Documentation of ReLaTIve [1].

LUISA D'AMORE
ROSANNA CAMPAGNA
VALERIA MELE
University of Naples Federico II

ALMERICO MURLI
CMCC - Lecce and SPACI

MARIAROSARIA RIZZARDI
University of Naples Parthenope

Contents

Chapter 1

Introduction

The Package contains, in addition to this documentation, the README.txt file and three subfolders:

- `SRC` with source codes of the ReLaTIve software. Users who want use ReLaTIve in their own software really only need this directory that contains the routine to call.
- `Sample_Main` with a sample main to call the software (`SampleMain.c`),
- `ARTICLE_TESTS` with the driver used by authors for testing the software on functions of a provided database (results in [1]).

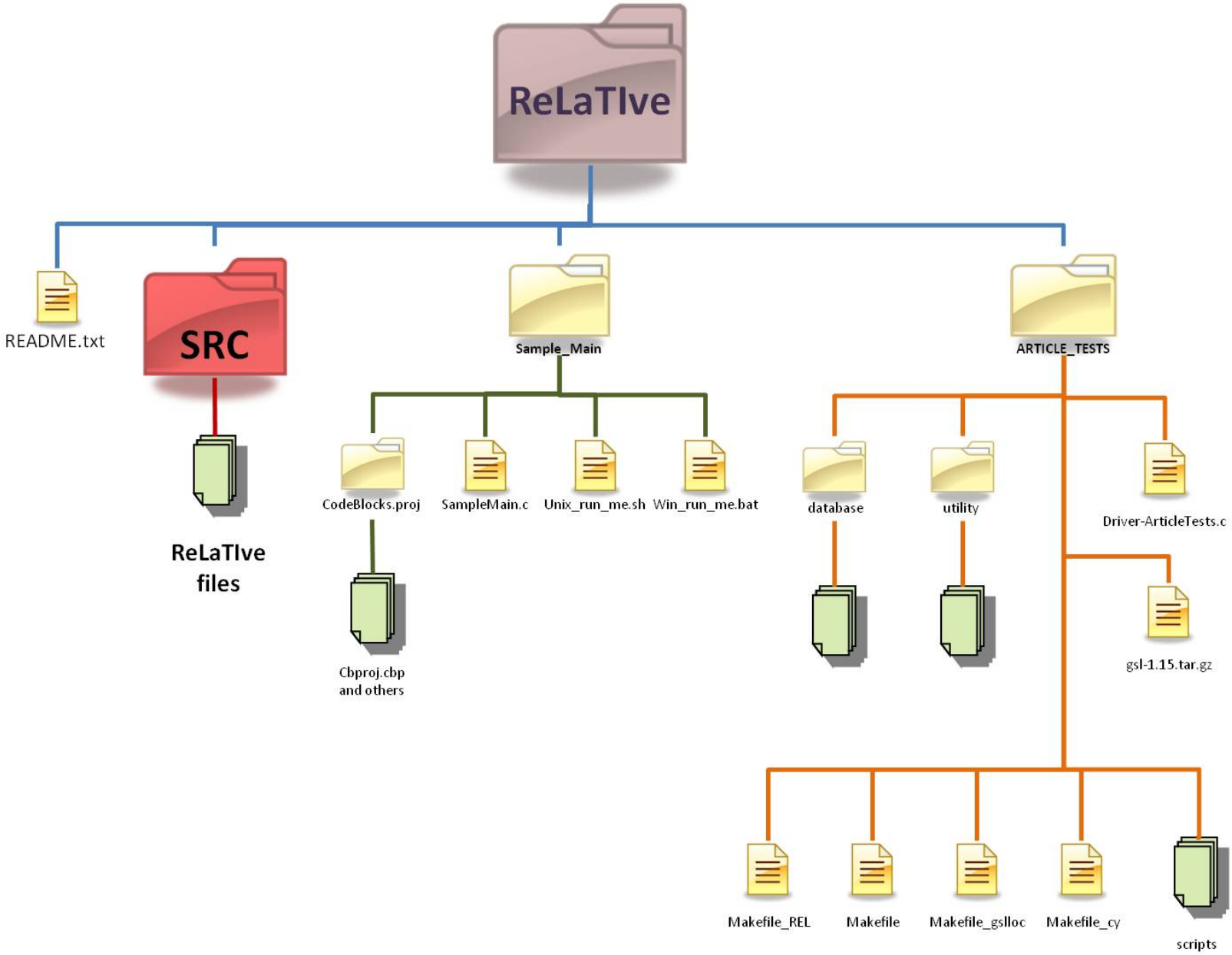

Figure 1.1: Relative Package.

Chapter 2

SRC: ReLaTIve

ReLaTIve is a fully automatic software to compute the inverse (f(x)) of a Laplace transform function (F(z)) computable on the real axis with a limited precision. It is written in ANSI C.
The mathematical background and general information about its performance are described in [1].

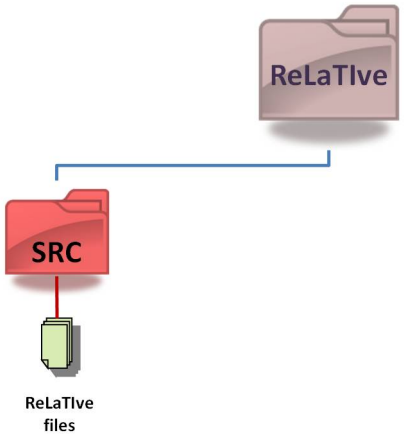

Figure 2.1: `SRC` folder. Users who want use ReLaTIve in their own software really need only this directory that contains the routine to call.

2.1 Content

The folder contains 3 files:

ReLaTIve.c: the routine.

ck_calc.c: containing the routines needed to compute the Taylor coefficients.

ReLaTIve.h: containing some header inclusions (stdio, math,...), and the prototypes of the functions. Any application using the ReLaTIve tool needs to include this file.

2.2 Routine specifications

F function Laplace Transform

- such that it can be expressed as $F(z) = z^{-1}G(z)$, where G is analytic at infinity,
- without a singularity at zero neither a singularity at infinity,
- with abscissa of convergence $sigma0 \in \Re$,
- such that it may be evaluated on the real axis with a pre-assigned limited precision, at most equals to the machine precision;

f function Inverse Laplace Transform

- infinitely differentiable for all $x > 0$.

Besides the function to invert F, the user has to provide:

- the value of $x \geq 0$ where the inverse function f has to be computed,
  - x should be relatively small, if $x > 14$ other methods could work better,
- (optional) the value of $sigma0 \in \Re$ (or an approximated value of it), the abscissa of convergence of F,
- (optional) a tolerance tol to the accuracy on $f(x)$,
- (optional) the maximum number of series coefficients,
- (optional) the singularity at infinity of F.

The software provides

- the inverse Laplace function f evaluated at a single point,
- the best number of Laguerre series coefficients to use,
- the number of Laguerre series coefficients calculated,
- the estimate of the absolute error on f,
- the estimate of the relative error on f,
- some diagnostics parameters.

2.3 Parameters details

The calling sequence of ReLaTIve is the following.

```
int ReLaTIve (double x,
              /* where the inverse function f has to be computed */
              double (*fz)(double),
              /* the function to invert, F */
              int sinf
              /* theres or not a singularity at infinity for F */
              double *sigma0,
              /* the value of sigma0 */
              int sflag
              /* default value of sigma0 or not */
              double *tol,
              /* a tolerance to the accuracy on f(x) */
              int *nmax,
              /* the maximum number of series coefficients */
              int *nopt,
              /* the best number of series coefficients to use */
              int *ncalc,
              /* the number of series coefficients calculated */
              double *absesterr,
              /* the estimate of the absolute error on f */
              double *relesterr,
              /* the estimate of the relative error on f */
              int *flag,
              /* some diagnostics parameters */
              double *ILf
              /* the inverse f evaluated at a single point */
             )
```

2.3.1 Input parameters

x: double precision array

On entry it contains value(s) at which the Inverse Laplace function is required. Each component of x has to be greater or equal to zero.

fz: double precision function pointer

On entry it contains the name of the Laplace Transform function.

sinf: integer

On entry it says if the Transform has a singularity at infinity.

sflag: integer

On entry it means if user is giving the abscissa of convergence of F.

2.3.2 Input/Output parameters

sigma0: double precision

On entry

- if $sflag > 0$, it contains the abscissa of convergence of F:

  - if it is less than zero, it will be posed to zero,
- if $sflag = 0$, it will be ignored and posed to the default value.

tol: double precision
On entry it contains the required accuracy on f.

- if $tol < 0$, it will be posed to the default value,
- if $tol = 0$, it will be posed to the default value,
- if $tol > 1$, it will be posed to the default value,
- if $tol < 10^{-7}$, it will be $tol = 10^{-}7$.

nmax: integer
On entry it contains the maximum number of Laguerre series terms.
If $nmax < 8$, it is posed at a default value ($nmax = 40$).

2.3.3 Output parameters

nopt: integer
On output it contains the optimal number of Laguerre series terms.

ncalc: integer
On output it contains the max number of terms calculated to find nopt.

absesterr: double precision
On output it contains the estimate of the absolute error on f.

relesterr: double precision
On output it contains the estimate of the relative error on f.

flag: integer
On output it contains a diagnostic parameter.

ILf: double precision:
On output it contains the computed value of the inverse function at x.

return value: integer. It contains a diagnostic on the input data.
If it is equal to 1, it means that the routine termined without any output because the input x was less than zero.

2.3.4 Error indicators and warnings

flag: integer
On output it contains a diagnostic about the behaviour of ReLaTIve.

- $flag = 0$: absolute error $< tol$ and relative error $< tol$
  Both the absolute and the relative error estimates are smaller than tol. This occurs if tol is greater than maximum attainable accuracy.

- $flag = 1$: both absolute error and relative error minimum attainable.
  Within nmax terms of the series expansion, the algorithm cannot satisfy the user's required accuracy. Instead, the algorithm provides numerical result within the maximum attainable accuracy. The maximum attainable accuracy is computed as the minimum value of estabserr (and estrelerr). Nopt is the number of terms at which such minimum is reached. This occurs if the series seems to converge too slowly or to diverge. User is invited to verify if the test function satisfies algorithm's requirements: overall, user should verify the convergence abscissa and the singularity at infinity given as input.
- $flag = 2$: absolute error $< tol$.
  Only the absolute error estimate is smaller than the user's required accuracy tol. This occurs if tol is greater than the maximum attainable accuracy and if the inverse function rapidly decreases towards zero.
- $flag = 3$: relative error $< tol$.
  Only the relative error estimate is smaller than the user's required accuracy tol. This occurs if tol is greater than the maximum attainable accuracy and the inverse function rapidly increases.

return value: integer

On output it contains a diagnostic on the input data.

- =1: $x < 0$, the run stopped without working.
- =0: ReLaTIve worked properly.

2.3.5 Relation between nopt and ncalc

ncalc is the maximum number of terms of the Laguerre series expansion calculated by the algorithm.

nopt is the number of terms of the Laguerre series expansion that gives the numerical result within the maximum attainable accuracy, less or equal to *nmax* (see section 2.3.2).

It can be:

- $nopt = ncalc < nmax$
  The computed value of the inverse Laplace function agrees with the true one within $log(tol)$ significant and decimal digits. This value is obtained calculating *nopt* terms of the Laguerre series expansion. It should be $flag = 0$
- $nopt < ncalc = nmax$
  Within *nmax* terms of the Laguerre series expansion, the algorithm cannot satisfy the user's required accuracy. Instead, the algorithm provides numerical result within the maximum attainable accuracy, and *nopt* is the number of terms at which such maximum is reached. This occurs if the series seems to diverge.
- $nopt = ncalc = nmax$
  This occurs if one of following conditions happens:

– within $nmax$ terms of the Laguerre series expansion, the algorithm cannot satisfy the user's required accuracy, but the series seems to converge, even if very slowly: the algorithm provides numerical result within the maximum attainable accuracy, reached within $nmax$ terms of the Laguerre series expansion.
Or:

– within $nmax$ terms of the Laguerre series expansion, the algorithm cannot satisfy the user's required accuracy and the series seems to diverge. Instead, the algorithm provides numerical result within the maximum attainable accuracy, and $nopt$ is the number of terms at which such maximum is reached that accidentally corresponds with $nmax$.
Or:

– within $nmax$ terms of the Laguerre expansion, the algorithm can satisfy the user's required accuracy, so the series seems to converge, even if quite slowly: the algorithm provides numerical result within the maximum attainable accuracy, reached within $nmax$ terms of the Laguerre series expansion. It should be $flag = 0$.

