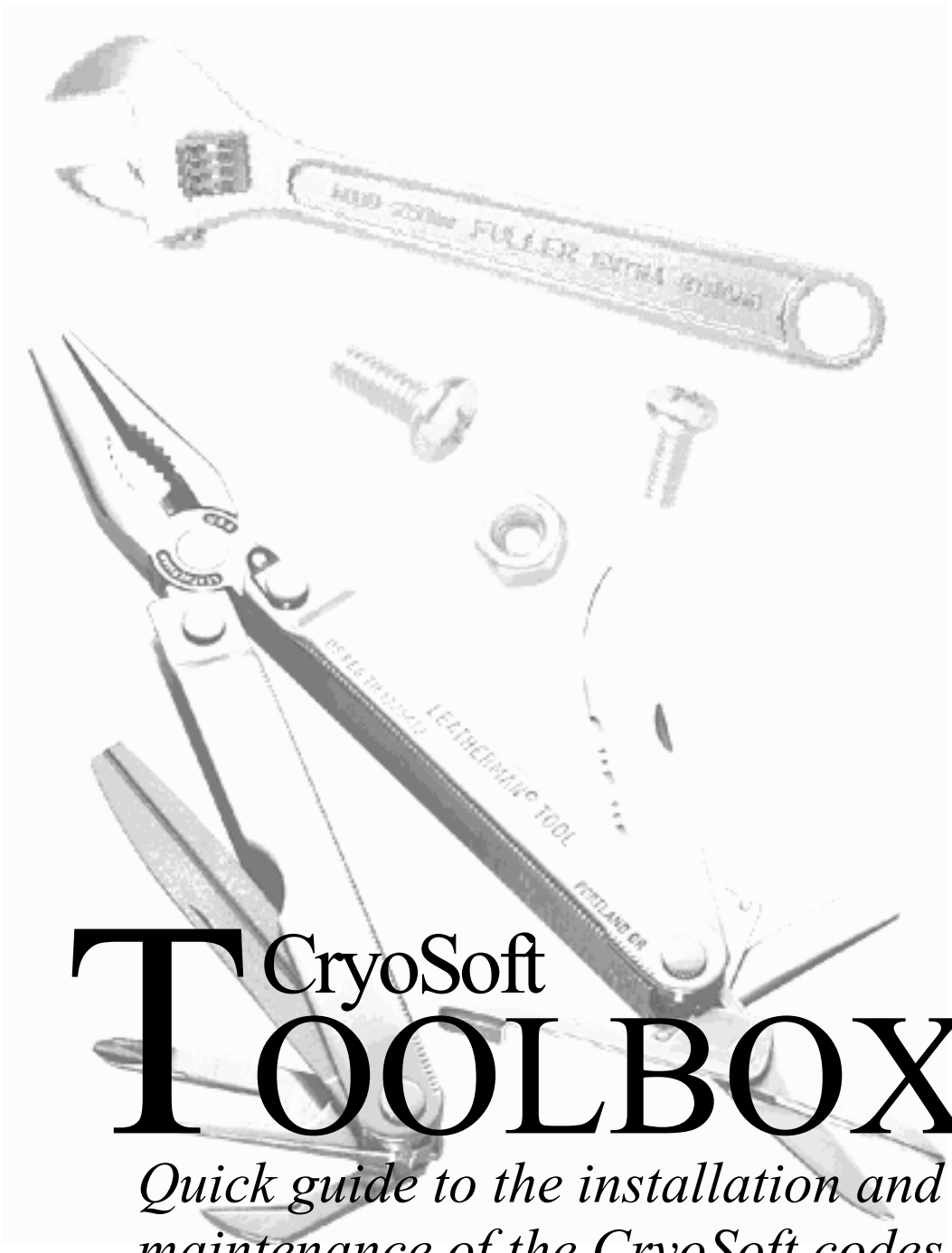
Installation Guide

Version 8.2
January 2021



CryoSoft
TOOLBOX

Quick guide to the installation and maintenance of the CryoSoft codes

Overview

CryoSoft has produced an integrated suite of codes for the analysis of electromagnetics, cryogenics and thermo-hydraulics of superconducting magnets. The system consists of a collection of applications and libraries organized in different layers. The user has in general access to the top-most layer only, that consists of the stand-alone applications for the solution of specific problems related to the design and analysis of superconducting magnet systems, and the associated post-processors for the generation of plots and reports.

