

# CUTEr Installation Manual for LINUX platform under Ubuntu 9.04

Carlos Quintero  
Carlos Ramirez  
Reinaldo Sanchez

Miguel Argaez  
Leticia Velazquez<sup>1</sup>

The University of Texas at El Paso  
Program in Computational Science

May 2009

<sup>1</sup>The authors thank the financial support from the ARL Grant No. W911NF-07-2-0027 and the Program in Computational Science, and the office space provided by the department of Mathematical Sciences.

CUTEr is a versatile testing environment for optimization and linear algebra solvers. The package contains a collection of test problems, along with tools intended to help developers design, compare and improve new and existing solvers. It allows the interfacing with solvers written in C, C++, Fortran 77, Fortran 90/95 and MATLAB.

For more information go to <http://hsl.rl.ac.uk/cuter-www/index.html>

The following installation manual is based in an updated version of CUTEr that is now available featuring simplified tools and a new MATLAB interface with dynamic allocation.

### 1. *Setting up the installation*

You will install SifDec2 and CUTEr2 (development versions), but first you probably need to install **subversion** (SVN), which it is used to maintain current and historical versions of files such as source code, web pages, and documentation. To do this, open a terminal and type

```
sudo apt-get install subversion
```

We create the folder CUTErInstallation by typing in the terminal

```
mkdir $HOME/CUTErInstallation
```

Now, we edit the file `~/.bashrc` which determines the behavior of interactive shells (in this tutorial we assume you use the bash as shell). In this case, we use the text editor *gedit*

```
gedit $HOME/.bashrc
```

and we need to add the variables CUTErDIR, CUTEr and SIFDEC. For this, at the end of the file add the following lines:

```
export CUTErDIR=$HOME/CUTErInstallation
export CUTEr=$CUTErDIR/cuter2
export SIFDEC=$CUTErDIR/sifdec2
```

Save and close the file. After this, we need to reload the environment variables, by executing in the terminal the following instruction

```
source $HOME/.bashrc
```

We will install CUTER using the C compiler **gcc 4.1**. In case, you do not have it already installed, execute the following

```
sudo apt-get install gcc-4.1
```

It will also be necessary to install the Fortran compiler **g95**, since GFORTRAN is not officially supported to create MEX files and often results in a hard MATLAB crash. For this, in the command line, write the following

```
cd $CUTERDIR
mkdir g95
cd g95
```

and then download the compiler *g95* by executing

```
wget http://ftp.g95.org/v0.91/g95-x86-linux.tgz
tar -zxvf g95-x86-linux.tgz
```

Now, we copy the binary *g95* to leave it available:

```
sudo cp $CUTERDIR/g95/g95-install/bin/i686-suse-linux-gnu-g95 /usr/local/bin/g95
```

as well as the *g95* libraries

```
sudo cp -R $CUTERDIR/g95/g95-install/lib/gcc-lib/ /usr/local/lib/
```

## 2. Download SIFDEC and CUTER

The CUTER test problems provided are written in so-called Standard Input Format (SIF). SIFDEC is a decoder to convert from this format into well-defined Fortran 77 and data files.

To download the development version of SifDec, use

```
svn co https://magi-trac-svn.mathappl.polymtl.ca/SVN/cuter/sifdec/branches/SifDec2 ./sifdec2
```

To download the development version of CUTER, which contains many enhancements and much simplified tools, use

```
svn co https://magi-trac-svn.mathappl.polymtl.ca/SVN/cuter/cuter/branches/CUTER2 ./cuter2
```

## 3. Install SIFDEC

We go to the *SIFDEC* folder and run the installation. In the terminal, type

```
cd $SIFDEC
./install_sifdec
```

We now need to define new environment variables: *MYSIFDEC*, *MANPATH*, and *PATH*. In order to do this, edit again the file `~/.bashrc` by adding the following lines

```
export MYSIFDEC=$SIFDEC/SifDec.large.pc.lnx.gfo
export MANPATH=$SIFDEC/common/man
export PATH=$SIFDEC/SifDec.large.pc.lnx.gfo/bin
```

#### 4. *Install CUTER*

Since we need to use **gcc-4.1**, it is necessary to add a new entry for the option of the C compilers in the installation of CUTER. Edit the file `$CUTER/build/arch/c.arch` by typing in the terminal

```
gedit $CUTER/build/arch/c.arch
```

and add the following liner right after `"# All platforms and operating systems"`

```
all ; all ; gcc-4.1 ; -DIsgcc ; GNU_gcc-4.1
```

Now, we proceed with the installation of CUTER: execute the following instructions

```
cd $CUTER
./install_cuter
```

Again, we need to add a new environment variable *MYCUTER*, and update *MANPATH*, *PATH* and *LIBPATH*: edit the file `~/.bashrc` and add the following lines at the end

```
export MYCUTER=$CUTER/CUTER.large.pc.lnx.g95
export MANPATH=$MANPATH:$CUTER/common/man
export PATH=$PATH:$CUTER/CUTER.large.pc.lnx.g95/bin
export LIBPATH=$LIBPATH:$CUTER/CUTER.large.pc.lnx.g95/double/lib
```

#### 5. *Setting up MATLAB*

Open MATLAB and add the following paths (you need to start MATLAB as root in order to save the paths. To do this just type in the terminal: `sudo matlab`)

In MATLAB, go to the menu *File* and then click on *Set Path*. Add the following paths

```
/home/username/CUTErInstallation/cuter2/CUTEr.large.pc.lnx.g95/double/bin
```

```
/home/username/CUTErInstallation/cuter2/common/src/matlab
```

Save the changes and close MATLAB.

