FEATFLOW in a Windows environment

Michael Köster[1]

24.03.2002

[1]Thanks to Matthias Grajewski for his intensive tests and for the first version of the section about the GMV.

Contents

Chapter 1

Preface - Overview and platform dependent modifications

FEATFLOW is the main part of a software package for doing computations with finite elements. The whole package consists of three different software modules which can be downloaded from different locations from the Internet. These software packages are:

1. The grid generator "`DeViSoR`" for creating and editing coarse grids. This is available on the FEATFLOW homepage together with a startup script for Windows.

2. The main computation package "`FEATFLOW`". The package consists of two files - on the one hand the main sourcecode and on the other hand a Windows patch file to make it work in a Windows environment. The files are available on the FEATFLOW homepage.

3. The visualization utility *General Mesh Viewer* or short: "`GMV`". This utility can be downloaded from its own homepage.

The installation of FEATFLOW in a Windows environment is not such an easy task as on UNIX stations. The source code is written in FORTRAN 77, but in contrast to a UNIX environment good fortran compilers are rare in the Windows world.

But there are still some possibilities to transfer the package to Windows. The first possibility is to use the free CYGWIN package, which emulates a UNIX environment. The installation with this is possible and described in Chapter 3.

If you have your own FORTRAN compiler that is able to compile and link Windows applications for the command line, you can use it, too. The procedure to install FEATFLOW with such a compiler is described in chapter 5.

If you are interested in writing your own installation scripts, you should look at chapter 6 which gives a more detailed overview about the structure of FEATFLOW and how to compile it.

Computation is not everything - for that reason chapter 4 gives a description how to install the visualization utility *GMV* in a CYGWIN environment. Furthermore there is a module needed to generate the coarse grids on which computations should be done - so chapter 7 gives a description how to install the *DeViSoR* module.

Before we start with a description of the installation procedures of the three modules there are some modifications to the system configuration to be done. Because this is platform dependent and important for all installation steps it will be described here. Additionally we

describe some details about the system configuration of Windows. These details are important to know if you want to install FEATFLOW successfully for you have to change these things frequently in the installation procedure.

1.1 Special modifications: Windows 95/98

The following section is only important if you are using Windows 95/98.

There is a problem in using these operating systems. A lot of installation scripts use temporary variables, the so called *environment variables*, which require an amount of storage in the main memory. In its default setting Windows 95/98 does not reserve enough storage to store all the information !

You can solve this problem with a little modification to your system configuration. Launch a text editor of your choice and open the file

"`C:\CONFIG.SYS`"

Make sure that the following line exists at the end of the file (compare picture 1.1):

```
SHELL=C:\COMMAND.COM /E:3072 /P
```

Save the file and restart your system. If anything goes wrong and you want to restore your old system configuration you can delete the line without being worried about it.

Figure 1.1: Modify your CONFIG.SYS in Windows 95/98

To be more precise: The above line points to the file "`COMMAND.COM`" which is responsible for interpreting any commands in a DOS box. The file resists in the directory `C:\`. The parameter "*/E:3072*" tells the system to increase the storage for environment variables from 512 bytes to 3072 bytes. The Parameter "*/P*" tells the system to execute the file "`AUTOEXEC.BAT`" when the computer is started.

Alternatively if you don't want to change your system configuration you can launch a command line by clicking on `START/EXECUTE` and entering (compare picture 1.2)

command /E:3072

Then you have a DOS box with enough memory available for the installation process. The disadvantage is you can't execute any applications by clicking on them in the Windows Explorer. You have to execute everything in *this* DOS box.

Figure 1.2: How to open a DOS box with enough memory for environment variables

1.2 The Windows environment variable `PATH`

Now we come to a very important thing in the system configuration: The `PATH` environment variable of Windows. You will have to change this frequently within the installation so we will describe here where you can find it and how to change it.

The `PATH` environment variable is a collection of directories, seperated by a ";" sign:

```
PATH=[1st directory];[2nd directory];[3rd directory];...
```

where "[1st directory]" stands for the first directory where Windows searches for files, "[2nd directory]" stands for the second directory and so on; i.e.:

```
PATH=C:\WINDOWS;C:\CYGWIN\BIN;C:\CYGWIN\USR\X11R6\BIN;C:\SUNJAVA
```

When you enter a command at the command line prompt Windows will search these directories to find a file with the name you entered. When such a file can be found Windows will execute it.

Depending on which version of Windows you use you can find the `PATH` environment variable at different places. First we describe how to change it in Windows 2000. Double click the the `MY COMPUTER` icon and select `CONTROL PANEL`. Double click the icon `SYSTEM`, change to the `ADVANCED` tab and click on the button `ENVIORONMENT VARIABLES`. Here you can find all environment variables of Windows 2000 (compare picture 1.3). The variables in the upper part of that tab belong to you, the variables in the lower part are valid for all users that can log on your computer. We suggest changing the `PATH` variable in the lower part (be aware that you need Administrator rights if you want to change anything here). If you want to add a directory to this simply add a semicolon to the end of the line and write the directory behind that. Afterwards commit all the dialogs with `OK`; that's all.

In a Windows 98 environment the `PATH` environment variable can be found at a very different place. Open a text editor of your choice and open the file

```
C:\AUTOEXEC.BAT
```

Figure 1.3: Where to find the `PATH` variable in Windows 2000

Figure 1.4: In Windows 95/98 you can change your `PATH` variable the file `C:\AUTOEXEC.BAT`

This file is the Windows startup script that's executed when your computer starts. To add directories to the `PATH` variable append the following line to the end of the file[1] (compare picture 1.4):

```
PATH=%PATH%;[1st directory];[2nd directory];[3rd directory]
```

where "[1st directory]" stands for the first directory you want to add, "[2nd directory]" for the second directory and so on; i.e.:

```
PATH=%PATH%;C:\CYGWIN\BIN;C:\CYGWIN\USR\X11R6\BIN;C:\SUNJAVA
```

In contrast to Windows 2000 after you change and save this file you have to restart your system before the changes take effekt ! Additionally you mustn't use long filenames in this line which means that something like

```
PATH=%PATH%;C:\Programs\Microsoft Visual Studio\VC98\BIN
```

is not allowed. Instead of that you have to use the *short names* of the directories (which you can see if you enter the `DIR` command on your command line), i.e. something like:

```
PATH=%PATH%;C:\Programs\Micros~2\VC98\bin
```

[1] The expression "`%PATH%`" means that the directories you enter to that line will be appended to your `PATH` variable. If this expression is not given the content of the `PATH` variable will be overridden - as a result some of your previosly installed applications in Windows might not run !

Chapter 2

Directory structure of FEATFLOW

This chapter gives a brief description about the main directory structure of the main computation package FEATFLOW. Use it as an introduction and later as a short reference where to find what. A more detailed view of the contents of the directories can be found in the main manual of FEATFLOW.

After unpacking / untaring the FEATFLOW packages you should have a structure like this:

.....

`/featflow1.2`
This is the installation directory of FEATFLOW. Here you can find two installation files:

- "`install_featflow.bat`" - This is a Windows batch file. Call this file if you want use your own compiler for compiling the whole code. Be aware of modifying the file "`FEAT_GLOBALS.BAT`" in the `MAKEFILES/INTELWIN` directory ! For details about using your own compiler see chapter 5.
- "`install_featflow`" - This is the UNIX / CYGWIN installation bash script for installing FEATFLOW. In a UNIX / CYGWIN environment call this to start the installation. See section 3 for details.

`/featflow1.2/featflow`
Here you can find readme files. If you installed the package by the above Windows batch file, you can also find a file here named "`FEAT_GLOBALS.BAT`" after installation. This contains the installation parameters. See chapter 5 for details.

`/featflow1.2/featflow/application`
This is the application directory of FEATFLOW. Here you can find test applications, examples and standard benchmarks. Especially:

`/featflow1.2/featflow/application/comp`
The *benchmark computation suite* of FEATFLOW. Here you can find two files:

- "`comp_start.bat`" - A batch file for compiling the benchmark ("*benchmark computation suite*") in Windows. Call it if you want to use your own FORTRAN compiler.

- "`comp.start`" - A bash shell script for compiling and running the benchmark in a UNIX / CYGWIN environment.

`/featflow1.2/featflow/application/example`
An example application of FEATFLOW. Here you can find one file:

- "`example.bat`" - A batch file for compiling the application. Call it if you want to use your own FORTRAN compiler.

