Short description of SimPack

Wolfram Hergert

27. Januar 2023

The group theory package GTPack contains a sub-package `SimPack.m`. This package organizes the connection to the external tight-binding SimPack programs. A detailed introduction to GTPack is given in [7].

If you have questions or you detect errors or inconsistencies,oooooo please contact: `wolfram.hergert@physik.uni-halle.de`

Inhaltsverzeichnis

1 Distribution

This distribution contains the following subdirectories:

- `doc`:
  Contains this documentation.
- `example`:
  Contains a simple example, mainly to explain the GTPack commands.
- `methods`:
  Contains in subdirectories tests of the genetic algorithm (directory `Pikaia`) and of the simplex method (directory `Nelder_Mead`). Those methods are explained in more detail in contrast to the least squares method.

  The directories contain subdirectories `docu, exe, math, src` and `test`. In `docu` the documentation and publications to the method are stored. `src` contains the source code. In `exe` executables will be stored that can be generated by a `makefile`. `test` contains the testdata. If available, `math` contains Notebooks to evaluate the tests.
- `applications`:
  In this directory applications for three Hamiltonians are given: simple TB s-band in fcc structure (directory `sb`), Hamiltonian with d-bands only for Cu (directory `Cu_d`) and finally the full Cu Hamiltonian (directory `Cu_spd`).

  All directories contain subdirectories for band structure calculation (`bands`) and calculation of densities of states (`dos`). Executables, generated by means of the `makefile`, are stored in directory `exe`. The results of the application of three fitting methods are stored in subdirectories of `fit`. Finally `math` contains data and Notbooks to evaluate the results by means of GTPack.
- `math`:
  The directory contains some data to be used by means of GTPack.
- `src`:
  This directory contains in several subdirectories the FORTRAN source code.

The main directory contains also the `makefile`. (cf. 4.1)

2 Introduction

Previous releases of GTPack contain already modules for the export of tight-binding Hamiltonians, constructed by GTPack, in FORTRAN form. Those modules are presented in Tab. 1.

Tab. 1: Commands to generate FORTRAN code in GTPack.

Module	*Task*
`GTTbToFortran`	exports Hamiltonian as a FORTRAN module
`GTTbRSToFortran`	transformation of a real space tight-binding Hamiltonian in a FORTRAN module
`GTTbToFortranList`	prints a tight-binding Hamiltonian as FORTRAN code

For details see the corresponding documentation pages of GTPack. Additionally we present now a suite of simple FORTRAN programs which can be used for band structure calculations, calculation of densities of states and fitting of TB parameters to *ab initio* band structure data. This document contains a short introduction to the codes.

3 The sub-package `SimPack.m`

This sub-package of GTPack contains modules for the interaction with the FORTRAN programs. On the one hand, data, generated by GTPack will be exported to be used in the FORTRAN code. On the other hand, results of the FORTRAN calculations can be read into GTPack to be used in models and for further calculations. Tab. 2 gives an overview.

Tab. 2: Commands to connect GTPack and SimPack.

Module	*Task*
`GTTbParmExport`	Export of TB-parameters to FORTRAN
`GTTBParmImport`	Import of a parameter set from FORTRAN
`GTTbFitExport`	Export of band structure to be used mainly in fit tests
`GTTbReadBands`	Read band structures
`GTTbReadDOS`	Read density of states
`GTBZTbPointMesh`	export of **k**-meshes

Documentation pages are available for all commands in the GTPack documentation.

3.1 A simple example

The directory `example` contains a complete example to explain the use of the above mentioned GTPack commands. Details can be found in the Notebook `fcc_s_band.nb`. All the necessary input files for the FORTRAN calculations are provided in this directory. Simply follow the remarks in the Notebook. You can perform the calculations easily by yourself. The example is about a simple s-band in fcc-structure. Fitting strategies for this simple examples are discussed in section 6.1.

4 The FORTRAN programs

4.1 Makefile

The `makefile` is prepared to generate all executables that you need for calculations. Calling `make` or `make help` shows all possibilities to generate executables. You will find an option to fit TB-parameters with various methods, to calculate the band structures or densities of states.

In `makefile` the correct Hamiltonian will be set with the variable `HAMIL`. The variable `DIREC` defines the directory, where the executable will be stored. `NAME` adds some extension to the name of the executable. This can be used to distinguish executables.

You can switch in the moment between `gfortran` and the `ifort` compiler by (out)commenting the corresponding lines (Variables `F90C`, `FFLAGS`, `LFLAGS`). Both versions work well. Note, this part of the `makefile` depends on the installations on your computer. All was tested on Macs. The GNU FORTRAN-Compiler was installed as part of `gcc` by means of `Homebrew` or `MacPorts`. `ifort` was installed as part of INTELs oneAPI Toolkits.

Of course you can try your favorite FORTRAN compiler. Note, some parts of the codes are constructed with respect to older FORTRAN versions. Be careful!

4.2 The different FORTRAN codes

Most of the FORTRAN programs contain a RoboDoc-like header which gives some details about the program.[1] Only the main programs will be discussed here. Input and output of the programs are controlled by tokens. The data sets consist of a list of such tokens which has to be finished by the special token: `FINAL_CARD`. In general you can add comment lines as much as you want after `FINAL_CARD`. The tokens will be listed with some explanations.

Be careful with empty space! The tokens have to be written right-aligned in the first 10 positions of the corresponding „card“. The eleventh position has to be empty space on all cards.

Complete examples will be discussed in Sec. 6.

4.2.1 Band structure calculation

As it is shown in Tab. 3, some tokens expect data, others not. In the Table *In* means a file name for input of data is expected and *Out* stands or output of the results. *Data* marks that the Token contains some data items. In general all file name tokens contain as argument the corresponding file name. Some additional remarks are given here:

- `VERBOSE` - should be used as the first card. An `Integer` defines the verbosity level. The additional information is stored in `verbose.bnd`. If a verbosity level `iverb`

[1] RoboDoc itself is not applied to the package. The structure of the headers is used only to provide a unified form of description in all codes.

is defined the tokens are stored in that file, otherwise the tokens are printed on standard output. Choose `iverb=1` to get an output of a list of **k**-vectors. With `iverb=2` a list of calculated eigenvalues is printed additionally. This information is printed in `verbose.bnd` to keep the general protocol of the calculation compact.

- `DOS_CALC - (GAUSS, TETRA)`
  This token decides, how the band structure data for DOS calculations are calculated. If the token is `GAUSS`, Gaussian smearing is used and an external file with **k**-vectors is expected. You can generate the **k**-mesh by means of GTPack. `TETRA` means that a version of the tetrahedron method will be used and the **k**-vectors are calculated internally.
- `KMESH_GEN - ist, nt, rgx`
  This has to be used, if the method for DOS calculation is `TETRA`. This version of the tetrahedron method is based on [8][2] `ist`=1, 2, 3 stands for sc, fcc, bcc lattice. `nteil` defines the subdivision of the mesh. The number of **k**-points in the mesh will be calculated from `nt`. The following formulas give the number of **k**-points:

$$\begin{aligned} \text{sc} &: nkp = nt(nt+1)(2+nt)/6 \\ \text{fcc} &: nkp = 9nt^2 + 1 + 2nt(8nt^2+7)/3 \\ \text{bcc} &: nkp = (nt+1)(nt+2)(2nt+3)/6 \end{aligned}$$

  Calculated for some values of nt we get

N	1	2	3	4	5	6	7
sc	1	4	10	20	35	56	84
fcc	20	89	240	505	916	1505	2304
bcc	5	14	30	55	91	140	204

Tab. 3: Tokens for band structure calculation.

Token	*I/O/D*	*Content*
file names		
`F_ETB_PARM`	In	Parameters for the TB-Hamiltonian
`F_KVECTORS`	In	**k**-vectors calculated in GTPack
`F_BNDSTRUC`	Out	band structure
`F_PROTOCOL`	Out	protocol of the calculation
`F_MATELM`	Out	weight factors for DOS calculation
other control data		
`PROBLEMTXT`	Data	title for the calculation
`VERBOSE`	Data	controls verbosity
`DOS_CALC`	Data	method for DOS calculation
`KMESH_GEN`	Data	internal generation of **k**-mesh
Continue on next page		

[2]This method works only for fcc, bcc and sc lattice.

