General **Romber**g FORTRAN95 code - **RomberG**

Guillem Pey
gpeycosta@gmail.com
gpey@iciq.es

February 26, 2026

Author's note: *This documentation and software are based on the paper "M. Medved et al. / Journal of Molecular Structure: THEOCHEM 847 (2007) 39–46"*[1]

1 A summary of the Rutishauser-Romberg methodology and its generalization

The original Romberg methodology was originally developed for numerical integration to improve the accuracy of the trapezoidal rule by linearly combining lower-order trapezoids. Following Romberg's work, Rutishauser proposed applying the latter to numerical differentiation to eliminate the truncation error arising from higher-order derivative terms in the central difference formula.

Let us consider the Taylor series expansion of a function f(x) evaluated at two nearby points displaced by h, *i.e.* f(x+h) and f(x-h):

$$\begin{cases} f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f^{(3)}(x) + \frac{h^4}{4!}f^{(4)}(x) + \dots \\ f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f^{(3)}(x) + \frac{h^4}{4!}f^{(4)}(x) + \dots \end{cases} \tag{1}$$

Subtracting f(x-h) from f(x-h) yields:

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{2h^3}{3!}f^{(3)}(x) + \frac{2h^5}{5!}f^{(5)}(x) + \dots \tag{2}$$

$$D(h) = \frac{f(x+h) - f(x-h)}{2h} = f'(x) + \frac{h^2}{3!}f^{(3)}(x) + \frac{h^4}{5!}f^{(5)}(x) + \dots \tag{3}$$

$$f'(x) \simeq D(h) + \mathcal{O}(h^2) \tag{4}$$

Consequently, the truncation error is of order h^2. It can then be eliminated by appropriately combining the expansions of D(h) and D(h/2).

$$D(h) = f'(x) + Ah^2 + Bh^4 + \dots \tag{5}$$

$$D(h/2) = f'(x) + A\frac{h^2}{4} + B\frac{h^4}{16} + \dots \quad \rightarrow \quad 4D(h/2) = 4f'(x) + B\frac{h^4}{4} + \dots \tag{6}$$

$$4D(h/2) - D(h) = 3f'(x) + \mathcal{O}(h^4) \tag{7}$$

$$f'(x) = \frac{4D(h/2) - D(h)}{3} + \mathcal{O}(h^4) \tag{8}$$

Let the smallest initial step size be h. We define an increasing sequence of step sizes by doubling h at each row:

$$h_j = 2^{j-1}h \tag{9}$$

where $j = 1, 2, 3, \ldots$ represents the row.

For a first-order derivative the first column of the Romberg triangle is computed using the standard central difference formula for the first derivative:

$$D_{j,1} = \frac{f(x + h_j) - f(x - h_j)}{2h_j} \tag{10}$$

Generalizing this, to eliminate the $O(h^{2k})$ error in column k, the iterative formula becomes:

$$D_{j,k} = \frac{4^{k-1} D_{j-1,k-1} - D_{j,k-1}}{4^{k-1} - 1} \tag{11}$$

where j is the current row index and k is the column index also referred to as the extrapolation level. This can also be written in the additive form for easier computation:

$$D_{j,k} = D_{j-1,k-1} + \frac{D_{j-1,k-1} - D_{j,k-1}}{4^{k-1} - 1} \tag{12}$$

The approximations may be built row by row like in the below table

Row (j)	**Step Size**	**O($\mathbf{h^2}$)**	**O($\mathbf{h^4}$)**	**O($\mathbf{h^6}$)**	**O($\mathbf{h^8}$)**
1	h	$D_{1,1}$			
2	$2h$	$D_{2,1}$	$D_{2,2}$		
3	$4h$	$D_{3,1}$	$D_{3,2}$	$D_{3,3}$	
4	$8h$	$D_{4,1}$	$D_{4,2}$	$D_{4,3}$	$D_{4,4}$

- **Column 1 ($k = 1$):** Central differences.
- **Column 2 ($k = 2$):** First extrapolation. $D_{j,2} = \frac{4D_{j-1,1} - D_{j,1}}{3}$.
- **Column 3 ($k = 3$):** Second extrapolation. $D_{j,3} = \frac{16D_{j-1,2} - D_{j,2}}{15}$.
- **Column 4 ($k = 4$):** Third extrapolation. $D_{j,4} = \frac{64D_{j-1,3} - D_{j,3}}{63}$.

