User's Guide

fast :

Fast Absorption Simulation by TDDFT

Version 1.1

Juin, 2016

Ross Brown, Cédric Castagne, Olivier Coulaud, Dietrich Foerster, Peter Koval.

Acknowledgements

It is a pleasure to thank Professor James Talman, University of Western Ontario, who kindly donated important routines for 3j, 6j, 9j symbols and a fast Hankel transform. The ongoing encouragement and advice of the SIESTA development team, particularly Alberto García, Institut de Ciència de Materials de Barcelona and Daniel Sanchez-Portal, Centro de Física de Materiales Donostia-San Sebastián, and of Luc Giraud, Inria-Bordeaux Sud-Ouest, is gratefully acknowledged. This work was supported by A.N.R. grant CIS 2007 NOSSI. —

Contents

1 About fast

FAST is a linear response time dependent density functional program for computing the electronic absorption spectrum of molecular systems. It uses an $O(N^3)$ linear response method based on finite numerical atomic orbitals and deflation of linear dependence in atomic orbital product space. This version is designed to work with data produced by the SIESTA DFT code. The code produces as principal output a numerical absorption spectrum (complex part of the polarisability, loosely called the polarisability below) and a list of transition energies and oscillator strengths deduced from fitting Lorentzians to the numerical spectrum. Considering the absence of hybrid functionals in SIESTA and that concerning calculation of spectra, generalised gradient Hamiltonians are not usually considered to be notably better than the local density approximation, the present release of FAST works only with LDA, which despite its limitations, has provided useful results on the systems to which the present authors have applied it.

This Reference Manual contains descriptions of all the input, output and execution features of FAST*, but is not intended as a tutorial introduction to the program. Potential users should consult the papers listed below for details of the methods employed.*

The code is written in Fortran 95 with dynamic memory allocation. It uses routines from the BLAS, LAPACK and FFTW package (see http://www.fftw.org/) for all FFT routines. It may be compiled for serial or parallel execution (under OpenMP, MPI and MPI+OpenMP). Installation is illustrated in section 2. Owing to licence related reasons, users should first obtain and install either the SIESTA code or the standalone SIESTA XC exchange-correlation library (http://www.icmab.es/siesta) . The present code includes two pieces of software under licence :

- Routines for numerical quadrature on the sphere, distributed through CCL http://www.ccl.net/.
  V.I. Lebedev, and D.N. Laikov "A quadrature formula for the sphere of the 131st algebraic order of accuracy" *Doklady Mathematics*, Vol. 59, No. 3, 1999, pp. 477-481.
- A GMRES solver available at http://www.cerfacs/algor/Softs.

  V. Frayssé, L. Giraud, S. Gratton, and J. Langou, A set of GMRES routines for real and complex arithmetics on high performance computers, CERFACS Technical Report TR/PA/03/3, 2003.

Papers on FAST.

Users of this code should include references [1–3] below in their publications, and in the user- and programmers'-manuals describing their codes.

1 P. Koval, D. Foerster, O. Coulaud. Fast construction of the Kohn–Sham response function for molecules, in "Physica Status Solidi B", February 2010, vol. 247, no 8, p. 18411848. [DOI : 10.1002/PSSB.200983811], http://hal.inria.fr/inria-00457652/en.

2 P. Koval, D. Foerster, O. Coulaud. A Parallel Iterative Method for Computing Molecular Absorption Spectra, in "Journal of Chemical Theory and Computation", 2010, vol. 6, no 9, p. 26542668 [DOI : 10.1021/CT100280X], http://hal.inria.fr/inria-00488048/en.

3 Olivier Coulaud, Patrice Bordat, Pierre Fayon, Vincent LeBris, Isabelle Baraille, Ross Brown. Extensions of the Siesta DFT code for simulation of molecules. http://hal.inria.fr/hal-00787088.

4 Inria research report 8221, included in the documentation of FAST.

2 Sample installation

This section illustrates configuration, installation and running the code on a Unix-like operating system, such as Linux or Mac OS X.

2.1 Prerequisites

Before installing FAST obtain and install the following :

- Intel Fortran and C++ compilers (Intel version 15.0 or higher), including the OpenMP, MKL and FFTW libraries.
- CMake version 3.5.
- The SIESTA XC exchange-correlation library (typically the SiestaXC sub-directory in the Obj of the SIESTA build tree).
- The GNU scientific library version 1.15 or higher is required.