2.4 How to compile ReLaTIve

To compile ReLaTIve and any software using ReLaTIve **you need a gcc compiler** (see chapter 3 to know how to compile a main program calling ReLaTIve). If you do not have a gcc compiler:

- On a **Linux system** we suggest to install GCC
(GNU Compiler Collection,`http://gcc.gnu.org/install/index.html`).

- On a **Windows system** we suggest to install Mingw/GCC
(`http://www.mingw.org/wiki/InstallationHOWTOforMinGW`).
REMARK: After installing it, you have to tell the command line interpreter where to find them. This is usually accomplished by adding the appropriate directory names to the PATH variable in your user environment. The installers will not do this for you. So, the steps are [3]:

  1. Right-click on your "My Computer" icon and select "Properties".
  2. Click on the "Advanced" tab, then on the "Environment Variables" button.
  3. You should be presented with a dialog box with two text boxes. The top box shows your user settings. The PATH entry in this box is the one you want to modify. Note that the bottom text box allows you to change the system PATH variable. You should not alter the system path variable in any manner, or you will cause all sorts of problems for you and your computer.
  4. Click on the PATH entry in the TOP box (or create it), then click on the "Edit" button
  5. Scroll to the end of the string and at the end add
  `;<mingw-installation-directory>\bin`
  REMARK: Substitute `<mingw-installation-directory>` with the absolute path name of the installation target directory you chose
  (ie `C:\MinGW`, or `C:\Program File\CodeBlocks\MinGW` if you installed it with Code::Blocks as we suggest in section 3.3.3);
  6. press OK and you are done (see figure 2.2).

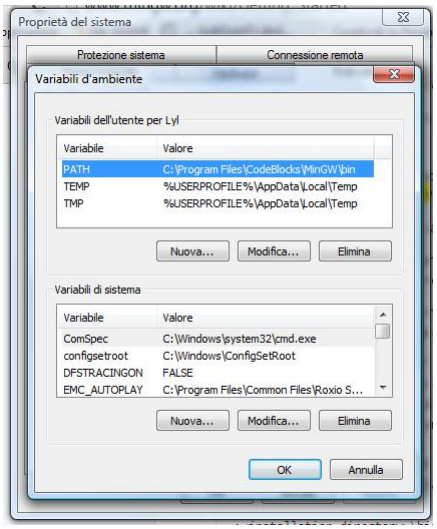

Figure 2.2: Set the PATH entry as `;<mingw-installation-directory>\bin` (in this case `C:\Program File \CodeBlocks\MinGW\bin`).

- On **both systems**, you can install Code::Blocks (see section 3.3.3).

Chapter 3

Sample_Main:
The Sample Main Program

An example of calling program for ReLaTIve is provided.

3.1 Content

The `Sample_Main` folder consists in:

- a sample main to call the software (`SampleMain.c`), that is a simple example of calling program for the ReLaTIve routine,
- a shell-script (`Unix_run_me.sh`) to build and run the executable under Unix,
- a batch file (`Win_run_me.bat`) to build and run the executable under Windows,
- a folder (`CodeBlocks_proj`) with files for a project that user can open with the IDE Code::Blocks, to build and run the executable under both Unix and Windows systems.

3.2 Program Specifications

The program `SampleMain.c`:

1. defines a Transform function
2. sets input
3. calls ReLaTIve
4. prints the calculated Inverse value(s) and all the ReLaTIve outputs

To compile it user needs a gcc compiler.
If you do not have a gcc compiler:

- On a **Linux system** we suggest to install GCC
(GNU Compiler Collection,`http://gcc.gnu.org/install/index.html`).

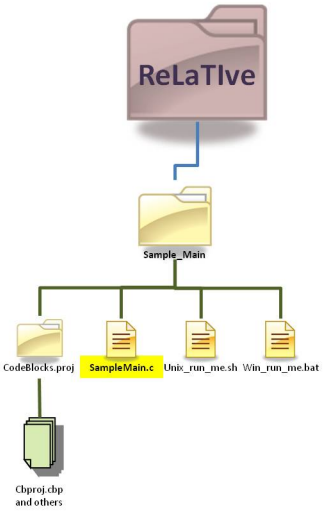

Figure 3.1: `Sample_Main` folder.

- On a **Windows system** we suggest to install Mingw/GCC (`http://www.mingw.org/wiki/InstallationHOWTOforMinGW`).
  REMARK: After installing it, you have to tell the command line interpreter where to find them. This is usually accomplished by adding the appropriate directory names to the PATH variable in your user environment. The installers will not do this for you. So, the steps are [3]:

  1. Right-click on your "My Computer" icon and select "Properties".
  2. Click on the "Advanced" tab, then on the "Environment Variables" button.
  3. You should be presented with a dialog box with two text boxes. The top box shows your user settings. The PATH entry in this box is the one you want to modify. Note that the bottom text box allows you to change the system PATH variable. You should not alter the system path variable in any manner, or you will cause all sorts of problems for you and your computer.
  4. Click on the PATH entry in the TOP box (or create it), then click on the "Edit" button
  5. Scroll to the end of the string and at the end add
     `;<mingw-installation-directory>\bin`
     REMARK: Substitute `<mingw-installation-directory>` with the absolute path name of the installation target directory you chose
     (ie `C:\MinGW`, or `C:\Program File\CodeBlocks\MinGW` if you installed it with Code::Blocks as we suggest in section 3.3.3);
  6. press OK and you are done (see figure 2.2).

- On **both systems**, you can install Code::Blocks (see section 3.3.3).

3.2.1 The program

The example is a simple calling program to invert the function

$$F(z) = \frac{1}{z^4 - a^4} \qquad \text{with } a = \frac{3}{5} \tag{3.1}$$

without giving any optional input and printing results at screen. It is listed below.

```
#include "../SRC/ReLaTIve.h"

double fzTest(double z){
        return 1./(pow(z,4)-pow(3./5.,4));
}

int main(){
  /*ReLaTIve INPUT*/
    double *x=NULL; /*Inverse Function evaluation point(s)*/
    double (*fz)(double); /*Function F Pointer*/
    int sinf=0; /*optional: there's or not a singularity at infinity for F*/
    int  sflag=1; /*optional: user will provide abscissa of convergence*/

  /*ReLaTIve INPUT-OUTPUT*/
    double sigma0=0.6; /*optional: abscissa of convergence of F */
    double  tol=0; /*optional: tolerance on accuracy */
    int  nmax=0; /*optional: maximum number of series coefficients*/

  /*ReLaTIve OUTPUT*/
    int  nopt;
    int  ncalc;
    double *absesterr=NULL; /*absolute error estimate*/
    double *relesterr=NULL; /*relative error estimate*/
    double  ILf; /*Inverse Function f computed */
    int  *flag=NULL; /*diagnostics on the result */
    int  ierr=0; /*diagnostics on the software work */

  /*AUXILIARY VARIABLES*/
    int  dim=0;
    int  i;
    double a,b,step;
    int  start=0;

    /* ---------------------------------- x values*/
    a=1.;
    b=15.;
    step=0.5;
    dim=(int)(((b-a)/step)+1);
    x = (double*)calloc(dim,sizeof(double));
    if ( x == NULL ){
            fprintf(stderr, "\nERROR: DYNAMIC ALLOCATION FAILED.\n");
            exit(1);
```

```
  }
  for (i=0; i<dim; i++)
      x[i] = a + i*step;
/* ------------------------------------------------*/

/* ------------------- memory space allocations*/
  flag=(int*)calloc(dim,sizeof(int));
  if ( flag == NULL ){
          fprintf(stderr, "\nERROR: DYNAMIC ALLOCATION FAILED.\n");
          exit(1);
  }
  relesterr=(double*)calloc(dim,sizeof(double));
  if ( relesterr == NULL ){
          fprintf(stderr, "\nERROR: DYNAMIC ALLOCATION FAILED.\n");
          exit(1);
  }
  absesterr=(double*)calloc(dim,sizeof(double));
  if ( absesterr == NULL ){
          fprintf(stderr, "\nERROR: DYNAMIC ALLOCATION FAILED.\n");
          exit(1);
  }
/* ------------------------------------------------*/

  fz = fzTest;
  for(i=0; i<dim; ++i){
      /******************** CALLING ReLaTIve ***********************/
      ierr=ReLaTIve(
            x[i],
            fz,
            sinf,
            &sigma0,
            sflag,
            &tol,
            &nmax,
            &nopt,
            &ncalc,
            &absesterr[i],
            &relesterr[i],
            &flag[i],
            &ILf);
      /*****************************************************************/
      switch(ierr){
      case 1:
              printf("\nx=%f - ReLaTIve termined: x<0.0!\n It must be
                      x>=0\n\n",x[i]);
              break;
      default:
        if (!start){
          start=1;
          printf("Used Tolerance on accuracy: %e;\n",tol);
          printf("Used abscissa of convergence on F: %e;\n",atof(sigma0));
          printf("Used maximum number of Laguerre series coefficients: %d;
```

```
                \n",nmax);
            printf("\n TABLE\n");
            printf("-------------------------------------------\n");
            printf("| x | f_comp | estabserr | estrelerr | Nopt | Ncal |FLAG|\n");
            printf("-------------------------------------------\n");
          }
            printf("| %4.2e | %14.8e | %9.3e | %9.3e | %5d | %5d | %2d |\n",x[i], ILf, absesterr[i],
                   relesterr[i], nopt, ncalc, flag[i]);
            break;
        }/*endswitch*/
  /* UPDATE OF i (and x)*/
    }/*endfor on x*/

    free(flag);
    free(relesterr);
    free(absesterr);
    return 0;
}
/*END OF MAIN*/
```

3.2.1.1 Output of the Sample Main Program

Running the example program users will obtain at screen (and in a file) an output similar to the one in figure 3.2.

```
Used Tolerance on accuracy: 1.000000e-03;
Used abscissa of convergence on F: 6.000000e-01;
Used maximum number of Laguerre series coefficients: 40;

                              TABLE
-------------------------------------------------------------------------------
|     x     |      f_comp      |  estabserr |  estrelerr |  Nopt |  Ncal |FLAG|
-------------------------------------------------------------------------------
| 1.00e+000 | 1.66739873e-001 | 4.007e-004 | 2.403e-003 |    16 |    16 |  2 |
| 1.50e+000 | 5.63001415e-001 | 5.330e-004 | 9.466e-004 |    16 |    16 |  0 |
| 2.00e+000 | 1.33670963e+000 | 7.067e-004 | 5.287e-004 |    16 |    16 |  0 |
| 2.50e+000 | 2.61998482e+000 | 9.593e-004 | 3.662e-004 |    16 |    16 |  0 |
| 3.00e+000 | 4.55622252e+000 | 2.348e-004 | 5.154e-005 |    18 |    18 |  0 |
| 3.50e+000 | 7.31156335e+000 | 3.171e-004 | 4.338e-005 |    18 |    18 |  0 |
| 4.00e+000 | 1.10895737e+001 | 4.280e-004 | 3.859e-005 |    18 |    18 |  0 |
| 4.50e+000 | 1.61546029e+001 | 5.778e-004 | 3.577e-005 |    18 |    18 |  0 |
| 5.00e+000 | 2.28625625e+001 | 7.801e-004 | 3.412e-005 |    18 |    18 |  0 |
| 5.50e+000 | 3.17029288e+001 | 1.886e-004 | 5.948e-006 |    20 |    20 |  0 |
| 6.00e+000 | 4.33519261e+001 | 2.545e-004 | 5.871e-006 |    20 |    20 |  0 |
| 6.50e+000 | 5.87475595e+001 | 3.431e-004 | 5.841e-006 |    20 |    20 |  0 |
| 7.00e+000 | 7.91836712e+001 | 4.634e-004 | 5.852e-006 |    20 |    20 |  0 |
| 7.50e+000 | 1.06436759e+002 | 6.292e-004 | 5.911e-006 |    20 |    20 |  0 |
| 8.00e+000 | 1.42933903e+002 | 8.556e-004 | 5.986e-006 |    20 |    20 |  0 |
| 8.50e+000 | 1.91976136e+002 | 2.045e-004 | 1.065e-006 |    22 |    22 |  0 |
| 9.00e+000 | 2.58040912e+002 | 2.828e-004 | 1.096e-006 |    22 |    22 |  0 |
| 9.50e+000 | 3.47182082e+002 | 3.769e-004 | 1.086e-006 |    22 |    22 |  0 |
| 1.00e+001 | 4.67575256e+002 | 4.882e-004 | 1.044e-006 |    22 |    22 |  0 |
| 1.05e+001 | 6.30250435e+002 | 8.440e-004 | 1.339e-006 |    22 |    22 |  0 |
| 1.10e+001 | 8.50081857e+002 | 1.652e-004 | 1.944e-007 |    24 |    24 |  0 |
| 1.15e+001 | 1.14712589e+003 | 6.640e-004 | 5.788e-007 |    24 |    24 |  0 |
| 1.20e+001 | 1.54842898e+003 | 2.650e-004 | 1.711e-007 |    26 |    26 |  0 |
| 1.25e+001 | 2.09046975e+003 | 2.942e-004 | 1.407e-007 |    28 |    28 |  0 |
| 1.30e+001 | 2.82245915e+003 | 5.784e-004 | 2.049e-007 |    28 |    28 |  0 |
| 1.35e+001 | 3.81079611e+003 | 9.948e-004 | 2.610e-007 |    30 |    30 |  0 |
| 1.40e+001 | 5.14508934e+003 | 2.835e-003 | 5.510e-007 |    30 |    40 |  3 |
| 1.45e+001 | 6.94628015e+003 | 2.903e-003 | 4.179e-007 |    30 |    40 |  3 |
| 1.50e+001 | 9.37761474e+003 | 1.400e-002 | 1.493e-006 |    34 |    40 |  3 |
```

Figure 3.2: Output of the Sample Main Program.