This manual provides information on:

- the hardware and software requirements for the installation of the CryoSoft package of codes and for the use of the applications;
- how to install the applications and the libraries using the script provided with the package;
- details on the organization, location and functions of the applications and the libraries;
- maintenance and customization of the applications, and in particular how to link user's defined external routines.

Details on each element of the suite can be found in the corresponding user's manuals. consult our on-line library at:

`http://cryosoftsupport.wix.com/cryosoft`

We assume in this guide that you have familiarity with the operating system and the compilers of your machine. For more clarifications and in case of any problems please consult your system manager or contact us at at:

`cryosoft.support@gmail.com`

System requirements

Platforms and operating systems At the time of writing this guide, the native platform for the development and test of the CryoSoft codes is:

- Macintosh running MacOS-X (10.8.5 and higher) under XQuartz,(2.7.8) gcc (5.1) with gfortran.

At different time of the development and production, the code system has been installed and tested on the following platforms:

- Mac-OS X (10.2 and higher) operating system;
- GNU/Linux operating system (most distributions).
- INTEL PC's running RedHat Linux OS;
- IBM-RISC workstations running the AIX-V4 operating system and later;
- SUN-SPARC workstation running the Solaris OS operating system;
- DEC-ALPHA workstation running the OSF-1 operating system;
- HP workstations running HP-UX OS;
- Windows-2000 and Windows-XP operating system, with an installed CYGWIN environment[1] (the reference version tested is CYGWIN 1.5.24-2).

The majority of the codes are written in the generally accepted FORTRAN-77 dialect, while graphics and a limited set of system routines are written in C. For this reason we require that a FORTRAN and a C compiler are properly installed, and available in the executable PATH.

In case several compilers are installed, precedence for FORTRAN is given to gfortran[2], followed by g77[3] and f77[4]. In the case of C compilers, precedence is given to gcc[5]. vs. cc[6]. The f77 and cc compilers are usually the native compilers in a specific architecture. The gfortran, g77 and gcc compilers are freely-distributed GNU[7] compilers.

[1] CYGWIN is a Windows environment that provides all core functionality of Linux, as well as a port of the GNU compilers gcc and g77, and the X11 graphic API. Details on CYGWIN can be found at `http://www.cygwin.com`. See Appendix II for a short guide to the installation of CYGWIN.
[2] Gfortran is the FORTRAN wrapper of the GNU gcc C compiler, compliant with the '77 rules.
[3] g77 is the GNU version of the FORTRAN compiler, compliant with the '77 rules.
[4] f77 is the standard UNIX FORTRAN compiler, compliant with the '77 rules.
[5] gcc is the GNU version of the C compiler.
[6] cc is the standard UNIX C compiler, compliant with the ANSI rules.
[7] GNU is the project of an operating system upward compatible with UNIX, mainly supported by the Free Software Foundation. GNU provides, for free download, all UNIX tools and main applications such as compilers, editors, graphic libraries. Details on the GNU project can be found at `http://www.gnu.org`.

A standard installation is performed using a shell script delivered with the software, `CSmake`, that uses the bash shell[8]. We hence require that bash, or a suitable dialect, is available (or linked) as `/bin/bash`.

The code compilation is performed using makefiles, which require that make[9] is installed.

The interactive graphics for all installations relies on the presence of the X Window System[10] (X11.R5 and later). X11 is available in all UNIX and Linux flavors with standard installation. It can be installed on all other systems listed above.

The following table gives the overview of the requirements on compilers and libraries installed:

	Required software
Configuration and installation script (CSmake)	bin/bash
Compilation	make
Compilers	gfortran, g77 or f77 gcc or cc
Graphics API	X11.R5 or later

As a final remark, for the installation you will require a fully functional UNIX or Linux shell, with the standard set of commands to copy, move, archive, remove files and directories. Refer to your system guide or system manager for this matter.

Disk space and memory The disk space required for the complete installation is modest, approximately 30 MB including executable and object files. The space required for execution depends on the size and number of cases run, and is typically of the order of 10 to 100 MB.

The memory needed for running the codes depends also on the size of the problem you are solving. Minimum of 128 MB RAM available may be adequate for simple analyses and small-size problems. Memory intense runs of large models (e.g. with `thea`) can require as much as 0.5 GB of RAM. System simulations involving the concurrent processes in a multi-core machine (e.g. under `supermagnet`) require several GB of memory for efficient usage of CPU.