Mathematically, it extracts the combined information from h, $2h$, $4h$, and $8h$ to cancel out terms related to the truncation error. Using the previous methodology of subtracting Taylor series expansions, the following set of formulae can be derived up to the fourth-order derivative:

$$f^{(1),k} \approx \frac{f(2^{k-1}h) - f(-2^{k-1}h)}{2^k h} \tag{13}$$

$$f^{(2),k} \approx \frac{f(2^{k-1}h) - 2f(0) + f(-2^{k-1}h)}{(2^{k-1}h)^2} \tag{14}$$

$$f^{(3),k} \approx \frac{f(2^k h) - 2f(2^{k-1}h) + 2f(-2^{k-1}h) - f(-2^k h)}{2(2^{k-1}h)^3} \tag{15}$$

$$f^{(4),k} \approx \frac{f(2^k h) - 4f(2^{k-1}h) + 6f(0) - 4f(-2^{k-1}h) - f(-2^k h)}{(2^{k-1}h)^4} \tag{16}$$

where these formulae can only be applied for step sizes in increasing in powers of two. The general set of formulae applicable for step sizes a$\in$[1,2], and the ones implemented in this code and computed

on the fly based on the data, are the following:

$$f_a^{(1),k} \approx \frac{f(a^{k-1}h) - f(-a^{k-1}h)}{2a^k h} \tag{17}$$

$$f_a^{(2),k} \approx \frac{f(a^{k-1}h) - 2f(0) + f(-a^{k-1}h)}{(a^{k-1}h)^2} \tag{18}$$

$$f_a^{(3),k} \approx 3 \cdot \frac{f(a^k h) - af(a^{k-1}h) + af(-a^{k-1}h) - f(-a^k h)}{a(a^2-1)(a^{k-1}h)^3} \tag{19}$$

$$f_a^{(4),k} \approx 12 \cdot \frac{f(a^k h) - a^2 f(a^{k-1}h) + 2(a^2-1)f(0) - a^2 f(-a^{k-1}h) - f(-a^k h)}{a^2(a^2-1)(a^{k-1}h)^4} \tag{20}$$

Since this software is aimed at computing the electric field derivatives of non-linear optical properties, *i.e.* the electronic energy, the dipole moment, the polarizability matrix, the first hyperpolarizability tensor, and so on, the definitions of the just stated properties read as

$$\mu_i := -\frac{\partial E(F)}{\partial F_i} \tag{21}$$

$$\alpha_{ij} := -\frac{\partial^2 E(F)}{\partial F_i \partial F_j} = \frac{\partial \mu_i}{\partial F_j} \tag{22}$$

$$\beta_{ijk} := -\frac{\partial^3 E(F)}{\partial F_i \partial F_j \partial F_k} = \frac{\partial^2 \mu_i}{\partial F_j \partial F_k} = \frac{\partial \alpha_{ij}}{\partial F_k} \tag{23}$$

$$\gamma_{ijkl} := -\frac{\partial^4 E(F)}{\partial F_i \partial F_j \partial F_k \partial F_l} = \frac{\partial^3 \mu_i}{\partial F_j \partial F_k \partial F_l} = \frac{\partial^2 \alpha_{ij}}{\partial F_k \partial F_l} = \frac{\partial \beta_{ijk}}{\partial F_l} \tag{24}$$

Eqs.(17-20) when differentiating from the energy are automatically toggled to be negative to match with the electronic energy definition of the non-linear optical properties.

2 Prerequisites of RomberG.exe

2.1 Compiling RomberG.f95

RomberG.exe is a FORTRAN95 flag-based script. FORTRAN95 itself is not capable of reading flags; that is why this code has been built using this module. Therefore, the first requirement is to have installed this module, or downloaded, in your working directory before compiling RomberG.f95. Running RomberG does not require having the aforementioned module in the working directory.

Next is compiling RomberG.f95 along with the getopt module with the following command line:

```
$ gfortran f90getopt.F90 -ffree-line-length-none -o ROMBERG.exe ROMBERG.f95
```

The "-ffree-line-length-none" compiling flag is mandatory to surpass FORTRAN's notorious limit of 80 characters per line. Once the previous command line is executed, check the executable ROMBERG.exe is in your working directory.

