
Recursive calculation of the Green's function

Release 1.2.0

A. Del Monte, N. Manini

October 04, 2015

Contents

1	Purpose	1
2	Required software	2
2.1	Installing Lapack	2
2.2	Installing TNT	2
3	Compiling	3
4	Running	3
5	Output	3
6	Testing	4
7	Reference	4
8	License	5
9	Development notes	5
10	To-do list	5
11	History	6
12	About this document	7

1 Purpose

The executable `eval-Green-func` calculates the Green's function of a quantum system of coupled anharmonic oscillators, put initially in some arbitrary excited state. From the Green's function it is straightforward to get the

spectrum of the system, as explained below in section [Output](#). In particular this method can be used to calculate the vibrational spectrum of a molecule.

The program requires as input the polynomial expansion of the vibrational potential in normal modes (up to any finite order) of the molecule to be analysed.

For more information about the purpose of this software see section [Reference](#).

2 Required software

To compile and run the program you need for:

1. An ANSI C++ compiler. The program has only been tested with GNU g++.
2. Lapack, the Fortran library for linear algebra, needed for matrix inversion. It can be freely downloaded at <http://www.netlib.org/lapack/index.html>; it is also found as a .deb or .rpm package for some Linux distributions.
3. The Template Numerical Toolkit library (version 1.2.6) developed at NIST, available for download at the link <http://math.nist.gov/tnt/download.html>.
4. A make utility is useful. The file Makefile has been tested only with the GNU make and contains some GNU extensions: it should be probably adapted to work with similar tools provided by other vendors.

2.1 Installing Lapack

On Debian and derived Linux distributions, like Ubuntu, you can install the official package of the Lapack library with the command:

```
sudo apt-get install liblapack-dev
```

(the development “dev” package is required to compile the executable).

Alternatively you can compile the Lapack library from sources. You need for the GNU Fortran compiler. We used the following commands to download and build Lapack 3.5 on Cygwin ¹:

```
wget http://www.netlib.org/lapack/lapack-3.5.0.tgz
tar -xvzf lapack-3.5.0.tgz
cd lapack-3.5.0
cp INSTALL/make.inc.gfortran make.inc
make blaslib
make lapacklib
```

We got the two files librefblas.a and liblapack.a.

2.2 Installing TNT

Since the TNT library is composed by header files only, you have simply to download the package for version 1.2.6 (available at the web link specified above) and unpack it in some directory.

¹ Cygwin is a porting of UNIX/Linux to Windows, <https://www.cygwin.com/>.

3 Compiling

In order to compile the software, do the following from a UNIX/Linux shell:

1. Extract the zip archive `eval-Green-func-1_2_0.zip` to some directory with the command `unzip eval-Green-func-1_2_0.zip`.
2. Go to the directory `eval-Green-func`, created after the zip archive has been extracted.
2. Update the Makefile with the proper settings for the TNT and Lapack libraries, depending on how they have been installed on your system.
3. Run the make utility, without any argument. If your computer has no make utility, guess what to do from the Makefile.

The build process will create the executable file `eval-Green-func` (or `eval-Green-func.exe` on Cygwin).

The program has been compiled and tested with Lapack version 3.5.0 and Template Numerical Toolkit 1.2.6 on the following operating-system configurations:

- Linux Ubuntu version 14.04 for architecture `x86_64` with `g++` compiler 4.8.4;
- Linux XUbuntu 15.04 for `x86_64` with `g++` 4.9.2;
- Cygwin version CYGWIN_NT-6.3 for `x86_64` with `g++` 4.9.3.

The compiler didn't give any error or warning.

4 Running

On UNIX/Linux, run the command:

```
./eval-Green-func INPUTFILE
```

where `INPUTFILE` is path-name of the configuration file. An example of the configuration can be found in the file `test/test_config.txt`, that provides the general scheme for the input of the program with some explanation.

The output data is printed at standard output, while some information about the program execution is written to standard error. On UNIX/Linux you can redirect the output to a file by a command like this:

```
./eval-Green-func INPUTFILE > OUTPUTFILE
```

The program is single-threaded but it is suited for distributed calculation: several instances can be launched in parallel to calculate a different part of the spectrum.

