binary inputs for the **AERO-F** and **AERO-S** codes. The options are the same for either code.

- To generate the binary input files for the **AERO-F** code

```
sower    -fluid                             chooses AERO-F
         -mesh <mesh_geometry_file>         fluid nodal/element geometry and bcs
         [-dec <decomposition_file>]        element-based decomposition
         [-cpu <number_of_cpus>]            default is number of subdomains. This
                                            option can be repeated several times in
                                            the same command to pre-process for
                                            multiple CPU scenarios
         [-cluster <number_of_clusters>]    default is 1 cluster
         [-output <output_file_prefix>]     default is OUTPUT
         [-match <matcher_file>]            required for aeroelasticity
```

The `sower` command is followed by several options which can be in any order. In general, the only required information is the switch `-fluid` which indicates the pre-processing is for the **AERO-F** code, and the switch `-mesh`

`<mesh_geometry_file>`, which provides the geometry file. The other options are necessary only for producing distributed input files that are required by simulations using the MPI protocol. In that case, the number of CPUs and the number of clusters must be specified.

A matcher file should also be specified when pre-processing an aeroelastic simulation.

It is also recommended that an output file prefix be specified so that the files are more easily recognizable.

For example, the following command generates the binary input files with the filename prefix `myfluidmesh` for the **AERO-F** code in aeroelastic mode using 4 CPUs, 2 clusters, and an arbitrary domain decomposition

```
sower -fluid -mesh my_mesh_file -dec my_decomposition_file -cpu 4 -cluster 2 -output
myfluidmesh -match my_matcher_file
```

The above command produces the following output files

- myfluidmesh.msh1
- myfluidmesh.msh2
- myfluidmesh.dec1
- myfluidmesh.dec2
- myfluidmesh.con
- myfluidmesh.match1
- myfluidmesh.match2
- myfluidmesh.4cpu

- To convert a vector result (for example, displacement or displacement sensitivity) associated with the wall boundary surface from the ASCII **XPost** format to the **AERO-F** binary distributed format, for example, for the purpose of initializing an **AERO-F** simulation

| | | |
|---|---|---|
| `sower` | `-fluid -split` | `chooses` **AERO-F** `in split mode` |
| | `-mesh <mesh_geometry_file>` | `fluid nodal/element geometry and bcs` |
| | `-con <connectivity_file>` | `cluster-sub, sub-sub data` |
| | `[-cluster <number_of_clusters>]` | `same number of clusters as that used for generating the binary distributed files mesh_geometry_file` |

| | |
|---|---|
| -result <skin_result> | ASCII input file containing the vector result associated with the wall boundary surface |
| -ascii | requests post-processing of **SDESIGN** data written in ASCII format |
| -bc <boundary code> | specifies that <skin_result> pertains to the surfaces that have the specified boundary code |
| -output <output_prefix> | default is OUTPUT |

This file conversion entails padding with zeroes the value of the displacement field at the interior CFD mesh nodes.

- To generate the binary input files for the **AERO-S** code

| | | |
|---|---|---|
| sower | -struct | chooses **AERO-S** |
| | -mesh <mesh_geometry_file> | structural nodal/element geometry and bcs |
| | [-dec <decomposition_file>] | element-based decomposition |
| | [-cpu <number_of_cpus>] | default is number of subdomains. This option can be repeated several times in the same command to pre-process for multiple CPU scenarios |
| | [-cluster <number_of_clusters>] | default is 1 cluster |
| | [-output <output_file_prefix>] | default is OUTPUT |
| | [-match <matcher_file>] | required for aeroelasticity |