If you want to start the example application in a UNIX / CYGWIN environment, use the installation script or the file:

`/featflow1.2/featflow/object/makefiles/[architecture]/example.m`

`/featflow1.2/featflow/application/user_start`
A good starting point for modifying and writing your own applications with FEATFLOW.

`/featflow1.2/featflow/application/workspace`
This directory contains some include files for different applications. Depending on how the FEATFLOW package is compiled (for *benchmark*, for *example* or what else) different include files are used. The basic versions of these include files are placed here.

`/featflow1.2/featflow/HelpApps`
This directory contains some applications that are only used in the installation process with a user defined compiler.

`/featflow1.2/featflow/graphics`
Some graphic programs that are commonly used in a UNIX environment: GMV, GNUPLOT and others.

`/featflow1.2/featflow/manuals`
A directory which contains GZipped versions of the FEATFLOW manuals. Extract it in a directory and use LaTeX to compile them.

`/featflow1.2/featflow/windows_manuals`
A directory which contains Zipped versions of the installation manual for FEATFLOW in a Windows environment.

`/featflow1.2/featflow/object`
The basic directory for binary libraries and all the makefiles. To be more precise:

`/featflow1.2/featflow/object/libraries`
After installation you can find here a subdirectory with the name of the architecture you compiled FEATFLOW for, containing all the libraries of FEATFLOW in a binary version. I.e. if you install FEATFLOW in a CYGWIN environment, you will get a `LIBWIN` subdirectory here with all the libraries.

`/featflow1.2/featflow/object/makefiles`
This subdirectory contains the makefiles for the different machine architectures.

- `SOURCE` - Contains template files for the compilation process in a UNIX / CYGWIN environment. The UNIX installation script takes these files and generates special versions of them depending on the target architecture you choose at the beginning of the installation. These "derived" versions will be placed in a similar directory like `SOURCE`. I.e. if you install FEATFLOW in a CYGWIN environment for *Intel/x86 Windows (CYGWIN)*, you will get a `CYGWIN` subdirectory on the same level as the `SOURCE` directory containing the installation scripts for the CYGWIN installation.
- `INTELWIN` - contains batch files for the compilation process in Windows with your own Windows compiler. If you want to use your own compiler, you must look into this directory for the files you have to modify. See chapter 6.

`/featflow1.2/featflow/source`
This subdirectory containes the real source code of FEATFLOW. All subdirectories of this directory contain the same structure. To be more precise here is an example of the `PP2D` subdirectory in this `SOURCE` directory:

`/featflow1.2/featflow/source/pp2d`
This is the directory of PP2D containing the following subdirectories:

- `DEV` - The development version of the application. Here is the basic code for compiling an executable, and here you can find a good starting point for programming your own applications.
- `MG` - This is the home for the MULTIGRID library of this application. The library that is compiled from this code is placed in the directory

  `/featflow/object/libraries/[architecture]`

  and used by a couple of other applications.
- `SRC` - This is the home for the general library of the application. The library that is compiled of this code is placed in the directory

  `/featflow/object/libraries/[architecture]`

  and used by a couple of other applications.

`/featflow1.2/featflow/utility`
A directory with some additional utilities for UNIX environments. Look into the main manual of FEATFLOW for the contents.

Chapter 3

FEATFLOW and CYGWIN

If you don't have a FORTRAN compiler for Windows you can compile the FEATFLOW package using the CYGWIN environment which emulates a UNIX environment in Windows. Beneath CYGWIN there exist also some more UNIX emulators in the Windows world like the utility INTERIX from Microsoft. But the advantage using CYGWIN is that you can dowload it from the Internet it supports the XFree86 X-Server which can be used to execute the visualtisation utility GMV. The installation of CYGWIN and FEATFLOW will be described here, the installation of XFree86 and the steps to run GMV in the next chapter.

3.1 Installation of CYGWIN

The following is only important if you use Windows 2000: Before the installation make sure that you have Administrator rights in Windows. If you use Windows as a terminal over the network by the Windows terminal services, make sure that you have installation rights - that means enter the command "`change user /install`" at the Windows command prompt !

Now to the installation: First of all call the URL

`http://www.cygwin.com`

and get the file "`SETUP.EXE`", the setup routine of CYGWIN. After downloading the file execute it to start the installation process of CYGWIN. Follow the instructions on the screen:

You should tell the setup application do install CYGWIN directly from the internet - the necessary files are then automatically downloaded from a server and installed. Then set the *default text file type* to *UNIX* and the *install for* item to *all*. The installation root directory of CYGWIN should be set to `C:/cygwin` (compare picture 3.1).

Figure 3.1: The SETUP application

You can tell the setup application the server where the files should be downloaded from and where they should be stored on your local hard disk (compare picture 3.2 and 3.3)

Figure 3.2: Download the files to the local hard drive

Figure 3.3: Choose the server to download from

After these steps the setup application wants to know, which packages to install. You have to add at least the following packages if they are not in the list to install (compare picture 3.4) - the sourcecode can also be installed but is not needed:

category	package	later used for
archive	unzip	packing / unpacking
archive	zip	packing / unpacking
base	gzip	packing / unpacking
devel	gcc	compiling the code
devel	make	compiling the code
devel	mingw-runtime	compiling help applications
graphics	opengl	GMV
net	openssh	connections to UNIX servers for GMV
net	openssl	connections to UNIX servers for GMV
shells	tcsh	compiling the code
utils	bzip2	packing / unpacking
utils	time	running the sample applications

After you click *NEXT* the setup application will install the files. When SETUP has finished you should have a link to the bash shell in your START menu.

Figure 3.4: Installation of the packages

After CYGWIN has been installed to your system there are still two important modifications to do. On the one hand you have to modify the `PATH` variable of CYGWIN so that the shell will search in the current directory if you want to start an application. On the other hand you have to add the path of the CYGWIN binary files to the `PATH` variable of windows.

The first thing can easily be done with a text editor. Open the file

"`/cygwin/etc/profile.`"

in a text editor of your choice (in this case WORDPAD is preferred instead of NOTEPAD because the file is written in a UNIX environment - WORDPAD can handle UNIX files, NOTEPAD not). Find the line

```
PATH="/usr/local/bin:/usr/bin:/bin:$PATH"
```

at the very beginning of the file. Change it to

```
PATH=".:/usr/local/bin:/usr/bin:/bin:$PATH"
```

and save the file.
The second and last thing is to make sure that the environment variable `PATH` points to the dynamic link libraries of CYGWIN.

- In Windows 2000 open to the *environment* tab of Windows and add the path of the CYGWIN libraries/binaries to your `PATH` variable (seperated by a ";", compare picture 3.6), i.e. `C:\CYGWIN\BIN`.
- If you use Windows 95/98 you have to add the path of the CYGWIN libraries / binaries to your `PATH` variable in the file "`C:\AUTOEXEC.BAT`" (seperated by a ";", compare picture 3.7).

Figure 3.5: Change the path variable of CYGWIN

3.2 Installation of FEATFLOW

The installation procedure in Windows is almost identical that in UNIX which is described in the main manual. You need two files:

1. the FEATFLOW package ("`featflowXXXX.tar.gz`") and
2. the windows installation patch ("`featflowwinXXXX.zip`").