3.3 How to compile and execute Sample Main

3.3.1 On a Linux system, with a gcc compiler installed

Open a shell prompt, enter the `Sample_Main` folder, then type and run the command
`./Unix_run_me.sh`

REMARK: set `Unix_run_me.sh` permission to "execute" (that is `chmod 755 Unix_run_me.sh`) before to run it.

3.3.2 On a Windows system, with a gcc compiler installed

Enter the `Sample_Main` folder and double click the file `Win_run_me.bat`

3.3.3 On both systems (by Code::Blocks)

3.3.3.1 Download and install Code::Blocks: Windows operating systems

User can find the tool at `http://www.codeblocks.org/` , more precisely the last binary release is at `http://www.codeblocks.org/downloads/26`.

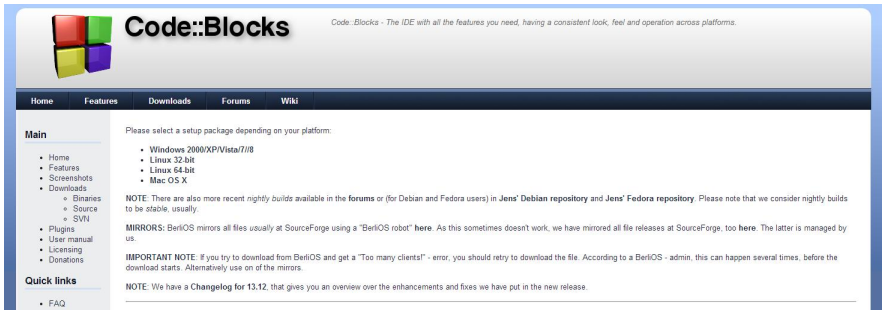

Figure 3.3: Open the CODE::BLOCKS download page and choose your operating system.

Here, choose the Windows platform and the suitable file to download.

The file to download depends on MinGW: if it is already on the system, choose the first file (as in figure 3.5), otherwise choose the second one (as in figure 3.4).
If unsure, use `codeblocks-13.12mingw-setup.exe` , that is the second one.

Then run the downloaded file and follow the setup wizard (that is click `Next` or `I Agree` till the button `Finish`).

Figure 3.4: If MinGW is not already istalled, choose the second file.

Figure 3.5: If MinGW is already istalled, choose the first file.

3.3.3.2 Download and install Code::Blocks: Linux (Ubuntu) operating systems

For a Linux system (Ubuntu like[1]) the right version of Code::Blocks can be found in the Software Center of the system. That is

1. search the Software Center and launch it (figure 3.6 and 3.7),

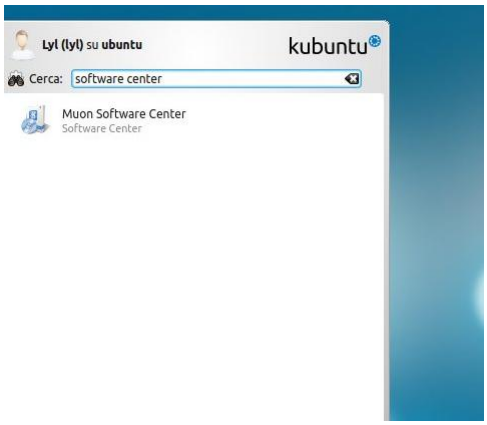

Figure 3.6: Search the Software Center.

2. here, search Code::Blocks, typing the name in the search bar (figure 3.8),

[1]For a different Linux distribution the procedure could be different, but in the most of cases it should be very similar to the one described here. For example, in case of Scientific Linux, users shall click on "Add/Remove software" between steps 1. and 2. For any other distribuition, they can follow instructions at `http://www.codeblocks.org/downloads/26`.

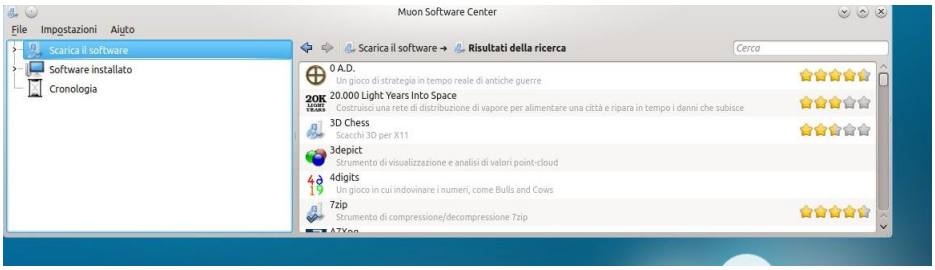

Figure 3.7: Launch the Software Center.

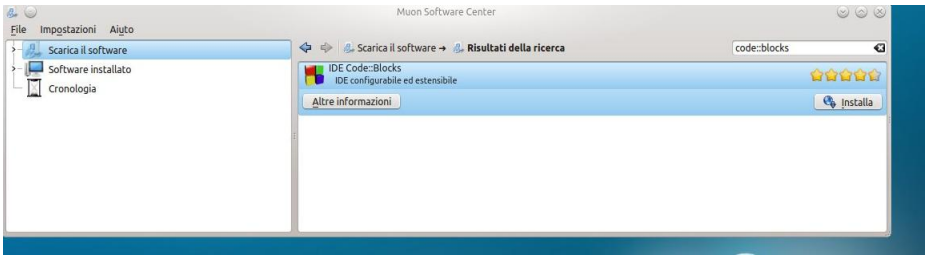

Figure 3.8: Search Code::Blocks.

3. then select it and click Install (eventually give the root password) (figure 3.9).

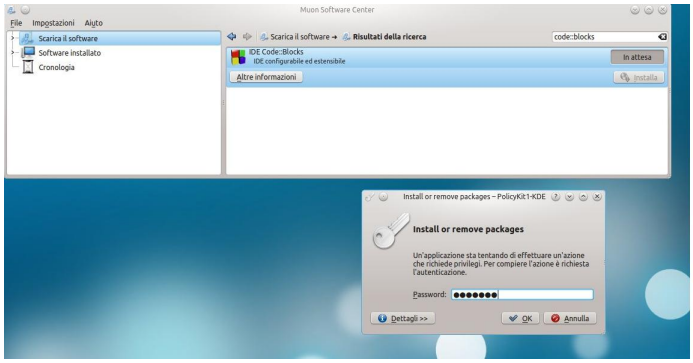

Figure 3.9: Select and click Install.

3.3.3.3 Open the sample main project in Code::Blocks

Run the Code::Blocks program.
The first time you run it, you should choose the default compiler and click “Ok” (figure 3.10).
Then you must choose if set C/C++ files associations with Code::Blocks (figure 3.11).
When the program is running, you can open an existing project. This is what you should do to use the Sample Main provided with ReLaTIve, that is

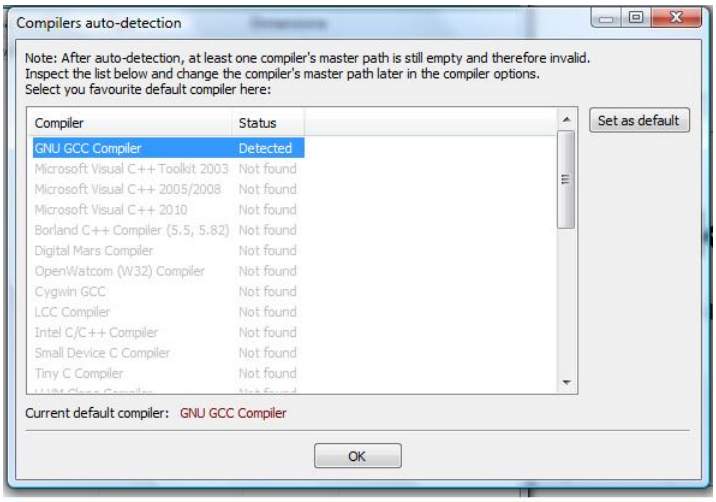

Figure 3.10: Set the compiler.

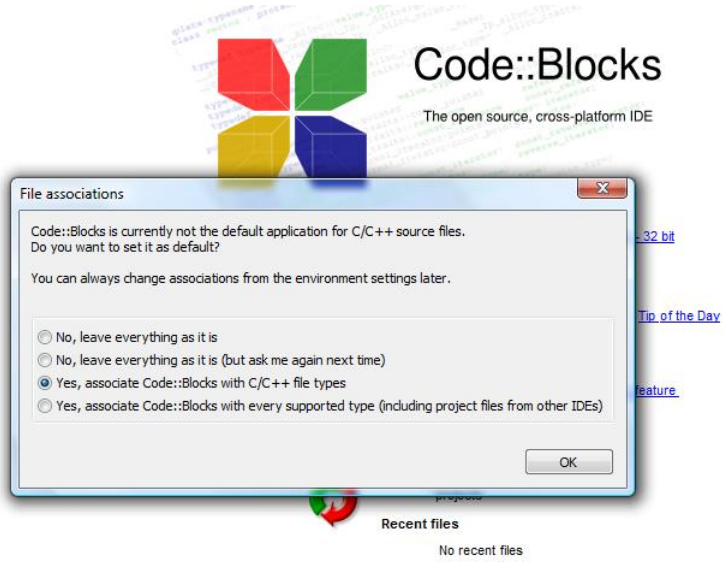

Figure 3.11: Choose the association or not.

- select the .cbp file that contain the project
  (i.e.: `ReLaTIve/Sample_Main/CodeBlocks_proj/CBproj.cbp`)
  (figure 3.13, 3.14 and 3.15),
- click the build and run button, so you will see the results window (figure 3.16 and 3.17).

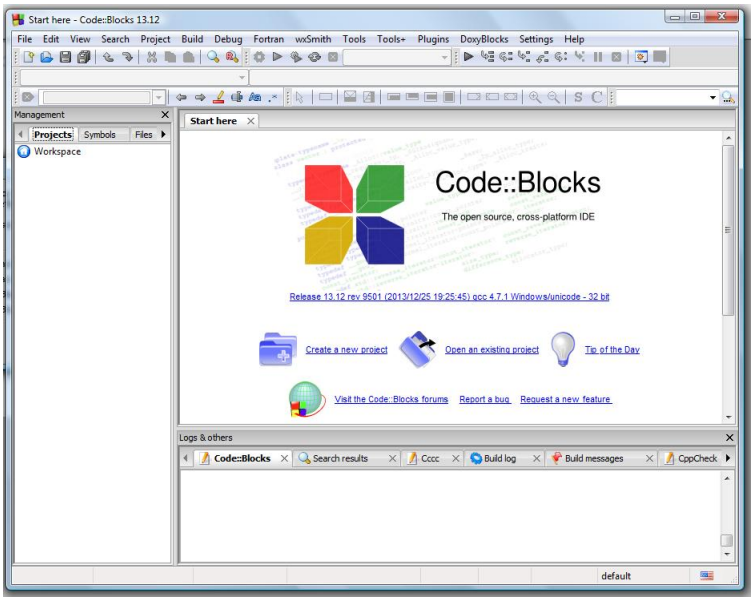

Figure 3.12: Code::Blocks is running.