Continued from previous page		
Token	*I/O/D*	*Content*
HAM_DIMENS	Data	dimension of the Hamiltonian
FINAL_CARD		close input data set

4.2.2 Calculation of density of states (DOS)

The tokens for the DOS calculations are given in Tab. 4

- VERBOSE - should be used as the first card, an Integer defines the verbosity level.
- DOS_METHOD
  If it is GAUSS, smearing with a GAUSS function is performed, the width of the Gaussian is given by SMEARING. Furthermore, a version of the tetrahedron method (TETRA) can be used for cubic structures. Some details for DOS calculations are explained in section 6.3.3.
- SMEARNING - width σ of the Gaussian in the smearing method
- N_ELECTRON - number of electrons for the DOS calculation, used to calculate the FERMI energy.
- ENERGY_AXIS - emin,de,emax
  minimum energy emin, step with de, and maximum energy emax of the energy window.
- PART_DOS - partial DOS
  We have to specify, which partial DOS we want to calculate. If we want to calculate the partial s, p, t_{2g} and e_g DOSes for Cu the input for the token PART_DOS would be:

```
PART_DOS    4
            's'     1
                    1
            'p'     3
                    2  3  4
            't2g'   3
                    5  6  8
            'eg'    2
                    7  9
```

  The first line tells, how many partial DOSes will be calculated. For each partial DOS follows the name and how many orbitals will contribute. The next line specifies the position of the orbitals in the model Hamiltonian.

Tab. 4: Tokens for DOS calculations.

Token	*I/O/D*	*Content*
file names		
F_BNDSTRUC	In	band structure
F_PROTOCOL	Out	protocol of the calculation
F_MATELM	Out	weight factors for DOS calculation
F_DENSITY	Out	Density of states
other control data		
PROBLEMTXT	Data	title for the calculation
VERBOSE	Data	controls verbosity
N_ELECTRON	Data	number of electrons
ENERGY_AXIS	Data	definition of the energy axis
BANDS_NUMB	Data	number of bands
FERMI_ENE	Data	FERMI energy
SMEARING	Data	smearing for DOS calculation
DOS_METHOD	Data	method DOS calculation
PART_DOS	Data	definition partial DOS
FINAL_CARD		close input data set

4.2.3 Fitting of *ab initio* band structures with TB models

Three different fitting methods are implemented. A genetic algorithm search or a simplex search could be used in a first step to avoid a quick stop in a local minimum. Such a fit could be followed by a least squares fit.

The input dataset contains data that are common for all methods, but also specific control data for the methods. The Tab. 5 contains the input, common for all methods.

Tab. 5: Tokens commmon for all fit methods.

Token	*I/O/D*	*Content*
file names		
F_ABINITO	In	*ab inito* or test band structure data
F_START	In	start values of TB-parameters for fitting
F_FIXPARM	In	Fix some parameter during fit
F_FINAL	Out	fitted TB-parameters
F_FITCONT	Out	fitted TB-parameters for restart of fit
F_PROTOCOL	Out	protocol of the calculation
other control data		
PROBLEMTXT	Data	title for the calculation
VERBOSE	Data	controls verbosity
FIT_METHOD		fit method: LEASTSQ, GENALG, SIMPLEX
DATA_FORM	Data	control data structure STANDARD or VASP
HAM_DIMENS	Data	dimension of the Hamiltonian
Continue on next page		

Continued from previous page		
Token	*I/O/D*	*Content*
WEIGHT_BND	Data	weights for bands
FINAL_CARD		close input data set

The general tokens correspond mainly to those, given for band structure and DOS calculations.

DATA_FORM controls the structure of the input data. STANDARD is used, if the data set is output of GTPack. Accordingly VASP is used to read data from the VASP code. More methods might be implemented in tb_abinitio.f90.

It is interesting to use the fitting methods successively. You could start with a genetic algorithm followed by the least squares method. The token F_FITCONT is used to store the results of the fit in a form, that can be used directly in the next fitting step.

Customize your fitting procedure

If a complex band structure is given, it might be meaningful to include only certain bands in the fit. By means of the token WEIGHT_BND a weight can be defined for each band separately. If the token is not defined, all bands are included with weight factor 1.0.

Tab. 6: Special tokens for the least squares method.

Token	*I/O/D*	*Content*
file names		
DNLQ1_EPS	In	bound for relative change of x
DNLQ1_EGR	In	minimal value for gradient
	Out	norm of gradient
DNLQ1_EFX	In	bound for defect
	Out	final value for defect
DNLQ1_ITM	In	maximum number of iterations
	Out	number of iterations
DNLQ1_IPR	In	control of print (0- no print, 1- start and final values)
	Out	number of calls of F
DNLQ1_IST	In	storage control
	Out	error indicator

By means of F_FIXPARM a file is defined that controls the handling of the parameters. After a title lines occur containing in INTEGER and a REAL number. The INTEGER is the parameter number and the REAL number to which the parameter should be fixed. If the INTEGER is negative that parameter is fixed to the value in the parameter set defined by F_START, i.e. the REAL number is meaningless in this case.

Least squares

The least squares fit is a well known method. Some details can be found in [11]. We use an old routine from the 1970's [12]. Tab. 6 lists the tokens for this method. The header of the source code in dnlq1.f90 contains more information.

Genetic algorithm

The genetic algorithm part is controlled by the the array `ctrl(12)`. The data are loaded in the array by means of a set of tokens. Most of the values have a default value. The default values are already set, i.e. only those values you want to modify, must be given in the input.

Token	*default*	*Content*
		file names
GEN_INDIV	100	number of individuals in population
GEN_GENER	500	number of generation over which solution is to evolve
GEN_NUMB	6	number of significant digits
GEN_CROSS	0.85	crossover probability
GEN_MUTMOD	2	mutation mode: 1/2= steady/variable
GEN_MUTINI	0.005	initial mutation mode
GEN_MUTMIN	0.0005	minimum mutation rate
GEN_MUTMAX	0.25	maximum mutation mode
GEN_RELFIT	1.	relative fitness differential
GEN_REPROD	3	reproduction plan: 1/2/3= full generational replacement/ steady-state-replace-random/steady-state-replace-worst
GEN_ELITIS	0	elitism flag: 0/1=off/on
GEN_PRINT	0	printed output: 0/1/2=None/Minimal/Verbose
GEN_XMIN	-	minimal value scaling
GEN_XMAX	-	maximal value scaling
GEN_SCALE	-	list of scaling values
GEN_SEED	-	seed for random number generator

The algorithm works with variables $x_k \in [0.0, 1.0]$. Thus a scaling to this interval is necessary. Two methods are provided. Defining `GEN_XMIN` and `GEN_XMAX` allows to scale according to

$$X_k = \frac{x_k - x_{\texttt{min}}}{x_{\texttt{max}} - x_{\texttt{min}}} \ . \tag{1}$$

By means of the token `GEN_SCALE` a list of scaling factors can be defined.

The source code `pikaia.f` does not contain a RoboDoc-like header header, but more explanations to the control array may be found there. Detailed information can be found in [4, 2, 3, 1]

Simplex method

The downhill simplex method was developed by Nelder and Mead [9]. Derivatives are not used in this method. A *simplex* is a geometrical object consisting of $N + 1$ points in N dimensions. The number of parameters to be found defines the dimension N of the space. A starting approximation defines a point $\mathbf{P}_0$ in this space. The other N are defined as

$$\mathbf{P}_i = \mathbf{P}_0 + \lambda_i \mathbf{e}_i \,, \quad i = 1, \ldots, N \ . \tag{2}$$

The token SIM_STEP is used to define the λ_i. The $\mathbf{e}_i$ are N unit vectors. A detailed explanation may be found in [11].

The tokens for this method are summarized in Tab. 7. The default values will be automatically set, i.e. it is only necessary the include those tokens in the input to be redefined.

Also the source code asa047.f90 does not contain a RoboDoc-like header, but more explanations can be found at the beginning of this file.

Tab. 7: Special tokens for the simplex method.

Token	*default*	*Content*
		file names
SIM_STEP	1.0	initial step sizes
SIM_MAXFN	max*NPARM	maximum number of function evaluations
SIM_IPRINT	-1	print control: -1/0/1 = no printing/parameters and function values after convergence/additional progress report
SIM_STOP	-	stopping criterion
SIM_NLOOP	2*NPARM	stopping rule applied after NLOOP function evaluations
SIM_IQUAD	1	1/0 =fitting for quadratic surface/ not
SIM_SIMP	-	criterion expansion simplex
ifault	out	success of calculation
	0	successful calculation
	1	max. number of function evaluation exceeded
	2	information matrix not +VE semi-definite
	3	number of parameters <1
	4	NLOOP < 1

5 Test for simplex method and genetic algorithm

Before we go to examples in connection to GTPack, examples from the distributions of the simplex method and the genetic algorithm are presented as a reference here. The subdirectories Pikaia and Nelder_Mead in methods contain the corresponding tests. Both subdirectories contain a makefile to generate the executables, in docu the documentation is given and in test the input files for the tests the are stored. Executables will be stored in exe.