The files can be downloaded from the FEATFLOW homepage:

`http://www.featflow.de`

First create a directory for FEATFLOW and copy the two packages into this - for example

`/cygwin/usr/featflow`

Launch the BASH shell of CYGWIN, change to this directory and extract the files - first the FEATFLOW package and then the patch file:

```
Administrator@WINDOWSHOST ~
$ cd /usr/featflow

Administrator@WINDOWSHOST /usr/featflow
$ gunzip featflow1.2d.tar.gz

Administrator@WINDOWSHOST /usr/featflow
$ tar -xvf featflow1.2d.tar
...

Administrator@WINDOWSHOST /usr/featflow
$ unzip -o featflowwin1.2d.zip
...

Administrator@WINDOWSHOST /usr/featflow
$
```

The main difference to the installation of FEATFLOW in a UNIX environment is that CYGWIN doesn't support the "`F77`" and "`CC`" compiler and the "`CSH`" shell. But fortunately the "`G77`", "`GCC`" and the "`TCSH`" does the same job as well. First copy the above files to another name:

Figure 3.6: Add the path of the CYGWIN binaries to the environment variable `PATH` of Windows 2000 (seperated by a ";")

Figure 3.7: Add the path of the CYGWIN binaries to the environment variable `PATH` in the file `AUTOEXEC.BAT` of Windows 95/98 (seperated by a ";")

Figure 3.8: The files of the FEATFLOW package, ready to be decompressed

```
Administrator@WINDOWSHOST /usr/featflow
$ cp /bin/g77 /bin/f77

Administrator@WINDOWSHOST /usr/featflow
$ cp /bin/gcc /bin/cc

Administrator@WINDOWSHOST /usr/featflow
$ cp /bin/tcsh /bin/csh

Administrator@WINDOWSHOST /usr/featflow
$
```

Afterwards you can launch the install script:

```
Administrator@WINDOWSHOST /usr/featflow
$ cd featflow1.2

Administrator@WINDOWSHOST /usr/featflow/featflow1.2
$ ./install_featflow



BE AWARE THAT YOU HAVE PROVIDED THE FORTRAN COMPILER OF YOUR
CHOICE
   UNDER F77, THE C COMPILER UNDER CC, AND THAT YOU USE THE RIGHT
        TIMING ROUTINES (ZTIME.F, TIMEWRAP.C) IN FEAT2D/SRC.

         UNDER PPC-LINUX: YOU HAVE TO UPDATE THE RESOURCES.
...
```

Now a menu will be displayed where you can choose the type of installation:

```
FEAT/FEATFLOW Release 1.2 installation program    V1.2
-------------------------------------------------------
Configuration:
-0- general platform
-1- SUN Sparcstation
-2- SUN Ultra1
-3- SUN Ultra2
-4- IBM RS/6000
-5- IBM RS/6000 PPC
-6- IBM RS/6000 Power2
-7- SGI R4000
-8- SGI R8000
-9- SGI R10000
-a- General Unix/Linux (G77/EGCS)
-b- Intel/x86 Linux (EGCS)
-c- MAC/PPC   Linux (EGCS)
-d- HP C110
-e- DEC Alpha Workstation
-f- DEC Alpha Linux
-g- Intel/x86 Windows (CYGWIN)
-h- Intel/x86 Windows (CYGWIN) max. optimization
-i- User defined
---------------------------------------
Selection:
```

At this point choose "g" for the CYGWIN installation or "h" for an optimized CYGWIN installation. If you choose the optimized installation the applications will run much faster than in normal mode. The disadvantage is you need an Intel Pentium III or above and the installation step will need more time.

Then you will be asked if you have *performance libraries* - choose "n" because the G77 doesn't have performance libraries.

The next question is whether to compile *AVS modules* or not. Choose "n" because the AVS modules are not part of the FEATFLOW package and so compiling them would not be possible.

Now follow the instructions on the screen. When you are asked whether to install FEATFLOW or not, choose "y" and FEATFLOW will be installed. That's all.

Chapter 4

XFree86, GMV and CYGWIN

If you want to visualize the results of your computations you can use the *General Mesh Viewer* or short: *GMV*. This application can be downloaded from the URL:

http://www-xdiv.lanl.gov/XCM/gmv/GMVHome.html

In a Windows environment you have two possibilities to use this software:

1. Use the commercial UNIX emulator *Interix* from Microsoft together with a commercial X-Server.

2. Use the XFree86 X-Server together with CYGWIN.

There exist binaries for both situations at the above URL. In this chapter the installation of XFree86 is described as well as the installation of GMV in this environment. The installation of GMV within *Interix* is not described here.

Additionally if you have installed an X-Server (commercial or XFree86) on your computer and your computer is connected to a UNIX workstation via a network, you have the possibility to run GMV on the UNIX machine and redirect the output to your screen in Windows. You can use this method for example in a university if you have a slow computer in a UNIX network with very fast UNIX workstations. The necessary steps to do this is described later in section 4.5.

4.1 Installation of XFree86 for CYGWIN

If you want to use the GMV you first have to install the XFree86 X-Server which gives a graphical interface to any application that runs on or is ported from UNIX. The installation and configuration of the XFree86 package is described here.

You first need the precompiled binaries of the package - they can be downloaded from the CYGWIN homepage or from the XFree86 homepage itself:

`http://www.xfree86.org`

There's also an installation manual of the XFree86 project on the CYGWIN homepage so this manual gives only a brief description of how to install it.

Copy the compressed libraries to your `/cygwin/tmp` directory - there should be the following files:

filename	size	status	function
extract.exe.gz	119 KB	required	used by the installation script
startup-scripts.tgz	1.4 KB	required	example X-Server and client startup scripts
Xbin.tgz	12.1 MB	required	executables for xterm & twm & etc.
Xdoc.tgz	35.2 KB	required	formatted documentation
Xetc.tgz	593 KB	required	configuration files
Xf100.tgz	12.1 MB	optional	100 dpi & 75 dpi fonts are used by default
Xfcyr.tgz	368 KB	optional	Cyrillic fonts
Xfenc.tgz	311 KB	required	font encodings
Xfnts.tgz	13.9 MB	required	75 dpi fonts
Xfscl.tgz	1.6 MB	optional	Speedo and Type1 scalable fonts
Xfsrv.tgz	248 KB	optional	X Font Server
Xhtml.tgz	703 KB	optional	documentation html format
Xinstall.sh	34 KB	required	XFree86 installation script
Xjdoc.tgz	107 KB	optional	XFree86 Japanese documentation
Xlib.tgz	95 KB	required	X11R6/lib/ files
Xman.tgz	616 KB	required	man pages
Xnest.tgz	1.2 MB	optional	Xnest nested X-Server
Xprog.tgz	553 KB	optional	additional files needed to compile programs
Xprt.tgz	846 KB	optional	X Print & Xprt server
Xps.tgz	5.9 MB	optional	documentation in Postscript format
Xvfb.tgz	1.3 MB	optional	X Virtual Frame Buffer & Xvfb server
Xxserv.tgz	1.3 MB	required	the Cygwin/XFree86 X-Server

Launch the bash shell of CYGWIN and enter the following commands:

```
Administrator@WINDOWSHOST ~
$ cd /tmp

Administrator@WINDOWSHOST /tmp
$ gunzip extract.exe.gz

Administrator@WINDOWSHOST /tmp
$ cp extract.exe /bin

Administrator@WINDOWSHOST /tmp
$ ./Xinstall.sh

                Welcome to the XFree86 4.2.0 installer

You are strongly advised to backup your existing XFree86
installation before proceeding.  This includes the /usr/X11R6 and
/etc/X11 directories.  The installation process will overwrite
existing files in those directories, and this may include some
configuration files that may have been customised.

If you are installing a version different from 4.2.0, you may need
an updated version of this installer script.

Do you wish to continue? (y/n) [n]
```

Press the "y" key and the installation will start. Follow the instructions on the screen.

After the install script has completed you have to install the startup scripts. To install them, enter the following commands:

```
Administrator@WINDOWSHOST /tmp
$ cp startup-scripts.tgz /usr/X11R6/bin/

Administrator@WINDOWSHOST /tmp
$ cd /usr/X11R6/bin/

Administrator@WINDOWSHOST /usr/X11R6/bin
$ tar -xzf startup-scripts.tgz
```

The next step in installing XFree86 is only important if you installed CYGWIN to a nonstandard directory, that means to a directory different to

`[drive]:\CYGWIN`

In this case you have to modify the startup batch script of XFree86 or it cannot be executed. Open the file

"`startxwin.bat`"

in a text editor of your choice. Find the line

```
SET CYGWIN_ROOT=\cygwin
```

As you can read from the comments in the file you have to change this line to point to the root directory of CYGWIN (compare picture 4.1).

Figure 4.1: Place the root directory of CYGWIN into the startup script of XFree86 if it's different to `[drive]:\cygwin`

The last step in installing XFree86 is to add the directory of the XFree86 binaries to the `PATH` environment variable of Windows - Look at section 1.2 how to do that. For example if CYGWIN is installed to `C:\CYGWIN` the path you have to add is:

`C:\CYGWIN\USR\X11R6\BIN`

That's all - the installation of XFree86 is now complete.

4.2 How to launch XFree86

Launching the X-Server is easy. There are two ways:

1. With the Windows Explorer double click on the file

"`/cygwin/bin/X11R6/bin/startxwin.bat`"

2. If you want to launch XFree86 with your CYGWIN BASH enter the following command:

"`startx`"

That's all.

