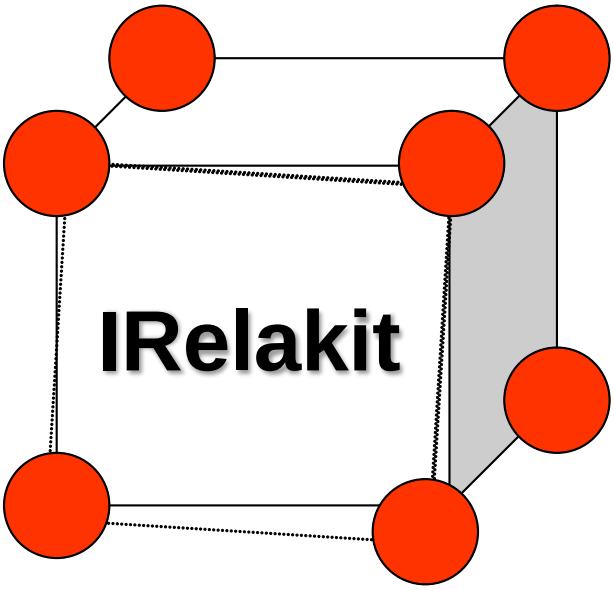
IRelakit Package

By Morteza Jamal

IRelakit enables the representation of elastic properties in both 3D and 2D visual formats, making them easier to visualize.

[01/07/2026]



IRelakit

+

WIEN2k

A Package for representation of elastic properties in both 3D and 2D visual formats

User's guide, IRelakit_26.1 (Release 04.02.2026)

Morteza Jamal

Ghods City-Tehran-Iran

MANDATORY CONDITIONS:

In any publication in the scientific literature please reference the program as follows:

M. Jamal, IRelakit, http://www.wien2k.at/ (2026).

ACKNOWLEDGMENT

For suggestions or bug reports please contact the author by email:

m_jamal57@yahoo.com

IRelakit Overview

IRelakit enables the representation of elastic properties in both 3D and 2D visual formats, making them easier to visualize. This allows for the conversion of the elastic stiffness matrix into a function of spherical coordinates (either 3D or 2D representations). To achieve this, the system transitions to spherical coordinates, where the unit vectors are perpendicular, similar to principal axes. To facilitate this, IRelakit defines two unit spherical vectors, **a** and **b**, in the new basis set and transitions from the original basis set to the new one in spherical coordinates. In this way, IRelakit connects elastic properties with spherical coordinates as functions of **θ**, **φ**, and **γ**, making them visualizable.

Program Structure

The IRelakit package consists of several scripts, each with a specific purpose:

- **IRelakit**: This is the primary script responsible for preparing the entire calculation. It must be executed in a directory that contains a valid elastic constants matrix (ELC-matrix file) and mass density (.rho file). Alternatively, the program can calculate the mass density of the compound based on the number of atoms and the unit cell volume. The script takes in the number of mesh points for **θ** and **φ**, as well as the Miller index for the 2D cut. The mesh number controls the scanning loops during which elastic properties are calculated, and it determines the grid used for graphical representations. From this, the program calculates the optimum values and averages. It creates forth main types of files:
  - `*.dat` (e.g., Shear-Rati.dat)
  - `*2D.dat` (e.g., Poisson-Rati2D.dat)
  - `*2DB.dat` (e.g., Young-Modu2DB.dat)
  - `ThPhCalP.dat` file

The `*.dat` and `*2D.dat` files are used for 3D and 2D-cut plots, respectively. The `*2DB.dat` file is used for plotting the convex hull (edge border) of the 2D-cut, while the `ThPhCalP.dat` file is for generating color/gray maps and heat maps using the Gnuplot program. At the end of the process, the calculated parameters are stored in the `IRelakit.dat` file.

If the convex hull is not sufficiently accurate, the `-d/-D ds` option (default value of `ds = 4 degrees`) can be used to improve the convex hull by reducing the polar angle rotation parameter (for example: IRelakit -d 3). For further details on how to use the program, execute the following:

```
IRelakit -h
```

- **xyz2wrl**: This Fortran-based program is used to generate 3D representations of elastic properties in the **wrl** format. This format can be visualized in a VRML browser, such as the well-known **View3dscene** program. It reads data from either `*.dat` (for 3D) or `*2D.dat`

(for 2D-cut) files, depending on the input parameters. The output file will be in the **wrl** format (e.g., `*.wrl` or `*2D.wrl`). For more details on how to use the program, execute:

```
xyz2wrl -h
```

- **xyz2x3d**: This Fortran-based program is used to generate 3D representations of elastic properties in the **x3d** format. This format can be visualized in a VRML browser, such as the well-known **View3dscene** program. It reads data from either `*.dat` (for 3D) or `*2D.dat` (for 2D-cut) files, depending on the input parameters. The output file will be in the **x3d** format (e.g., `*.x3d` or `*2D.x3d`). For more details on how to use the program, execute:

```
xyz2x3d -h
```

