Documentation of ReLIADiff. A C++ Software Package For Real Laplace transform Inversion based on Algorithmic Differentiation *

LUISA D'AMORE†, ROSANNA CAMPAGNA†, VALERIA MELE†
ALMERICO MURLI‡

† University of Naples Federico II, Via Cintia, Naples, Italy
‡ CMCC -Lecce, Italy and SPACI

DEMOS User Guide

*Accompanying the paper in [1]

Contents

1 Introduction

In the directory `C++_files/demos-Linux_g++` users find demos to compile and execute on a Linux system with a g++ compiler installed.

> *See the* ***RELIADIFF_userguide.pdf*** *file to know about suggested compilers.*

Demos use RELIADIFF

1. On a function of the Laplace Transform/Inverse database (in `DEMO1-Functions_From_DATABASE`)
   - User may give different arguments in the provided shell scripts.
   - Demo generates a file with results.
2. On a Laplace Transform, to be defined in the demo with its Inverse (in `DEMO2-Function_To_Give`)
   - User may change the Transform/Inverse definition (and the related parameter szero in the main program) <u>before</u> to compile to test RELIADIFF on his own.
   - Demo generates a file with results.

2 Content of DEMOS directory

The directory `C++_files/demos-Linux_g++` contains three directories and two files:

- DEMO1–Functions–From–DATABASE

  > *To know about the functions in the database, look at the Database_function_list.pdf file.*

- DEMO2–Function–To–Give

  > *To change the function to test RELIADIFF on, user must edit the file FunctionForTest.c, before to compile the demo. See the proper section in this document.*

  – The demo will generate a file with results for the defined function.

- utility: files containing utility routines needed to execute functions in the database and the demos. Here are also functions to print in a file the role of the *flag*, *Ncalc* and *Nopt* variables returned by RELIADIFF.

*A description of the **utility** directory is in the Appendix.*

- gsl–1.15.tar.gz: a compressed file containing the installation files for the GNU Scientific Library (GSL), that user could need to install.
- Makefile_gsl: makefile installing the GSL locally, if it is needed.

The files about the GSL are useful if there isn't a GSL version previously installed in the GSL default path.

3 DEMO 1

3.1 Purpose

In the directory `DEMO1-Function_From_DATABASE` users find files to build a demo–program driving tests of RELIADIFF routine on the functions in the database.

*View the **Database_Function_list.pdf** file in the Doc directory to know details about the functions in the database.*

3.2 Description

This main program:

- obtains the needed parameter from the user
- sets the abscissa of convergence depending on the function chosen by the user
- sets the *szero* parameter as needed by the function chosen by the user
  - *szero=1* if the Transform has a singularity at zero
- calls RELIADIFF routine for each evaluation point required by the user
- compare the results with the known inverse values
- prints the output in a txt file named with the number of the function

The user can choose how many functions and which ones to use.

*View the **Database_Function_list.pdf** file in the Doc directory to know numbers corresponding to functions in the database.*

3.3 Content of directory

The directory contains:

- **TEST_ON_DATABASE.c**: the main program
- **Makefile**: to compile the demo with a GNU Scientific Library (GSL) already installed (as root in the default path).
  - The command `make` compiles the c file linking the needed, to obtain the executable file `test_on_database`
  - The command `make clean` deletes executable file, the txt files in the directory and the created libraries
- **database**: files needed to generate a library of 89 functions that authors used to test RELIADIFF.

*A description of the **database** directory is in the Appendix.*

- **scripts**: 6 shell scripts running the program with different kinds of input.
  - **TESTONE_defaultParameters.sh**: a shell script running the program passing no arguments (user shall explicitly choose a function at runtime, giving the corresponding number when required)
  - **TESTONE_givenPoints.sh**: a shell script running the program on a single function with all default parameters but explicitly giving some evaluation points (user shall explicitly choose a function at runtime, giving the corresponding number when required)
  - **TESTONE_tol–3.sh**: a shell script running the program on a single function providing the required accuracy (tol=1^{-3}) and giving a range for the evaluation points (user shall explicitly choose a function at runtime, giving the corresponding number when required)
  - **TEST_10_tol–4.sh**: a shell script running the program with tol=1^{-4}, on 10 functions from database (the user shall explicitly choose the functions at runtime, giving a number a time when required)
  - **TESTALL_tol–4.sh**: a shell script running the program with tol=1^{-4}, on all the functions in the provided database, giving a range for the evaluation points and printing all the used coefficients for each evaluation point.
  - **TESTALL_defaultParameters.sh**: a shell script running the program on all the functions in the database without any other argument