5 Output

The output data contains several commented infos starting with character '#', followed by the spectrum in the notation:

```
<energy>  <-Im G00 (energy)> <Re G00 (energy)>  <-Im G01 (energy)> <Re G01 (energy)> ...
```

(row by row, the whole $n \times n$ Green's matrix, with n = number of states in tier 0).

If $n = 1$, the imaginary part of the Green's function represents the computed spectrum. Otherwise you need to choose the linear combination $a_1 |1\rangle + a_2 |2\rangle + \dots$ of the states $|1\rangle, |2\rangle, \dots$ of the tier 0, representing the initial excitation; then the spectrum is the imaginary part of the matrix product:

```

(a1)
(a2)
(a1,a2,...)^* . G . (.) = sum_i sum_j ai^* Gi j (energy) aj
(.)
(.)

```

where * indicates the complex conjugate.

In the $n = 1$ case you may visualize the output - that is the spectrum - with `gnuplot` by entering the `gnuplot` command-line session and issuing the command:

```
plot "OUTPUTFILE" with lines
```

The utility will display a graphic using the first column for the X-axis and the second column for the values of the Y-axis. For some explanation about how to use `gnuplot`, see for example <http://lowrank.net/gnuplot/datafile-e.html>. To install `gnuplot` on Debian-derived Linux distributions, use the command:

```
sudo apt-get install gnuplot-x11
```

Alternatively you can use the utility `xmgrace`; you have just to pass the output file name as command line parameter:

```
xmgrace OUTPUTFILE
```

6 Testing

In the distribution we include the file `test_output.txt` produced by running:

```
$ ./eval-Green-func test/test_config.txt > test/test_output.txt
```

Before doing any serious calculation, check that your executable produces the same output. We got exactly the same data for all the operating-system configurations listed in section [Compiling](#).

We tried the performances of the test on two different machines with installed Ubuntu 14.04. The time needed to complete the run (redirecting the output to file) was:

- 0.22 seconds with a CPU Intel Core i5-2500 (3.30GHz \times 4) and 16 GiB of RAM;
- 0.40 seconds with an Intel Core2 Duo P9500 (2.53GHz \times 2) with 8 GiB of RAM.

7 Reference

This software was developed at the Physics Department of the “Universita’ degli Studi di Milano” (State University, Milan, Italy) by Alessio Del Monte and Nicola Manini, initially for the Alessio’s thesis, then for the article “*Low-energy unphysical saddle in polynomial molecular potentials*” published in *Molecular Physics*, Volume 103, Number 5, March 10, 2005, pp. 689-696(8). The article can be purchased at the website [ingentaconnect](#).

For more details see the webpage [IVR in polyatomic molecules](#) created by Nicola, where you can download the latest version of the software and Alessio’s thesis.² The latter provides a presentation of the physics theory underlying the program’s calculations, a description of the algorithms (in particular in sections 3.3.3 and 4.2) and some notes regarding the difficulties of the numerical computation (sec. 4.3).

Send bugs, comments and suggestions to alessiodelmonte@gmail.com or to nicola.manini@fisica.unimi.it.

² Note that this webpage is cited by the article in the section “References” with an old URL that is not accessible any more.

8 License

This software was developed by Alessio Del Monte and Nicola Manini. It is not subject to copyright protection and is in the public domain: permission is granted to any individual or institution to use, copy, modify or redistribute it. The authors make no guarantees about this software and assume no responsibility for its use by other parties.

Whoever makes use of it is asked to cite “A. Del Monte, N. Manini, L.G. Molinari, and G.P. Brivio, Mol. Phys. 103, 689 (2005)” and the URL <http://materia.fisica.unimi.it/manini/ivr.html>.

This license statement should be provided in derived software.

9 Development notes

The software has been developed in the C++ programming language using the 1998 ISO standard, but it is compatible with the more recent 2011 ISO standard (usually referred as C++11). We have widely employed the Standard Template Library (STL) whenever this was possible without compromising the performance of the numerical computation.

The program makes calls to a few external routines of the Lapack library for linear algebra written in Fortran. Anyway the linking to Fortran libraries is not a problem since the compiled objects have the same linking format as they were modules developed in C; thus to call Lapack routines from C++ code it is possible to use the `extern "C"` directive. Moreover note that Fortran for multidimensional arrays uses the column-major ordering convention instead of the row-major convention adopted by C and C++; anyway this is not a problem since the Lapack routines are called by the program only for symmetric matrices.