4.3 How to configure XFree86 for GMV

Under normal circumstances you can use the X-Server "as is" without any modifications. But perhaps you want to customize it. Examples for such situations where you have to change the standard configuration of the X-Server are the following:

- You want to change the size of the window of the X-Server or you want it to run in full screen mode.
- You want to change the background color.
- Your Windows desktop runs only with 256 colors and if you start GMV you get an error message (compare section 4.5):

  "Cannot allocate enough colors for material buttons.'

  which means that your desktop doesn't have enough colors.

All modifications can be done in the startup files. Open the Windows startup script of the X-Server

"`/cygwin/bin/X11R6/bin/startxwin.bat`"

in a text editor of your choice. (For simplicity only the modifications the the file "`startxwin.bat`" are described here - changes to the UNIX startup script `startx` are similar but a little bit more complicated.)

4.3.1 Change the window size of the X-Server

If you use Windows 2000 find the line:

```
start /B XWin -screen 0 1024 768
```

If you use Windows 95/98/Me find the line:

```
start XWin -screen 0 1024 768
```

In this line you can change the size of the window. I.e. if you want the X-Server to run in Windows 2000 with a 800x600 window change it to

```
start /B XWin -screen 0 800 600
```

If you want the X-Server to run with a maximized window in Windows 2000 change it to

```
start /B XWin
```

and if you want to run it in fullscreen in Windows 2000 change it to

```
start /B XWin -fullscreen
```

If you want to know all the parameters that can be used in the call of "`XWin`" start a DOS box and type the following command:

"`XWin /?`"

Then a list with all possible parameters is displayed.

4.3.2 Change the background color of the X-Server

The background color of the X-Server can be changed with the appropriate UNIX command. Find the line

```
xsetroot -solid aquamarine4
```

at the very end of the file and change it the way you like, for example to

```
xsetroot -solid darkblue
```

4.3.3 XFree86 with 256 colors

If you run XFree86 on a desktop with 256 colors and you have an application like GMV that requires true colors, this application might not start. In this case you can try to launch the X-Server in the so called *compatibility mode*. Normally XFree86 uses DirectX to display graphical output. With the compatibility mode you can force the X-Server to use the Windows GUI. This can help to use applications that require true colors in an environment with only 256 colors, but the disadvantage is that the X-Server will take a long time to start and there will be intensive color flashing later when you move your mouse in the window of the X-Server.

To force the X-Server to compatibility mode use the parameter

"`-engine 1`"

in the call of the file "`XWin`". That means in Windows 2000 find the line

```
start /B XWin -screen 0 1024 768
```

in the file "`startxwin.bat`" and change it to

```
start /B XWin -engine 1 -screen 0 1024 768
```

4.4 How to install and start GMV in Windows

There exists a version of the mesh-viewer GMV which can be executed in a CYGWIN environment. To benefit from this version (beta 2.9), download the file "`cygwinogl.tar.Z`" from

`http://www-xdiv.lanl.gov/XCM/gmv/GMVHome.html`

Copy the file into a directory of your choice - for example

`/cygwin/usr/gmv`

Start the CYGWIN BASH, change the current directory to this and type the following:

```
Administrator@WINDOWSHOST ~
$ cd /usr/gmv

Administrator@WINDOWSHOST /usr/gmv
$ gunzip cygwinogl.tar.Z

Administrator@WINDOWSHOST /usr/gmv
$ tar -xvf cygwinogl.tar

...

Administrator@WINDOWSHOST /usr/gmv
$ mv cygwinogl gmv.exe
```

That's all. If you now want to start the GMV first launch the XFree86 X-Server, change into the above directory and type

`gmv`

GMV should start now. If not there are two typical errors which can easily be avoided:

1. There's an error reporting that the file "`LIBICE.DLL`" can not be found. In this case please make sure that the directory of the XFree86 binaries is added to the `PATH` variable of Windows as described in section 4.1, because this file resists in this directory !

2. There is the error message

   "Error: no RGB visual with depth buffer"

   This means you tried to start GMV on a desktop with 256 colors. This is not possible within Windows - you have to change your desktop at least to HiColor !

Remark: GMV requires three mouse buttons. If you use a scroll mouse try to press your scroll wheel - it can normally be used as the third mouse button !

4.5 GMV in a UNIX network

As mentioned above if your Windows computer is connected to a UNIX network you have the possibility to run GMV on a UNIX workstation and redirect the output to your Windows desktop. For this purpose you first have to launch your X-Server on the Windows PC and start a shell (this is done automatically if you use "`startxwin.bat`" of XFree86). The output of the UNIX workstation will later be displayed in the window of your X-Server.

Because the UNIX workstation you want GMV to run on wants to use your X-Server, you first have to allow the workstation to connect to it. To simplify that process for a first test you should allow all workstations to connect to your Windows X-Server:

```
bash-2.05a$ xhost +
access control diabled, clients can connect from any host
bash-2.05a$
```

Now connect to your UNIX workstation and set the `DISPLAY` environment variable to your windows server. Afterwards you can start the GMV on that machine:

```
bash-2.05a$ ssh myunixworkstation -l myname
myname@myunixworkstation's password:
Authentication successful.
Last login: Thu Dec 13 2001 19:30:03 +0100 from myunixworkstation
Sun Microsystems Inc.   SunOS 5.7       Generic October 1998
No mail.
bash-2.05a$ setenv DISPLAY WINDOWSHOST:0.0
bash-2.05a$ gmv
```

That's all. GMV should now start on the UNIX workstation. If not there are three typical errors:

1. There is an error message

   "Segmentation fault"

   Please check the content of the `DISPLAY` environment variable. The error message means that GMV can't find an X-Server to display its output.

2. There is an error message

   "XLib: Client is not authorized to connect to Server"

   Please make sure that the UNIX workstation you are connected to has the rights to connect to the XFree86 X-Server. This is done by the `xhost` command as mentioned above.

3. There is an error message

   "Cannot allocate enough colors for material buttons.'
   "New colormap created, color flashing might occur."
   "X error of failed request: BadAccess ..."

You tried to start GMV on a desktop with 256 colors. GMV requires true colors ! You should switch your desktop at least to HiColor mode to avoid this error. Alternatively you also have the possibility to force GMV on the UNIX machine to use 256 colors. This can be done by switching your X-Server to *compatibility mode* - see section 4.3.3 how to do this.[1]

[1] This trick works only with true UNIX machines - it doesn't work if you use the CYGWIN binary of GMV in Windows as described in section 4.4 !

Chapter 5

FEATFLOW with a user defined compiler

To compile FEATFLOW with a user defined compiler is a little bit tricky. In contrast to the world of UNIX there is no standard FORTRAN compiler available on every system and no "`MAKE`" command which allows easy compiling. One way out of this is to use so called *batch files* that can be executed within the command line console of Windows. Such files are equivalent to the so called *makefiles* that are commonly used in UNIX environments.

Of course these files can only be templates so that they can be used with as many compilers as possible. This makes it necessary that the end user - that means you - has to do some pre-installation steps so that the scripts will work in the right way. This means you must have a compiler installed to your system, which at least understands FORTRAN 77 and you have to know how to use it !

5.1 Introduction into compiling

The main installation script of FEATFLOW is the file

"`INSTALL_FEATFLOW.BAT`"

which you can find in your FEATFLOW directory. First of all a small introduction what will happen when you call this file from the command line:

1. The file generates another batch file named "`FEAT_GLOBALS.BAT`" with the current installation settings / directory.
2. A couple of "`MAKE_xxx.BAT`" files are called. They copy templates of batch files into all the source code directories and call them afterwards. These scripts are responsible for compiling all the code. You don't have to worry about that.

The main important role in the compiling the code with the right settings plays the file "`FEAT_GLOBALS.BAT`" which will be generated during the installation. You will have to change the template for this file before installation so that it matches to your configuration.