The directory contains also:

- **Makefile_gslloc**: just to compile the demo with a GNU Scientific Library (GSL) installed locally, if user does not already have one installed in the default path.
  - command `make -f Makefile_gslloc` compiles the c file linking the libraries, to obtain the executable file `test_on_database`

3.4 How to compile

3.4.1 GSL already installed in /usr/local/

By default, the GSL is installed in

- `/usr/local/bin`,
- `/usr/local/man`,
- `/usr/local/lib`,
- `/usr/local/include`,
- ...

That is in `/usr/local/`. So, if you previously installed the GSL without changing path:

1. Enter the directory `demos-Linux_g++/DEMO1-Functions_From_DATABASE`
2. Type the command `make`

3.4.2 GSL already installed in a different path

When installing GSL user can specify an installation prefix other than '`/usr/local`' by giving `configure` the option '`--prefix=PATH`'. So, if you previously installed the GSL changing path:

1. Find the path of the GSL: `/yourpath/` (example: `/usr/`).
2. Enter the directory `demos-Linux_g++/DEMO1-Functions_From_DATABASE`.
3. Open the file `Makefile`.
4. Edit only the path for the *gsllib* variable (as shown in the figure 1).

```
In file Makefile (if GSL is already installed in /yourpath/):
...
## CHECK
##########################################################################################
### GSL building directory: change it if you know a different one in your environment ###
ifndef gsllib

export gsllib = /yourpath/
endif
##########################################################################################
...
```

Figure 1: In red the path to change in the Makefile.

5. Type the command make.

3.4.3 GSL not installed

1. Install the GSL locally (in the `C++_files/demos-Linux_g++/GSL` directory), that is
   (a) Enter the `demos-Linux_g++` directory
   (b) Type the command `make -f Makefile_gsl`
2. Enter the directory `demos-Linux_g++/DEMO1-Functions_From_DATABASE`
3. Type the command `make -f Makefile_gslloc`

3.5 How to execute

After compiling, with the makefile, in this directory there will be an executable file named `test_on_database`.
To execute it, user should only provide it the arguments he/she wants, as described at the previous point:

```
./test_on_database [ordered list of arguments]
```

or he/she may use one of the provided scripts, described at the beginning of this section.

3.6 Arguments

The user may execute `test_on_database` with 6 kinds of INPUT:

1. **none argument**
2. **one argument**
3. **two arguments**
4. **three arguments**
5. **four arguments**
6. **more than seven arguments**

Where:

1. **none argument**: The driver uses all default values:

tol	required accuracy
ntf= 1	number of functions testing RELIADIFF with
nmax= 2000	maximum number of series coefficients
pcoeff= 0	do not print coefficients
x= 1, 5, 10, 15	evaluation points

2. **1 argument**: *tol* (if it is a string "n" it is posed equal to the default value).
   The driver will use other default values:
   ntf= 1
   nmax= 2000
   pcoeff= 0
   x= 1, 5, 10, 15

3. **2 arguments**: *tol*, *ntf* (if one is a string "n" it is posed equal to the default value; if *ntf* is the string "a" it is posed equal to the total number of functions in the database).
   The driver will use other default values:
   nmax=2000
   pcoeff= 0
   x= 1, 5, 10, 15

4. **3 arguments**: *tol*, *ntf*, *nmax* (if one is a string "n" it is posed equal to the default value; if *ntf* is the string "a" it is posed equal to the total number of functions in the database).
   The driver will use other default values:
   pcoeff= 0
   x= 1, 5, 10, 15

5. **4 arguments**: *tol*, *ntf*, *nmax*, *pcoeff* (if one is a string "n" it is posed equal to the default value; if *ntf* is the string "a" it is posed equal to the total number of functions in the database).
   The driver will use other default values.
   x= 1, 5, 10, 15