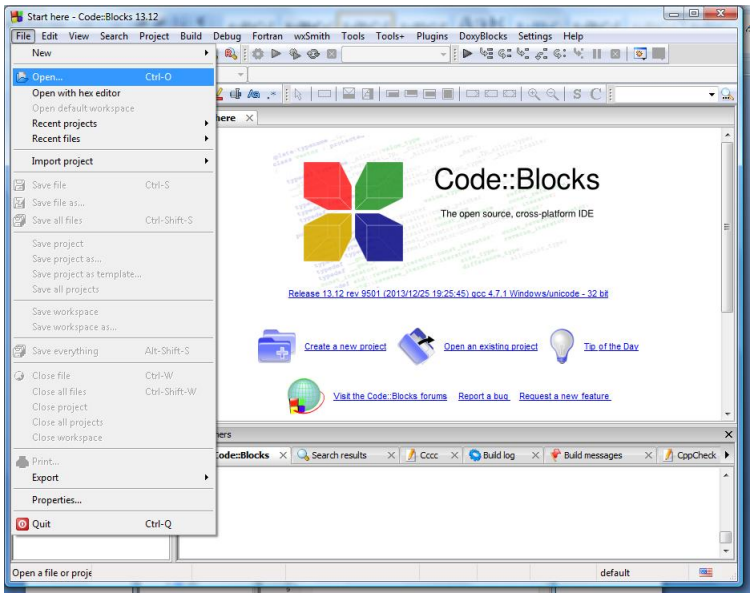

Figure 3.13: Open a file.

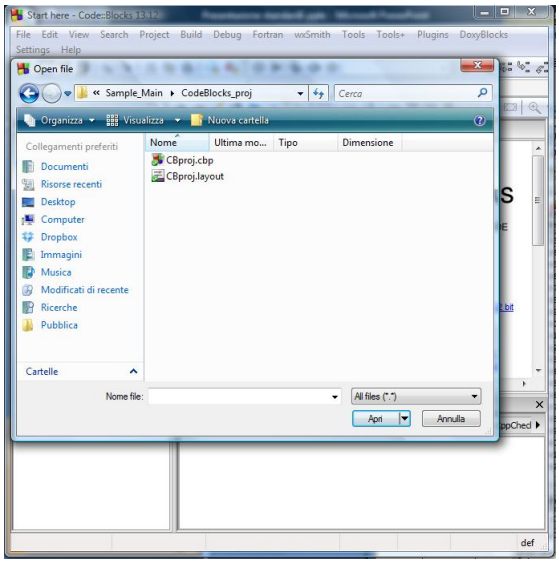

Figure 3.14: Choose the project `CBproj.cbp`.

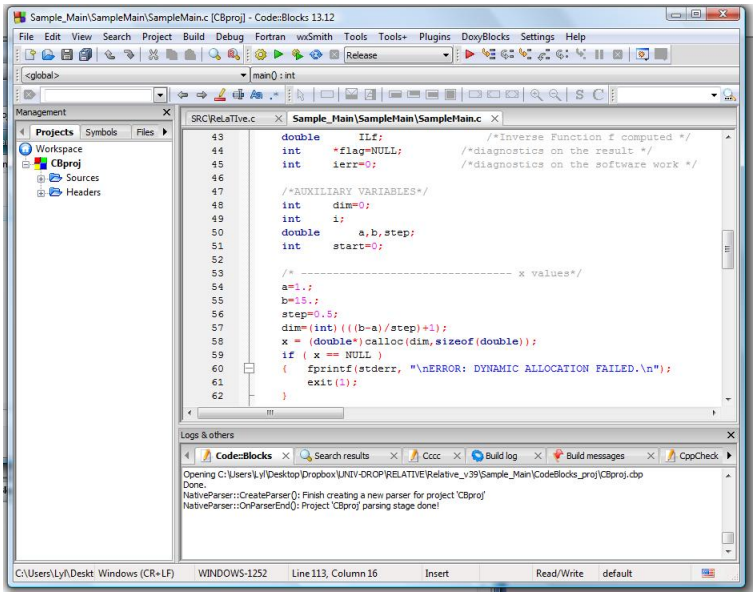

Figure 3.15: The project is open.

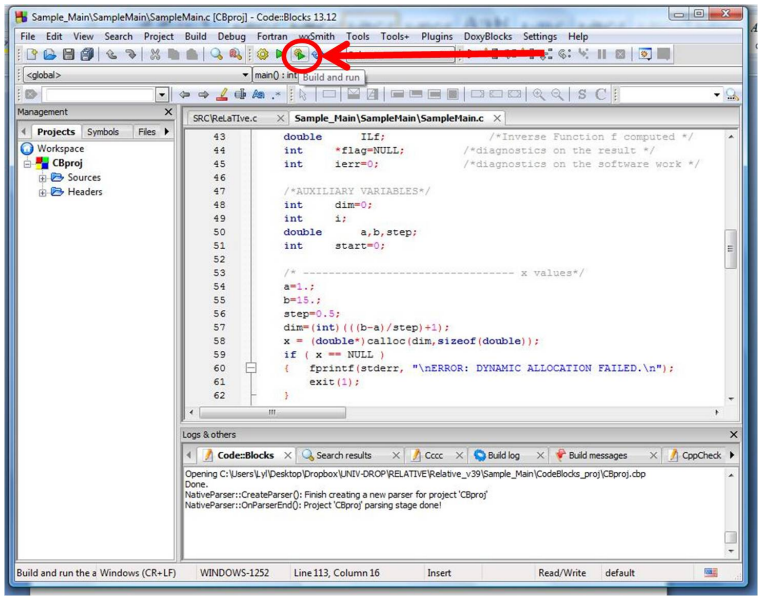

Figure 3.16: Build and run.

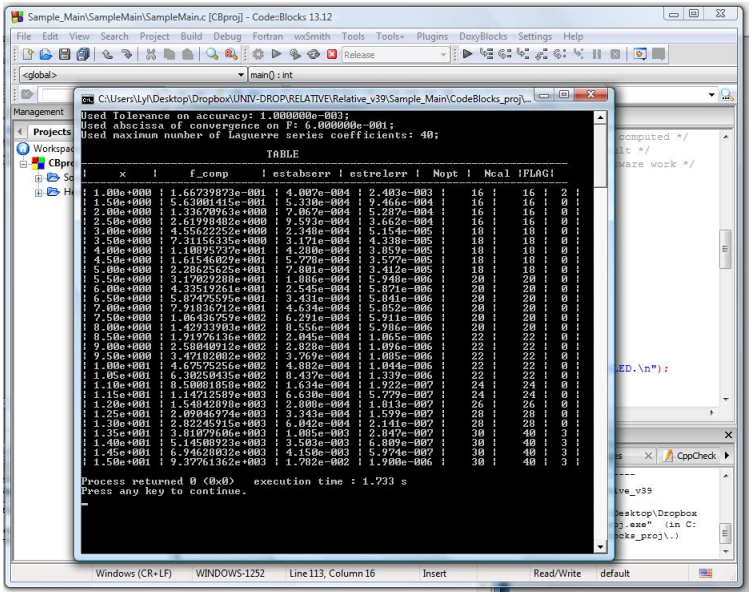

Figure 3.17: Program output.

Chapter 4

ARTICLE_TESTS: Accompanying paper tests

4.1 Content

The driver is in the ARTICLE_TESTS folder that contains:

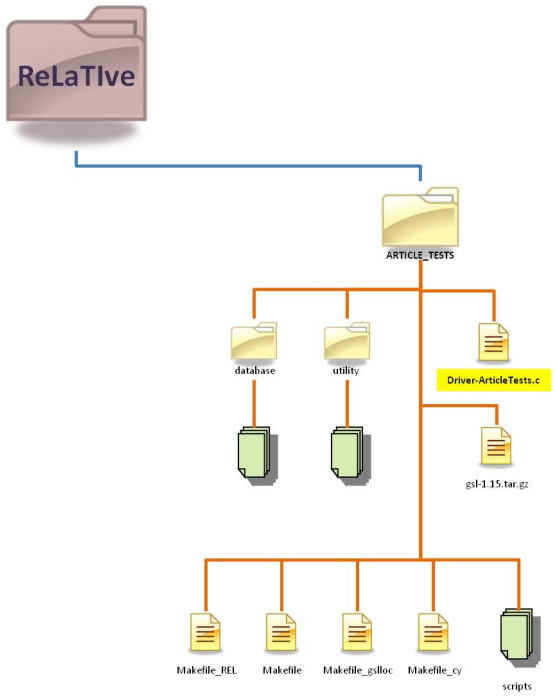

Figure 4.1: ARTICLE_TESTS folder.

- the GSL package

  gsl-1.15.tar.gz: a compressed file containing the installation files for the GNU Scientific Library (GSL), that user could need to install.

- two subfolders:

  `database`**:** files to generate a library of 49 functions that authors used to test the Laplace inversion automatic tool ReLaTIve (see Appendix to know about the functions and section 4.2.1.1 to know details about the folder).

  `utility`**:** files containing utility routines needed to execute functions in the database and the driver (see ection 4.2.1.2 to know details about the folder).

- the driver:

  Driver-ArticleTests.c: the main program.

- four makefiles:

  Makefile: to compile the driver with the GNU Scientific Library (GSL) installed as root in the default[1] path (`/usr/local/`).

  - The command `make` compile the c file linking all the needed libraries, to obtain the executable file Driver,
  - The command `make clean` deletes executable file, all the txt files in the directory and the created libraries.

  Makefile_REL: to create a library to link to when compiling an application calling ReLaTIve.

  - The command `make -f Makefile_REL` creates the library librelative.a
  - The command `make -f Makefile_REL` clean deletes the libreative.a file

  Makefile_gslloc: to compile the driver with a GNU Scientific Library (GSL) installed locally.

  - The command `make f Makefile_gslloc` install the GSL, and compile the c file linking all the needed libraries, to obtain the executable file Driver,
  - The command `make cleangsl -f Makefile_gslloc` deletes the GSL locally installed,
  - The command `make clean -f Makefil_gslloc` executable file, all the txt files in the directory and the created libraries.

  Makefile_cy: to compile the driver on Cygwin (Windows systems).

  - The command `make -f Makefile_cy` compile the c file linking all the needed libraries, to obtain the executable file Driver,
  - The command `make clean -f Makefile_cy` deletes executable file, all the txt files in the directory and the created libraries.

- three shell scripts to show how to give arguments:

  TESTONE_givenPoints.sh: a shell script to run the program on a single function with all default parameters but explicitly giving some evaluation points (user shall explicitly choose a function at runtime).

[1] see the INSTALL file in the provided GSL package.

TESTONE_tol-3.sh: a shell script to run the program on a single function providing the required accuracy $tol = 10^{-3}$ and giving a range for the evaluation points (user shall explicitly choose a function at runtime).

TESTALL_tol-4.sh: a shell script to run the program with $tol = 10^{-4}$, on all the functions in the provided database and giving a range for the evaluation points.

4.2 Description

Authors include in the package the driver they used to test the ReLaTIve routine. It was written using some timing functions that need to be compiled and executed on a Linux system with gcc compiler installed.
This driver tests the ReLaTIve behaviour on 49 functions defined in an included database[2], that was written using the GNU Scientific Library (GSL) [8], free software under the GNU General Public License, available for Linux systems [3].
Then, to compile and execute the test driver, user needs

- a Linux system with a gcc compiler and GSL installed (see section 4.4.1), or
- a Windows system with Cygwin[4] [4] installed (see sections 4.4.2.1 and 4.4.2).
  NOTE: the used timing C function (`gettimeofday()`) on Windows doesn't reach the microsecond resolution that it has on Linux systems, so the execution times returned by the driver can be a different by those reported in the paper.