[8] Bash is the acronym for the Bourne Again Shell, the command language interpreter that appears in the GNU operating system. Bash is an sh-compatible shell that incorporates useful features from the Korn shell (ksh) and C shell (csh). It is intended to conform to the IEEE POSIX P1003.2/ISO 9945.2 Shell and Tools standard. Details on bash can be found at `http://www.gnu.org/software/bash/`.

[9] Make is a tool which controls the generation of executables and other non-source files of a program from the program's source files. Details on the GNU version of make can be found at `http://www.gnu.org/software/make/`.

[10] The X Window System (or X, or X11) is a network transparent window system which runs on a wide range of computing and graphics machines. Details on X11, and on the freely redistributable open-source implementation of X11, can be found at `http://www.xfree86.org/`.

Installation

The following sections report the step-by-step procedure to be followed to install the package if you have purchased the codes in source form, in this case follow steps 1 through 7. If you have purchased the codes in binary, pre-compiled form, it may be advisable to perform a remote installation for which we would request you to provide a temporary remote login access to your machine (`ssh`). The result after installation is the same for both options, apart for the `src/` directory that is empty in the case that you have purchased pre-compiled codes.

Unpacking the codes

For a source-code installation you have received the CD with the code sources, the compilation script and examples in tar format (UNIX and Linux) or as a WinZip archive. Alternatively the codes may have been transferred by remote copy (`scp`) onto your home directory in a `tar` archive. For installation you should:

Step 1. create a directory that will contain all codes and will eventually become the root directory for the installation. We will refer to this directory as `CryoSoft/` in the following, although the name of this directory can be different. Copy the `tar` or self-extracting archive file `CryoSoft.tar` in this directory;

Step 2. de-compress the archive file with the command

```
tar –xvf CryoSoft.tar
```

The archive will expand into the `CryoSoft/` directory that will be the root location for the installation.

Note We use through this guide the UNIX concepts of file and directory. A file name is indicated in `Courier`. If the file name is terminated with a slash `/` it indicates a directory in your disk. A file name terminated with a star `*` indicates that the file is executable by the shell. Variations applicable to Windows-2000 system are detailed whenever applicable.

The directory `CryoSoft/` is the root for the system of codes. After unpacking it contains the following files and directories:

`CryoSoft/`

`CSmake*`	installation script. See Appendix I for a description of the options of this script
`etc/`	directory containing the makefiles (compilation scripts) for each single code.
`src/`	directory containing the source files for the libraries and analysis codes.
`usr/`	directory containing the standard source files for the user's defined external routines that we provide in dummy form.
`xample/`	directory containing examples and test programs.

Note that the scripts provided, as well as the makefiles and libraries are for all platforms supported. Obviously, only a part of them will be used on a specific platform.

Environment setting The executable script `CSmake` that we provide for installation for UNIX or Linux requires the definition of the environment variable `CRYOSOFT`. This variable contains the absolute path of the `CryoSoft/` directory where the archive has been unpacked. We recommend that you define this variable in the shell resources file (e.g. `.cshrc` if you are running `csh`, `.bashrc` if you are running `bash`).

The steps 3 and 4 below describe the setting of this environment variable. As an example, assume that you have unpacked the codes following the instructions in the previous section starting from your home directory `/usr/home/` thus generating a directory `/usr/home/CryoSoft/` and you are running `bash` as your UNIX shell. You then need to:

Step 3. add the following statements in your `.bashrc` file:

```
CRYOSOFT=/usr/home/CryoSoft
export CRYOSOFT
```

Step 4. start a new shell or source the .bashrc file with the command

```
source .bashrc
```

so that the `CRYOSOFT` variable is defined in the environment. Steps 3. and 4. above must be adapted to the actual location of the `CryoSoft/` directory in your file system, and to the shell that you are running.

Note The name of the `CryoSoft/` directory can be changed, e.g. to keep track of subsequent releases. In this case you must update the `CRYOSOFT` environment variable accordingly to guarantee the functionality of the installation and compilation scripts.