- **Heatmap_lapw**: This shell script is used for generating palette-mapped 3D color/gray maps or 2D polar heat maps (for 2D-cut plots) in which each "pixel" corresponds to a pre-determined range of **θ** and **r** for the elastic properties. The program uses Gnuplot for plotting. It reads from the `ThPhCalP.dat` or `*2DB.dat` files and outputs **png** images (e.g., `*HM.png` or `*HM2DC.png`). For further details, run:

```
Heatmap_lapw -h
```

- **Polarplot_lapw**: This script provides an interface for plotting 2D-cut elastic properties (convex hull) as polar or Cartesian coordinates using Gnuplot. It uses data from `*2DB.dat` and generates output images such as `*2DB.png` (r vs. **θ**) for polar coordinates, and `*2DBrxy.png` (-rxy option, r vs. **θ**) as well as `*2DBxy.png` (-xy option, ry vs. rx) for Cartesian coordinates. For more information on options, use:

```
Polarplot_lapw -h
```

- **xyz2gnu**: This program is an interface for plotting elastic properties in 3D (x, y, z) or as **Pm3d** plots, including monochrome options (`-mc`). The `-mv` option can be used to project the 3D plot onto a 2D map view, effectively flattening the Z coordinate onto the XY plane. The program reads data from the `ThPhCalP.dat` file and outputs **png** or **pdf** images. For details, run:

```
xyz2gnu -h
```

- **IRelakit_lapw**: This shell script automates the process of calling all the necessary programs (IRelakit, xyz2wrl, xyz2gnu, Heatmap_lapw, and Polarplot_lapw) for all elastic properties. It organizes the output files in appropriate subdirectories, generating **png** and **wrl** formatted output.

IRelakit Installation Guide

1. Makefile Configuration

Before building IRelakit, the user should edit the Makefile to match the target Linux environment. The appropriate Fortran compiler, compiler flags, and loader (linker) options must be specified according to the available compiler and system configuration. Once the Makefile has been updated, the program can be compiled by executing the **make** command.

The following parameters must be specified:

Fortran Compiler

Specify the Fortran compiler used for compilation.

Examples:

```
FC = gfortran
```

or

```
FC = ifort
```

or (Intel oneAPI)

```
FC = ifx
```

Fortran Compiler Flags

Set the compiler options and optimization flags according to the selected compiler and system requirements.

Examples:

```
FFLAGS = -O3
```

or

```
FFLAGS = -O3 -Wall
```

or

```
FFLAGS = -O2 -fopenmp
```

or for debugging:

```
FFLAGS = -O2 -g
```

Loader / Linker Options

Specify linker options and external libraries required during the linking stage.

Example without external libraries:

```
LDFLAGS =
```

Example using mathematical libraries:

```
LDFLAGS = -llapack -lblas
```

Example with OpenMP support:

```
LDFLAGS = -fopenmp
```

Example from my WIEN2K-OPTIONS:

```
#---------------- compiler -----------------
FC = ifort
#------------- compiler options ------------
FOPT = -O -FR -mp1 -w -prec_div -pc80 -pad -ip -DINTEL_VML -traceback -
assume buffered_io -I$(MKLROOT)/include
#-------------- loader options -------------
LDFLAGS = $(FOPT) -L$(MKLROOT)/lib/$(MKL_TARGET_ARCH) -pthread
R_LIBS =  -lmkl_lapack95_lp64 -lmkl_intel_lp64 -lmkl_intel_thread -
lmkl_core -openmp -lpthread
```

After modifying the Makefile, compile the program:

```
make
```

This will build the IRelakit executable for the selected Linux environment.

---

2. Environment Variables

To use IRelakit conveniently from any directory, define the installation path in the Linux shell startup file:

```
~/.bashrc
```

Assume that IRelakit is installed in:

```
/home/user/IRelakit
```

Add the following lines to the end of the `~/.bashrc` file, if the executable file is located directly in the main directory:

```
export IRELAKIT=/home/user/IRelakit
export PATH=$IRELAKIT:$PATH
```

You can verify that the environment variable has been defined correctly with (Activate the environment):

```
source ~/.bashrc
```

Check the installation path as:

```
echo $IRELAKIT
```

If the correct path is displayed, the environment configuration is complete.

References

[1] Jan W. Jaeken,Stefaan Cottenier, Comp. Phys. Comm. 207 (2016) 445.

[2] Nye, J. F., Physical properties of crystals: their representation by tensors and matrices OXFORD Science publications. (1985)

[3] Arnaud Marmier, Zoe A.D. Lethbridge, Richard I. Walton, Christopher W. Smith, Stephen, C. Parker and Kenneth E. Evans, Comp. Phys. Comm. 181 (2010) 2102.

[4] Shahram Yalameha, Zahra Nourbakhsh, and Daryoosh Vashaee, Comp. Phys. Comm. 271 (2022) 108195.