2.2 System-based prerequisites

- RomberG.exe is strictly prepared to read the Gaussian16 fchk files, and single-column files containing numerical values. For single-column files, to avoid inconsistencies with Gaussian signs (*vide infra*), it is encouraged filtering the output fchk files with `$ ls -v` and getting the energy, or the desired property/number, following said order.

- Wherever this code is executed, please check the location of the grep command. Depending on whether ROMBERG.exe is executed in a cluster (HPC), it may be located in '/bin/grep' or in a personal laptop/desktop in '/usr/bin/grep'. These two scenarios are already considered, the source code is also given to include other situations.

- ROMBERG.exe has been tested for 'GNU Fortran (Debian 4.4.7-2) 4.4.7' and 'GNU Fortran (GCC) 4.8.5 20150623 (Red Hat 4.8.5-36)' compilers. Other compilers, such as Ubuntu's default FORTRAN compiler, may output gibberish.

- Although the code is rather robust at reading the properties and "ignoring" the file names, it is strongly encouraged that the folder that contains the .fchk files is the name of the molecule, and the name of the .fchk files are named after something such as 'file' to avoid any kind of conflicts. For single-column files, such requirement is not mandatory yet the input file still requires of an extension to distinguish from a directory.

2.3 File organization

As the last point of the previous section stresses, it is necessary to have all fchk files of the molecule in a directory named after the molecule. It is recommended to name the fchk files as:

```
basename_field-direction_magnitude-of-the-field
```

For instance, a valid fchk filename is "mol3_X-0.0004" or "ethaney+0.0064".

If the user desires to generate RomberG-ready .fchk files without any trouble, there is also the 'makeromberg.py' Python script. This script works under the assumption that there is a com and log file in the directory under this name-template 'sp(*name-of-the-molecule*)0.com' with the keyword 'Field=X+000':

```
%nproc=4
%mem=8GB
%chk=spH2O0.chk
%oldchk=H2O.chk
# B3LYP/6-311+G(d,p) Polar NoSymm guess=read geom=check Field=X+000
...
```

This simple example uses the name of the molecule 'H2O'. For further details, the user may read the 'makeromberg' script.

WARNING! RomberG works under the "chemistry convention" of electric fields - the electric field in the route section must be of opposite sign to the name of the file. Hence, the input files will be generated accordingly.

3 Executing of RomberG.exe

The options of RomberG.exe can be displayed by executing the following command line:

```
$ ./ROMBERG.exe -h {name_of_the_molecule}
```

and it is going to print in screen the following:

Several options are available for RomberG.exe:

- -i, –input : Input property - Energy/energy/E ; Dipole/dipole/M ; Alpha/alpha/A ; Beta/beta/B. Character-string expected.
- -o, –output : Output property - Dipole/dipole/M ; Alpha/alpha/A ; Beta/beta/B ; Gamma/gamma/G. Character-string expected.
- -F, –totalfields : Number of total fields probed. Integer expected.
- -I, –isotropic : Toggle the calculation of isotropic properties.
- -S, –simple : Set to zero the calculated properties under 10^{-6} with Romberg errors lower than 10%.
- -p, –printP : Print the derivatives when computing the Romberg triangle for each component.
- -Q, –init : Different first field. Default $1.0 \cdot 10^{-4}$.
- -Y, –sqrt : Activate the step of $\sqrt{2}$ to compute the derivatives.
- -J, –force : Force the complete calculation of isotropic properties despite non-computed values. TO BE IMPLEMENTED
- -h, –help : Displays this message.

REMEMBER: The execution of RomberG.exe requires the name of the molecule to display the help message.

For instance, a regular command-line execution could be

```
$ ./ROMBERG.exe -i alpha -o gamma -F 13 -S {name_of_the_molecule}
```

this example of command line is going get the elements of the polarizability matrix of all .fchk, compute the elements of $\boldsymbol{\gamma}$ by computing the second derivative of each element of $\boldsymbol{\alpha}$ with respect to X, Y, and Z. With those values it is going to print them in screen, compute several quantities related to the second hyperpolarizability, and display zeroes when necessary.

WARNING! There are combinations of properties, i.e `-i dipole -o gamma`, that for computing isotropic properties are going to assume that non-computed terms are zero. Activating the `--force` flag bypasses over the rule.