N.B. Although FAST is a parallel code, all the above routines are used in **sequential** mode in the tddft code.

2.2 Configuration and compilation

The project is packaged under CMake (http://www.cmake.org). General information about CMake, as well as installation of binaries and CMake source code is available from the CMake homepage.

The first step is to get the source of FAST and uncompress it:

```
shell$ tar zxf FAST-1-1.tar.gz
shell$ cd fast
```

The second step is to configure the installation using CMake:

```
shell$ mkdir build
shell$ cd build/
```

shell$ CC=icc FC=ifort CXX=icpc cmake -DSIESTA_XC_DIR=/ABSOLUTEPATH/..../Obj/SiestaXC ../

where /ABSOLUTEPATH/..../Obj/SiestaXC is the absolute path of the SIESTA exchange-correlation directory.

Next compile the code:

shell$ make

shell$ cmake .

Before installing, adjust the installation directory (absolute path) in file cmake_install.cmake. Then install:

shell$ make install

Options can be activated by setting flags during compilation. Several options may be set to use cmake. These options define library paths used in the code and the type of implementation (sequential or parallel for example).

- **CMAKE_INSTALL_PREFIX:** should be set to the path of the installation directory (*i.e.* prefix).
- **FAST_OPENMP:** Activate (set ON, default: OFF), to compile the OpenMP version of FAST.
- **FAST_MPI:** Activate (set ON, default: OFF), to compile the MPI version of FAST. In this case MPI variables and functions are accessed through module `mpi` rather than *via* an include file. Moreover the compiler must be `mpif90`.
- **FAST_DISPLAY_PROD:** Activate (set ON, default: ON) to display all information about orbital products.
- **FAST_CHECK_NBITER_GMRES:** Activate (set ON, default: ON), to verify the number of iterations in the GMRES algorithm.
- **FAST_USE_MKL:** Activate (set ON, default: ON) to link FAST with the Intel MKL library. If disabled, linking to third-party librairies (BLAS, LAPACK and FFTW3) has to be set manually, namely :
  - **FAST_BLAS_PATH:** path to the BLAS library.
  - **FAST_BLAS_FLAGS:** flags needed to link with BLAS.
  - **FAST_LAPACK_PATH:** path to the LAPACK library.
  - **FAST_LAPACK_FLAGS:** flags needed to link with LAPACK.
  - **FAST_FFTW3_PATH:** path to the FFTW3 library.
  - **FAST_FFTW3_FLAGS:** flags needed to link with FFTW3.

  - **FAST_FFTW3_INCLUDE_PATH:** path to the include file for FFTW3.

- **FAST_SIESTA_COUPLING:** Activate (set ON, default: OFF) to enable coupling with SIESTA through mpicpl. Requires FAST_MPI options enabled.
- **FAST_PRINT_SPECTRUM_ITER:** Activate (set ON, default: OFF) to active print-out of spectrum and integral during each iteration.

2.3 Running FAST

FAST requires a parameter file and several pre-computed files from SIESTA. Several output files are produced. The simplest way to run the code is to put the SIESTA input files in an *input* directory and to create an *ouput* directory before running the code.

For example, on changing directory to one of the examples in the file tree installed by the last step above, the command line is

```
../../bin/fast input_file_name
```

where `input_file_name` is the name of the file of input parameters, *e.g.* water.inp, see the next section.

3 Detailed description of program options

This section describes all the parameters that define a FAST run, with their data types and default values.

3.1 General system descriptors

SystemLabel (*string*): A **single** word (max. 20 characters **without blanks**) containing a nickname of the system. This name must be **exactly** the same (*e.g.* case sensitive) as the generic name of all SIESTA input files (see input files section, 3.2).

iv (*integer*): Controls verbosity for debugging; default 0 (low); set to 1 or 2 for higher verbosity.

input_directory (*string*): The name of the directory where FAST reads all SIESTA files. Set to '.' to use the working directory.

Default: `input`

output_filename (*string*): Generic name (less than 120 characters) for all output files. For example 'calculation_case1/case1'. N.B. The directory should be created before runing FAST, otherwise the code crashes with error (under the Intel compiler) *forrtl: No such file or directory.*

Default: `output/str_SystemLabel` where str_SystemLabel is the string set in the SystemLabel keyword.

Default: `1`

io_units (*string*): Units of frequency for the spectra. Input of frequency limits will be interpreted according to this option. Output of absorption spectra depends on this option.