6. > 7 **arguments**: *tol*, *ntf*, *nmax*, *pcoeff* (if one is a string "n" it is posed equal to the default value; if *ntf* is the string "a" it is posed equal to the total number of functions in the database); the 5–th argument *range* should be 0 or greater:

   - IF *range>0* the inverse functions are computed on a set of equispaced points belonging to the interval [a,b] with step size "step". In this case:
     6–th argument: *a*,
     7–th argument: *b*,
     8–th argument: *step*

   - IF *range=0* the inverse functions are computed on a given set of values. In this case:
     6–th argument: *dim*, is the number of evaluation points
     7–th until to (6+dim)–th argument: *the evaluation points*.

3.6.1 Return value

Return value: 0 if the program runs without any problem

4 DEMO 2

4.1 Purpose

In the directory `DEMO2-Function_To_Give` users find files to build a demo–main program driving tests of RELIADIFF routine on a provided Laplace Transform with its Inverse provided too.

4.2 Description

This main program:

- obtains the needed parameter from the user
- sets the *szero* parameter as needed by the function

  – *szero* should be 1 if the Transform has a singularity at zero

- calls RELIADIFF routine
- compares the result with the known inverse value
- prints the output in a txt file named `output_demo2.txt`

4.3 Content of directory

The directory contains 4 files and a directory:

- **TEST_FunctionToGive.c**: main program
- **FunctionForTest.c**: containing the definition of the function to invert and of the true inverse to compare to (the user should modify this function in this file as well as he needs).

To write the function for test, see the proper section in this guide.

- **Makefile**: compiling the demo with a GNU Scientific Library (GSL) installed as root in the default path.
  - The command `make` compiles the c file linking the libraries, to obtain the executable file `test_functiontogive`
  - The command `make clean` deletes executable file, the txt files in the directory and the created libraries
- **Makefile_gslloc**: compiling the demo with a GNU Scientific Library (GSL) installed locally.
  - The command `make f Makefile_gslloc` compiles the c file linking the libraries, to obtain the executable file file `test_functiontogive`
- **scripts**: 6 shell scripts to run the program with different kinds of input.
  - **TEST_default.sh**: a shell script running the program with no argument
  - **TEST_givenpoints.sh**: a shell script running the program providing explicitly some evaluation points
  - **TEST_somedefaultvalues.sh**: a shell script running the program with some default values in arguments

4.4 How to write the function to test

User should write the Transform, named **fzE**, and its Inverse function, named **gzE**, in the file `TEST_FunctionForTest.c`, replacing those given as example.
fzE must require in input:

- *z:* (`TADIFF`) double precision: the evaluation point of the Transform

and returns:

> *The C/C++ function implementing F(s) shall take in input a T<double> value [2] and return a T<double> value, that will contain the required function value and related Taylor coefficients, calculated by Taylor arithmetic.*
> *To obtain this, all the floating point variables inside the function shall be of T<double> type (not the constants).*

- (`TADIFF`) double precision: the evaluation of the Transform at *z:*

gzE must require in input:

- t: double precision: the evaluation point of the Inverse function

and returns:

- double precision: the evaluation of the Inverse function at t

4.5 How to compile

4.5.1 GSL already installed in /usr/local/

By default, the GSL is installed in

- `/usr/local/bin`,
- `/usr/local/man`,
- `/usr/local/lib`,
- `/usr/local/include`,
- ...

That is in `/usr/local/`. So, if you previously installed the GSL without changing path:

1. Enter the directory `demos-Linux_g++/DEMO2-Function_To_Give`
2. Type the command `make`

4.5.2 GSL already installed in a different path

When installing GSL user can specify an installation prefix other than '`/usr/local`' by giving '`configure`' the option '`--prefix=PATH`'. So, if you previously installed the GSL changing path:

1. Find the path of the GSL: `/yourpath/` (example: `/usr/`).
2. Enter the directory `demos-Linux_g++/DEMO2{Function_To_Give`.
3. Open the file Makefile.
4. Edit only the path for the *gsllib* variable (as shown in the figure 2).
5. Type the command `make`.

```
In file Makefile (if GSL is already installed in /yourpath/):
...
## CHECK
############################################################################################
### GSL building directory: change it if you know a different one in your environment ###
ifndef gsllib

export gsllib = /yourpath/
endif
############################################################################################
...
```

Figure 2: In red the path to change in the Makefile.