The complete set of tests of the documentations is discussed here, because the application of the simplex method or of genetic algorithms seems to be sometimes tricky. So you can find out by yourself, where a possible problem might be.

5.1 Simplex method according to Nelder-Mead

The tests of this method are summarized in directory methods/Nelder_Mead. The subdirectory src/orig contains the original code according to [9, 10]. In this subdirectory

you will find a test program `asa047_test.f90` for the subroutine `nelmin` in `asa047.f90` together with a text file `asa047_test.txt` containing the results of the tests.

To perform the tests as close as possible to the later use in the TB fitting procedure a new test program is given in `src` (`nm_test.f90`) The original test program is split in parts and the original test functions are used.

The executables for the original test and also the new test will be generated by a `makefile`.

The following tests are proposed to find a minimum (cf. [5, 10, 9][3])

1. **Rosenbrocks parabolic valley**

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{3}$$

   Obviously we get $f(1.0, 1.0) = 0.0$. For the standard test a starting point $\mathbf{x}_s = (-1.2, 1.0)$ is used.

2. **Powells quartic function**

$$f(\mathbf{x}) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4 \tag{4}$$

   For the standard test $\mathbf{x}_s = (3, -1, 0, 1)$ is used at the beginning.

3. **Fletcher and Powells helical valley**

$$\begin{aligned} f(\mathbf{x}) &= 100\big[x_3 - 10\,\theta(x_1, x_2)\big]^2 + \big[\sqrt{x_1^2 + x_2^2} - 1\big]^2 + x_3^2 \\ 2\pi\theta(x_1, x_2) &= \begin{cases} \operatorname{atan}(x_2/x_1) & x_1 > 0 \\ \pi + \operatorname{atan}(x_2/x_1) & x_1 < 0 \end{cases} \end{aligned} \tag{5}$$

   For this test the starting point $\mathbf{x}_s = (-1, 0, 0)$ is used.

4. **Quartic function**

$$f(\mathbf{x}) = \sum_{i=1}^{10} x_i^4 \tag{6}$$

   The starting point $\mathbf{x}_s \equiv (1, 1, \ldots, 1)$ was used.

Input files and results can be found in the subdirectory `test`. Files `#.in` are input files that use the values of the original test program. The input file `rosen1.in` together with `rosen1.ip` demonstrates the use of modified test parameters. The file `asa047.txt` is the output of a test run with the compiled original test program.

The results are as follows:

- The original test program leads to exact the same output as delivered with the code.
- The new test program leads also to the same test results.

[3]The corresponding pdf files can be found in `docu`.

5.2 Pikaia

`Pikaia` will be used to apply a genetic algorithm (GA) to get TB parameters from a fit to a given band structure. The directory `methods/Pikaia` contains the tests of the method given in the documentation. In subdirectory `docu` you will find the relevant information [2, 4, 3, 1].[4] The `makefile` can be used to generate executables for the calculations.

The subdirectory `math` contains the notebook `pikaia.nb`, used to generate test data and the figures for this section.

As for the simplex method, also for `Picaia` the original test programs have been modified. The original test codes can be found in `src/orig`. An input subprogram is used to modify the standard control variables of the test (`ga_input.f90`). If the input file does not contain modifications of the control variables, the standard test will be used.

Encoding, decoding and scaling of parameters

Encoding and decoding is described in section 3.5 of the User's guide [4]. Most important is, that the algorithm performs the encoding as

$$x_k \in [0.0, 1.0] \quad \Rightarrow X_k = (X_1, X_2, \ldots, X_{nd})_k \tag{7}$$

The number of digits in encoding/decoding is nd. The vector of variable $\mathbf{x} \equiv (x_1, x_2, \ldots, x_n)$ used to minimize $f(\mathbf{x})$ has to be scaled accordingly. The scaling factors are defined in files `#.scl`.

5.2.1 Maximizing a function of two variables

This test can be found in subdirectory `test/fit0`. The function is

$$f(x,y) = \cos^2(n\pi r)e^{(-r^2/\sigma^2)}\,,\; r^2 = (x-0.5)^2 + (y-0.5)^2\,,\; x,y \in [0.0,1.0]\;. \tag{8}$$

Fig. 1 gives an impression of the function. The figure reveals, that the maximum cannot be easily found by a least squares method. The test with the GA demonstrates, that the correct maximum at $\mathbf{x}$=(0.5,0.5) will be found.

5.2.2 A linear least square fit

The function used for tests of linear and non-linear least square fits is given by:

$$f(t,a,b,\mathbf{A},\mathbf{P},\mathbf{s}) = at + b + \sum_{m=1}^{M} A_m \sin\left[2\pi\left(\frac{t}{P_m} + s_m\right)\right]\;. \tag{9}$$

The linear test can be found in subdirectory `test/fit1a`. Fig. 2 shows function (9) with added noise. Normally distributed noise with $\sigma = 5$, as proposed in the description, is

[4] See also the Pikaia page.

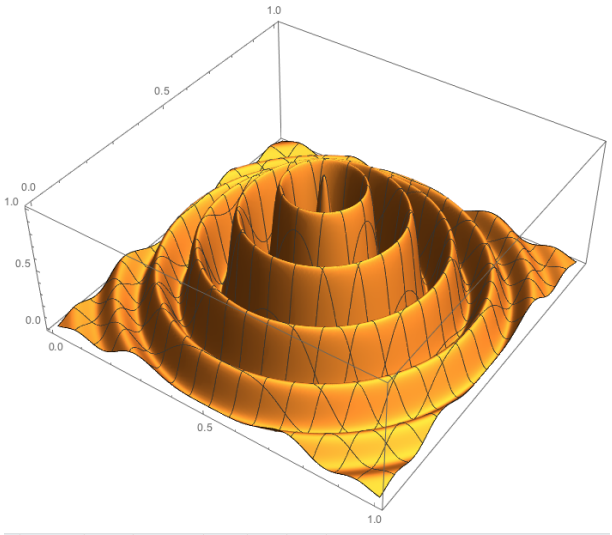


Fig. 1: Function $f(x, y)$ according to (8) for $n = 9, \sigma^2 = 0.15$.

used. To generate `test1a.dat` the same parameters as given in the documentation are

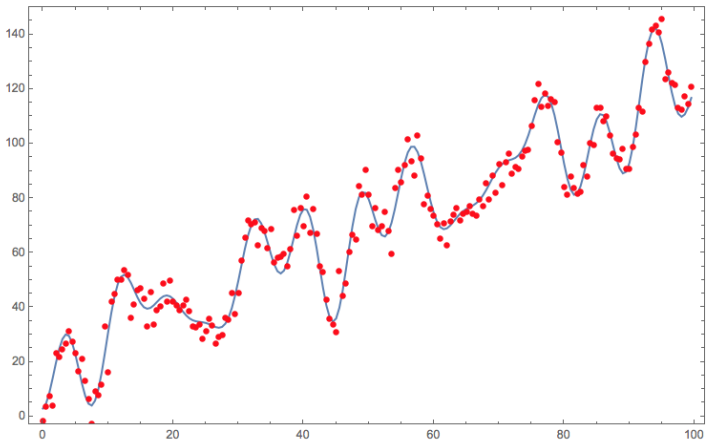


Fig. 2: Synthetic data set generated by means of (9). The red points indicate the data with added noise.

used. The parameter set is presented in Tab. 8.

The linear fit, i.e. only the variables a, b are involved, leads to $a = 1.0365, b = 17.786$ using the genetic algorithm. Fig. 3 illustrates the result. A direct fit in MATHEMATICA leads to the model $y = 17.7591 + 1.037\,x$. The agreement is very good without any change of the standard setting of the GA .

5.2.3 A non-linear least square fit

The data set from section 5.2.2 is used also in a nonlinear fit. Tab. 8 summarizes the results from the User's guide.

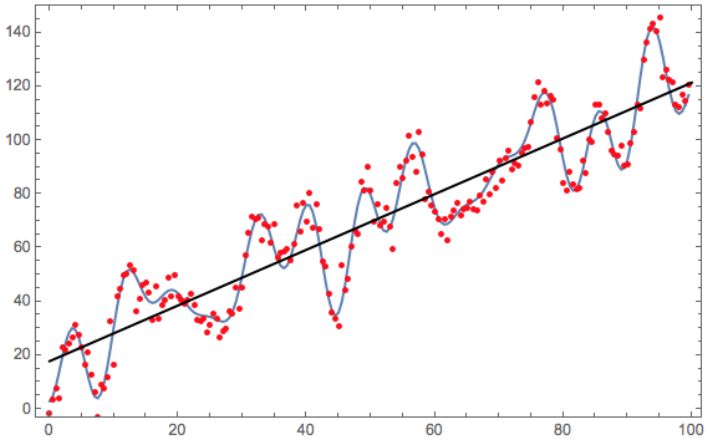


Fig. 3: Result of the linear least squares fit.