5.2 Preparing the installation: Specify the compiler

After unpacking / untaring the FEATFLOW packages change to the directory

`featflow1.2/featflow/object/makefiles/INTELWIN`

In this directory you will find a lot of batch files. They all are used in the compilation process. Find the file

"`FEAT_GLOBALS.BAT`"

in this directory. This file is a template for another file which will be created later in the installation process. Open it in a text editor of your choice. It should contain the following statements:

```
:FSTD

set F95=ifl.exe /4Yportlib /nologo

set F95OBJONLY=/c

set LINKEXE=link.exe
set LINKLIB=lib.exe

set EXENAME=/out:
set LIBNAME=/out:

set LIBDIR=%FEATFLOW%\object\libraries
set LIBTITLE=INTELWIN
set FLIBRARIES=%LIBDIR%\%LIBTITLE%

set MAKEDIR=%FEATFLOW%\object\makefiles
set MAKENAME=INTELWIN

set OBJEXTENSION=.OBJ
set LIBEXTENSION=.LIB

set COMPLIBEXT=.LIB

set USELIBPARAM=

set LIBBLAS="%FLIBRARIES%\libblas%LIBEXTENSION%"

set FINCLUDE=%FEATFLOW%\application\workspace
```

The options above are usable for the INTEL FORTRAN 95 compiler in conjunction to the Microsoft linker of the Visual C++ 6.0 package. If you have this combination you don't need to change anything and you can proceed to section 5.5. If not, please change the the following definitions for environment variables:

- `F95` - This environment string contains the command line that is used for calling the compiler. As you can see above the compiler named "`IFL.EXE`" is called together with some standard options. Change this line so that it matches to your compiler !
- `F95OBJONLY` - When compiling the source code the compiler should produce .OBJ files which can then be linked together by a linker to an executable or a library. This environment variable contains the parameter for the compiler to produce only an .OBJ file and not an executable when called.

- `LINKEXE` - This string contains the command line that is used to call the linker to produce an executable file.
- `LIBNAME` - This string contains the command line that is used to call the linker to produce a library file.
- `EXENAME` - This is the parameter which tells the linker the name of the output file for an executable file. Normally this should be "`/out:`" or "`/o `". If you want to use a standalone parameter like the "`/o`" that is not concatted with the filename be aware to put a space character at the end of the line because the parameter here is directly concatted with the filename ! (That means use "`/o `" and not "`/o`" except you have to call the compiler the way "`F95.EXE /oname`" instead of "`F95.EXE /o name`")
- `LIBNAME` - This parameter tells the linker the name of the output file for a library. It is used in the same way as the parameter `EXENAME` and contains the same string. But perhaps you have to change it if you want to use another linker for library files than for executable files.
- `OBJEXTENSION` - This environment variable contains the extension of the object files. Normally this is `.OBJ` but in some cases you have to change it to the right string. I.e. if you want to use the CYGWIN compiler without the BASH you have to change this to "`.o`" (see section 5.4).
- `LIBEXTENSION` - This environment variable contains the extension of the library files. Normally this is `.OBJ` but in some cases you have to change it to the right string. I.e. if you want to use the CYGWIN compiler without the BASH you have to change this to "`.a`" (see section 5.4).

There are also some special options that only have to be changed in very special environments of compiler and linker:

- `LIBBLAS` - This environment variable points to the BLAS library. A standard BLAS library is shipped with the FEATFLOW package and placed in the standard library path (which you can find in the "`LIBDIR`" variable). Some fortran compilers provide a special BLAS library that is optimized for a special machine architecture. If you have such a special (precompiled) BLAS library you can change this variable to point to that library.
- `MAKEDIR` - This environment variable describes the directory path where the makefiles for all the processor architectures are placed in.
- `MAKENAME` - This environment variable is the name of the subdirectory in the `MAKEFILES` directory that holds the makefiles for the INTEL architecture.
- `COMPLIBEXT` - This setting should only be changed when you have a linker for libraries that doesn't allow an extension for the target file when being called. Normally this needn't to be changed. But if you have a linker that is called the way "`LINK.EXE abc *.obj`" to produce the file "`abc.lib`" you have to delete the "`.LIB`" string here.

- `USELIBPARAM` - This has to be used if you must tell your linker which files are libraries and which files are OBJECT files. Normally this string hasn't to be changed because the linker can identify the type of the file on its extension. But if you have to call your linker the way

  "`LINK.EXE abc.exe *.obj /lib:*.lib`"

  to tell it which files are OBJECT files and which are libraries, you can change this here (i.e. to "`/lib:`" in this example).

5.3 The file `FEAT_GLOBALS.BAT` for the INTEL FORTRAN compiler

As a first example how to use the file "`FEAT_GLOBATS.BAT`" we first show the options that can be used when you are using the INTEL FORTRAN compiler. Take a look at the content of this file[1]:

```
set F95=ifl.exe /4Yportlib /nologo

set F95OBJONLY=/c

set LINKEXE=link.exe
set LINKLIB=lib.exe

set EXENAME=/out:
set LIBNAME=/out:

set LIBDIR=%FEATFLOW%\object\libraries
set LIBTITLE=INTELWIN
set FLIBRARIES=%LIBDIR%\%LIBTITLE%

set MAKEDIR=%FEATFLOW%\object\makefiles
set MAKENAME=INTELWIN

set OBJEXTENSION=.obj
set LIBEXTENSION=.lib

set COMPLIBEXT=.lib

set USELIBPARAM=

set LIBBLAS="%FLIBRARIES%\libblas%LIBEXTENSION%"

set FINCLUDE=%FEATFLOW%\application\workspace
```

The first line defines the call to the compiler together with two standard options. The INTEL compiler has the name "`IFL.EXE`" and is installed in a subdirectory of the folder `C:\Programs` on your hard disk.

The second line defines the option that must be used for the compiler to generate object files instead of executable binaries. The third and the fourth line define the name of the two

[1] The file "`FEAT_GLOBALS_INTEL.BAT`" contains a backup of this configuration.

linker applications - one for linking executable files and one for linking executable binaries. These two files are contained in the Visual C++ package of Microsoft and can be found in a subdirectory of the folder `C:\Programs`, too.

The environment strings `EXENAME` and `LIBNAME` define the strings that must me used with the linker to define the name of the output file that should be generated. Note that there is no space character at the end of these lines because the parameter uses a ":" to seperate the parameter from the filename !

The next lines define some directories where the generated libraries are placed and where all the scripts can be found.

The three environment variables `OBJEXTENSION`, `LIBEXTENSION` and `COMPLIBEXTENSION` now define the extensions that are used by the compiler/linker for generated files - in this case object files have the extension "`.OBJ`" and libraries have the extension "`.LIB`".

The `USELIBPARAM` is undefined in the case of the INTEL compiler. After all the environment variable `LIBBLAS` is set to the path containing the BLAS library (usually the same directory where the other libraries are, too), and the `FINCLUDE` environment variable is set to the folder containing the include files that are used for the different configurations of FEATFLOW for the example applications and the *computation suite*.

Remark: If you want to use the INTEL FORTRAN compiler to compile the FEATFLOW source code you should make sure that both the compiler and the Visual C++ package of Microsoft are installed properly and working from the command line. We'll not give a description here how to install these applications since this would be much too complex. But we'll give some hints you should pay attention to before compiling FEATFLOW:

- You should make sure that the directories containing the compiler "`IFL.EXE`" and the two linker applications "`LINK.EXE`" and "`LIB.EXE`" are contained in your `PATH` environment variable so that these applications can be found during the compiling process.
- You should make sure that the `PATH` variable contains the directories where the dynamic link libraries of Visual C++ and those of the INTEL compiler resist in. For example the "`LINK.EXE`" and "`LIB.EXE`" applications search for the file "`MSPDB60.DLL`" which resists in the `Common\MsDev98\Bin` subdirectory of your Visual C++ directory so you have to add this directory to the `PATH` variable, too.
- You should make sure that the environment variable `LIB` contains the directories where the FORTRAN libraries of the INTEL compiler and the Visual C++ libraries of the Visual Studio resist in. (This variable can be changed at the same point as the `PATH` variable, only the name is different.) Otherwise the linker would give you an error message that a library file cannot be found !

5.4 An example of a modified `FEAT_GLOBALS.BAT`

Here is an example of modifying the file "`FEAT_GLOBALS.BAT`" for another compiler than the one of INTEL:

If you have installed the CYGWIN environment you can install FEATFLOW within the BASH shell as described in chapter 3. But it is also possible to use the G77 without the BASH - you can install it using the installation technique that is described here. All you have to do is to modify the following lines of the file "`FEAT_GLOBALS.BAT`":

```
set F95=g77

set F95OBJONLY=-c

set LINKEXE=g77
set LINKLIB=ar rv

set EXENAME=-o
rem           ^- There is a space character here !
set LIBNAME=

set OBJEXTENSION=.o
set LIBEXTENSION=.a

set COMPLIBEXT=.a
```

(of course you don't need to enter the line with the `REM` command in the front - but don't forget to enter the space character at the end of the line "`set EXENAME=-o ` " !)

Alternatively you can rename the file "`FEAT_GLOBALS_CYGWIN.BAT`" to "`FEAT_GLOBALS.BAT`" - this file contains the above changes.