User can give different arguments (as shown in the provided shell scripts) to test in many way the method on the function of the database.
The driver will generate a file with results for each function.
`Driver-ArticleTests.c` program:

1. obtains all the needed parameter from the user
2. sets the abscissa of convergence depending on the function chosen by the user
3. sets the sinf parameter as needed by the function chosen by the user
   - sinf=1 if the Transform has a singularity at infinity
4. calls ReLaTIve routine
5. compares the result with the known inverse value
6. 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.

[2] see Appendix.
[3] If GSL is not in the system, user can install it locally using the provided package
[4] Cygwin is a large collection of GNU and other tools which provide Linux distribution-like functionality to Windows. It currently works with all recent, commercially released x86 32 bit and 64 bit versions of Windows, starting with Windows XP SP3. For more information visit `http://cygwin.com/`.

4.2.1 Subfolders details

User does not need to know details about the subfolders `utility` and `database`. Anyway, here authors give a brief description of them.

4.2.1.1 `database` folder

Purpose: Library of Laplace Transforms and of the related Inverse functions. The database contains 49 functions.

Content: The subdirectory `database` of ReLaTIve package contains 5 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`.

Makefile to compile 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 to compile the database library with the locally installed GSL.

- The command `make -f Makefile_gslloc` creates the library `libdatabase.a`.

Makefile_cy to compile the database library with Cygwin on Windows systems.

- The command `make -f Makefile_cy` creates the library `libdatabase.a`.
- The command `make clean -f Makefile_cy` deletes the `libdatabase.a` file.

Specifications and parameters of functions: View the Appendix to know details about the functions.

4.2.1.2 `utility` folder

Purpose: In this directory users can find some utility tools, that is some functions to compute

- The Laguerre polynomial.
- The Hermite polynomial.

and two routines to write on a file the explanation of the role of the $flag$, $ncalc$ and $nopt$ variables returned by ReLaTIve.

Content: The directory `utility` contains five files which are:

Util.c containing functions needed by the functions of the database and routines needed to print in a file the role of the flag, ncalc and nopt variables returned by ReLaTIve.

Util.h containing some header inclusions, and the prototypes of the functions defined in `Util.c`.
REMARK: It includes ReLaTIve.h, the GSL headers and the header for timing.

Makefile to compile the utility 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 to compile the utility library with the locally installed GSL.

- The command `make -f Makefile_gslloc` creates the library `libutil.a`.

Makefile_cy to compile the utility library with Cygwin on Windows systems.

- The command `make -f Makefile_cy` creates the library `libutil.a`.
- The command `make clean -f Makefile_cy` deletes the `libutil.a` file.

Specifications and parameters of functions: The functions are:

double Laguerre(int K, double x)

K: integer: degree of the Laguerre polynomial to calculate.

x: double: point of evaluation.

return value: double: Laguerre polynomial value in x.

double Hermite(int K, double x)

k: integer: degree of the Hermite polynomial to calculate.

x: double: point of evaluation.

return value: double: Hermite polynomial value in x.

void print_flags_file(FILE *fp)

fp: file handler: handler of the file where to write the $flag$ explanation.

print_N_file(FILE *fp, int nmax)

fp: file handler: handler of the file where to write the *nopt/ncalc* relation.

nmax: integer: maximum number of Taylor coefficients to calculate.

4.3 Specifications and parameters

The user can execute Driver with 5 kinds of INPUT:

NONE ARGUMENT: in this case the software uses the following default values:

tol depends on the function (default tolerance).

ntf = 1 (default number of test function).

nmax = 40 (default maximum number of series cofficients).

x = 1, 5, 10, 15 (the points at which the inverse function is evaluated).

ONE ARGUMENT: in this case the software uses the following values:

tol given by user.
If user provides the value n as first argument, the software will use the default value.

ntf = 1 (default number of test function).

nmax = 40 (default maximum number of series cofficients).

x = 1, 5, 10, 15 (the points at which the inverse function is evaluated).

TWO ARGUMENTS : in this case the software uses the following values:

tol given by user.
If user provides the value n as first argument, the software will use the default value.

ntf given by user.
If user provides the value a as second argument, the software will use the default value.

nmax = 40 (default maximum number of series cofficients).

x = 1, 5, 10, 15 (the points at which the inverse function is evaluated).

THREE ARGUMENTS: in this case the software uses the following values:

tol given by user.
If user provides the value n as first argument, the software will use the default value.

ntf given by user.
If user provides the value a as second argument, the software will use the default value.

nmax given by user.
If user provides the value n as second argument, the software will use the default value.

x = 1, 5, 10, 15 (the points at which the inverse function is evaluated).

MORE THAN SIX ARGUMENTS: in this case the software uses the following values:

tol given by user.
If user provides the value n as first argument, the software will use the default value.

ntf given by user.
If user provides the value a as second argument, the software will use the default value.

nmax given by user.
If user provides the value n as second argument, the software will use the default value.

range that can be greater or equal to zero. That is:

- if $range > 0$ user will give other three arguments:
  a : given by user,
  b : given by user,
  step : given by user.

Inverse function will be computed at uniformly distributed points on [a,b] with step size *step*.

- if $range = 0$ user will give other arguments:
  dim : the number of points, given by user,
  x_1,...,x_{dim} : a list of dim points, given by user.
  Inverse function will be computed at dim points x_1,...,x_{dim}, directly given by user.

The driver will give:

return value: 0 if the program runs without any problem.

4.4 How to compile the driver

4.4.1 Linux

Before compiling the driver, user must check if and where GNU Scientific Library (GSL) is installed in the system. Than he has three options to compile the driver:

1. If GSL was installed as root in the default[5] path (/usr/local/), user will:
   - Open a shell prompt,
   - Enter the `ARTICLE_TEST` folder,
   - Type the command `make`.
2. If GSL was installed in a different path, user will:
   - Check what are the directories containing GSL include and libraries files.
     WARNING: the absolute installation path of GSL must be without any space!
   - Open the `ARTICLE_TEST/Makefile` file,
   - Change GSL include and libraries directories paths, that is type your paths for enviroment variables `gsldirinc` and `gsldirlib`.
     WARNING: the absolute paths for `gsldirinc` and `gsldirlib` must be without any space!
   - Save and close `Makefile`,
   - Open the `ARTICLE_TEST/utility/Makefile` file,
   - Change GSL include and libraries directories paths, that is type your paths for enviroment variables `gsldirinc` and `gsldirlib`,
     WARNING: the absolute paths for `gsldirinc` and `gsldirlib` must be without any space!
   - Save and close `Makefile`,
   - Open the `ARTICLE_TEST/database/Makefile` file,

[5] see the INSTALL file in the provided GSL package.

- Change GSL include and libraries directories paths, that is type your paths for enviroment variables `gsldirinc` and `gsldirlib`,
  WARNING: the absolute paths for `gsldirinc` and `gsldirlib` must be without any space!
- Save and close `Makefile`,
- Open a shell prompt,
- Enter the `ARTICLE_TEST` folder,
- Type the command `make`.

3. If GSL is not installed (or the driver was already compiled in this way), user will

   - Open a shell prompt,
   - Enter the `ARTICLE_TEST` folder,
   - Type the command `make -f Makefile_gslloc`. The first time this will install the GSL in the directory `ARTICLE_TEST/GSL`: this could take several minutes.

4.4.2 Windows

4.4.2.1 Cygwin installation

Windows systems users need to install Cygwin[6] to compile and execute the tests driver. Cygwin installation could take some hours to complete.
To obtain Cygwin setup program, visit `http://cygwin.com/`. (see figure 4.2).

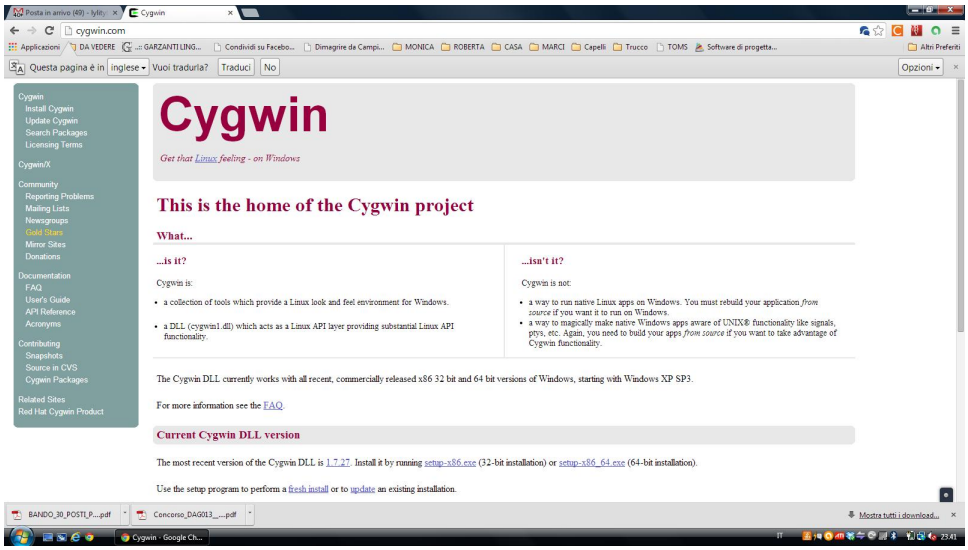

Figure 4.2: `http://cygwin.com/` web page.

To use Cygwin the steps are:

[6]Cygwin is a large collection of GNU and other tools which provide Linux distribution-like functionality to Windows. It currently works with all recent, commercially released x86 32 bit and 64 bit versions of Windows, starting with Windows XP SP3. For more information visit `http://cygwin.com/`

1. Download and run the `setup-x86.exe` (or `setup-x86-x64.exe`) file (see figure 4.3).
2. Choose *Install from internet* and click Next (see figure 4.4).
3. Choose the path where to install Cygwin (we suggest `C:\cygwin`) and click Next (see figure 4.5).
4. Select a directory where you want to store installation files (we suggest to create a new folder called `C:\CygwinLocalPackage`), then click Next (see figure 4.6 and 4.7).
5. Choose a site from the list and click Next (see figure 4.8).
6. Download will start (see figure 4.9).
7. Setup program will ask what packages to install (see figure 4.10).
8. Write *gcc* in the search box, then click on *Default* for each item (it will become *Install*) (see figure 4.11 and 4.12).
9. Write *gsl* in the search box, then click on *Default* for each item (it will become *Install*) (see figure 4.13 and 4.14).
10. Write *make* in the search box, then open the *Debug* item and click on *Skip* for the following item:
    - make-debuginfo.
11. Then open the *Devel* item and click on *Skip* for the following item:
    - make.
12. Click Next (see figure 4.15).
13. Click Next again to satisfy dependencies (see figure 4.16).
14. Download and installation will start (see figure 4.17, 4.18 and 4.19).
    NOTE: it can take some hours to complete!
15. Choose the icons to create and click End (see figure 4.20).
16. To launch Cygwin, click on the Cygwin icon (on the desktop or in the Start Menu) (see figure 4.21 and 4.22). If you choose the suggested installation path, your home will be in `C:\cygwin\home\nameofyourpc`.
17. Move (or copy) the ReLaTIve package in the Cygwin installation folder (for example in `C:\cygwin`, or in your home `C:\cygwin\home\nameofyourpc`).

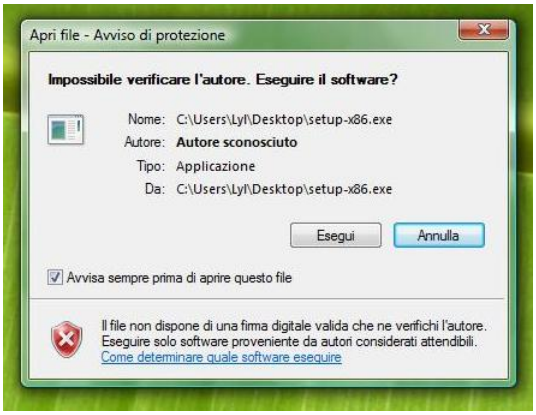

Figure 4.3: Run `setup-x86.exe` (or `setup-x64.exe`) file.

