Install Instructions for `dcicpp`, the algorithm for the Dynamic Control of Infeasibility method

Abel Soares Siqueira
Advisor: Francisco A. M. Gomes

10 de março de 2012

Obs. 1: This is a provisory document.
Obs. 2: This guide is for linux.
Obs. 3: The file `downall.sh` *inside dcicpp will download all dependencies, except base_matrices, Goto BLAS and CUTEr*
Obs. 4: The file `downcuter.sh` *inside dcicpp will download CUTEr*

All the libraries were downloaded to the same folder in the home folder. For simplicity, we will assume the user do the same. Hence, the first thing to do will be create such folder.

```
$ mkdir $HOME/Libraries
$ cd $HOME/Libraries
```

1 Obtaining

You will need many things to install dcicpp.

1. The source for dcicpp itself

```
$ git clone git://github.com/abelsiqueira/dcicpp.git
```

2. base_matrices

```
git clone git://github.com/abelsiqueira/base_matrices.git
```

3. CHOLMOD (e outras bibliotecas do Tim Davis)

```
http://www.cise.ufl.edu/research/sparse/cholmod/current/CHOLMOD.tar.gz
http://www.cise.ufl.edu/research/sparse/amd/current/AMD.tar.gz
http://www.cise.ufl.edu/research/sparse/camd/current/CAMD.tar.gz
http://www.cise.ufl.edu/research/sparse/colamd/current/COLAMD.tar.gz
http://www.cise.ufl.edu/research/sparse/ccolamd/current/CCOLAMD.tar.gz
http://www.cise.ufl.edu/research/sparse/UFconfig/current/UFconfig.tar.gz
```

4. Metis

```
http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/metis-5.0.2.tar.gz
```

5. Goto BLAS. Download from the site

```
http://www.tacc.utexas.edu/tacc-projects/gotoblas2
```

Uncompress everything. And get ready to start installing. **Note: we will install everything in 32 bits**.

2 Installing

1. Goto BLAS The following command will install Goto BLAS with 32 bits.

```
$ make BINARY=32
```

If needed, use TARGET option too. We needed to set to CORE2, because our processor was not supported.

```
$ make BINARY=32 TARGET=CORE2
```

2. Metis You will need to add `-m32` to the compiler. Edit the file `GKlib/GKlibSystem.cmake` and add `-m32` to the line 31. It shoul become

```
set(GKlib_COPTIONS "{GKlib_COPTIONS} -m32 -std=c99 -fno-strict-aliasing")
```

Then, enter the commands (`#` means root access)

```
$ make config
$ make
# make install
```

3. CHOLMOD Edit `UFconfig/UFconfig.mk`:

- Add `-m32` to `CF` and `F77FLAGS`
- Comment out lines

  ```
  BLAS = -lblas -lgfortran
  LAPACK = -llapack
  ```

- Add line

  ```
  BLAS = -L$(HOME)/Libraries/GotoBLAS2/ -lgoto2 -lgfortran -lgfortranbegin
  ```

- Edit the variable `METIS_PATH` and `METIS` to reflect your configurations. In our case

```
    METIS_PATH = $(HOME)/Libraries/metis-5.0.2
    METIS = /usr/local/lib/libmetis.a
```

Now, go to the folder CHOLMOD and enter

```
$ make all
```

4. **base_matrices** add the lines from the file in the `addtobash.rc` to your configuration file `$HOME/.bashrc`

   Remove the CUTER parts, if you do not intend to use CUTEr.

   Open a new terminal or use the command

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

   If needed, edit `make.inc` and enter

   ```
   $ make all
   ```

5. **dcicpp** If needed, edit `make.inc` and enter

   ```
   $ make all
   ```

Your installation is now complete. You can test and compare this algorithm with CUTEr. To do so, proceed to the next section.

3 CUTEr

First, download CUTEr using svn

```
$ svn co https://magi-trac-svn.mathappl.polymtl.ca/SVN/cuter/sifdef/branches/SifDec2 ./sifdec2
$ svn co https://magi-trac-svn.mathappl.polymtl.ca/SVN/cuter/cuter/branches/CUTEr2 ./cuter2
```

Now download the SIF 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
```

Uncompress all files

```
$ tar -zxf mastsif_small.tar.gz
$ tar -zxf mastsif_large.tar.gz
```

We need to add some lines to `$HOME/.bashrc` that depend on your configuration. We will use the following options with CUTEr, if you need to change them, then the lines could change too.

- PC

- Linux
- gfortran
- gnu g++
- double
- large

In file `$HOME/.bashrc`, add before the `addtobash.rc` file.

```
ROOTCUTER="$HOME/Libraries"

export CUTER="$ROOTCUTER/cuter2"
export MYCUTER="$CUTER/CUTEr.large.pc.lnx.gfo"
export SIFDEC="$ROOTCUTER/sifdec2"
export MYSIFDEC="$SIFDEC/SifDec.large.pc.lnx.gfo"
export MASTSIF="$ROOTCUTER/mastsif"
export MANPATH="$CUTER/common/man:$SIFDEC/common/man:$MANPATH"
export PATH="$MYCUTER/bin:$MYSIFDEC/bin:$PATH"
```

Open a new terminal or use the command

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

Now, go to the sifdec2 folder and edit the file `config/linux.cf`.

Search for Isxxx, where xxx reflects your choice of fortran compiler. In your case, we used `gfortran`, so we search for Isgfo. Then, add -m32 to `FortranFlags`. Also, edit the file `config/all.cf` Search for Isgpp and add -m32 to CFlags

Note: In the latest version, you must edit the file config/Umake.tmpl, line 46, and fix FFLAGS (missing S).

Back in the sifdec2 directory, enter

```
$ ./install_sifdec
```

When asked if you want to run `install_mysifdec`, press enter (only if you already modified .bashrc, otherwise modify it now, as proposed by the installer). When asked if you want to `make all`, press enter.

Go to cuter2 directory and edit the file `config/linux.cf`. Repeat the procedure above to add -m32 to the fortran compiler. Now edit the file in `cuter2/config/all.cf` Repeat the procedure above to add -m32 to the `C++` compiler. Now open the file `config/Umake.tmpl`, find the line 43 with

```
CFLAGS     = CFlags
```

and add below it the line

```
CXXFLAGS   = CFlags
```

Now go to the file `$CUTER/common/include/cuter.h` There, change the definition os integer and logical to long int, i.e.

```
typedef long int integer
typedef long int logical
```

Finally, go to the cuter2 directory and enter

```
$ ./install_cuter
```

and follow the same instruction as above.

Create a directory and test cuter with the command

```
$ runcuter -p gen -D ROSENBR
```

Now, compile dcicpp for cuter. In the dcicpp folder, enter

```
$ make cuter
```

See `TESTING.cuter` inside dcicpp to proceed with testing dcicpp with CUTEr