Allowed values : `Rydberg`, `cm-1`, `Angstrom`, `Hartree`, `nm`, `eV`.

Default: `Rydberg`

3.2 Input files

A FAST TDDFT calculation needs information on the system from the underlying DFT calculation, such as the geometry, the molecule wavefunctions, the pseudo potentials, the Hamiltonian and overlap matrices *etc.* When not compiled to be run coupled to SIESTA *via* MPICPL, the FAST program gathers this information from the .DIM (geometry), .WFSX (packed wavefunctions), .HSX(Hamiltonian,. . .) and .PLD (charge density) SIESTA output files. To generate these files the following two keywords must be present in the input file of SIESTA.

```
COOP.Write     .true.
WriteDenchar   .true.
```

The reader is referred to the SIESTA manual for further details.

3.3 Output files

`input_outputFileName`_polarizabilitySpectra.txt : The polarisability spectra. On each line : ω, $P(\omega)$ (unpolarised *i.e.* sum of components), the x, y and z components. Lines starting with a '#' show frequencies where there was a convergence problem or a negative polarisability.

`input_outputFileName`_dipol_core.txt : the dipole core.

`input_outputFileName`_numof_iter.txt : iterations needed to achieve convergence at each frequency.

`input_outputFileName`_OddFrequency.txt : frequencies where the GMRES had difficulty.

In additon, if the Lorentzian fit to the numerical spectrum is activated :

`input_outputFileName`_transitions.txt : transition energies and the oscillator strengths.

`input_outputFileName`_lorentzians.txt : spectrum and the fitted spectrum at the frequencies at which the polarisability was calculated.

`input_outputFileName`_modelledSpectrum.txt : fitted spectrum on a set of *uniformly* spaced points, not necessarily those where the spectrum was computed.

3.4 Base compression parameters

At the heart of the FAST program is the suppression or reduction of linear dependence in the space of atomic orbital products used to describe the linear response (see papers 1 & 2 above). Dominant products are defined by diagonalisation of the Coulomb metric in the space of products of atomic orbitals and retention of a sub-set of eigenvectors with the largest eigenvalues. Several parameters control this process. Default values below have produced good results on test systems, but users are encouraged to experiment.

eigmin_local (*real*): Defines the threshold for eigenvalues of the metric of orbital products on the same atom. Eigenvectors with smaller eigenvalues will be dropped in the computation of the sigma-matrices and spectrum.

Default: `1.0 E-3`

eigmin_bilocal (*real*): Defines the threshold for eigenvalues of the metric of orbital products on different atoms. Eigenvectors with smaller eigenvalues will be dropped in the computation of the sigma-matrix and spectrum.

Default: `1.0 E-4`

use_eigmin_local_H (*boolean*): If `.true.` use a special threshold for hydrogen local products.

Default: `.false.`

eigmin_local_H (*real*): Threshold for hydrogen local products.

Default: `eigmin_local/10`

use_rcut (*boolean*): If `.true.`, drop orbital products for pairs of atoms with small geometrical orbital overlap, using criterion r_{cut} set with the **rcut** keyword.

Default: `.false.`

rcut (*real*): Allows suppression of further products, based on the thickness of the lens shaped region defined by the intersection of the largest orbitals on a pair atoms A and B. All orbital products of a pair of atoms are dropped from the metric when

$$R_{max}(a) + R_{max}(b) - distance(a, b) < r_{cut},$$

where $R_{max}(a)$ (resp. b)is the radius of the support of all atomic orbitals on atom A (resp. B).

Default: `3.0U` where U is the unit of length for atomic positions the atoms stored in the siesta .DIM output file read by FAST.

write_all_eigens (*boolean*): If `.true.` write all eigenvalues for all pairs of atoms in separate files. File names are constructed according to the rules :

For species the file name is construct

```
input_outputFileName //'_eigen_' // species //".txt")
```

For atoms pairs (*e.g.* atoms n1 and n2) the eigenvalues are stored in file

```
input_outputFileName//'_eigen_' // n1n2,//".txt"
```

Default: `.false.`

3.5 Spectrum parameters

The basic task accomplished by FAST is to compute the absorption spectrum $P(\omega)$ of a molecular, *i.e.* finite system at a set of frequencies ω in some user-defined interval $[\omega_{\min}, \omega_{\max}]$. Such a spectrum is not in itself very useful since it is a set of discrete resonances with width and heights dependent on the regularisation parameter ϵ. Paper 3 therefore describes an adaptative algorithm to extract transition frequencies, Ω_I, and oscillator strengths, f_I, from the raw polarisability spectrum calculated by FAST. The algorithm proceeds by a two-tier iteration.

