

This chapter provides an overview of using TGrid and instructions for starting it.

- Section 2.1: Using TGrid
- Section 2.2: Starting TGrid
- Section 2.3: Starting Dual Process Build of TGrid

2.1 Using TGrid

Starting from a given boundary mesh (or a boundary mesh with some 2D quadrilateral or 3D hexahedral cells), TGrid generates an unstructured triangular or tetrahedral (or hybrid) grid. In 2D, the boundary mesh consists of nodes and straight edges, whereas in 3D, it consists of nodes and triangular and/or quadrilateral faces.

A boundary mesh file can be obtained from a number of sources including GAMBIT. TGrid creates an interior mesh, from the boundary mesh and writes it to a new file. This file can then be read into the solver, where the solution process and postprocessing occur.

2.1.1 Grid Generation Steps

The basic steps for using TGrid are as follows:

1. Read a simple boundary mesh file or a boundary mesh containing some 2D quadrilateral or 3D hexahedral cells into TGrid.
2. Examine the boundary mesh for topological problems such as free edges and duplicate nodes. When the boundary is topologically correct, you can check a 3D surface mesh for poor face quality.

Many quality-related problems can be solved easily with edge swapping, but more difficult problems may require direct manipulation of the faces and nodes.

3. Generate the volume mesh. You can do this automatically or by proceeding through a series of steps.
 - For hybrid grids, first generate prisms or pyramids, and then generate the triangular or tetrahedral volume cells. If required, you can extend the computational domain by generating more prisms.

- For grids containing only triangles or tetrahedra, you can either use the automatic mesh generation procedure, or perform each step yourself.
4. Check the mesh for problems (quality and location).

The presence of degenerate cells will prevent you from obtaining a solution, and poor cells in critical areas will cause serious accuracy and convergence problems. If such cells cannot be removed or improved, you will have to generate a new mesh by modifying the boundary mesh and/or using different mesh parameters.
 5. Write the mesh to a new file for input to the solver.

2.2 Starting TGrid

The installation process (described in a separate booklet) is designed to ensure that the TGrid is installed properly so that the program can be launched when you follow the appropriate instructions. If not, consult your computer systems manager or your support engineer. The way you start TGrid will be different for UNIX and Windows systems.

Starting TGrid on UNIX system

To start TGrid, type `tgrid 2d` or `tgrid 3d` in the command line of an xterm window. When TGrid starts, it will display a text window and the main menu bar. This is called the console window.

If you start TGrid on UNIX systems without specifying `2d` or `3d`, press the return key to see the following list of options:

```
TGrid version>
2d                custom                host
3d                executable             listen
```

`custom` prompts you for a version that is not in the menu, i.e., a version that is not part of the standard distribution. You can also use it to add options to a standard version name (e.g., `custom "2d -option"`).

`executable` sets the name of the program to be run. The default is `"tgrid"`. You can use this option to give a full pathname if the executable is not in your search path (e.g., `executable "/Fluent.Inc/bin/tgrid"`).

`host` sets the machine and username on which TGrid is to be run. If necessary, you will be prompted for a password in the window in which Cortex was started. The default is to run on the same machine on which Cortex is running.

`listen` tells Cortex to wait a specified period of time for a TGrid session to start up instead of starting TGrid itself.

Starting TGrid on Windows system

There are two ways to start TGrid on a Windows system:


- Click on the **Start** button, select the **Programs** menu, select the **Fluent.Inc** menu, and then select the **TGrid** program item. If the default **Fluent.Inc** program group name was changed when TGrid was installed, you will find the TGrid menu item in the program group with the new name that was assigned, rather than in the **Fluent.Inc** program group.
- Start from an MS-DOS Command Prompt window by typing `tgrid 2d` or `tgrid 3d` at the prompt.

Before doing so, first modify your user environment so that the MS-DOS command utility will find TGrid.

2.3 Starting Dual Process Build of TGrid

The dual process build allows you to run **Cortex** on your local machine and **TGrid** on a remote machine. The advantage of using dual process build is faster response to graphic actions performed when you use TGrid from a remote machine.

For example, if you are handling a big mesh (e.g., underhood mesh) and running TGrid on a faster remote machine with only the display set to your local machine. In this case, the graphics actions (e.g., zoom-in, zoom-out, opening a panel, etc.) can be slow if the remote machine is located very far away or if the network connectivity is slow. To avoid the slow response of the graphics actions, run the dual process build of TGrid.

 When running a dual process build, ensure that the both machines (remote and host) have similar platforms (either both IBM platforms or both non-IBM platforms). You can not use dual process build for IBM host machine and non-IBM remote machine or vice versa.