Build All the codes provided are recognized, compiled and linked automatically using the script `CSmake` that you find in the `CryoSoft/` directory. To do so you should have a running shell. This is the standard shell under UNIX or Linux. Now:

Step 5. change the directory to `CryoSoft/`

Step 6. execute the script `CSmake` a first time with the option `–configure`:

```
CSmake –configure
```

this will recognize the architecture and will find the location and type of compilers available. This information is stored in the file `arch.conf` generated automatically by `CSmake` in the directory `CryoSoft/etc/`. In addition, a directory `CryoSoft/inc/` is created, containing the header files `c2f_call.h` and `c2f_vars.h` for C compilations.

Step 7. execute the script `CSmake` without options

```
CSmake
```

this will compile all library sources, create the library archives, compile the source codes of the applications and generate the executable applications.

At the end of the execution three new directories are generated, containing the libraries, the object modules of the codes compiled and the applications as executable binaries:

`CryoSoft/`	
`obj/`	directory containing the object modules that are used for successive compilations (e.g. when customizing the applications using modified external routines)
`lib/`	directory containing the binary libraries needed for linking
`bin/`	directory containing the applications as binary executables

To ease the use of the codes we suggest that you add the `CryoSoft/bin` directory to your path.

Organization of applications and libraries

After a successful installation the set of executable applications will be available in the `CryoSoft/bin/` directory. These are the stand-alone applications for analysis, tests and post-processing. In the case of a complete installation of the CryoSoft package of codes, the directory `CryoSoft/bin/` will contain the following applications:

`solids`	facility to perform direct calls to the library `solids.a`.
`fluids`	facility to perform direct calls to the library `fluids.a`.
`cid`	cable interactive designer.
`flower,flowerpost`	cryogenic hydraulic networks analysis.
`heater,heaterpost`	heat conduction solver.
`mac`	field, inductance, forces and AC loss analysis of a superconducting magnetic system.
`power,powerpost`	electrical circuit analysis.
`supermagnet`	multi-tasking code manager for the integration of codes.
`thea`, `theapost`	thermohydraulic and electric analysis of superconducting cables.
`zerodee`, `zerodeep`	0-D stability analysis.

A set of libraries are generated during installation in the directory `CryoSoft/lib/`. These libraries are only necessary at compilation time, when they are linked to the object modules of the codes in order to generate the executable applications. Therefore they are not necessary to run the applications. However they are saved for later use and in particular the creation of customized executables (see later).

The libraries are organised in layers. The first layer is a set of properties for solid materials and fluids:

`solids.a`	collection of thermophysical and electrical properties of solid materials (metals, alloys, superconductors, composites and insulators). The stand-alone application `solids` is provided to allow direct calculation of properties.
`fluids.a`	collection of thermodynamic, thermophysical and transport properties of cryogenic fluids (helium and nitrogen). The stand-alone application `fluids` is provided to allow direct calculation of fluid properties.
`friction.a`	collection of correlations for the calculation of the friction factor coefficient in a pipe.
`heattransfer.a`	collection of correlations for the calculation of the heat transfer coefficient in a pipe.

The second layer is a set of mathematical libraries and graphic libraries. These are organised as follows:

`FE.a`	finite elements library containing shape functions definitions.
`PDE1D.a`	1-D PDE solver library.

`PDE3D.a`	3-D PDE solver library.
`IC.a`	intercommunication routines SuperMagnet and children.
`algebra.a`	linear algebra library.
`dBase.a`	data-base management library.
`geometry.a`	library of geometric operations in 2-D and 3-D space.
`magnetic.a`	magnetic calculation routines.
`maths.a`	mathematical functions library.
`miscellanea.a`	miscellaneous functions library.
`parser.a`	string parser library.
`postpro.a`	post-processing data handling and plotting library.
`storage.a`	i/o storage functions library.
`GX.a`	graphic library.
`plt_PS.a, xll_PS.a`	low-level PostScript file generation libraries.
`plt_IGX.a`	low-level interactive graphics.
`3dworld.a`	3-D graphic functions library.

The lowest layer is provided by the system calls:

`system.a`	library for interfacing with system specific calls

Finally during the compilation described previously, object modules are generated for all the codes, and stored in the directory `CryoSoft/obj/`. As for the libraries, these object modules are not necessary to run the applications. It is however useful to store them for later use in the creation of customized executables.