As the basis for the linear algebra calculations we used the classes `TNT::Vector` and `TNT::Matrix` of the Template Numerical Toolkit library, that in the meanwhile have been superceded by the `Array` classes of the same library, as documented here: <http://math.nist.gov/tnt/overview.html>. These legacy classes are retained in TNT version 1.2.6 just for backward compatibility.

In the comments to the source code we often refer to the Italian translation of the book “*The C++ Programming Language - 3rd Edition*”, by Bjarne Stroustrup.

Some points to be reviewed or improved are marked with “%%”. In the files `tnt_cmat_modified.h` and `tnt_094_fortran_modified.h`, the modifications to the original code of the TNT library are marked with the acronym “ADM”, for Alessio Del Monte.

10 To-do list

There are several improvements that can be done on the software, in particular:

- Review and improve the comments to the source code; update the references.
- Add the proper markup to generate automatic documentation with Doxygen.
- Improve and make more uniform the indentation of the code.
- Add the files to some source code management tool like Subversion or GIT.
- Generate a portable executable by statically linking the needed libraries (see option `-static` of `g++`)
- Add more detailed output informations, providing timing for basis generation and Green’s function calculation.
- Improve the handling of errors and exceptions.
- Add a smarter mechanism for basis generation, with a floating threshold.

- Substitute the legacy classes `TNT::Vector` and `TNT::Matrix` of the TNT library with either `TNT::Array2D` or `TNT::Fortran_Array2D`. Probably also the the design of classes in `lib_sparsemat.hpp` should be evolved.

In general all the code should be reviewed and probably improved and refactored in several places.

Note that this software currently is *not* under active development, thus may be that only a very few of the listed items will be completed in the future by the authors.

11 History

- Release 1.2.0, 4 October 2015 - We brought back to life the software, that was not even possible to compile with the latest version of the GNU `g++` compiler. In particular we:
 - resolved all the compiler errors and warnings;
 - fixed a bug that made the program to crash while reading input parameters;
 - upgraded the TNT library to the latest stable version (1.2.6), eliminating some customized files that were not necessary any more;
 - changed the program in order to read the configuration from a file instead of from standard input;
 - reorganized the sources by renaming them, refactoring a few classes and functions, simplyfing the inclusion of header files;
 - improved and updated the `Makefile`;
 - renamed the target executable to the more significant name `eval-Green-func`;
 - automated the generation of make dependencies by means of the `-MMD` compiler option;
 - tested the executable in the operating-system configurations listed above, obtaining the new test output data included in the package;
 - updated and extended the `README` file and converted it to the *reStructuredText* markup format;
 - generated the PDF version of the `README` using *Sphinx*;
 - rewritten the license statement to make it more precise.

Note that we didn't made any change to the logic of the program, that thus should provide exactly the same results of the previous version 1.1. Indeed when we compared the output of the test with version 1.2.0 with the output data included in package 1.1, we found only differences in the very last decimal digits; since the two versions were built and executed with different versions of the operating system, of the architecture (from 32 to 64 bit), of the compiler and of the Lapack library, some differences in the numerical approximation can be expected.

- Release 1.1, 25 November 2003 - A very few improvements:
 - Added the possibility of setting a maximum size for each tier, selecting states according their cumulative effective coupling strength (c.c.s.);
 - redesigned the code for tier basis generation.
- Release 1.0, 3 November 2003 - First public version released on the web.
- Beta 0.1, 2 April 2003 - “Dirty” version developed for Alessio’s thesis.

12 About this document

This documents has been created by A. Del Monte and N. Manini. It is written using the *reStructuredText* markup, <http://docutils.sourceforge.net/rst.html>. The source file `README.rst` has been converted to PDF and other formats by means of the *Sphinx* tool, <http://sphinx-doc.org>. The configuration files used to convert it are included in the ZIP package. See the `Makefile` for some more information.

For a quick reference to the *reStructuredText* markup, see <http://docutils.sourceforge.net/docs/user/rst/quickref.html>.