5.5 Start of the installation

Now we can start the installation script. Change to the directory

`featflow1.2/`

and call the file

"`INSTALL_FEATFLOW.BAT`"

to start the installation process. This batch script will produce all the libraries and executables of FEATFLOW. After it has finished look into the following directories to find the compiled files:

- `/featflow1.2/featflow/object/libraries/INTELWIN`
  This directory contains all the libraries of FEATFLOW (`.LIB`-files). You can add this path to your "`INCLUDE`" environment variable of your linker if you want to use the libraries frequently.

- `/featflow1.2/featflow/application/comp`
  This directory should contain the executable files for the *computation suite*:

  "`CC2D.EXE`", "`PP2D.EXE`", "`CC3D.EXE`", "`PP3D.EXE`".

  You can start the computation suite by launching the "`COMP_START.BAT`" batch file (see section 5.6) or by calling these files directly.

- `/featflow1.2/featflow/application/example`
  This should contain the executables for the FEATFLOW example application:

  "`CC2D.EXE`", "`PP2D.EXE`", "`CC3D.EXE`", "`PP3D.EXE`".

In contrast to the computation suite these files are compiled with a slightly different sourcecode. If you want to know the exact differences, look at the directory

`/featflow1.2/featflow/application/workspace`

which holds different include files. You can figure out which include files are used in which configuration if you look into the "`INC_xxx.BAT`" files in this directory. Take a look into section 6.6 about the background of the include files.

- `/featflow1.2/featflow/application/user_start/input_files`
  Contains the above 4 files, too, but this time compiled with different code. Look at the "`.F`" files in this directory to figure out the changes and modify them if you want.

Remark: Don't wonder about some warning messages or messages like *"No files found"* within the installation process - it's normal. Also take a look to section 6.8 about system dependent restrictions and effects.

5.6 Compiling and executing the computation suite and example programs

The computation suite and the example programs are build in the installation step but not executed. This is because they will take a very long time to execute - almost a couple of hours.

If you want to execute the *computation suite* or the example application you should use the batch file that is stored in the directory of the application - that is the file "`COMP_START.BAT`" in the directory of the *computation suite* and the file "`EXAMPLE.BAT`" in the directory of the example application.

You should start these files in this way:

"`COMP_START.BAT > output.txt`"

or

"`EXAMPLE.BAT > output.txt`"

so that the output is redirected into a text file. Afterwards you can analyze the output of the application in this file.

Chapter 6

A more detailed description about compilation and other problems

This chapter gives a more detailed view of compiling the FEATFLOW source code and the example applications with a user specified compiler without CYGWIN. Additionally there will be described some parameters for the G77 to improve the performance in the CYGWIN environment and some hints for compiling your own code.

6.1 Steps in the installation procedure

If you start the installation batch file "`INSTALL_FEATFLOW.BAT`" the following things will be done:

1. An environment variable `FEATFLOW` is created which points to the subdirectory

   `./featflow`

   in the current directory. This is then the main installation path of FEATFLOW. The content of this variable is derived from the current directory. To get the current directory the install script uses the help application "`GETCWD.EXE`", which you can find in the directory `featflow/HelpApps` together with its sourcecode - it can be compiled with the "`g++`" of CYGWIN if the MINGW32 package of CYGWIN is installed.

2. The installation path is written to the file "`./featflow/FEAT_GLOBALS.BAT`" together with the information in the template file

   "`./featflow/object/makefiles/INTELWIN/FEAT_GLOBALS.BAT`"

   which was modified in 5.2. The resulting file is then frequently used in every installation step because it holds all the information about the installation path and the compiler options.

3. The file "`./featflow/object/makefiles/INTELWIN/MAKE_COPY.BAT`" is called. This batch file copies the standard compilation batch files from the directory

   `/object/makefiles/INTELWIN`

into all the subdirectories of the `/source` directory.

4. The file "`./featflow/object/makefiles/INTELWIN/MAKE_LIB.BAT`" is called. This file calls all the batch files in the subdirectories of the `/source` directory which are responsible for building the libraries.

5. The file "`./featflow/object/makefiles/INTELWIN/MAKE_STD_APPS.BAT`" is called. This file calls all the batch files in the subdirectories of the `/source` directory which are responsible for building the standard applications.

6. The file "`./featflow/application/comp/COMP_START.BAT`" is called. This file is responsible for building the code for the *computation suite*. The code is not executed !

7. The file "`./featflow/application/example/EXAMPLE.BAT`" is called. This file is responsible for building the code for the examples. The code is not executed !

8. The file "`./featflow/application/user_start/USER.BAT`" is called. This file is responsible for building the standard code for the user applications. You can use this file, too, if you took modifications to the files in this directory and you want to recompile them. The code is not executed !

As you can see all the files that are responsible for compilation are put together in the directory

`./featflow/object/makefiles/INTELWIN/`

Even the files "`COMP_START.BAT`", "`EXAMPLE.BAT`" and "`USER.BAT`" can be found in this path - they are copied to their "home" directories before they are called. If you want to modify the way the code is compiled look into this directory to find the desired file.

Additionally this directory contains the file "`CREATE_STD_DIRS.BAT`". This file is called by all those batch files which compile an application and not a library. It is responsible for creating some subdirectories which are used by FEATFLOW. This is necessary because a directory is not automatically created when an application wants to use it. The following list gives an overview about the subdirectories that must exist in a directory before FEATFLOW is executed there:

- `./#adc`
- `./#avs`
- `./#data`
- `./#film`
- `./#gmv`
- `./#ns`
- `./#points`
- `./#pre`
- `./#tria`
- `./#tries`

6.2 Structure of the batch files for generating all the files

Now we take a look at the those batch files that are responsible for compiling and linking the code. All these files are copied from the `makefiles/INTELWIN/` directory to their *home* directory, which is contained in the directory `/SOURCE` of the FEATFLOW package, before they are executed in the installation process. There are two types of such batch files: Those that are responsible for building libraries and those that are responsible for building executable files. We demonstrate these two types on two examples[1]:

First we take a look on: "`feat2d.bat`"

```
@echo off
echo -----------------------------------
echo FEAT2D.BAT running...
echo -----------------------------------
call ..\..\..\feat_globals.bat

echo Deleting rubbish:
del *.o

echo Compiling:
for %%9 in (*.f) do g77 -c %%9

echo Linking:
ar rv -o libfeat2d.a *.o
move /Y *.a "%FLIBRARIES%"

echo Deleting rubbish:
del *.o
echo Ok...
```

This file is responsible for building the basic library of the 2D routines of FEATFLOW. As you can see the first thing this script does is calling the file "`FEAT_GLOBALS.BAT`". Afterwards it's sure that the necessary environment variables like the FEATFLOW home directory and others are set.

In the second step the batch file deletes old object files to make sure that there is no *rubbish* from an older installation. Afterwards the compiler and the linker are called for all "`.F`" files in the directory, and the built library is copied to the directory of the libraries. At a last step the temporary object files that are generated during compilation are deleted.

Now we take a look at the file: "`CC2D.BAT`"
This file is responsible for building an executable application:

```
@echo off
echo -----------------------------------
echo CC2D.BAT running...
echo -----------------------------------
call ..\..\..\feat_globals.bat
```

[1]For readability all uses of environment variables in these files are replaced by the strings that are defined in the file "`FEAT_GLOBALS.BAT`" when using the compiler options for CYGWIN

```
set SRCCC2D=%FEATFLOW%\source\cc2d\src

call "%MAKEDIR%\%MAKENAME%\create_std_dirs.bat"

echo Deleting rubbish:
del *.o

echo Compiling:
g77 -c "indat2d.f"
g77 -c "parq2d.f"
g77 -c "%SRCCC2D%\cc2d.f"

echo Linking:
g77 -o cc2d.exe indat2d.o parq2d.o cc2d.o "%FLIBRARIES%\libcc2d.a"
       "%FLIBRARIES%\libcc2dmg.a" "%FLIBRARIES%\libfeat2d.a"
       "%FLIBRARIES%\libblas.a"

echo Deleting rubbish:
del *.o
echo Ok...
```

The script behaves in a similar way like above: At first the file "`FEAT_GLOBALS.BAT`" is called to get the installation parameters. But before old temporary files are deleted the file "`create_std_dirs.bat`" is called. This script makes sure that all the subdirectories are created that are necessary for executing the application.