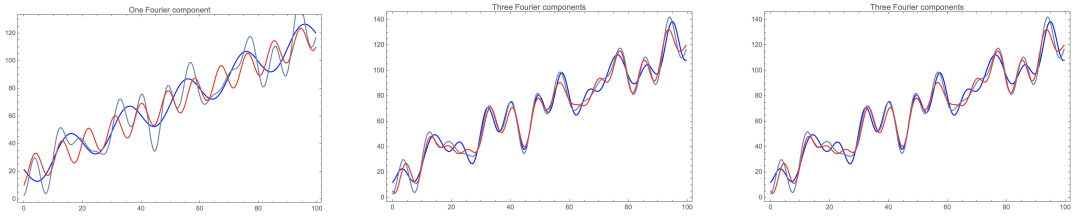
Tab. 8: Non-linear least squares fits.

M	a	b	m	A_m	P_m	Φ_m
original signal						
3	1.00	20.00	1	12.00	20.00	0.5000
			2	10.00	9.00	0.8000
			3	8.00	7.50	0.8000
tests from documentation						
1	0.9999	19.21	1	10.30	9.044	0.8346
3	0.9961	19.30	1	9.998	20.03	0.5124
			2	7.346	8.927	0.7008
			3	6.770	7.373	0.7011
5	0.9998	19.37	1	9.997	20.01	0.4999
			2	9.828	9.020	0.8115
			3	6.945	7.472	0.8001
			4	20.01	44.60	0.6462
			5	19.76	45.10	0.1643
own tests						

The tests result in larger deviations in this case. This can be seen in Fig. 5 Our results are a bit different, because we do not use the data set from the User's manual. Anyway we get a similar structure of the result.

5.2.4 Generalized least squares fitting by distance regression

In Fig. 5a an ellipse with added noise is shown. The ellipse is found from an ellipse in normal position by shifting and rotating.



(a) M=1 (b) M=3 (c) M=5

Fig. 4: Non-linear least squares fit (blue - this test, red - User's guide)

For the fit the following expressions are used (see User's guide):

$$r^2(\theta) = \frac{a^2 b^2}{a^2 \cos^2(\theta - \theta_0) + b^2 \sin^2(\theta - \theta_0)} \tag{10}$$

a, b are the semi-major and semi-minor axes and θ_0 is the inclination of the major-axis with respect to the x-axis. The points on the ellipse are finally given by

$$x_j = x_0 + r(\theta_j)\cos(\theta_j) \tag{11}$$
$$y_j = y_0 + r(\theta_j)\sin(\theta_j) \tag{12}$$
$$\tag{13}$$

where $\theta_j = \text{atan}(y_j/x_j) \in [0, 2\pi]$.

In Fig. 5b the results of the fit are presented.

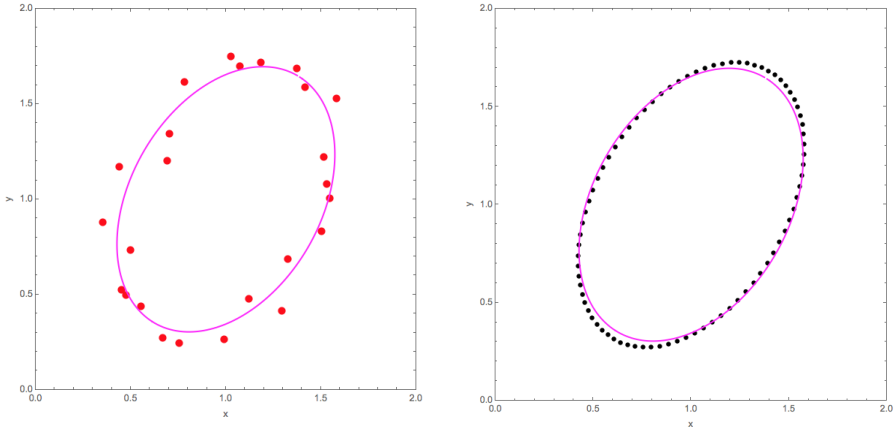
Tab. 9: Results of the fit.

	x_0	y_0	a	b	θ
construction	1.00000	1.00000	0.75000	0.50000	$1.00000(\pi/3)$
User's guide	0.97076	0.97996	0.77182	0.50538	$0.96357(\pi/3)$
this fit	1.00064	0.99898	0.79032	0.48722	$1.05705(\pi/3)$

Be aware, that the genetic algorithm is a probabilistic technique. This means the result given here, is not necessary the best one. Modifications in the algorithm (mutation, elitism, ...) might lead to better results. Usually it is not a one shot calculation. Starting the random number generator differently leads to different results. It is possible that you get a and b interchanged and a rotation angle of $5\pi/6$ from the fit, which defines the same ellipse.

5.2.5 Data modeling using robust estimators

The test program `xpk3.f` is only slightly modified to allow the input of the data constructed with MATHEMATICA and to rescale the parameters. The program is `xpk3m.f90`. The executable will be generated by the `makefile`. See `test/fit3` for data and results. The center of the unit circle is at $(1.5, 1.5)$. According to the documentation, similar



(a) ellipse with noisy data (b) result of the fit

Fig. 5: Fitting an ellipse.

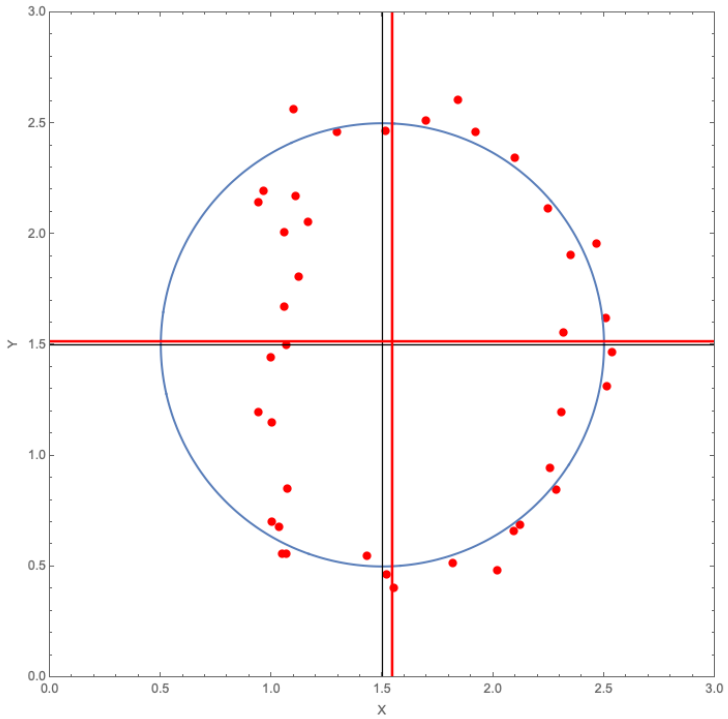


Fig. 6: Result of the unit circle center search (red dots - data points, blue cross - center of circle, red cross - estimated center)

data are modeled. A normal distribution is used here for the noise generation. The GA gives $(1.544, 1.516)$ as an estimate for the circle center. Fig. 6 represents the input data and the fit.

6 Examples for fitting bands

The sub-directory `examples` contains a set of complete examples. In `sb` a single s-band is investigated in sc-, fcc- and bcc-structure. In `Cu_d` a TB-Hamiltonian for Cu is used, but for d-bands only. Finally `Cu_spd` contains the complete Hamiltonian for Cu. All examples contain the following subdirectories: `bands, dos, exe, fit, math`. The names stand for the content.

Not all examples and calculations are done up to the end, but a series of input files are presented. It should be possible to extend the calculations to the other cases.

The calculation of bands and DOS is more or less straightforward. Therefore such calculations will be discussed only for the complete Cu-Hamiltonian.