Maintenance and customization

Ports The most efficient method to re-install the codes on a different machine, or architecture, is to use the tar option of the script `Csmake`. The command:

```
CSmake –tar
```

Builds an archive file, `CryoSoft.tar`, containing all source codes, the scripts and configurations files necessary to re-install following the procedure described earlier. It is possible to perform a selective port by specifying a code target to the above command, e.g.

```
CSmake –tar thea
```

to pack only source codes relative to a given application (`thea` in the above example).

Patches If for any reason the source codes are modified, e.g. after applying patches that we may provide to improve functionality or to correct bugs, you will need to re-compile the codes affected by the modifications. This can be done simply by launching the script `CSmake` without options:

```
CSmake
```

The script will recognize the modified codes and proceed to the compilation of the necessary libraries and applications as needed.

Clean installation We suggest that you use the `CSmake` script also for cleaning and re-installing the code system if and when needed. As an example it may be necessary to perform a new installation after a change of operating system version. It is possible to completely clean the installed codes by using the `CSmake` command with the `–clean` option:

```
CSmake –clean
```

This erases all binaries produced by the compilation and the system configuration file `arch.conf`. After cleaning the installation, you can proceed to a new installation by launching once the `CSmake` script with the `–configure` option:

```
CSmake –configure
```

to recognize the system architecture and the compilers, and finally without options:

```
CSmake
```

This will generate new libraries, object modules and executable applications, as for the virgin installation described previously.

Warning The `-clean` option cannot be used for maintenance in the case that you have purchased a pre-compiled installation of the code system. This option will cause the irreversible removal of all executable applications, libraries and object modules. As no source is provided in this case, it will not be possible to re-install the system of codes.

Customization You can generate your own, customized applications by programming dedicated functions in the External Routines provided with most applications in the CryoSoft suite of codes.

We always provide dummy External Routines that can be used as a template for programming your own version. These are generally FORTRAN subroutines or functions contained in files located in the `CryoSoft/usr/` directory, and need to be compiled and linked with the object modules of the corresponding application in order to generate an executable binary. See the manuals for each application for details on nature and structure of the External Routines.

For any customization we suggest that you generate your own set of External Routines, compilation scripts, customized objects and applications in separate directories. This avoids conflicts with the standard code and the directory structure can be tailored to your own needs. The compilation scripts (makefiles, with extension `.make`) that we provide with the standard installation in the `CryoSoft/etc/` directory can be easily used as templates for the generation and maintenance of the applications thus generated. The scripts support a structure where the customized code, the object modules and the resulting application are in three different directories. Three variables are defined to this purpose in the makefile *`appl`*`.make` for an application *`appl`* (e.g. `thea.make` for `thea`):

`USER`	directory containing the External Routines. By default `USER` is the directory containing the standard External Routines (e.g. `CryoSoft/usr/thea/code_`*`X.x`* for `thea` version *X.x*)
`UOBJ`	directory containing the object modules for the External Routines after compilation. By default `UOBJ` is the standard directory (e.g. `CryoSoft/obj/thea/code_`*`X.x`* for `thea` version *X.x*)
`EXE`	directory containing the binary executable application. By default `EXE` is the directory `CryoSoft/bin`

You should make a copy of the makefile and modify the three variables above in accordance with the directory structure that you have generated for the modified source codes, objects and applications.

Note Any modification of standard source code or scripts can impair the functionality of the installation scripts and of the applications concerned. We therefore strongly suggest that you back-up the installation before modifying standard code so that, should a problem arise, you can return to a clean condition at any time.

The makefile for the application *`appl`* is launched under UNIX and Linux with the command:

```
make –f appl.make
```

The compilation scripts are configured to be fully portable through your file system. They make reference to the installation directory through the environment variable `CRYOSOFT`. This is necessary to link the libraries and the object modules for the parts of the codes that are not customized.

Warning External Routines give unlimited access to the data structure used by the programs. Improper programming of External Routines can therefore corrupt operation and lead to evident or concealed malfunctions and generate manifest or hidden errors in the computed results. IN NO EVENT WILL CRYOSOFT BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY AUTHORISED OR UNAUTHORISED USE OF THIS FEATURE, even if advised of the possibility of such damages.

Problems, bugs and queries

Please address all questions you may have, problems you may find, and bug reports to:

`cryosoft.support@gmail.com`

Also, visit regularly our web site at:

`http://cryosoftsupport.wix.com/cryosoft`

as well as the web site of our official and exclusive distributor, Horizon Technologies, at:

`www.htess.com`

for an update on the codes, their capabilities and documentation.

