MAGIX

Modelling and **A**nalysis **G**eneric **I**nterface for e**X**ternal numerical codes

Thomas Möller

Manchester, 6th November 2012

Overview

- Why MAGIX ?
- Requirements, Installation
- Start MAGIX
- Registration process
- Optimization algorithms, error estimation, algorithm chain
- Examples
- Summary

Why MAGIX ?

- Many physical and chemical models depend on a set of parameters.
- The best description of observational data using a certain model requires the determination of a parameter set that most closely reproduces the data, by some criteria.
- MAGIX modifies in an iterative procedure the parameter set to improve the quality of the description, i.e. reducing the value of χ^2.
- MAGIX provides the best-fit parameters, within the framework of the model, to a particular data set, including confidence intervals for the parameters.
- (M)any models can be plugged into MAGIX so as to explore their parameter space and find the set of parameter values that best fits observational data.
- MAGIX contains so-called global optimizer which can be used to explore the whole parameter space and find multiple minima of the χ^2 distribution without an initial guess.
- MAGIX program can be called by other programs (for example CASA)

Requirements

Requirements for **MAGIX**:

- Python 2.6 (or later)
- Numpy 1.3 (or later)
- gfortran 4.3 (or later)
- scipy 0.9 (or later)
- matplotlib 0.99 (or later)
- subversion 1.6.6 (or later)

Installation

Installation of **MAGIX**:

- There are NO precompiled releases of MAGIX.
- Download the install script

  ```
  install_magix.sh
  ```

  from our website

  ```
  http://www.astro.uni-koeln.de/projects/schilke/MAGIX_NR
  ```

  which download automatically the source code from the repository using subversion
- In order to install MAGIX simply start the following bash-script at the command prompt

  ```
  ./install_magix.sh
  ```

Start MAGIX

Working with **MAGIX**:

- start **MAGIX** with:

```
./magix_start.py <path>/io_control.xml
```

- get extended help

```
Documentation/MAGIX_Manual.pdf
```

- example runs for the different algorithms are located in the subdirectory

```
run/examples/
```

- To start the Levenberg-Marquardt example type

```
./magix_start.py run/examples/Levenberg-Marquardt/io_control.xml
```

at the command prompt.

Start MAGIX

XML files needed by MAGIX to work:

- The registration file, which contains a description of the structure of the input and output file(s) of the external model program.
- The so-called instance file, which includes the names, initial values (and ranges) for all model parameters. It works in conjunction with the registration file and indicates the model parameters which should be optimized by MAGIX and which are hold fixed.
- The XML file containing settings for the import/export of experimental data, i.e. path(s) and name(s) of the data file(s), format(s), range(s) etc.
- The so-called algorithm control file, defines which algorithm or algorithm sequence MAGIX should use for the optimization together with the settings for each algorithm.
- The I/O control file, including the paths and file names of the aforementioned XML files.

Registration

Registration process:

- MAGIX has to know the structure of the input and output file(s).
- How many input files have to be created?
- What's the name(s) and format (ASCII or FITS) of the input file(s)?
- Which parameter has to be written to which position within the input file(s) ?
- What's the format of each parameter (string, integer, float)?
- How many output files are created by the external model program and what's the name and format of the each output file?
- How should MAGIX start the external model program, i.e. name and location of the external model program.
- Is it possible to execute the external model program at the same time at the same machine for more than one instances?
- As long as the structure of the input and output file(s) are not changed, the registration process has to be done only once !

Registration

Example of a registration XML-file:

```
<ModelProgramCall>
   <PathStartScript>Fit-Functions/ckRtm/magixRtmGreybody.py</PathStartScript>
   <ExeCommandStartScript>python magixRtmGreybody.py</ExeCommandStartScript>
   <ParallelizationPossible>Yes</ParallelizationPossible>
   <InputDataPath>data.dat</InputDataPath>
</ModelProgramCall>

<NumberInputFiles>1</NumberInputFiles>

<InputFile>
   <InputFileName>in.txt</InputFileName>

   <NumberLines>5</NumberLines>

   <line group="false">
      <NumberParameterLine>1</NumberParameterLine>
      <Parameter group="false">
         <NumberReplicationParameter> </NumberReplicationParameter>
         <Name>sourceSize</Name>
         <Format>ES30.15</Format>
         <LeadingString></LeadingString>
         <TrailingString></TrailingString>
      </Parameter>
   </line>
   . . .
```