Afterwards the compiler and the linker[2] are called for all "`.F`" files in the directory. The result is an executable file. Again the last step is to clean temporary files.

6.3 Internals about the batch files for building the examples

If you take a look in the batch files which are responsible for executing the *computation suite* and the example applications ("`COMP_START.BAT`" and "`EXAMPLE.BAT`") you could wonder about two parameters the files can be called with.

If these files are called without parameters they search for the files "`CC2D.EXE`", "`PP2D.EXE`", "`CC3D.EXE`" and "`PP3D.EXE`". If one file cannot be found the compilation process is started to build these files. Afterwards the files are executed - the output will be displayed on your screen.

The batch files can be called with following arguments:

- "`xxx.BAT BUILDONLY`"
  The applications are build whether they exist or not. They are not executed.
- "`xxx.BAT ONLYEXECUTE`"
  The applications are executed directly without building whether they exist or not.
- "`xxx.BAT NOTIME`"
  The applications are executed without stopping the time they need for execution. For

[2] In this case the linker is the "`G77`", too !

stopping the time the help application "`UTIME.EXE`" is needed which resists in the directory

`featflow/HelpApps`

together with its sourcecode - it can be compiled with the "`g++`" of CYGWIN if the MINGW32 package of CYGWIN is installed.

The first two parameters are only used within the installation routine to build the executables without executing them. The last parameter is used nowhere but you can use it optionally if you want.

If you have modified the code of the example applications you should simply delete the "`.EXE`"-files and call the batch file without any parameter. This will build the executable and execute it.

6.4 Additional parameters for the G77 in CYGWIN

If you use the CYGWIN environment and you have a Pentium III or a better processor you can try some parameters for the G77 to improve the performance. In our tests the optimized G77 code is better than the standard code of the original Intel FORTRAN compiler but not as good as the optimized code of the Intel compiler. In our opinion the best settings for the G77 are as follows:

`-O6 -march=pentiumpro`

There are two possibilities to enter these options:

On the one hand you can start the installation of FEATFLOW and choose the installation option "h":

```
FEAT/FEATFLOW Release 1.2 installation program    V1.2
-------------------------------------------------------
Configuration:
-0- general platform
-1- SUN Sparcstation
...
-g- Intel/x86 Windows (CYGWIN)
-h- Intel/x86 Windows (CYGWIN) max. optimization
-i- User defined
---------------------------------------
Selection:h
...
```

All applications will be rebuild and the optimized libraries will be put into the subdirectory

`/featflow1.2/featflow/object/makefiles/libwinmx`

On the other hand (i.e. if you want to play with the compiler options) you can modify the installation script. Open the file "`INSTALL_FEATFLOW`" in a text editor and find the following lines:

```
case g:
  echo "Installation for Windows platforms with CYGWIN:"

  set LIBDIR = $FEATFLOW"/object/libraries/libwin"
  set LIBTITLE = "win"
  set COMPOP = ""
  set ARFLAGS = "rv"
  echo "Standard libs are being used."
  set F77LIB = '$(FEATFLOWLIB)/libblas.a'
...
```

Change the environment variable `COMPOP` ("COMPiler OPtions") this way:

```
  set COMPOP = "-O6 -march=pentiumpro"
```

Afterwards start the installation script as described in section 3.2.

If you want to compile the FEATFLOW package with the Windows batch files as described in section 5.4 you have to add these parameters to the file "`FEAT_GLOBALS.BAT`". Simply open this file in a text editor and change the `F95` environment variable to the following:

```
set F95=g77 -O6 -march=pentiumpro
```

6.5 Distributing applications built with CYGWIN

If you have compiled FEATFLOW and you want to execute it on another computer that does not have CYGWIN installed you can still do that. But it's not enough to copy only the "`.EXE`" files to the other computer. If you want to execute them you also need the file "`CYGWIN1.DLL`" of CYGWIN - without this file the executables would not work !

You can find that file in the directory

`/cygwin/bin`

Simply copy this file together with the executables of FEATFLOW into the same directory on the target computer. Don't forget to build the directory structure of FEATFLOW (see section 6.1 for the subdirectories that are needed or use the batch file "`CREATE_STD_DIRS.BAT`" to build it automatically). Afterwards you can execute FEATFLOW.

6.6 Special code variants for different applications - the include files

If you take a look into the batch files which are responsible to build the *computation suite* or the example applications you can find the call of different "`INC_xxx.BAT`". After installation you can find these files in the directory `./featflow/application/workspace`.

These files are responsible for "using the right include files at the right time". FEATFLOW needs different amounts of memory when running. Because this memory can't be dynamically allocated an estimate of the memory needed has to be set before compilation - and this number resists in special include files belonging to the application that should be compiled.

The "`INC_xxx.BAT`" files now copy the needed include files into the directory

`./featflow/application/workspace`

Later the compiler will find them there. In this way this directory is a central point for the whole compiling process of the examples.

For example the file "`INC_COMP.BAT`" copies the files "`*.INC`" from the directory

`/comp/input_files`

(the directory of the *computation suite*) to this directory so that the compiler will use these files later when compiling the files for the *computation suite*.

6.7 Hints about linking your code

There's one point in compiling and linking you should be aware of: the order of the libraries in the linking process. All the libraries have a special importance and they must be given in the right order to the linker. If not this could lead to unpredictable results, depending on your linker.

The following table shows the importance of the different libraries. If you want to use the libraries enter the libraries in your make file in this order - from top to bottom ! The files at the top of the list are the most specialized and should be entered at the beginning of the command line for the linker (after your object files). The files at end of the list are the most general and should be entered at the end of the command line for the linker. Files in the same line have the same importance because they don't build upon each other - commonly they are not included within the same application because of the job they have to do.

- `LIBTR2TO3, LIBTRIGEN2D, LIBTRIGEN3D`
- `LIBCC3D, LIBPP3D`
- `LIBCC2D, LIBPP2D`
- `LIBCC3DMG, LIBPP3DMG`
- `LIBCC2DMG, LIBPP2DMG`
- `LIBFEAT3D`
- `LIBFEAT2D`
- `LIBBLAS`

For example the application "`PP3D.EXE`" is linked with the following command (compare "`PP3D.BAT`"):

```
link ...  libpp3d.lib libpp3dmg.lib libfeat3d.lib libfeat2d.lib libblas.lib
```

6.8 System dependent restrictions

If you compiled the *computation suite* or the example applications and you execute them within Windows 95/98 you may be surprised by the the following effect: The timing routines don't seem to work. The FEATFLOW applications always display "0" instead of the time they need for doing computations (compare picture 6.1).

Figure 6.1: Wrong timing in Windows 95/98

This behavior is not a problem of FEATFLOW - it's a problem of Windows. When you run the applications in Windows 2000 everything runs fine. In Windows 95/98 it doesn't work. FEATFLOW uses the FORTRAN command `ETIME` to measure the time the processor needs for execution of a specified process. Unfortunately neighter CYGWIN nor the INTEL FORTRAN compiler implement this command correctly in Windows 95/98.

If you want to modify the source code to use a system dependent timing routine find the file

"`featflow/source/feat2d/src/ztime.f`"

and change it as you like. In this file the timing routine is encapsuled.

Another remark: There are incompatibilities between Windows 98 and Windows 2000 in redirecting the standard output of applications into files. Because of this the installation log file "`featinstall.log`" will be crippled after installing when FEATFLOW with the installation batch files of chapter 5 in a Windows 98 environment: the output of the installation scripts cannot be redirected into the log file in the same way as in Windows 2000.

As for a similar reason you will often read messages like *"No files found"* on the screen when installing FEATFLOW with the installation batch files of chapter 5 in a Windows 2000 environment. This is no reason for panic: The batch files sometimes try to delete some temporary files which do not exist and so there's a message that these files cannot be found. Windows 2000 is not able to redirect this warning message to the installation log file.

All these effects do not influence the functionality of the software in any way !

Chapter 7

DeViSoR grid generator

The *DeViSoR* software module is a pure JAVA application. This guarantees its functionality on most platforms. But this makes it necessary that SUN JAVA is installed to the computer before the application can be used. The first section will give a description on where to download and how to install the SUN JAVA development package.

The DeViSoR software manual already gives a description on how to install the software package. But because this is originally written for UNIX systems this chapter takes a deeper look into the installation within the Windows world.

There are at least two versions of the DeViSoR software module available on the FEATFLOW homepage, which differ slightly in the way how to install. Section 7.2 will give a instruction how to install the original version 1.0 of DeViSoR and section 7.3 will give a description how to install version 2.X and above.