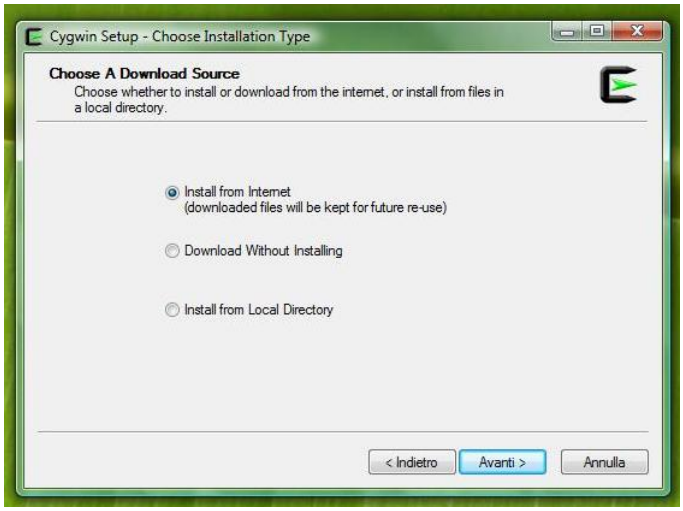

Figure 4.4: Choose *Install from internet* and click Next.

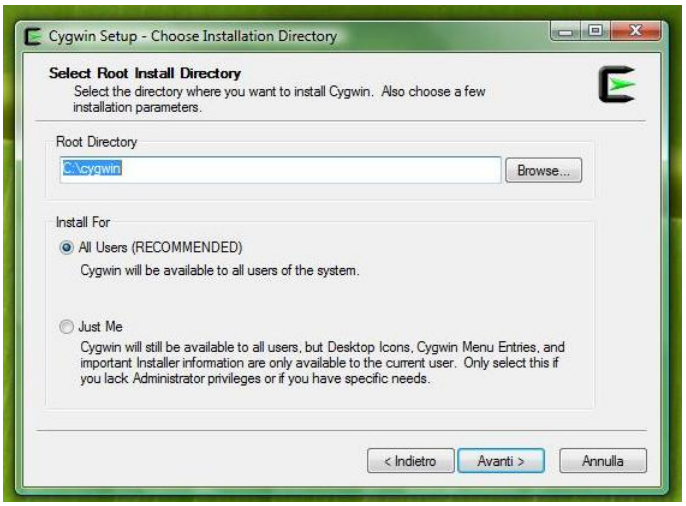

Figure 4.5: Choose the path where to install Cygwin (we suggest `C:\cygwin`) and click Next.

Figure 4.6: Choose the path where to install Cygwin (create a new folder called `C:\CygwinLocalPackage`).

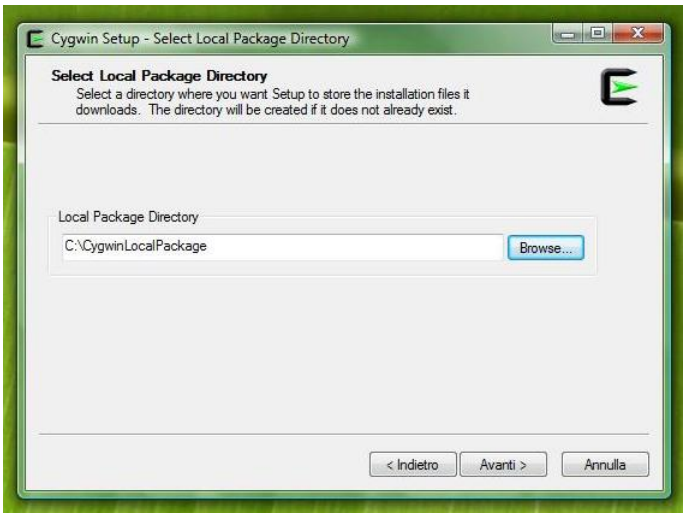

Figure 4.7: Click Next.

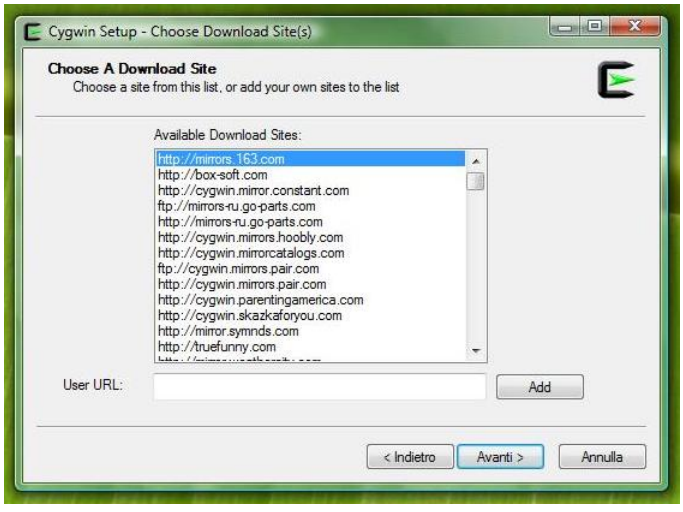

Figure 4.8: Choose a site from the list and click Next.

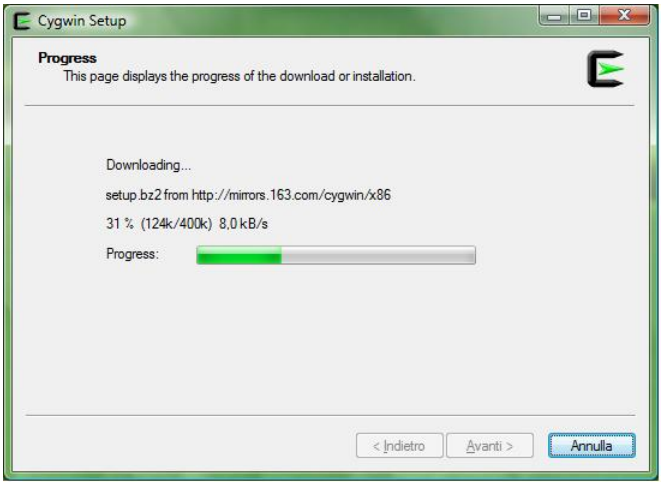

Figure 4.9: Starting download.

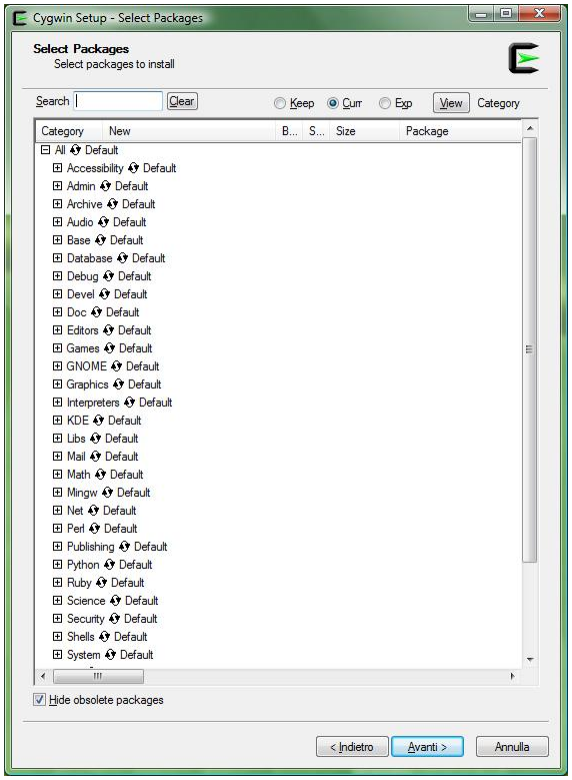

Figure 4.10: Setup program asks what packages to install.

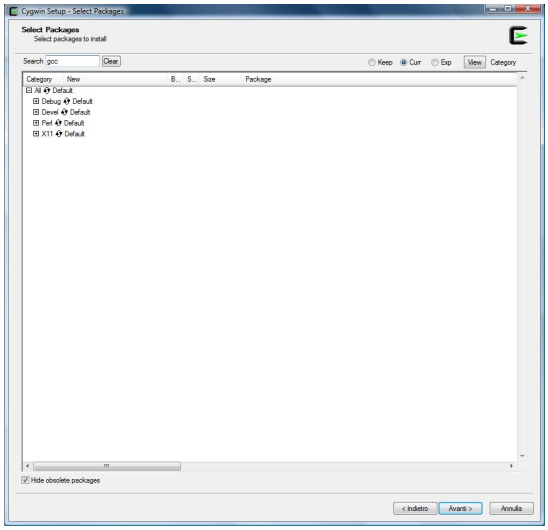

Figure 4.11: Write *gcc* in the search box, then click on *Default* for each item.

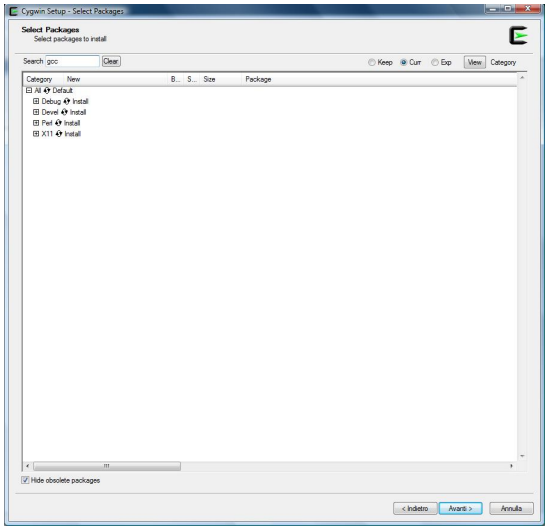

Figure 4.12: *Default* will become *Install*.

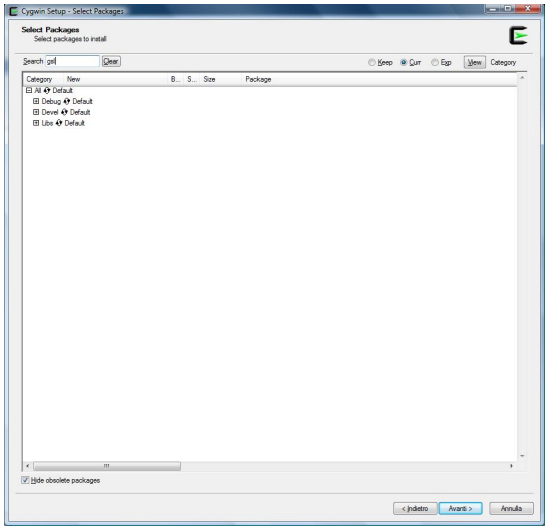

Figure 4.13: Write *gsl* in the search box, then click on *Default* for each item.

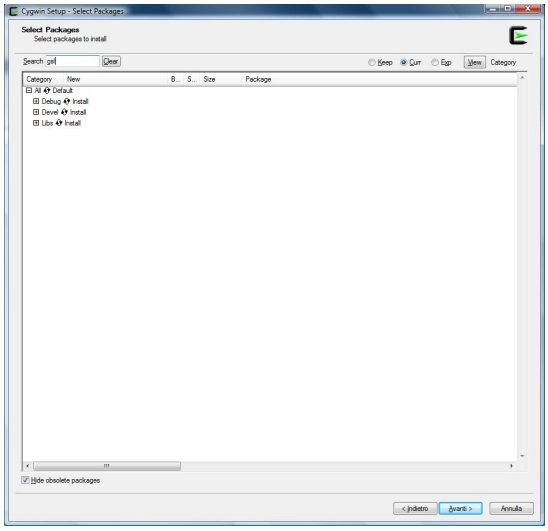

Figure 4.14: *Default* will become *Install*.

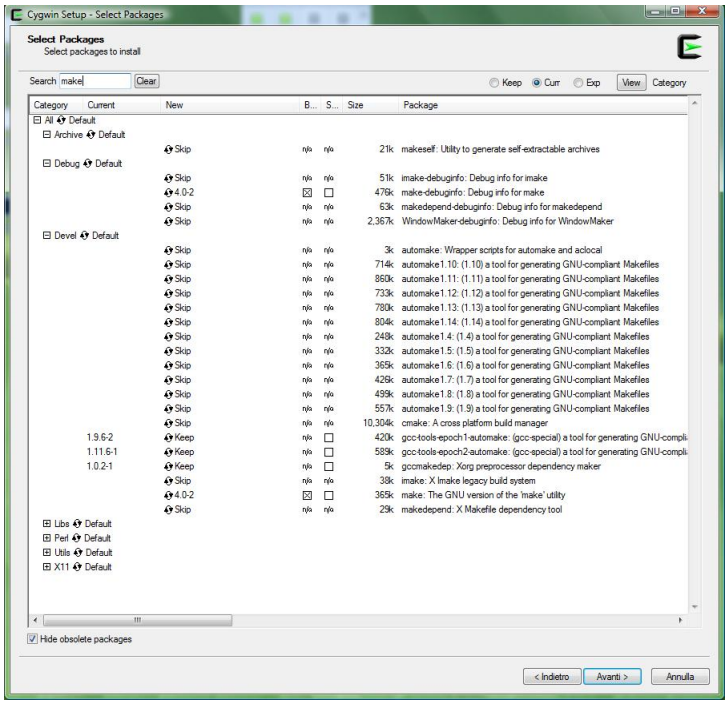

Figure 4.15: Write *make* in the search box, then click on *Skip* for two items (`make-debuginfo` and `make`). Then click Next.