4.5.3 GSL not installed

1. Install the GSL locally (in the `C++_files/demos-Linux_g++/GSL` directory), that is
   (a) Enter the `demos-Linux_g++` directory
   (b) Type the command `make -f Makefile_gsl`
2. Enter the directory `demos-Linux_g++/DEMO2-Function_To_Give`
3. Type the command `make -f Makefile_gslloc`

4.6 How to execute

After compiling, with the makefile, in this directory there will be an executable file named `test_functiontogive`.
To execute it, user should only provide it the arguments he/she wants, as described at the previous point:

```
./test_functiontogive [ordered list of arguments]
```

or he/she may use one of the provided scripts, described at the beginning of this section.

4.7 Arguments

The user may execute `test_functiontogive` with 7 kinds of INPUT:

1. **none argument**
2. **one argument**
3. **two arguments**
4. **three arguments**
5. **four arguments**

6. **five arguments**

7. **more than seven arguments**

Where:

1. **none argument**: The driver uses all default values:

tol= 10^{-3}	required accuracy
sigma0= 1	convergence abscissa for the Transform
nmax= 2000	maximum number of series coefficients
pcoeff= 0	do not print coefficients
szero= 0	the Laplace transform has not a singularity at zero
x= 1, 5, 10, 15	evaluation points

2. **1 argument**: *tol* (if it is a string "n" it is posed equal to the default value).
   The driver will use other default values:
   sigma0= 1
   nmax= 2000
   pcoeff= 0
   szero= 0
   x= 1, 5, 10, 15

3. **2 arguments**: *tol*, *sigma0* (if one is a string "n" it is posed equal to the default value).
   The driver will use other default values:
   nmax=2000
   pcoeff= 0
   szero= 0
   x= 1, 5, 10, 15

4. **3 arguments**: *tol*, *sigma0*, *nmax* (if one is a string "n" it is posed equal to the default value).
   The driver will use other default values:
   pcoeff= 0
   szero= 0
   x= 1, 5, 10, 15

5. **4 arguments**: *tol*, *sigma0*, *nmax*, *pcoeff* (if one is a string "n" it is posed equal to the default value).
   The driver will use other default values:
   szero= 0
   x= 1, 5, 10, 15

6. **5 arguments**: *tol*, *sigma0*, *nmax*, *pcoeff*, *szero* (if one is a string "n" it is posed equal to the default value).
   The driver will use other default values:
   x= 1, 5, 10, 15

7. > 8 **arguments**: *tol*, *sigma0*, *nmax*, *pcoeff*, *szero* (if one is the string "n" it is posed equal to the default value); the 6–th argument *range*, should be 0 or greater.

   - IF *range>0* the inverse functions are computed on a set of equispaced points belonging to the interval [a,b] with step size "step". In this case:
     7–th argument: *a*,

8–th argument: *b*,
9–th argument: *step*

- IF *range=0* the inverse functions are computed on a given set of values. In this case:
  7–th argument: *dim*, is the number of evaluation points
  8–th until to (7+dim)–th argument: *the evaluation points*.

4.7.1 Return value

Return value: 0 if the program runs without any problem

5 Appendix

In this section we will describe two other directories in the package.

5.1 Utility

5.1.1 Purpose

In this directory users find some utility tools.

5.1.2 Description

There are some function computing

- The integer power of a real number (in the `TADIFF` definition)
- The Laguerre polynomial
- The Hermite polynomial

There are also two routines writing on a file the explanation of the role of the *flag*, *Ncalc* and *Nopt* variables returned by RELIADIFF.