Appendix I

CSmake

We provide the script `CSmake` to configure, maintain and compile all or single codes. The script has the same syntax for all installations. The general syntax of the `CSmake` script is the following:

```
CSmake [targets] [-help] [-clean] [-configure] [-tar]
                 [-version] [-verbose]
```

where the various options have the following meaning:

`-clean` remove all CryoSoft libraries, objects and executables, targets are ignored

`-configure` configure the CryoSoft structure, targets are ignored

`-help` list the syntax parameters

`-version` displays the current versions of the programs

`-verbose` verbose, print running information

`-tar` build an archive of the installation (or selected targets)

`targets` a list of the codes to be compiled and linked. The list contains one or more of the following allowed keywords:

`system`	create the system library
`libraries`	create the generic libraries
`graphics`	create graphic libraries and test programs
`fluids`	create libraries of the fluid properties and test programs
`solids`	create libraries of solids properties and test programs
`cid`	create cid
`flower`	create flower and its postprocessor
`gandalf`	create gandalf and its postprocessor
`heater`	create heater and its postprocessor
`mac`	create mac
`opticon`	create opticon
`power`	create power and its postprocessor
`supermagnet`	create supermagnet
`thea`	create thea and its postprocessor
`zerodee`	create zerodee and its postprocessor

if no ***`targets`*** are specified (void list), all codes present in the `CryoSoft/src/` directory are compiled and linked.

Appendix II

CYGWIN

CYGWIN is a Linux-like environment for Windows, freely available for download at `http://www.cygwin.com/`, or commercially distributed by Red Hat (see details at `http://www.redhat.com/software/cygwin/`). CYGWIN consists of a collection of tools that provide a Linux look and feel, and a DLL (cygwin1.dll) that acts as a Linux API emulation layer. Specifically, once installed, CYGWIN provides a terminal with a running Linux shell. This shell executes scripts (such as `CSmake`) and commands (such as `tar` and `make`) in a fashion that makes the installation of the CryoSoft codes package under Windows practically identical to that under UNIX or Linux. In addition, CYGWIN delivers the open source standard GNU gcc and g77 compilers, as well as the X11 server, applications and API as a specific part of the project (see details at `http://x.cygwin.com/`).

CYGWIN is best obtained using the installer `setup.exe`, and following the installation manual, both obtained at `http://www.cygwin.com/`. Note that, when installing packages for the first time, `setup.exe` does not install every package. Only the minimal base packages from the CYGWIN distribution are installed by default. In addition to the minimal installation, for the installation and run of the CryoSoft codes you will need to install the following packages:

`devel`	
`make`	(the GNU make utility)
`X11`	
`X-startup-scripts`	(startup scripts for X11)
`xorg-x11-base`	(basic CYGWIN/X)
`xorg-x11-bin`	(X11 binaries)
`xorg-x11-bin-dlls`	(X11 binary DLLs)
`xorg-x11-devel`	(X11 header and import libraries)
`xorg-x11-fents`	(X11 font encodings)
`xorg-x11-fnts`	(X11 fonts)
`xorg-x11-xwin`	(CYGWIN X server)
`xterm`	(xterm emulator)

The CryoSoft codes that do not need interactive graphics can be started from the CYGWIN console terminal (i.e. without starting the X server) and run as from a UNIX or Linux shell. The compiled codes require that the run-time library `cygwin1.dll` (provided with the standard CYGWIN installation) is visible in the Windows PATH variable. With a standard installation, this DLL is located in `C:\cygwin\bin\cygwin1.dll`. To make sure that this DLL is found, it is possible to copy, or link this DLL library to one of the directories searched by Windows to find executables, e.g. `C:\Windows\System` for a standard installation of Windows on a `C:\` disk.

Codes that require interactive graphics need a running X server at run-time. CYGWIN provides a Windows .bat file to start the X server, `startxwin.bat`. In a standard CYGWIN installation this file is located in `C:\cygwin\usr\X11R6\bin\startxwin.bat`. Again, it is possible to copy, or alias this .bat file to ease its use. Note that the standard `startxwin.bat` starts an xterm that can be used to run the CryoSoft codes as in a UNIX or Linux environment.