In the top level iteration, *(i)* the number of frequency points used (and polarisabilities, $P(\omega)$ computed, beware the cost in CPU time), is increased by a constant fraction; or *(ii)* the regularisation parameter ϵ is reduced by a step towards its target value; or *(iii)* both changes are made. In between top level iterations, the lower level iteration bears on the distribution of the frequency points. It uses the current spectrum to cluster the frequency points around the resonances, which improves the estimates of the a sum of Lorentzians to the spectrum.

Tier-two clustering is iterated until the resonance parameters Ω_I and f_I are stable, after which tier-one update of the number of frequency points and the regularisation parameter, or both, are updated, and the cycle continues until convergence of the resonances between iterations. A combination of keywords provided at the end of this section can be used to invoke only tier two iterations.

The following parameters control the frequencies where FAST computes the optical polarizability $P(\omega)$, and how the adaptive algorithm described in paper 3 extracts the transition frequencies and oscillator strengths.

omega_min (*real*): Lower bound of the frequency range.

Default: `0.02 in io_units`

omega_max (*real*): Upper bound of the frequency range.

Default: `0.4 in io_units`

N.B. The default values of **omega_min** and **omega_max** are suitable for the visible spectral region if **io_units** are Rydbergs.

eps_units (*string*): Units for the regularisation parameter ε.

Allowed values : `Rydberg`, `cm-1`, `Angstrom`, `Hartree`, `nm`, `eV`.

Default: `Rydberg`

freq_eps (*real*): Regularization parameter ϵ in the response function which defines the apparent width of the resonances, *cf.*papers 1–3.

A **positive** value of **freq_eps** is interpreted as the *target* value towards which ϵ will be decreased during tier-one iterations of the adaptive algorithm. Making it small helps to identify resonances and specially to distinguish close resonances. But note that the linear response problem solved here describes *undamped* resonances (no relaxation, coupling to phonons, *etc.*). Therefore, resonances are in principle infinite, whence the presence of the regularisation parameter in the equations, to 'pseudo-damp' them to finite height. Thus, if ϵ is too small, the linear system to be solved by GMRES tends to singularity when a frequency point happens to lie less than a few ϵ from a transition, producing numerical instabilities, such as failure of GMRES to converge, or unphysical solutions, such as negative polarisabilities.

When choosing **freq_eps**, users should bear in mind that the fit procedure pin-points transitions much more accurately than the width of the resonances (ϵ), in the authors' experience typically to a precision of at least $\epsilon/100$.

File `input_outputFileName`_OddFrequency.txt provides a list of all frequencies which gave rise to a singular matrix.

A **negative** value of **freq_eps** causes FAST to deduce this parameter from the number of frequencies according to

$$freq\_eps = \frac{3 * (omega\_max - omega\_min)}{2(nff - 1)}$$

.

Default: +0.01 `Rydberg`

nff (*integer*):

A **positive** value of **nff** defines the *initial* number of frequency points at which to compute the spectra. The initial distribution of points is uniform in the target window $[\omega_{min}, \omega_{max}]$:

$$\omega_i = omega\_min + \frac{omega\_max - omega\_min}{nff - 1}(i - 1) \text{ with } i = 1..nff$$

A **negative** value of **nff** causes the number of points to be controlled by the *target* regularisation parameter :

$$nff = 1 + \frac{3}{2freq\_eps}(omega\_max - omega\_min) \tag{1}$$

Default: +257

distribution (*string*): The type of distribution used to compute the spectra. `uniform` for uniform distribution, and `adaptive` for non uniform distribution. In this case, the points will be localized near the peaks in the spectrum

Default: `adaptative`

dist_hierachic_nbiter (*integer*): Maximum number of (tier-one plus tier-two) iterations when the adaptive algorithm is chosen.

Default: `5`

dist_hierachic_nbtransitions (*integer*): The number of transitions to atttempt to extract. If this number is too large, the algorithm returns the number of transitions actually found by the fit algorithm. Otherwise, the algorithm stops when convergence is reached on the first **dist_hierachic_nbtransitions** transitions, sorted by energy Ω_I or oscillator strength f_I (see also **fit_lorentzian_sort**).