5.1.3 Content

The directory `C++_files/demos-Linux_g++/utility` contains 4 files which are:

- **Util.c** containing a library of functions needed by the functions of the database and useful to write a function to test the RELIADIFF routine. There are also routines writing the on a file the explanation of the role of the *flag*, *Ncalc* and *Nopt* variables returned by RELIADIFF.
- **Util.h** containing some header inclusions, and the prototypes of the functions defined in `Util.c`. It includes RELIADIFF.h and the GSL headers
- Two makefiles :
  - **Makefile** compiling the database library with GSL installed in the default path.
    - The command `make` creates the library `libutil.a`
    - The command `make clean` deletes the `libutil.a` file
  - **Makefile_gslloc** compiling the database library with a locally installed GSL.
    - The command `make f Makefile_gslloc` creates the library `libutil.a`

If user compiles the demos he doesn't need to compile directly the utilities.

5.1.4 Specification of Utility functions

There are 5 functions:

Integer power of a TADIFF–double precision variable

```
T<double> pow_int_T(T<double> a, int n)
```

`a`:	(TADIFF) double precision: the base
`n`:	integer: the exponent
return value:	(TADIFF) double precision: the n–power of a

Computation of the Laguerre Polynomial in `x` of degree `K`

```
double Laguerre(int K, double x)
```

`K`:	integer: degree of the Hermite polynomial to calculate
`x`:	double precision: point of evaluation
return value:	double precision: Hermite polynomial at `x`

Computation of the Hermite Polynomial in `x` of degree `K`

```
double Hermite(int K, double x)
```

`K`:	integer: degree of the Hermite polynomial to calculate
`x`:	double precision: point of evaluation
return value:	double precision: Hermite polynomial at `x`

Printing in file the diagnostics parameters interpretation: flag

```
void print_flags_file(FILE *fp)
```

`fp`:	file handler: handler of the file where to write the flag meaning in RELIADIFF

Printing in file the diagnostics parameters interpretation: ncalc and nopt

```
void print_N_file(FILE *fp, int nmax)
```

`fp`:	file handler: handler of the file where to write Nopt/Ncalc meaning in RELIADIFF
`nmax`:	integer: required maximum number of Taylor coefficients to calculate in RELIADIFF

5.2 DEMO1–Function_From_DATABASE/Database

5.2.1 Purpose

In this directory, users find a library of Laplace Transforms and of the related Inverse functions. The database contains 89 functions.

5.2.2 Content

The directory database contains 4 files which are:

- **dbLaplace.c**: containing the Laplace Transforms
- **dbInvLaplace.c**: containing the Inverse Laplace Transforms
- **dbL.h**: containing the prototypes of functions defined in `dbLaplace.c` and `dbInvLaplace.c`. It includes `Util.h`.
- Two makefiles :
  - **Makefile** compiling the database library with GSL installed in the default path.
    - The command `make` creates the library `libdatabase.a`
    - The command `make clean` deletes the `libdatabase.a` file
  - **Makefile_gslloc** compiling the database library with a locally installed GSL.
    - The command `make f Makefile_gslloc` creates the library `libdatabase.a`

> *If user compiles the demos he/she doesn't need to compile directly the database.*

5.2.3 Specification of Database functions

Each Transform function is of the kind

```
T<double> fzXX(T<double> z)
```

where XX is the number of the function in the database.

Each Inverse function is of the kind

```
double gzXX(double)
```

where XX is the number of the function in the database.

> *View the **Database_Function_list.pdf** file in the Doc directory to know about functions in the database and their numbers.*

The C/C++ function implementing F(s) shall take in input a T<double> value [2] and return a T<double> value, that will contain the required function value and related Taylor coefficients, calculated by Taylor arithmetic.
To obtain this, all the floating point variables inside the function shall be of T<double> type (not the constants).

References

[1] A. Murli, L. D'Amore, V. Mele, R. Campagna, *ReLIADiff. A C++ Software Package For Real Laplace transform Inversion based on Algorithmic Differentiation*, ACM Transactions on Mathematical Software, 0, 0, Article 0 (0000), 20 pages.

[2] C. Bendtsen, O. Stauning, *Tadiff, a flexible C++ package for Automatic Differentiation using Taylor series expansion*, technical report IMM-REP-1997-07, Department of Mathematical Modelling, Technical University of Denmark, 2800 Lyngby, Denmark, 1997.