This command is similar to the previous pre-processing command for the fluid code. The `sower` command is followed by several options which can be in any order. The only required information is the switch `-struct` which indicates that the preprocessing is for the **AERO-S** code, and the switch `-mesh <mesh_geometry_file>`, which provides the structural geometry file (**AERO-S**'s input file). The other options are necessary only for producing distributed input files that are required for simulations utilizing the MPI protocol. In that case, the number of CPUs and the number of clusters must be specified. A matcher file should also be specified when preparing an aeroelastic simulation. Finally, it is recommended that an output file prefix be specified so that the files are more easily recognizable. For example, the following command generates the binary input files with the filename prefix `myStruct` for the **AERO-S** code in aeroelastic mode using 2 CPUs, 4 clusters, and and an arbitrary domain decomposition

```
sower -struct -mesh my_mesh_file -dec my_decomposition_file -cpu 2 -cluster 4 -output
myStruct -match my_matcher_file
```

The above command produces the following output files

- myStruct.msh1
- myStruct.msh2
- myStruct.msh3
- myStruct.msh4
- myStruct.dec1
- myStruct.dec2
- myStruct.dec3
- myStruct.dec4
- myStruct.con
- myStruct.4cpu
- myStruct.match1
- myStruct.match2

---

Next: Other Processing Commands, Previous: Pre-Processing Commands,
Up: LineCommandInput

## 4.2 Post-Processing Commands

This section describes the commands for post-processing the output files from the **AERO-S** and **AERO-F** codes. Post-processing of the **AERO-F** data files is necessary for visualization with the **XPost** software. However, post-processing of the structure data files is only necessary if the input files were in binary format (i.e. **SOWER** was used to generate the input files for a simulation in distributed mode).

- To convert the **AERO-F** geometry and connectivity binary files into a **XPost** geometry and connectivity ASCII input file. In this case, the result of the conversion is stored in a file named *topo.xpost* and the `-output` option cannot be used to specify a different file name.

```
sower   -fluid -merge                chooses AERO-F
        -mesh <mesh_geometry_file>   fluid nodal/element geometry and bcs
        -con <connectivity_file>     cluster-sub, sub-sub data
```

- To convert **AERO-F** binary output files to the **XPost** ASCII format.

```
sower   -fluid -merge                  chooses AERO-F
        -mesh <mesh_geometry_file>     fluid nodal/element geometry and bcs
        -con <connectivity_file>       cluster-sub, sub-sub data
        -result <binary_result_prefix> specifies the data file prefix
        [-output <text_output_file>]   specifies the ASCII output file
```

| | |
|---|---|
| `[-binary]` | specifies to output in binary format |
| `[-skin <output_skin_name>]` | outputs data for the fluid-structure interface that can be defined by the boundary codes (see below). If the option -bc is not specified, the fluid-structure interface is defined as the collection of surfaces that have a negative bc code. |
| `[-bc <boundary code>]` | outputs data for the surfaces that have the specified boundary codes (see below) |
| `[-freq <output_frequency>]` | specifies the frequency of outputing results |
| `[-range <first# last#>]` | Requests to output results for a range of time-steps or iteration numbers delimited by first# and last#, with an increment that can be specified using the -freq switch. This switch can be repeated several times on the same command line. |
| `[-load <load_case_name>]` | changes the default load case name which is load |
| `[-nodes <node_set_name>]` | changes the default node set name which is either SkinNodes or FluidNodes or StructureNodes |
| `[-elements <element_set_name>]` | changes the default element set name which is SkinMesh or FluidMesh |

The boundary codes are as follows:

```
-5 = OutletMoving;
-4 = InletMoving;
-3 = StickMoving;
-2 = SlipMoving;
0 = Internal;
2 = SlipFixed;
3 = StickFixed;
4 = InletFixed;
5 = OutletFixed;
6 = Symmetry;
10= Missing;
```

The `sower` command is followed by several options which can be in any order. The only required information is the switch `-fluid -merge` to indicate that fluid data files will be processed, and the switches `-mesh <mesh_geometry_file>` and `-con <connectivity_file>` that are necessary to provide **SOWER** with the basic mesh data. The other switches are optional. The data can be reduced by selecting the frequency of the iterations or time-steps to be processed via the switch `[-freq <output_frequency>]`. Also, specific iterations or time-steps can be processed by using the switch `[-range <res# res#>]`. Finally, the **XPost** names of the load case, node set, and element set can be changed by using the last three options, respectively. For example, the following command generates the ASCII output file mydata.xpost, for use with **XPost** from the fluid data files fluidrun.sol.

```
sower -fluid -merge -mesh myfluidmesh.msh -con myfluidmesh.con -result fluidrun.sol
-out mydata
```

The above command produces the single output file `mydata.xpost`.