6.1 A single s-Band

The directory `examples/sb/fit` contains the test with all three fit methods. The fcc structure is studied. In general the band is given for all cubic structures by means of

$$\begin{aligned} E(\boldsymbol{k}) = (ss0) + 4(ss\sigma)_1(\cos(\pi\xi)\cos(\pi\eta) + \cos(\pi\xi)\cos(\pi\zeta) + \cos(\pi\eta)\cos(\pi\zeta)) + \\ + 2(ss\sigma)_2(\cos(\pi\xi)\cos(\pi\eta) + \cos(\pi\zeta)) + 8(ss\sigma)_3\cos(\pi\xi)\cos(\pi\eta)\cos(\pi\zeta)\ . \end{aligned} \tag{14}$$

For the fcc structure the parameter set

$$\{(ss0), (ss\sigma)_1, (ss\sigma)_2, (ss\sigma)_3\} = \{0., 1., 0., 0.\} \tag{15}$$

is chosen. To this parameter set noise is added to generate a starting approach for the fit.[5].The set of parameters is:

$$\{(ss0), (ss\sigma)_1, (ss\sigma)_2, (ss\sigma)_3\} = \{0.0536, 1.1315, 0.1943, 0.1566\}\ . \tag{16}$$

Fig. 7 presents a comparison of the original band structure and the band calculated with noisy parameters.

6.1.1 Results of the fits

The files to the different fits can be found in subdirectories `lsqr, genalg, simplex`. Tab. 10 summarizes the results of the fits. The least square method is a bit superior here, but all methods work very well.

6.2 Cu d-Hamiltonian

The Hamiltonian for the fcc structure with d electrons up to second nearest neighbors is used for Cu. This Hamiltonian contains 7 parameters. Tab. 11 contains the original parameters from [6]. To this parameters noise is added and those parameters are also given in the table. Equally distributed random numbers in the given intervals are used

[5]See Notebook `Testdata_s.nb` in `math`

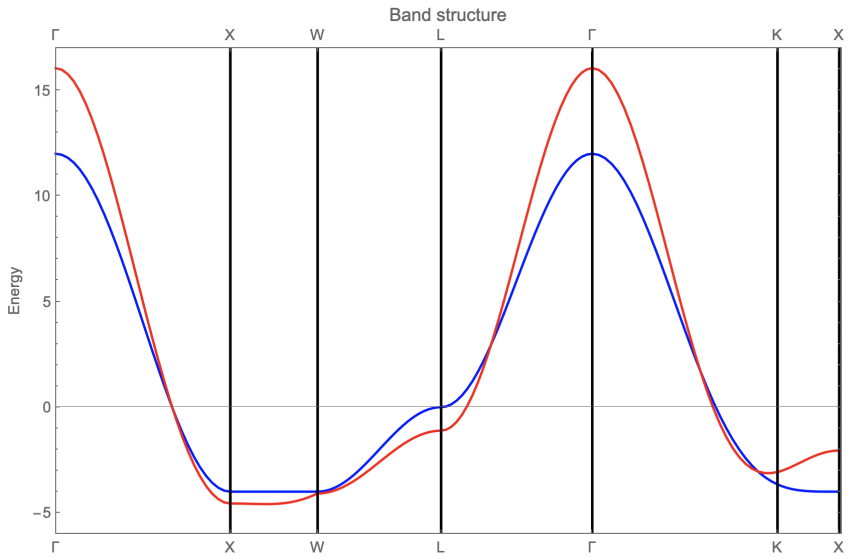


Fig. 7: Original band (blue) and band with noisy parameters.

Tab. 10: Results of the fits.

	$(ss0)$	$(ss\sigma)_1$	$(ss\sigma)_2$	$(ss\sigma)_3$
correct	0.0	1.0	0.0	0.0
start	0.0536	1.1315	0.1943	0.1565
`LEASTSQ`	-0.33E-15	1.0000	-0.14E-16	0.24E-16
`GENALG`	0.11E-05	1.0000	-0.21E-06	-0.21E-06
`SIMPLEX`	-0.41E-04	1.0000	0.22E-04	0.15E-04

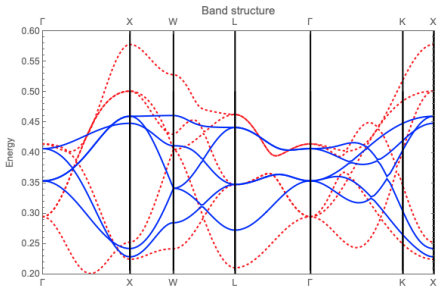
Tab. 11: Parameters of the Hamiltonian.

parm set	$(dd0)$	$(dd\sigma)_1$	$(dd\pi)_1$	$(dd\delta)_1$	$(dd\sigma)_2$	$(dd\pi)_2$	$(dd\delta)_2$
[6]	0.37	-0.02566	0.018	-0.00408	-0.00451	0.00241	-0.00029
$[-0.02, 0.02]$	0.38	-0.03206	0.0229	-0.01770	-0.00965	-0.00707	0.01778
$[-0.01, 0.01]$	0.366	-0.01820	0.0109	0.00218	-0.0112	-0.00473	0.00914

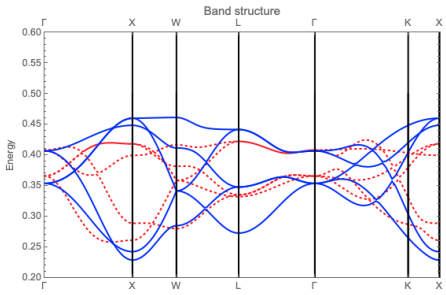
to simulate noise. This is a „real life“ experiment. We start with the noisy parameters, i.e. a first guess for the parameters, and have to find the correct TB parameters used to generate the bands to be fitted. In this experiment the fit should result in the parameters used to create the band structure.

Different strategies are possible:

- Use first only the symmetry points for the fit and than the data along the lines.
- It is also possible to fix some parameters during the fit. As an example use first only the nearest neighbor parameters and add the second nearest neighbors in a next step.
- Combine different fitting methods.



(a) (-0.02,0.02)



(b) (-0.01,0.01)

Fig. 8: Band structures with noisy parameters compared with the correct bands.

The fit program allows an output of the „best fit“ in a form, which can be used directly as input for the next fitting step. This can be also a different fitting method.

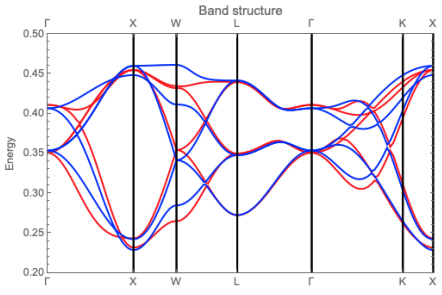
6.2.1 Pure Least squares fit

The data set with noise [-.02,.02] is used. For a first fit only the symmetry points are taken into account. Tab. 12 presents the parameters after the fit. Fig. 9a shows the

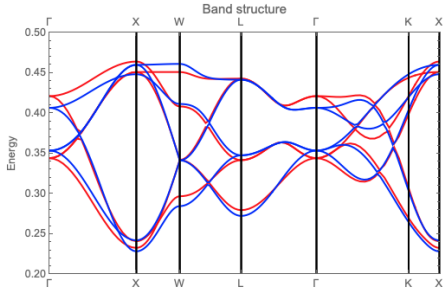
Tab. 12: Parameters of the Hamiltonian.

parm set	$(dd0)$	$(dd\sigma)_1$	$(dd\pi)_1$	$(dd\delta)_1$	$(dd\sigma)_2$	$(dd\pi)_2$	$(dd\delta)_2$
[6]	0.37	-0.02566	0.018	-0.00408	-0.00451	0.00241	-0.00029
[-0.02,0.02]	0.38	-0.03206	0.0229	-0.01770	-0.00965	-0.00707	0.01778
fit points	0.37	-0.02470	0.0182	-0.00474	-0.00888	-0.01359	0.00565
fit line	0.37	-0.02566	0.018	-0.00408	-0.00451	0.00241	-0.00029

quality of this first fit.



(a) symmetry points



(b) nearest neighbors

Fig. 9: First step with the least squares methods with two different approximations (blue - original band structure).

In the next experiment **k** vector along the symmetry lines are taken into account, but only the noisy parameters for nearest neighbors are used. (cf. Tab. 13) The result of this first approximation is given in Fig. 9b

After this first step in both calculations the data from the symmetry lines are used and the parameters of the first step of the fit are taken as starting values. As Tab. 12 reveals, that in both cases the exact parameters will be found.

Tab. 13: Parameters of the Hamiltonian.