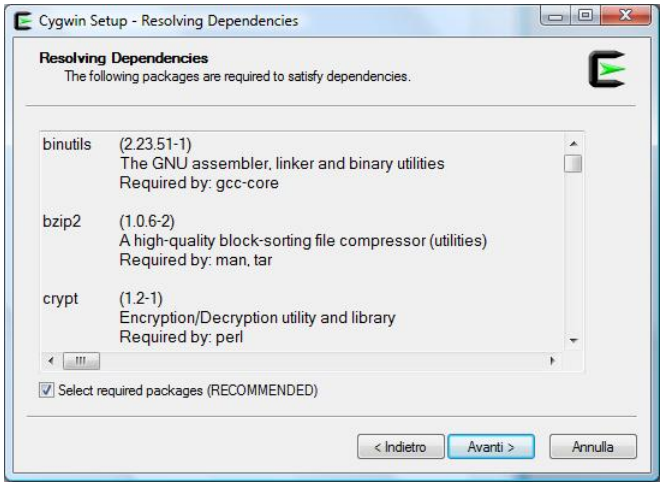

Figure 4.16: Click Next again to satisfy dependencies.

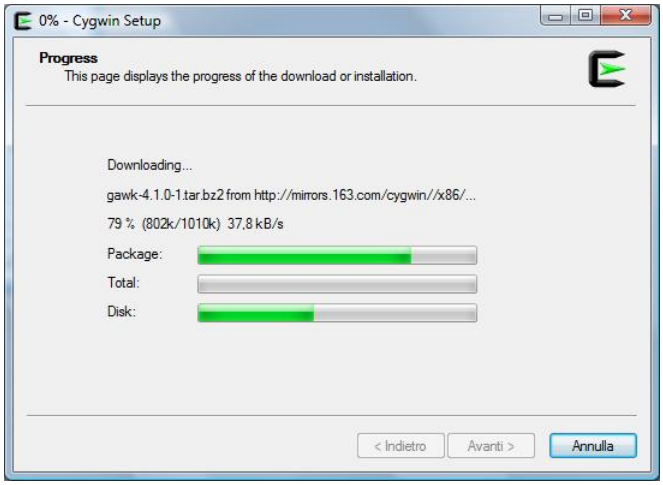

Figure 4.17: Downloading.

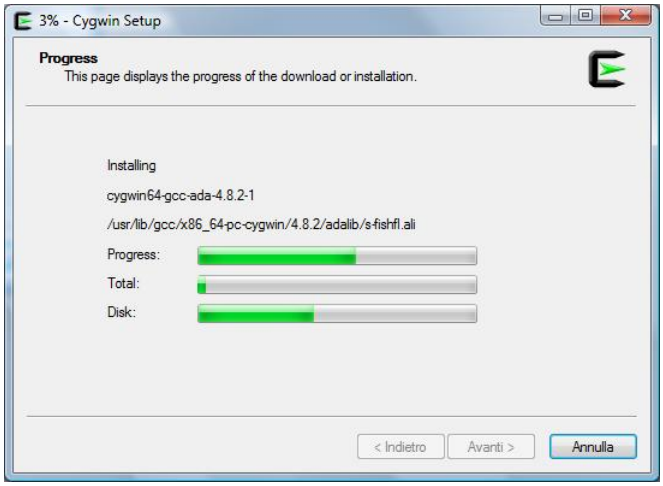

Figure 4.18: Installing.

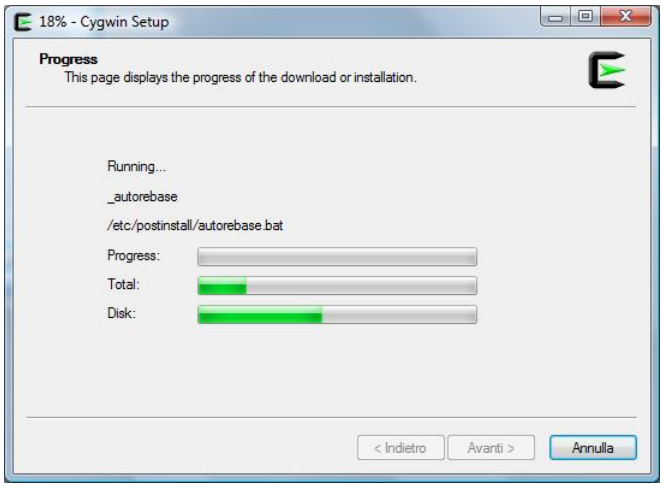

Figure 4.19: Running.

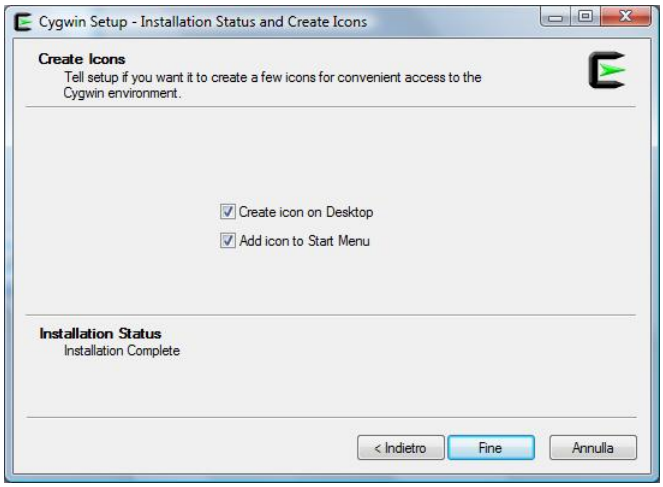

Figure 4.20: Choose the icons to create and click End.

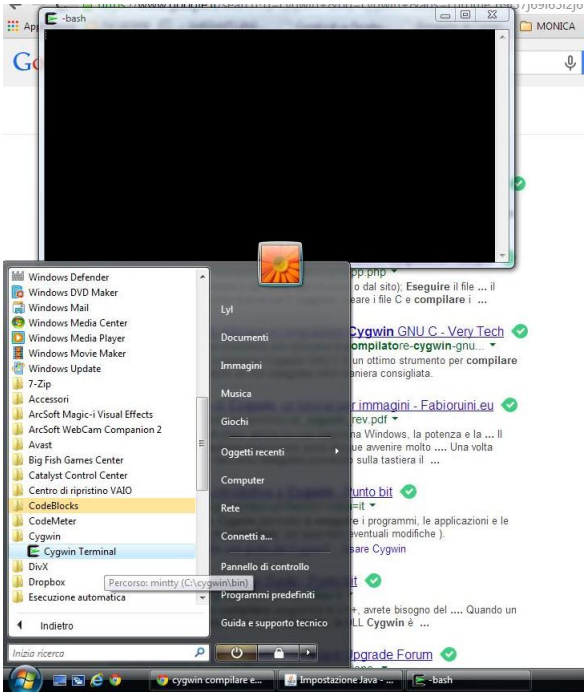

Figure 4.21: Click on the Cygwin icon in the Start Menu.

4.4.2.2 Compile

The steps are:

- Run Cygwin and use it as a Linux shell prompt,
- Enter the `ARTICLE_TEST` folder,

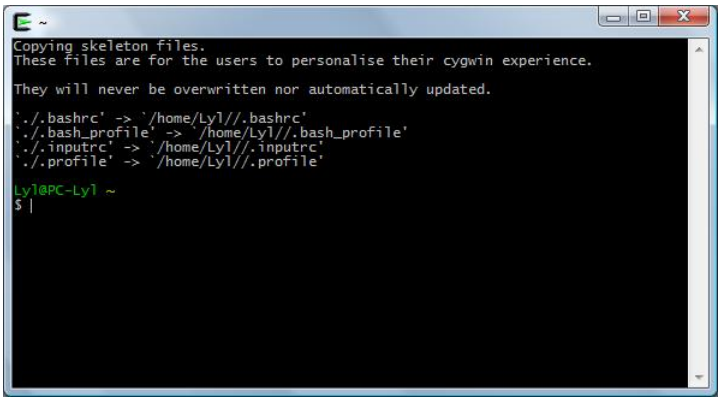

Figure 4.22: Cygwin starts for the first time.

- Type the command `make -f Makefile_cy`.

4.5 How to execute the driver

After compiling (both in Lunix and in Cygwin), in `ARTICLE_TESTS` folder there will be an executable file named *Driver* (or Driver.exe). To execute it, you should only provide it the arguments[7] you want, as described at the next point:

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

If you want to use it with all default parameters, you can just give the command

```
./Driver
```

The program will ask you what function of database you want to test ReLaTIve with. Type the corresponding number (see Appendix).

If you want to change some parameters, you can look at the provided test scripts as examples of different kind of executions.
REMARK: set scripts permission to "execute" (that is `chmod 755 *.sh`) before to run them.
REMARK: the execution of the script `TESTALL_tol-4.sh` can take a lot of time because of the computation of all the Inverse Function to compare to the ReLaTIve results.

[7] Info about parameters in the section 4.3.

Chapter 5

Appendix: Functions Database

5.1 Introduction

The files

1. dbLaplace.c,
2. dbInvLaplace.c

contain 49 Laplace Transforms with corresponding Inverses.
Each function is identified by a number so that the user, launching the Driver in the directory `ARTICLE_TESTS` can choose which function he/she wants to test ReLaTIve on, using the proper number.

Next section lists the Transforms (with the corresponding number) and their Inverses.
The last section shows the meaning of the used symbols.

The database is made of functions coming from [5], [6] and [7].
Each Transform function is of the kind

double fzXX(double z)

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

In the next, we will refer to a Laplace Transform as $F(z)$.

Each Inverse function is of the kind

double gzXX(double t)

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

In the next, we will refer to a Laplace Inverse Transform as $f(t)$.

Unless otherwise specified, it is:

$$a = 3/5, \qquad b = 5/7, \qquad n = 5, \qquad k = 9/11,$$
$$\nu = 3, \qquad r = 0.5, \quad c = 0.4, \qquad mu = 4$$

5.2 Functions

1. $F(z) = \frac{1}{z}$;

   $f(t) = 1.$

2. $F(z) = \frac{1}{z^2}$;

   $f(t) = t.$

3. $F(z) = \frac{1}{1+2z}$;

   $f(t) = 0.5e^{-0.5t}.$

4. $F(z) = \frac{1}{(z+2)^2}$

   $f(t) = te^{-2t}.$

5. $F(z) = \frac{z}{(z-1)^2}$;

   $f(t) = (1+t)e^{t}.$

6. $F(z) = \frac{1}{z-2}$;

$f(t) = e^{2t}$.

7. $F(z) = \frac{1}{(z+a)^n}$

$f(t) = \frac{t^{n-1}}{(n-1)!e^{at}}$.

8. $F(z) = \frac{1}{(z+a)(z+b)}$

$f(t) = \frac{e^{-at}-e^{-bt}}{b-a}$.

9. $F(z) = \frac{z}{(z+a)(z+b)}$

$f(t) = \frac{ae^{-at}-be^{-bt}}{a-b}$.

10. $F(z) = \frac{z^2-1}{(z^2+1)^2}$;

$f(t) = t\cos(t)$.

11. $F(z) = \frac{z}{z^2+a^2}$

$f(t) = \cos(at)$.

12. $F(z) = \frac{1}{z^2+1}$;

$f(t) = \sin(t)$.

13. $F(z) = \frac{1}{z^2+z+1}$;

$f(t) = \frac{2}{\sqrt{3}} e^{-t/2} \sin(\frac{t\sqrt{3}}{2})$.

14. $F(z) = \frac{1}{(z+2.5)^2+1}$

$f(t) = e^{-2.5t} sin(t)$.