At last section 7.5 will give a more general introduction how to set up SUN JAVA within Windows. This is not directly necessary for DeViSoR but it will show some aspects that are common to all JAVA applications.

7.1 SUN JAVA development engine

The SUN JAVA development engine is available on the SUN homepage:

`http://www.sun.com`

You need at least the following release:

JAVA 2 SDK 1.4 (or above)

The following example shows how to install JAVA2 SDK 1.4. If you have a newer version of JAVA the installation procedure should be equivalent.

From the SUN homepage you should get a file like

"`j2sdk-1_4_0-win.exe`"

The installation consists of two steps:

The first step: Execute the above file and follow the instructions on the screen. The only point you should change in the default values of the setup routine is the installation path. You should change the destination folder for the installation to the following (compare picture 7.2):

Figure 7.1: Installation of SUN JAVA

Figure 7.2: Change the destination directory !

`C:\sunjava`

All the following descriptions in this manual refer to this directory. If you don't change the destination folder or if you use another folder you'll have to work with that instead of `C:\sunjava`.

The second step: You should add the path to the file

"`JAVA.EXE`"

to the environment variable `PATH` of Windows so you can launch JAVA from every directory in your system. The JAVA executable resists always in the `BIN\` subdirectory of the directory you installed JAVA to. If you followed the instructions above the path you have to add is:

`C:\sunjava\bin`

Look at section 1.2 how to add a directory to the `PATH` variable of Windows.

Remark: Some internet browsers like Netscape 6.x ship the SUN JAVA package together with the browser. You can use this package, too, but you should be aware that it has two disadvantages:

1. It's only the runtime version of JAVA - this means it doesn't contain a compiler ! For that reason you are not able to *compile* `.java` files. In the case of the DeViSoR this does not matter because because the DeViSoR is already compiled.

2. You can't specify the path JAVA is installed to you, and so you have to search for the right path on yourself. Common default paths of such installations are:

   - `C:\Programs\JavaSoft\JRE\[version]`
   - `C:\Programs\Java\JRExxxx`
   - `C:\Programs\JDKxxxx`

   You have found the right folder if there is a `bin/` subdirectory in it, which holds the file "`JAVA.EXE`".

7.2 Installation procedure of DeViSoR, Version 1.X

First download the installation package from the FEATFLOW homepage:

`http://www.featflow.de`

There should be two files:

- "`devisor1.XX.tar.gz`" - The DeViSoR grid software module itself, and
- "`devisorwin1.XX.zip`" - The Windows installation patch.

Save the files in a folder of your choice, for example to the folder:

`C:\CYGWIN\USR`

Then extract them into the same directory. For example you can use CYGWIN for that task:

```
Administrator@WINDOWSHOST ~
$ cd /usr

Administrator@WINDOWSHOST /usr
$ gunzip devisor1.0.tar.gz

Administrator@WINDOWSHOST /usr
$ tar -xvf devisor1.0.tar
...

Administrator@WINDOWSHOST /usr
$ unzip -o devisorwin1.0.zip
...

Administrator@WINDOWSHOST /usr
$
```

A subdirectory `Devisor/` will be created where the DeViSoR software module resists in. It contains some more subdirectories with the sourcecode (compare picture 7.3).

Figure 7.3: After extracting DeViSoR - launch the software with the file `devisorgrid.bat`

Now you can launch the DeViSoR grid generator by double clicking on the file:

"`devisorgrid.bat`"

7.3 Installation procedure of DeViSoR, Version 2.X

The installation procedure of DeViSoR 2.X is similar to DeViSoR 1.X. Download the installation package from the FEATFLOW homepage; in the difference to the version 1.0 this should be only one file:

"`devisor2.tar.gz`"

Save the files in a folder of your choice, for example to the folder:

`C:\CYGWIN\USR`

Extract them into the same directory - i.e. using CYGWIN:

```
Administrator@WINDOWSHOST ~
$ cd /usr

Administrator@WINDOWSHOST /usr
$ gunzip devisor2.0.tar.gz

Administrator@WINDOWSHOST /usr
$ tar -xvf devisor2.0.tar
...

Administrator@WINDOWSHOST /usr
$
```

A subdirectory `Devisor2/` will be created where the DeViSoR 2.X software module resists in. It contains some more subdirectories with the compiled sourcecode (compare picture 7.4).

Figure 7.4: After extracting DeViSoR 2 - launch the software with the file `devisorgrid.bat`

Now you can launch the DeViSoR grid generator by double clicking on the file:

"`devisorgrid.bat`"

7.4 Common installation mistakes

Although the installation is very easy there can happen some typical mistakes which prevent the application from starting. These mistakes will be described here.

1st mistake: Not enough environment space.
You should be aware that there is enough environment space available. The batch file `devisorgrid.bat` uses some environment variables and if there's not enough space it can happen that JAVA or the DeViSoR can not be found. This is more a problem of Windows 95/98 than Windows 2000; look at section 1.1 how to make enough environment space available !

2nd mistake: Calling the batch file from the wrong directory
If there is an error message: *"! ERROR ! Can't find DeViSoR source code !"* you called the

batch file `devisorgrid.bat` from the wrong directory. The batch file is assumed to be called from the directory DeViSoR is installed to. Change into this directory before calling the batch file !

3rd mistake: JAVA cannot be found
You didn't add the right path to the JAVA executables to the environment variable `PATH` of Windows. Enter the correct path and it should work ! The path must contain two files: "`JAVA.EXE`" (the runtime machine) and "`JAVAC.EXE`" (the JAVA compiler).

For the 2nd and 3rd mistake there is still another solution possible. Open the batch file in a text editor of your choice and take a look at the first lines. There are two environment variables set: The `JAVAHOME` variable points to the directory with the executable files of JAVA (that's the `bin-` subdirectory of the directory you installed JAVA to) and the `DEVISORHOME` directory points to the directory of the DeViSoR.

Figure 7.5: Manual configuration in the startup script of DeViSoR

If you look at the file you will recognize that these lines are empty. This state is usable if the executable of JAVA can be found in one of the directories that are listed in the `PATH` environment variable and if the batch file is called from the directory the DeViSoR is installed to (this happens if you double click on the file in the Windows Explorer). In any other case you can enter the absolute path to the JAVA executables and the directory of DeViSoR into these lines (compare picture 7.5).

7.5 General configuration of SUN JAVA

This section gives a more generalized introduction how to configure SUN JAVA to execute a JAVA application, for example the DeViSoR grid module. Everything that is described here is not really necessary for executing the DeViSoR grid, but the DeViSoR grid module is a good example to show how JAVA has to be configured for another application.

SUN JAVA only needs one parameter to be set up before an application can be launched - this is the `CLASSPATH` environment variable, which is a collection of all directories where JAVA is searching for the subpackages of an application. The variable can be set up globally (i.E. in the "`AUTOEXEC.BAT`" or in the environment tab of Windows 2000) or locally to a command line.

First we will show how to change this in your system configuration: If you use Windows 2000 you have to enter this in the environment tab (the point where the `PATH` environment variable resists in, too; compare picture 7.6).

If you use Windows 95/98 you have to add the corresponding `SET` command to your startup file "`AUTOEXEC.BAT`" (compare picture 7.7) and restart your system.

Figure 7.6: Add the `CLASSPATH` to the environment of Windows 2000

For example if you want to use the DeViSoR module the variable has to point to the DeViSoR directory. If you followed the instructions in section 7.2 this is:

```
C:\cygwin\usr\Devisor
```

Afterwards you can launch your application. As an example the following line shows the command you can enter to launch the DeViSoR grid module:

```
java devisor.grid.base.Grid
```

Alternatively if you don't want to change your system configuration you can of course set up the `CLASSPATH` environment variable locally to a command line. To do this open a DOS box and enter the following command:

```
SET CLASSPATH=.;[path to 1st application];[path to 2nd application];...
```

(The "." is not really necessary but makes it possible to execute a JAVA application in the current directory.) Afterwards you can run your application with the corresponding call to JAVA. In the example of DeViSoR enter

```
SET CLASSPATH=.;C:\cygwin\usr\Devisor
```

and run the DeViSoR with the command above (compare picture 7.8).

Figure 7.7: Add the directory of JAVA and the `CLASSPATH` to your `AUTOEXEC.BAT` of Windows 95/98

Figure 7.8: How to start the DeViSoR module manually - first set up the `CLASSPATH`, then launch the application