- To convert **AERO-F** binary output files associated with $n_1$ subdomains to **AERO-F** binary input files associated with $n_2$ subdomains.

First, apply the previous command (see above) **with the** `-binary` **option** to the binary output files associated with $n_1$ subdomains (this is an assembly step). Then, apply the following command to the binary file resulting from the previous step to create the desired binary input file associated with $n_2$ subdomains.

| | | |
|---|---|---|
| sower | -fluid -split | chooses **AERO-F** |
| | -mesh <mesh_geometry_file> | fluid nodal/element geometry and bcs |
| | -con <connectivity_file> | cluster-sub, sub-sub data |
| | -cluster <number_of_clusters> | same number of clusters as that used for generating the binary distributed files mesh_geometry_file. |
| | -result <result_file> | specifies the binary solution file |
| | -output <output_file prefix> | specifies the binary output file prefix |

Note that the first step uses the connectivity and mesh files associated with $n_1$ subdomains while the second step uses the ones associated with $n_2$ subdomains.

- To convert the **AERO-S** binary output files to the **XPost** ASCII format.

| | | |
|---|---|---|
| sower | -struct -merge | chooses **AERO-S** |
| | -mesh <mesh_geometry_file> | structural nodal/element geometry and bcs |
| | -con <connectivity_file> | cluster-sub, sub-sub data |
| | -result <result_file> | ASCII output will have same file prefix |

The `sower` command is followed by several required options which can be in any order. The mesh geometry file and connectivity file are the same as those used during pre-processing. The output files will have the same filename prefix as the result of the **AERO-S** simulation. For example, the following command generates an ASCII output file called `disps.xpost` for use with **XPost**, from the **AERO-S** data files `disps.out`.

```
sower -struct -merge -mesh myStruct.msh -con myStruct.con -result disps
```

The above command produces the single output file `disps.xpost`.

Previous: Post-Processing Commands, Up: LineCommandInput

## 4.3 Other Processing Commands

This section describes other miscellaneous commands for pre/post-processing the input/output files from the **AERO-S** and **AERO-F** codes.

- To transform **XPost** ASCII output files into **AERO-F**-related binary output files

| | | |
|---|---|---|
| `sower` | `-fluid -split` | chooses **AERO-F** |
| | `-mesh <mesh_geometry_file>` | fluid nodal/element geometry and bcs |
| | `-con <connectivity_file>` | cluster-sub, sub-sub data |
| | `-cluster <number_of_clusters>` | same number of clusters as that used for generating the binary distributed files `mesh_geometry_file`. |
| | `-result <result_file>` | specifies the ASCII **XPost** result file |
| | `-ascii` | specifies that this result file is an ASCII file |
| | `-output <output_file_prefix>` | specifies the binary output file prefix |

In particular, this command can be used to convert a distance-to-the-wall ASCII file `filename.dwall` generated by **CD2TET** (see the **CD2TET** User's Reference Manual) to a binary distributed file.

Furthermore, since the **MESHTOOLS** software — which can be used to perform planar cuts in geometry and result files — operates only on binary **AERO-F** output files, the above command can also be useful when wanting to “cut” a result file available in the ASCII **XPost** format.

In the context of a sensitivity analysis, it is also useful for splitting and transforming the ASCII file containing the derivatives of the mesh position at a wall boundary with respect to a number of shape design variables.