4 Output of RomberG.exe

RomberG.exe is coded not only to compute the formulae on Eqs.(17-20) but also to compute isotropic properties.

Without specifying the flag of `--isotropic`, only the Romberg triangles are going to be printed along with the tensors composed with the properties with the minimum Romberg error using the Gaussian fchk format. Furthermore, when computing first(second) and second(third) order derivatives from alpha(dipole moment) the following properties are going to be computed:[2]

$$\mu = \sqrt{\mu_x^2 + \mu_y^2 + \mu_z^2} \tag{25}$$

$$\alpha = \frac{\alpha_{xx} + \alpha_{yy} + \alpha_{zz}}{3} \tag{26}$$

The output of either output property is printed following Gaussian's format-checkpoint format. Activating the `--isotropic` flag activates the calculation of the below formuli applicable for any general case:

$$\Delta\alpha = \frac{\sqrt{2}}{2}((\alpha_{xx}-\alpha_{yy})^2+(\alpha_{yy}-\alpha_{zz})^2+(\alpha_{zz}-\alpha_{xx})^2+3\cdot(\alpha_{xy}^2+\alpha_{yx}^2+\alpha_{yz}^2+\alpha_{zy}^2+\alpha_{xz}^2+\alpha_{zx}^2))^{1/2} \tag{27}$$

$$\beta_{vec} = \frac{1}{3}\sqrt{\sum_i^{x,y,z}(\sum_j^{x,y,z}\beta_{ijj}+\beta_{jij}+\beta_{jji})^2} \tag{28}$$

$$\beta_{||} = \frac{3}{5\mu}\sum_{i,j}^{x,y,z}\mu_i\cdot\beta_{ijj} \tag{29}$$

$$\beta_{\perp} = \frac{1}{5\mu}\sum_{x,y,z}^{i,j}\mu_i\cdot(2\beta_{ijj}-3\beta_{jij}+2\beta_{jji}) \tag{30}$$

$$\gamma_{||} = \frac{1}{5}\sum_{i,j}^{x,y,z}(\gamma_{ijji}+\gamma_{ijij}+\gamma_{iijj}) \tag{31}$$

Since the code is open-source, if the user wants to implement other formulae other than the ones presented here or correct the multiplying factors, they are free to modify the original code. For other further corrections, implementations, requests, etc., feel free to contact me through the email at the beginning of the document.

5 Summary of top-to-bottom execution of RomberG.exe

This section is a summary that covers all the required steps from the geometry optimization of the interest molecule to the execution of RomberG.exe

1. (*Recommended*) Optimization of the molecule. Use the basename of the molecule, *i.e.*, H2O or CH3OH, for the following steps. It is also recommended to move all the optimization files, except for the .chk, elsewhere.

2. Prepare the 'Polar NoSymm Field=X+000' calculation according to the last textbox in Section 2 and following the name convention - 'sp(NAME)0.com'. You may not read from the previous checkpoint file.

3. Execute makeromberg.py. For instance, `$ python makeromberg.py H2O`.

4. Submit the calculations to your HPC or local machine.

5. In the parent directory of the molecule calculations, compile RomberG.f95 as stated before and execute following the guidelines presented in Section 3.

6. You may delete the checkpoint files and the submission files to save on disk space. As long as all the necessary .fchk are present, all other files can be deleted.

7. Execute the desired command line of RomberG.

6 Possible conflicts with RomberG.exe and fchk files

The Gaussian computation of NLOPs at the DFT level is analytical. However, computing them at the post-HF level is analytical only until the polarizability matrix. Consequently, toggling as input NLOP the first hyperpolarizability in this second scenario **will return gibberish**.

7 Bibliography

[1] Miroslav Medveď et al. "A generalized Romberg differentiation procedure for calculation of hyperpolarizabilities". In: *Journal of Molecular Structure: THEOCHEM* 847.1 (2007), pp. 39–46. ISSN: 0166-1280.

[2] Robert W. Góra and Bartosz Błasiak. "On the Origins of Large Interaction-Induced First Hyperpolarizabilities in Hydrogen-Bonded π-Electronic Complexes". In: *The Journal of Physical Chemistry A* 117.31 (2013), pp. 6859–6866.