Default: `1`

dist_hierachic_threshold (*real*): The threshold, η, to stop the adaptive algorithm. The criterion is to stop at tier-one iteration k if

$$\max_I \left(\max \|\Omega_I^k - \Omega_I^{k-1}\|, \max \|f_I^k - f_I^{k-1}\| \right) < \eta$$

Default: `0.01`

dist_hierachic_epsiter (*integer*): Number of equal steps to reach the final value of ϵ `freq_eps`. (see **dist_hierachic_epsstart**).

Default: 1

dist_hierachic_epsstart (*real*): Start regularization parameter. During tier-one iteration i, the regularization parameter ϵ is given by the relation

$$\epsilon = dist\_hierachic\_epsstart + \frac{(freq\_eps - dist\_hierachic\_epsstart)}{dist\_hierachic\_epsiter - 1}(i-1)$$

Default: 0.01 Rydberg

dist_hierachic_increaserate (*real*): Specifys the ratio by which to increase the number of points between tier-one iterations.

Default: `1.0`

***N.B.* Updating frequency points only:** Tier-two iterations, *i.e.* optimisation of a fixed number of points with one value of ϵ can be achieved by choosing : **nff** > 0 and **freq_eps** $=$ **dist_hierachic_epsstart** > 0. An appropriate value of **nff** would be half the value given by eqn. (1).

3.6 Solver parameters

The GMRES method is used to solve the following linear system for the polarisability, $P(\omega)$,

$$\begin{cases} P(\omega) = \sum_{i=1}^{3} < d_i, X_i(\omega) > \\ \\ (1 - \chi^0(\omega)\Sigma)\, X_i(\omega) = \chi^0(\omega) d_i, \qquad i = 1, 2, 3 \end{cases} \tag{2}$$

where d_i is the dipole in the i-direction, χ_0 is the susceptibility of the non-interacting Kohn-Sham system and Σ the interaction kernel, *cf.* papers 1–3. The parameters of the method are:

solver_krylov (*integer*): Maximum dimension of the Krylov space. This parameter is also called the restart parameter and it controls the amount of memory required by the matrix in the Krylov space.

Default: 20

solver_itermax (*integer*): Maximum number of iterations to reach convergence.

Default: `100`

solver_verbose (*integer*): Controls output of information on convergence of the solver. If the value is non-zero, output is written in file `fort.solver_verbose`.

Default: `0`

solver_eps (*real*): tolerance for convergence.

Default: `0.001`

3.7 Fit parameters

These parameters control the fit of the spectrum by a sum of normalised (unit area) Lorentzian resonances, used to extract transition frequencies and oscillator strengths. The spectrum is approximated as:

$$spectrum(\omega) = B + \sum_{k=1}^{N_{\text{trans}}} \frac{\epsilon}{\pi} \frac{A_k}{(\omega - \omega_k^0)^2 + \epsilon^2} \tag{3}$$

where N_{trans} is the number of Lorentzians, B a constant to capture the background due to resonances outside the computed interval, ω_k^0 the k^{th} transition energy and $f_k = \frac{2}{\pi} A_k \omega_k^0$ its oscillator strength (see paper 3).

fit_lorentzian_sort (*integer*): Specifies whether to sort transitions in order of energy (=1) or oscillator strength (=2).

Default: `2`

fit_lorentzian_nbpoints (*integer*): Controls whether a model spectrum (eqn. 3) is output. **fit_lorentzian_nbpoints** $>$ 1 specifies the number of points output to file `input_outputFileName`_modelledSpectrum.txt. No output if **fit_lorentzian_nbpoints** ≤ 1 . *Default:* `-1`

3.8 Potential parameters

These parameters control the calculation of the interaction kernel in the linear response equations.

use_hartree (*integer*): Include/exclude (`1`/`0`) the Hartree potential in the interaction kernel.
Default: `1`

use_exchange (*integer*): Include/exclude (`1/0`) the exchange potential in the interaction kernel.

Default: `1`

use_correlation (*integer*): Include/exclude (`1/0`) the correlation potential in the interaction kernel.

Default: `1`

xc_authors (*string*): The type of exchange-correlation potential (*cf.* the SIESTA manual), currently **LDA only**, see the introduction.