## 6. Download the CUTEr problems collection

Among the CUTEr test problems are the CUTE test problems. These problems span a wide spectrum of difficulty, ranging from small-scale differentiable unconstrained minimization to large-scale equality and inequality-constrained dense and sparse problems, systems of nonlinear equations, network problems and so on.

First define the new variable *MASTSIF* where the problems will be located. Add the following line to the file `$HOME/.bashrc`

```
export MASTSIF=$CUTERDIR/mastsif
```

Again, reload the environment variables by executing in the terminal the instruction

```
source $HOME/.bashrc
```

Create the folder containing the problems: type the following

```
mkdir $MASTSIF  
cd $CUTERDIR
```

Now, you can download the problem set (you will download a set of small problems and another set of large problems)

```
wget ftp://ftp.numerical.rl.ac.uk/pub/cuter/mastsif_small.tar.gz  
wget ftp://ftp.numerical.rl.ac.uk/pub/cuter/mastsif_large.tar.gz
```

Decompress the files (once you do this, all the problem files will be in the folder *mastsif*)

```
tar -zxvf mastsif_small.tar.gz  
tar -zxvf mastsif_large.tar.gz
```

## 7. Compile Fortran subroutine using MATLAB by building MEX-file

Open MATLAB, type in the command line the following instruction

```
>>mex -setup
```

and choose to use “*mexopts.sh*” (in our case, this was the option #3)

Now you will need to edit (with your favorite text editor) the file `~/.matlab/mexopts.sh` (sometimes it is under `~/.matlab/R2008b/mexopts.sh` - assuming R2008b is your MATLAB version)

Edit the C compiler options so they look like:

```
CC='gcc-4.1'  
CFLAGS='-ansi -D_GNU_SOURCE'  
CFLAGS="$CFLAGS -fPIC -pthread -m32"  
CFLAGS="$CFLAGS -fexceptions"  
CFLAGS="$CFLAGS -D_FILE_OFFSET_BITS=64"  
CLIBS="$RPATH $MLIBS -lm"  
COPTIMFLAGS='-O -DNDEBUG'  
CDEBUGFLAGS='-g'  
CLIBS="$CLIBS -L/usr/local/lib/gcc-lib/i686-suse-linux-gnu/4.0.3/ -lf95"
```

The Fortran compiler options should look like

```
FC='g95'  
FFLAGS='-fexceptions'  
FFLAGS="$FFLAGS -fPIC"  
FLIBS="$RPATH $MLIBS -lm"  
FOPTIMFLAGS='-O'  
FDEBUGFLAGS='-g'
```

Once you restart MATLAB, you should be able to compile your MEX file.

## 8. *Modifying runcuter*

In the terminal execute

```
gedit $MYCUTER/bin/runcuter
```

and modify the line

```
command="$MEXFORTRAN -I${CUTER}/common/include -output ${outputName} ${CUTER}  
/common/src/tools/mcuter.c ELFUN.o GROUP.o RANGE.o ${EXTER} -L${LIBDIR} -lcuter  
${BLAS} ${LAPACK} ${PACKLIBS}"
```

to

```
command="${MEXFORTRAN} -I${CUTER}/common/include -output ${outputName} ${CUTER}/
common/src/tools/mcuter.c ELFUN.o GROUP.o RANGE.o ${EXTER} -L${LIBDIR} -lcuter ${BLAS}
${LAPACK}"
```

## 9. Test the new MATLAB interface

Before problem data may be accessed from Matlab, the problem must be decoded and linked with the main Matlab gateway interface. To do this, run the following command at the shell prompt. We use problem GENHS28 as an example.

```
runcuter --package mx --decode GENHS28
```

Now, you can open MATLAB, and at the Matlab command line, problem data is loaded using the instruction

```
>> prob = cuter_setup();
```

This creates a MATLAB structure stored in the variable *prob*, which holds problem-related data. Asking Matlab to print this structure, we get

```
>> prob
prob =
    n: 10
    m: 8
  nnzh: 19
  nnzj: 24
    x: [10x1 double]
    b1: [10x1 double]
    bu: [10x1 double]
    v: [8x1 double]
    c1: [8x1 double]
    cu: [8x1 double]
  equatn: [8x1 logical]
  linear: [8x1 logical]
    name: 'GENHS28'
```

The first four fields of *prob* are, in order, the number of variables of the problem, the number of constraints, the number of nonzero elements in a triangle of the Hessian of the Lagrangian and the number of nonzero elements in the Jacobian matrix of the constraints. Fields in Matlab are accessed using the dot operator (e.g. typing *prob.nnzh* in the MATLAB command line should return 19 as value)

More information about the new MATLAB interface can be found in the webpage <https://magi-trac-svn.mathappl.polymtl.ca/Trac/cuter/wiki/NewMatlabInterface>