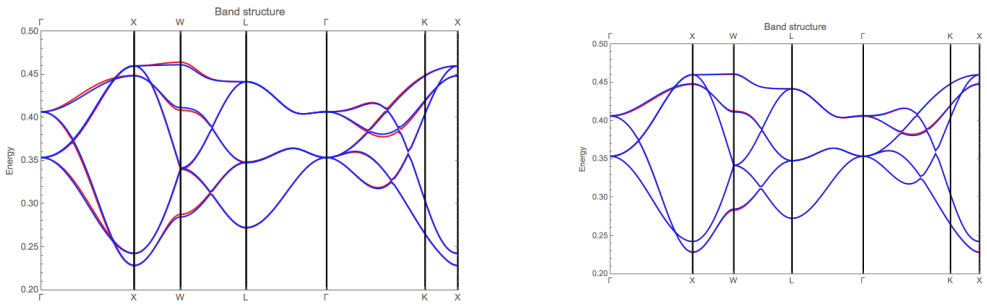
parm set	$(dd0)$	$(dd\sigma)_1$	$(dd\pi)_1$	$(dd\delta)_1$	$(dd\sigma)_2$	$(dd\pi)_2$	$(dd\delta)_2$
[6]	0.3700	-0.02566	0.0180	-0.00408	-0.00451	0.00241	-0.00029
[-0.02,0.02]	0.3800	-0.03206	0.0229	-0.01770	0.0	0.0	0.0
fixed	0.3699	-0.02567	0.0182	-0.00431	0.0	0.0	0.0
all	0.3700	-0.02566	0.01800	-0.00408	-0.00451	0.00241	-0.00029

6.2.2 Genetic algorithm

We start again from the same set of noisy parameters. In a first step only the symmetry points are used for fitting. In a second step the lines are used. The results of the fits are presented in Tab. 14 and Fig. 10.

Tab. 14: Parameters of the Hamiltonian.

parm set	$(dd0)$	$(dd\sigma)_1$	$(dd\pi)_1$	$(dd\delta)_1$	$(dd\sigma)_2$	$(dd\pi)_2$	$(dd\delta)_2$
[6]	0.3700	-0.02566	0.0180	-0.00408	-0.00451	0.00241	-0.00029
[-0.02,0.02]	0.38	-0.03206	0.0229	-0.01770	-0.00965	-0.00707	0.01778
points	0.3700	-0.02555	0.0181	-0.00419	-0.0037	0.0028	-0.0011
lines	0.3701	-0.02567	0.01798	-0.00405	-0.00500	0.00232	-0.00006



(a) symmetry points (b) symmetry lines

Fig. 10: Result of fit with genetic algorithm.

Already the first run with symmetry points only provides an excellent result. The result is not so much improved by taking the symmetry lines into account.

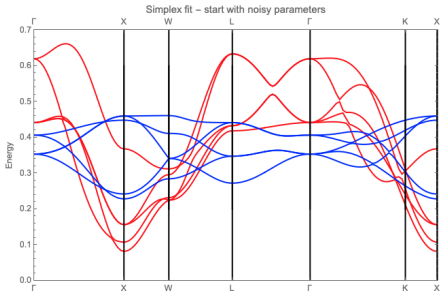
Note, the genetic algorithm depends on the initialization of the random number generator and the settings for the genetic algorithm (mutation rate, elitism, ...). See the documentation for more information and the input files for this example.[6]

6.2.3 Simplex method

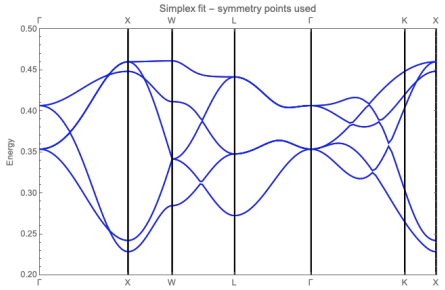
It is enough to take into account only the energy values at the symmetry points to find the correct TB-parameters. A different set of noisy parameters is used here. Fig. 11

Tab. 15: Fit by means of simplex method

parm set	$(dd0)$	$(dd\sigma)_1$	$(dd\pi)_1$	$(dd\delta)_1$	$(dd\sigma)_2$	$(dd\pi)_2$	$(dd\delta)_2$
[6]	0.3700	-0.02566	0.0180	-0.00408	-0.00451	0.00241	-0.00029
[-0.05,0.05]	0.38475	0.01593	0.04015	0.00481	0.01781	-0.02642	-0.03486
points	0.3700	-0.02566	0.01800	-0.00408	-0.00451	0.00241	-0.00029



(a) noisy parameters (red)



(b) fit with symmetry points only

Fig. 11: Band structures with noisy parameters compared with the correct bands.

reveals the excellent performance of the simplex method for this example.

6.3 Cu *spd*-Hamiltonian

The test with the full Hamiltonian for Cu is close to that what can be expected in using the package SimPack for fitting TB-parameters.

6.3.1 Band structure

We start with a test of the band structure.[7] The band structure is easily calculated in GTPack. The TB-parameters and the **k**-path are exported for the FORTRAN calculation. The results of the external FORTRAN calculation are compared with the GTPack result.

[6] `genalg1.in` and `genalg2.in` in `fit/genalg`.
[7] cf. Notebook `CU_bands.nb`.

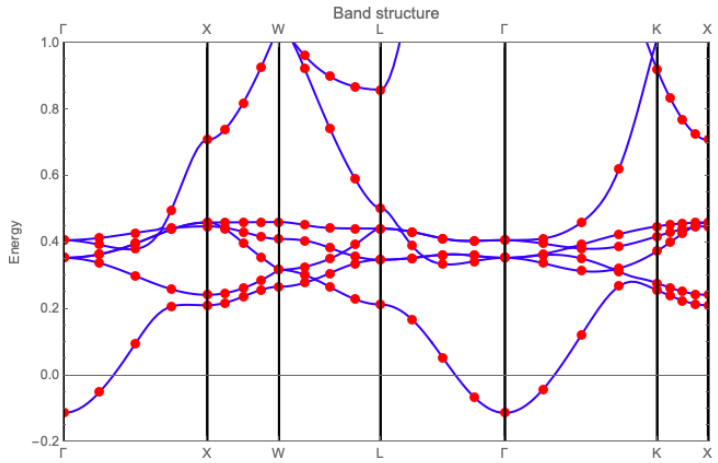
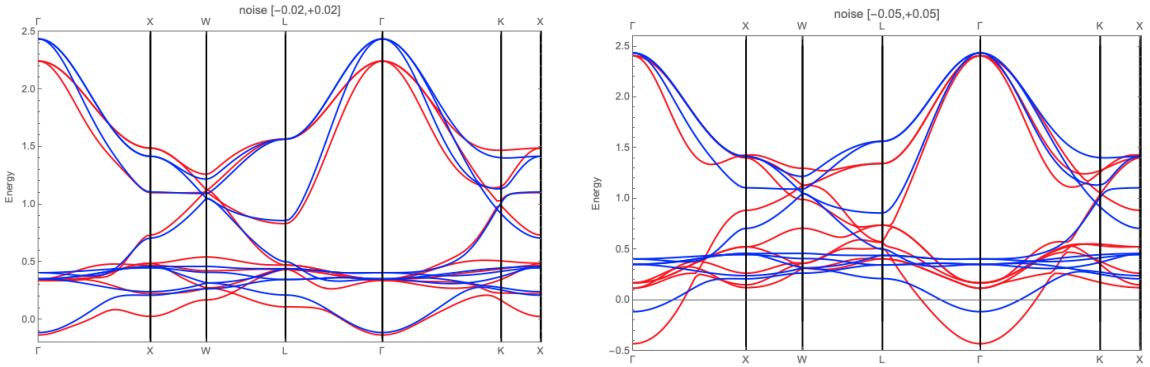


Fig. 12: gtp (blue) and FORTRAN (red).

Fig. 12 reveals a perfect agreement. Thus, the whole path of generating and exporting Hamiltonians to FORTRAN and the external calculation works well.

6.3.2 Results of the fits

Fig. 12 demonstrates, that the band structure is more complex than in the *d*-Hamiltonian case, due to the hybridization of the bands. We can again start with a fit using sym-



(a) small noise (b) larger noise

Fig. 13: Exact bands (blue) and bands with noisy parameters (red).

metry points only. In a next step symmetry lines should be used without too much hybridization. L-Γ-X might be a good choice. In Notebook `Testdata_Cu_spd.nb` sets of noisy parameters is generated by means of adding equally distributed random numbers from two ranges. The range $[-0.02, +0.02]$ produces moderate deviations from the exact band structure. The range $[-0.05, +0.05]$ produces strong deviations. Fig. 13 presents the corresponding band structures.