Default: `'PZ'`

xc_3d_order (*string*): Order of Lebedev grid used to evaluate the coefficient of the exchange-correlation interaction kernel. The order must be in the range `1 to 31.`

Default: `7`

xc_3d_nl (*string*): Order of Gauss-Legendre method used to evaluate the coefficients of the exchange-correlation matrix. The following order are available `6,12,24,48,96.`

Default: `24`

4 Coupling FAST to SIESTA *via* MPICPL

It should be clear that connecting two parallel computer codes such as FAST and SIESTA so that they will cooperate is not a trivial undertaking. Users of FAST who will be doing one-off or infrequent computations *e.g.* at a single geometry for different systems are best advised to keep to communication *via* SIESTA output files.

However, FAST can be coupled directly to SIESTA using the MPICPL (MPI Coupling) framework. MPICPL is dedicated to the coupling of parallel scientific codes, based on the well-known MPI standard. It is divided into several independent layers for coupling, data redistribution and steering. The codes to be coupled are launched and connections between them are set up by mpicpl, according to information derived from an xml file.

The direct coupling option might be attractive to perform a series of TDDFT calculations on the fly, for example during a SIESTA molecular dyanmics run. This is feasible once the range of frequencies and FAST parameters have been tuned to extract resonances in the desired frequency window, *e.g.* usually, experimental data will be available only for the first one or two excited states. Such repetitive calculations are handled on the SIESTA side of the coupling by a set of patches, in which a call to FAST, communicating *via* MPICPL is inserted in the 'move' loop of the SIESTA main program.

MPICPL is downloadable at https://gforge.inria.fr/projects/mpicpl/.

The SIESTA patches are available at https://gforge.inria.fr/frs/?group_id=1179 .

4.1 Compilation

To activate the coupling with SIESTAthe **FAST_MPI** option must be set to **ON**, then set **FAST_SIESTA_COUPLING** option to **ON**. After that the include directory of MPICPLmust be set in **MPICPL_INCLUDE_DIR** variable.

To construct the executable `tddft-qm` code, just do

```
make clean tddft-qm
```

4.2 Coupling parameters

Parameters below are available only if FAST was compiled with `NOSSI_COUPLING` defined.

nossi_qm_coupling (*boolean*): Flag to tell if the code is to attempt to connect itself to a running SIESTA computation.

Default: `.false.`

nossi_qm_binding_name (*string*): Name of the link between the FAST code and the SIESTA code. The name of this link must be the same as the in the xml file read by the mpicpl wrapper when it launches both codes.

Default: `'tddft-dft'`

nossi_code_name (*string*): Name of the FAST code in the xml file. See next section.

Default: `'tddft-nossi'`

4.3 Example

In order to use MPICPL the coupling to be set up must be described in an xml file (see mpicpl documentation). Here is an example for coupling SIESTA and FAST(siesta-fast.xml).

```
<coupling arguments="nossi_top">
  <!-- declare all codes -->
    <code np="1"
        cwd="${nossi_top}/Examples/tddft-qm/h2o/siesta"
     program="${nossi_top}/siesta/epsn/siesta.sh"
      name="siestaNOSSI"
       args="h2o_tddft.fdf" />
  <code np="1"
name="tddft_nossi"
cwd="${nossi_top}/Examples/tddft-qm/h2o/tddft"
program="${nossi_top}/fast/tddft_qm"
args=""/>

<!-- interconnect codes -->
   <binding name="tddft-dft" client="tddft_qm" server="siestaNOSSI" />

</coupling>
```

The coupled programs are launched by

```
$MPICPL_TOP/tools/mpicplrun -d ./nossi-tddft-dft.xml  $NOSSI_TOP
```

5 Troubleshooting

1. **Unphysical polarisability at some frequency points**: As explained above, linear response TDDFT is caught on the horns of a dilemma, between using a small enough regularisation parameter to make individual resonances visible, and not choosing it so small that the linear system defining the polarisability is singular. If a frequency point happens to lie too close to one of the resonances of the system, the matrix is numerically singular and GMRES may fail to converge, or return unphysical results such as a negative polarisability.

   Since such singularities point out an underlying resonance, it may be useful to increase the regularisation parameter ϵ (**freq_eps**), or to shift the bounds of the frequency interval, **omega_min**, **omega_max**, or to reduce the number of frequency points.

   The question of numerical instability is in part a question of computer word length. Considering that the fit procedure with $\epsilon = 10^{-3}\,\mathrm{Ry}$ typically extracts resonances to an accuracy of $10^{-5}\,\mathrm{Ry}$, way below experimental error in most practical situations and vastly below the inherent accuracy attainable with any flavour of DFT, FAST is written in single precision arithmetic, because that saves a factor of about two on the execution time.