To start the dual process build of TGrid, do the following:

1. Start TGrid on your local machine using the command `tgrid -serv`.

It will open the TGrid window with the version prompt in the console.

2. Type `listen` and press **Enter** on your keyboard.

You will be prompted for a timeout (the period of time to wait for a connection from remote TGrid). The default value is 300 seconds. You can also specify the timeout as per your requirement. Utilize this time to login to the faster machine and to start TGrid.

3. Press **Enter** on your keyboard again.

A message will prompt you to start TGrid on remote machine with the following arguments:

```
-cx host:p1:p2
```

where,

host is the name of the host (local) machine on which the Cortex is running on.

p1 and p2 are the two integers indicating the connecting port numbers that are used to communicate information between the Cortex on the host machine and TGrid on the remote machine.

4. Login to the remote machine and set the display to the host machine.
5. Start TGrid from remote machine using the following command:

```
tgrid version -cx host:p1:p2
```

Replace **version** by the version that you wish to run (2d or 3d). The host and port numbers are displayed in the TGrid console window.

The GUI commands related to the File menu (e.g., reading file, importing file) and other Select File dialogs do not work for dual process build. Therefore, use the TUI commands.

2.3.1 Startup Options

Before starting TGrid you can obtain information about the available releases. Type `tgrid -help` in an xterm window (on UNIX systems) or the MS-DOS Command Prompt window (on Windows systems).

The options available are:

Usage: [version] [-help] [options]

options:

-cl	following argument passed to tgrid,
-cxarg	following argument passed to cortex,
-cx host:p1:p2	connect to the specified cortex process,
-driver [gl opengl null pex sbx x11 xgl],	sets the graphics driver (available drivers vary by platform),
-env	show environment variables,
-g	run without gui or graphics,
-gu	run without gui,
-gr	run without graphics,
-help	this listing,
-i journal	read the specified journal file,

-n	no execute,
-nocheck	disable checks for valid license file and server,
-project x	write project x start and end times to license log,
-r	list all releases,
-rx	specify release x,
-v	list all versions,
-vx	specify version x,
version	if given, must be first.

i On Windows systems, only `-env`, `-gu` (with restrictions), `-help`, `-i journal`, `-r`, `-rx`, `-v`, and `-vx` are available.

The first three options are for specifying arguments for TGrid and Cortex. Cortex is a process that provides the user interface and graphics for TGrid. When you start TGrid, you actually start Cortex, which then starts TGrid.

On Windows systems, `tgrid -gu` will run TGrid keeping it in its iconified form. The GUI *will* be available if you expand it. This option can be used in conjunction with the `-i journal` option to run a job in background mode.

Options	Description
<code>-cx host:p1:p2</code>	Used only when you start TGrid manually (see Section 2.2: Starting TGrid).
<code>tgrid -driver</code>	Allows you to specify the graphics driver to be used in the TGrid session (e.g., <code>tgrid -driver xgl</code>).
<code>tgrid -env</code>	Lists all environment variables before running TGrid.
<code>tgrid -g</code>	Runs Cortex without graphics and without the graphical user interface. This option is useful if you are not on an X Window display or if you want to submit a batch job.
<code>tgrid -gu</code>	Runs Cortex without the graphical user interface.
<code>tgrid -gr</code>	Runs Cortex without graphics.

To start TGrid and immediately read a journal file, enter the command `tgrid -i journal`, replacing `journal` with the name of the journal file you want to read.

Options	Description
<code>-nocheck</code>	Speeds up the startup by not checking to see if the license server is running. This is useful if you know that the license daemon is running.
<code>-project x</code>	Allows you to record CPU time for individual “projects” separately.
<code>tgrid -project x</code> (where <code>x</code> is replaced by the name of the project)	Extra information related to CPU time will be written to the license manager log file (usually <code>license.log</code> in the <code>license</code> subdirectory of your TGrid installation directory). To determine the CPU time for the project, add the <code>USER CPU</code> and <code>SYSTEM CPU</code> values that appear in <code>license.log</code> . See the installation notes for more information about the license manager.
<code>tgrid version -r</code>	Replaces <code>version</code> with the desired version and lists all releases of the specified version.
<code>tgrid -rx</code>	Runs release <code>x</code> of TGrid. You may either specify a version or wait and specify it later, when prompted by TGrid.
<code>tgrid -v</code>	Lists the available versions.
<code>tgrid -vx</code>	Runs version <code>x</code> of TGrid.

Note: Type `tgrid -n` or use the `-n` option in conjunction with any of the others to see where the (specified) executable is without actually running it.

64-Bit Version

The 64-bit version of TGrid is required only when your problem size exceeds the 32-bit process address space. Most problems with less than two million cells will fit within a 32-bit address space. To use the 64-bit version, add the `-64` option to the command line when starting the program.