Pure least squares fit

The following tests will be performed for the two sets of parameters[8] In a first step

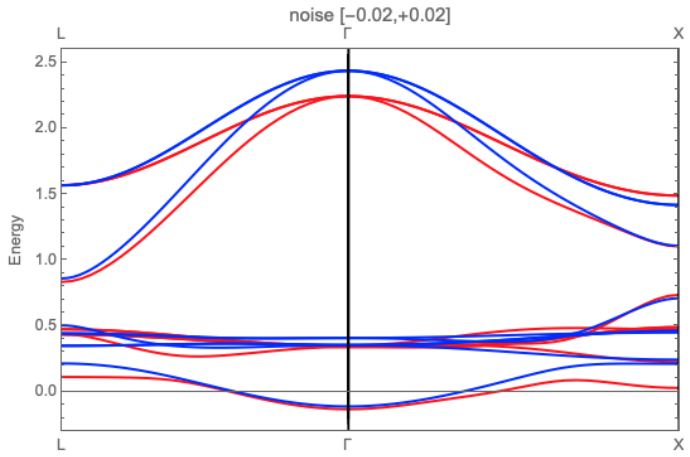


Fig. 14: Exact bands (blue) and bands with noisy parameters (red) along L-Γ-X.

we use only the symmetry points for the fit and switch off the second nearest neighbor interactions to improve our starting approximations. Fig. 15 represents the result for the complete band structure. If the fit is now performed with this set of parameters along two or all lines the exact band structure is reproduced.

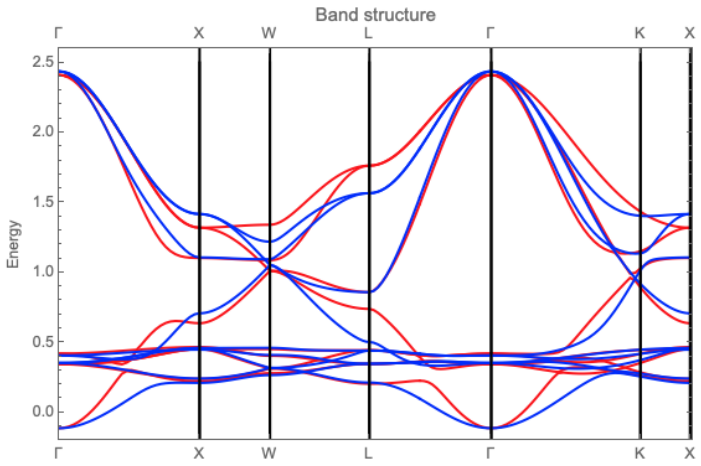
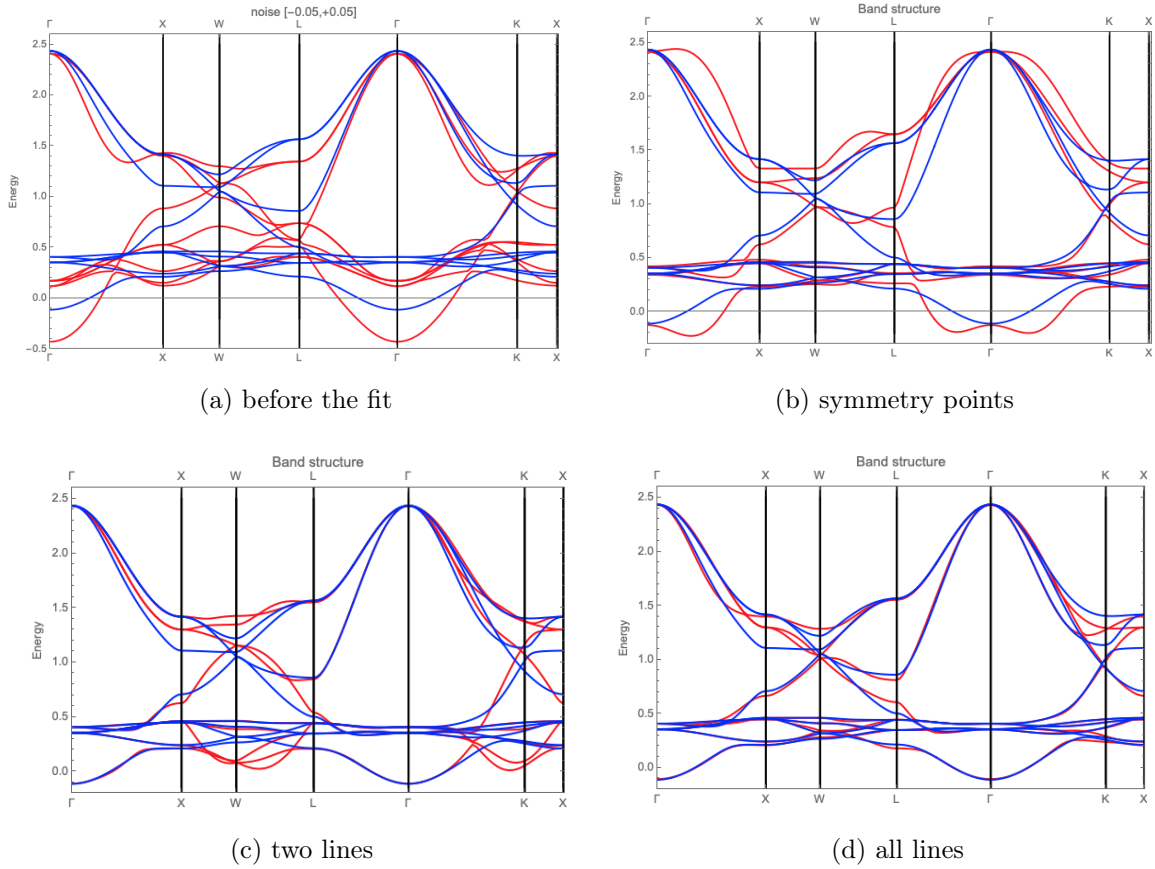


Fig. 15: Fit at symmetry points with nearest neighbors only [-0.02,+0.02]

- only the symmetry points are used for the fit,
- fit along the two lines ,

[8] All input files corresponding to [-0.02,+0.02] are named `lsqr1_#.in` and the input files to [-0.05,+0.05] are named `lsqr2_#.in`. All other file names are declared in the input files.

- fit using all bands (cf. Fig. 12),
- use the fixing of parameters or weights for the bands.



(a) before the fit

(b) symmetry points

(c) two lines

(d) all lines

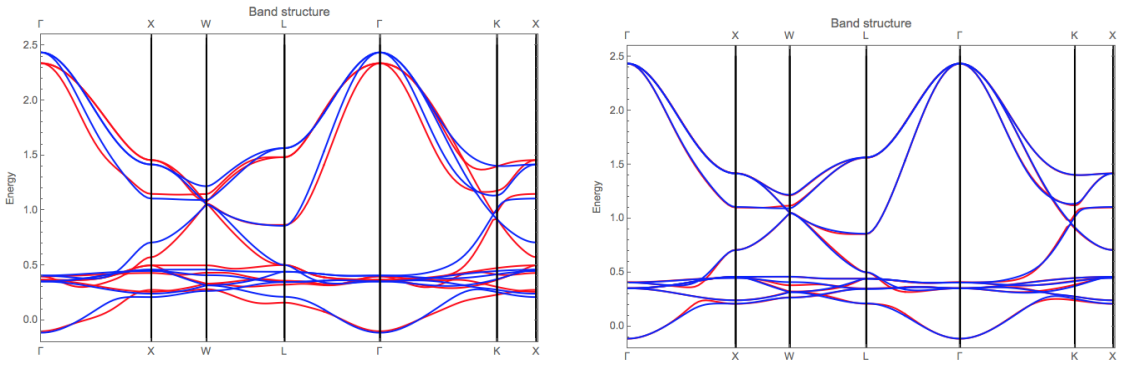
Fig. 16: Fit with parameter set [-0.05,+0.05].

In a next test we perform the same calculations, but start with the second set of of parameters, generated with larger noise. The results of this experiment are summarized in Fig. 16. The fit using the symmetry points only improves the correspondence at the symmetry points considerably. It is nice to see, that The use of L-Γ-X (cf. Fig. 16c) results in nearly perfect fits along those lines, but the other lines are represented badly. Now we use in Fig. 16d the complete band structure for fitting. The result improves again, but is still not perfect. In a next step we restrict the fit to the lowest 6 bands and the result is perfect.

Genetic algorithm

Here we demonstrate (cf. Fig. 17) only the application of two fit methods to one problem. In a first step the genetic algorithm is applied. In the second step the results of the genetic algorithm is used as input for the least squares fit.