15. $F(z) = \frac{1}{z(z^2+a^2)}$;

$f(t) = \frac{1-\cos(at)}{a^2}$.

16. $F(z) = \frac{1}{z^2(z^2+a^2)}$;

$f(t) = \frac{at-\sin(at)}{a^3}$.

17. $F(z) = \frac{1}{(z^2+a^2)^2}$;

$f(t) = \frac{\sin(at)-at\cos(at)}{2a^3}$.

18. $F(z) = \frac{z^2}{(z^2+a^2)^2}$;

$f(t) = \frac{\sin(at)+at\cos(at)}{2a}$.

19. $F(z) = \frac{8a^3z^2}{(z^2+a^2)^3}$;

$f(t) = (1 + a^2t^2)\sin(at) - at\cos(at)$.

20. $F(z) = \frac{z}{(z^2+a^2)(z^2+b^2)}$;

$f(t) = \frac{\cos(at)-\cos(bt)}{b^2-a^2}$.

21. $F(z) = \frac{z+a}{(z+a)^2+b^2}$;

$f(t) = \cos(bt)e^{-at}$.

22. $F(z) = \frac{(3a^2)}{z^3+a^3}$;

$f(t) = e^{-at} - e^{\frac{at}{2}}(\cos(\frac{1}{2}\sqrt{3}at) - \sqrt{3}\sin(\frac{1}{2}\sqrt{3}at))$.

23. $F(z) = \frac{r^2}{z^2(z^2-r^2)}$;

$f(t) = \frac{1}{r}\sinh(rt) - t$.

24. $F(z) = \frac{1}{z^2-a^2}$;

$f(t) = \frac{\sinh(at)}{a}$.

25. $F(z) = \frac{z}{z^2-a^2}$;

$f(t) = \cosh(at)$.

26. $F(z) = \frac{4a^4}{z^4+4a^4}$;

$f(t) = \sin(at)\cosh(at) - cos(at)\sinh(at)$.

27. $F(z) = \frac{z}{z^4+4a^4}$;

$f(t) = \frac{\sin(at)\sinh(at)}{2a^2}$.

28. $F(z) = \frac{1}{(z^4-a^4)}$;

$f(t) = \frac{\sinh(at)-\sin(at)}{2a^3}$.

29. $F(z) = \frac{z}{z^4-a^4}$;

$f(t) = \frac{\cosh(at)-\cos(at)}{2a^2}$.

30. $F(z) = \ln\left(\frac{z+a}{z+b}\right)$;

$f(t) = \frac{e^{-bt}-e^{-at}}{t}$.

31. $F(z) = \ln\left(\frac{z^2+a^2}{z^2}\right)$;

$f(t) = \frac{2(1-\cos(at))}{t}$.

32. $F(z) = \ln\left(\frac{z+r}{z-r}\right)$;

$f(t) = \frac{2\sinh(0.5t)}{t}$.

33. $F(z) = \frac{1}{\sqrt{z}z-a^2}$;

$f(t) = \frac{e^{a^2t}erf(a\sqrt{t})}{a}$.

34. $F(z) = \frac{(b^2-a^2)}{\sqrt{z}(z-a^2)(\sqrt{z}+b)}$;

$f(t) = e^{a^2 t}(\frac{b \cdot erf(a\sqrt{t})}{a} - 1) + e^{b^2 t} erfc(b\sqrt{t})$.

35. $F(z) = \frac{1}{\sqrt{z+a}(z+b)^{3/2}}$;

$f(t) = e^{-0.5(a+b)t} t \left[I_0(0.5(a-b)t) + I_1(0.5(a-b)t) \right]$.

36. $F(z) = \frac{\sqrt{z+2a}-\sqrt{z}}{\sqrt{z+2a}+\sqrt{z}}$;

$f(t) = \frac{I_1(at)}{te^{at}}$.

37. $F(z) = \frac{(a-b)^n}{(\sqrt{z+a}+\sqrt{z+b})^{2n}}$;

$f(t) = \frac{nI_n(0.5(a-b)t)}{te^{0.5(a+b)t}}$.

38. $F(z) = \frac{1}{(\sqrt{z+a}+\sqrt{z})^{2\nu}\sqrt{z+a}\sqrt{z}}$;

$f(t) = \frac{I_\nu(0.5at)}{a^\nu e^{(0.5at}}$.

39. $F(z) = \frac{(z-\sqrt{z^2-a^2})^\nu}{\sqrt{z^2-a^2}}$;

$f(t) = a^\nu I_\nu(at)$.

40. $F(z) = \frac{1}{\sqrt{z^2+a^2}e^{k(\sqrt{z^2+a^2}-z)}}$;

$f(t) = J_0(a\sqrt{t^2+2kt})$.

41. $F(z) = \frac{\sqrt{z+2a}}{\sqrt{z}} - 1$;

$f(t) = \frac{a[I_0(at)+I_1(at)]}{e^{at}}$.

42. $F(z) = \frac{1}{\sqrt{z^2+a^2}}$;

$f(t) = J_0(at)$.

43. $F(z) = \frac{1}{z^{n+1/2}}$;

$f(t) = \frac{t^{n-1/2}}{\Gamma(n+1/2)}$.

44. $F(z) = \frac{1}{z^{3/2}e^{k\sqrt{z}}}$;

$f(t) = \frac{2\sqrt{t}e^{-k^2/4t}}{\sqrt{\pi}} - k \cdot erfc(\frac{k}{2\sqrt{t}})$.

45. $F(z) = \frac{1}{z(z+1)}\left[\frac{1}{2z} - \frac{e^{-2z}}{1-e^{-2z}}\right]$;

$f(t) = \frac{1}{2} + e^{-t}\left[\frac{1}{2} - \frac{e^2}{e^2-1}\right] - \frac{1}{\pi}\sum_{n=1}^{\infty}\frac{\sin(n\pi t)-\arctan(n\pi))}{n\sqrt{n^2\pi^2+1}}$.

46. $F(z) = \frac{(100z-1)\sinh(\sqrt{z}/2)}{z[z\sinh(\sqrt{z})+\sqrt{z}\cosh(\sqrt{z})]}$;

$f(t) = -\frac{1}{2} + \sum_{n=1}^{\infty}\left(\left(100 + \frac{1}{b_n^2}\right)\frac{2b_n\sin(b_n/2)e^{-b_n^2 t}}{(2+b_n^2)\cos(b_n)}\right)$

With $b_n\tan(b_n) = 1$.

47. $F(z) = \frac{1}{z}e^{-r\sqrt{\frac{z(1+z)}{1+cz}}}$;

$f(t) = \frac{1}{2} + \frac{1}{\pi}\int_0^{\infty}\left[e^{[-rm\sqrt{u/2}(\cos(\theta)-\sin(\theta)]}\ \sin[tu - rm\sqrt{u/2}(\cos(\theta)+\sin(\theta))\right]\frac{du}{u}$.

With $m = \left[\frac{1+u^2}{1+c^2u^2}\right]^{1/4}$, $\quad 2\theta = \arctan(u) - \arctan(cu)$.

48. $F(z) = \frac{1}{z(\sqrt{1+z^2+\frac{z^4}{16}}+\frac{z}{4}\sqrt{16+z^2})^2}$

$f(t) = 1 - \frac{1}{\pi}\int_0^{u_1}[(\sin(ut+2k) - \sin(ut-2k)]\frac{du}{u} +$

$+\frac{1}{\pi}\int_{u_2}^{4}[(\sin(ut+2k) - \sin(ut-2k)]\frac{du}{u}.$

With $\cos(k) = \frac{1}{4}\sqrt{(u_1^2-u^2)(u_2^2-u^2)}$, $\quad u_1 = 2\sqrt{2-\sqrt{3}}$, $\quad u_2 = \sqrt{2+\sqrt{3}}$.

49. $F(z) = \frac{z-\sqrt{z^2-1}}{\sqrt{z}\sqrt{z^2-1}\sqrt{z-0.5\sqrt{z^2-1}}}$;

$G(t) = \frac{2}{\pi}\int_0^c \left[\cosh(tu)\frac{u\sqrt{(R+u)/2}+\sqrt{c^2-u^2}\sqrt{(R-u)/2}}{R\sqrt{c^2-u^2}\sqrt{u}}du\right] +$

$+\frac{2}{\pi}\left[\int_0^b \frac{u-\sqrt{c^2+u^2}}{\sqrt{u}\sqrt{c^2+u^2}\sqrt{N\sqrt{c^2+u^2}-u}}\cos(tu)du\right].$

With $R = \sqrt{u^2+N^2(c^2-u^2)}$, $\quad b = \sqrt{(1-N)/(1+N)}$, $\quad c = 1$, $\quad N = 0.5$.

5.3 Remarks

Some remarks about symbols used in functions definition.

Gamma function $\Gamma(t)$, subject to t not being a negative integer or zero:

$$\Gamma(t) = \int_0^\infty e^{-x}x^{t-1}dx$$

Bessel Function of the First Kind:

- regular cylindrical Bessel function of zeroth order, $J_0(z)$.
- regular cylindrical Bessel function of order n, $J_n(t)$.

- regular cylindrical Bessel function of fractional order ν, $J_\nu(t)$.

$$J_\nu(z) = \left(\frac{1}{2}\right)^\nu \sum_{k=0}^{\infty} (-1)^k \frac{(\frac{1}{4}z^2)^k}{k!\Gamma(\nu+k+1)} \quad arg(z) < \pi, \quad \nu \in \Re$$

Modified Bessel Function of the First Kind:

- regular modified cylindrical Bessel function of zeroth order, $I_0(t)$.
- regular modified cylindrical Bessel function of first order, $I_1(t)$.
- regular modified cylindrical Bessel function of order n, $I_n(t)$.
- regular modified Bessel function of fractional order ν, $I_\nu(t)$, for $t > 0$, $\nu > 0$.

$$I_\nu(t) = i^{-\nu} J_\nu(it), \quad t \in \Re, \quad t \geq 0, \quad \nu \in \Re$$

Error Function $erf(t)$:

$$erf(t) = \frac{2}{\sqrt{\pi}} \int_0^t e^{-x^2} dx$$

Complementary Error Function $erfc(t)$:

$$erfc(t) = 1 - erf(t) = \frac{2}{\sqrt{\pi}} \int_t^\infty e^{-x^2} dx$$

Dawson's integral for t:

$$I(t) = e^{-t^2} \int_0^t e^{y^2} dy = \frac{1}{2}\sqrt{\pi} e^{-t^2} erf(t)$$

used to compute:

$$g(t) = \frac{erf(\sqrt{a-b}\sqrt{t})}{\sqrt{a-b}e^{bt}} = e^{-bt} \frac{I(t)\frac{2}{\sqrt{\pi}e^{(\sqrt{b-a}\sqrt{t})^2}}}{\sqrt{a-b}}$$

since

$$erf(\sqrt{a-b}\sqrt{t}) = erf(i\sqrt{b-a}\sqrt{t}) = I(t)\frac{2}{\sqrt{\pi}} e^{(\sqrt{b-a}\sqrt{t})^2}$$

Some used constants:

- Euler-Mascheroni constant γ;
- Greek Pi constant π.

Bibliography

[1] L. D'Amore, R. Campagna, V. Mele, A. Murli and M. Rizzardi *ReLaTIve. An Ansi C90 software package for the Real Laplace Transform Inversion*, Numer. Algorithms 63, 1, 187-211. DOI=10.1007/s11075-012-9636-0 http://dx.doi.org/10.1007/s11075-012-9636-0, May 2013.

[2] `http://www.mingw.org/wiki/InstallationHOWTOforMinGW`.

[3] `http://www.mingw.org/wiki/Getting_Started`.

[4] `http://cygwin.com`.

[5] DAVIES, MARTIN. *Numerical inversion of Laplace Transform. A survey and comparison of methods*. J. of Comp. Phisics, 33, 1-32, 1979.

[6] P. VALKO, S. VAIDA, 2002. *Inversion of Noise-free Laplace Transforms: towards a standardized set of test problems*, Inverse problems in engineering, Vol. 10, 467-483.

[7] D.G. DUFFY, *On the numerical inversion of Laplace Transforms: Comparison of three new methods on characteristic problems from applications*. ACM Transactions on Mathematical Software, Vol. 19, No. 3, pp. 333-359, 1993.

[8] `http://www.gnu.org/software/gsl/`.