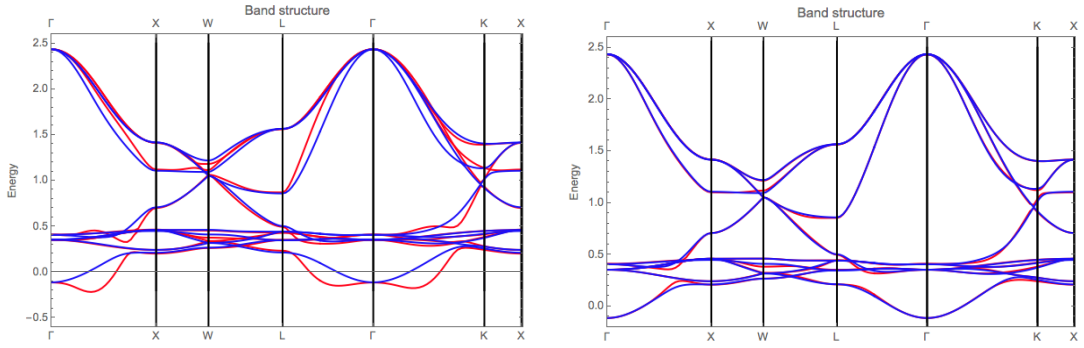
Simplex method



(a) genetic algorithm (b) least squares fit

Fig. 17: Combination of fit methods for the noisy set [-0.02,+0.02].

We perform tests with the simplex method.[9] In a first step only the symmetry points



(a) symmetry points (b) all lines

Fig. 18: Fit of parameters starting noisy set [-0.02,+0.02].

are used in the fit (cf. Fig. 18a). Next, the lines X-Γ-L are used. Finally all lines are included (cf. Fig. 18b). As demonstrated before you can switch to another method after each fitting step. Thus the final result from the simplex fit could be improved with the least squares method.

6.3.3 Calculation of DOS

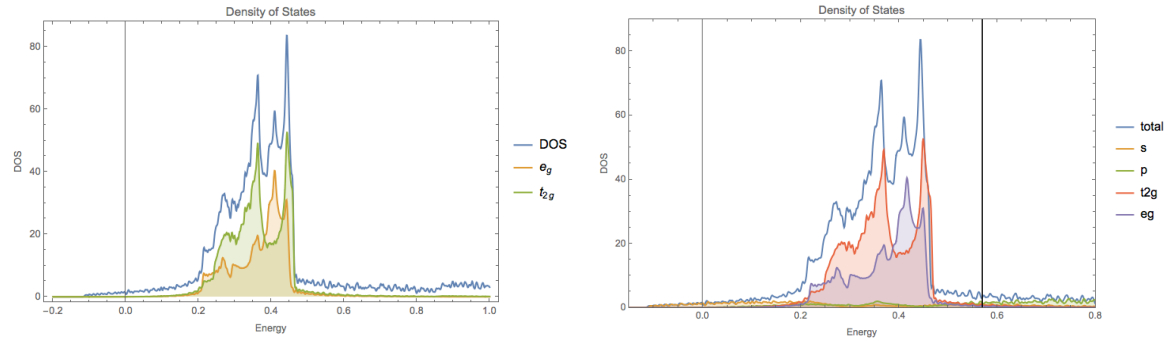
Finally some details of the calculation of densities of states, as implemented in this code should be discussed.[10]

Gaussian smearing

[9] See the Notebooks in `Cu_spd/math` and all data in `Cu_spd/fit/simplex`.

[10] See also the Notebook `Cu_DOS.nb` in `Cu_spd/math` and the corresponding input files in `Cu_spd/dos` to generate the data. See also `READ_ME.TXT` in this directory.

FORTRAN is a bit faster and can be preferential, if you need a large set of **k**-points for your calculation. Fig. 19 reveals, that both methods lead to the same result.



(a) GTPack calculation (b) SimPack calculation

Fig. 19: Calculation of partial DOS'es for Cu directly in GTPack or with SimPack by means of Gaussian smearing.

Tab. 16: Input to calculate the partial DOS

```
PART_DOS      4
              's'     1
                      1
              'p'     3
                      2  3  4
              't2g'   3
                      5  6  8
              'eg'    2
                      7  9
```

For the calculation of the partial DOS the token `PART_DOS` is used. After the token follows the number of partial DOS to be calculated. For each partial DOS the name and the number of orbitals that define the DOS are given. In the following line the orbitals that have to be summed up for that DOS are defined.

It is clear, that the meshes generated for the tetrahedron method can be used also for a calculation with Gaussian smearing. An example is given in the Notebook.

Tetrahedron method

The tetrahedron method is able to provide a DOS with finer details or to calculate the DOS with less **k**-points. A rather old implementation of this method are used. For the DOS calculation with the tetrahedron method the bands have to be calculated at the corresponding ***k***-meshes. Information about the mesh construction for TETRA can be found in `kpoints_3d.f90`.

emin	de	emax	Fermi energy E_F	electrons
-0.2	0.001	1.	0.5741	11.

number of densities 7 energy points = 1210

	total	s	p	t2g	eg
DOS	3.88875	0.947199	1.63813	0.879335	0.496828
N_{el}	10.9919	0.702778	0.493707	5.86315	3.94655

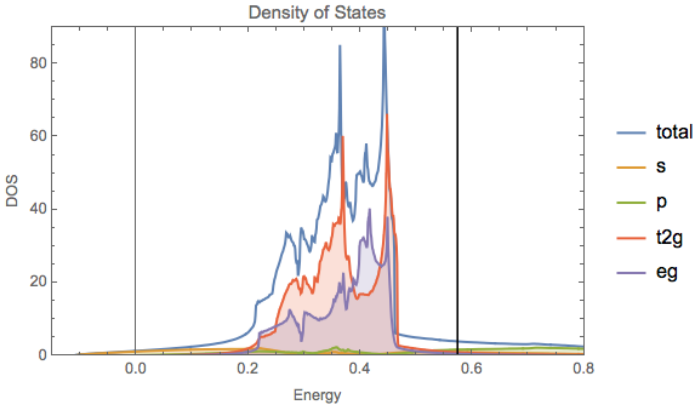


Fig. 20: Calculation of partial DOS with SimPack and method TETRA.

Fig. 20 reveals a calculation of partial DOSes by means of SimPack and the tetrahedron method (TETRA).

References

[1] P. Charbonneau. *An introduction to genetic algorithms for numerical optimization*. High Altitude Observatory. National Center for Atmospheric Reserach. Boulder, Colorado, 2002.

[2] P. Charbonneau. „Genetic Algorithms in Astronomy and Astrophysics“. In: *The Astrophysical Journal Supplement Series* 101 (1995), S. 309.

[3] P. Charbonneau. *Release notes for Pikaia 1.2*. High Altitude Observatory. National Center for Atmospheric Reserach. Boulder, Colorado, 2002.

[4] P. Charbonneau und B. Knapp. *A Users Guide to Pikaia 1.0*. High Altitude Observatory. National Center for Atmospheric Reserach. Boulder, Colorado, 1976.

[5] R. Fletcher und M. J. D. Powell. „A Rapidly Convergent Descent Method for Minimization“. In: *The Computer Journal* 6.2 (1963), S. 163–168.

[6] H. J. Gotsis, D. A. Papaconstantopoulos und M. J. Mehl. „Tight-binding calculations of the band structure and total energies of the various phases of magnesium“. In: *Phys. Rev. B* 65 (13 2002), S. 134101.

[7] Wolfram Hergert und Matthias Geilhufe. *Group Theory in Solid State Physics and Photonics. Problem Solving with Mathematica*. Wiley-VCH, 2018.

[8] G. Lehmann und M. Taut. „On the Numerical Calculation of the Density of States and Related Properties“. In: *physica status solidi (b)* 54.2 (1972), S. 469–477.

[9] J. A. Nelder und R. Mead. „A Simplex Method for Function Minimization“. In: *The Computer Journal* 7.4 (Jan. 1965), S. 308–313.

[10] R. O'Neill. „Algorithm AS 47: Function Minimization Using a Simplex Procedure“. In: *Journal of The Royal Statistical Society Series C-applied Statistics* 20 (1971), S. 338–345.

[11] W.H. Press u. a. *Numerical Recipes in FORTRAN -The Art of scientific computing*. Cambridge: Cambridge University Press, 1992.

[12] H. Schwetlick u. a. *Program DNLQ1*. Math. Bibl. FORTRAN. Computer station of MLU Halle-Wittenberg. Halle, 1976.