Registration

The corresponding input file “in.txt”:

4.000000000000000E+01
8.700000000000000E+02
3.2352196432795 49E+01
1.714226271972070E+00
1.239041499402653E+04

Instance file

Example of an instance XML-file:

```
<SubSection>
   <NumberParameters>5</NumberParameters>

   <Parameter fit="false">
      <name>sourceSize</name>
      <value>40.0</value>
      <error> </error>
      <lowlimit>0</lowlimit>
      <uplimit>80</uplimit>
   </Parameter>

   <Parameter fit="false">
      <name>WaveRef</name>
      <value>870.0</value>
      <error> </error>
      <lowlimit>0</lowlimit>
      <uplimit>100</uplimit>
   </Parameter>
   . . .
```

Experimental data

Example of experimental XML-file:

```
<ExpFiles>
   <Section>
      <SubSection>

         <NumberExpFiles>1</NumberExpFiles>

         <file>
            <FileNamesExpFiles>run/ckRtm_lm/experiment.dat</FileNamesExpFiles>
            <ImportFilter>ascii</ImportFilter>

            <NumberHeaderLines>0</NumberHeaderLines>
            <SeparatorColumns> </SeparatorColumns>

            <NumberColumnsX>1</NumberColumnsX>
            <NumberColumnsY>1</NumberColumnsY>

            <ErrorY>no</ErrorY>

            <NumberExpRanges>0</NumberExpRanges>
         </file>

      </SubSection>
   </Section>
</ExpFiles>
```

Fit control file

Example of a fit control XML-file:

```
<!-- set number of used algorithms -->
<NumberOfFitAlgorithms>1</NumberOfFitAlgorithms>

 <algorithm>
      <FitAlgorithm>Levenberg-Marquardt</FitAlgorithm>

      <!-- define method used for Levenberg-Marquardt-->
      <MethodLM>nr</MethodLM>

      <!-- define value of the variation -->
      <VariationValue>1e-5</VariationValue>

      <!-- set max. number of iterations -->
      <number_iterations>50</number_iterations>

      <!-- set max. number of processors -->
      <NumberProcessors>8</NumberProcessors>

      . . .
```

Algorithms

Available algorithms:

- Levenberg–Marquardt (local optimization, very fast)
- Simulated Annealing (local optimization, fast)
- Particle Swarm Optimization (global optimization, good convergence)
- Bees algorithm (global optimization, explore the landscape of the problem)
- Genetic algorithm (global optimization, good convergence)
- Nested Sampling (global optimization, good convergence)
- Interval Nested Sampling (global optimization, fast convergence)
- “Error estimation”
- Interface to make several algorithms included in the scipy package available

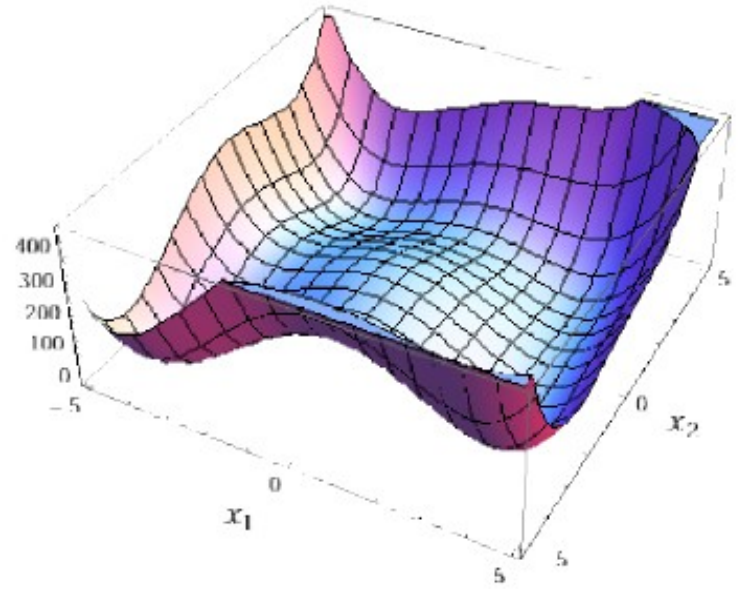
Algorithms

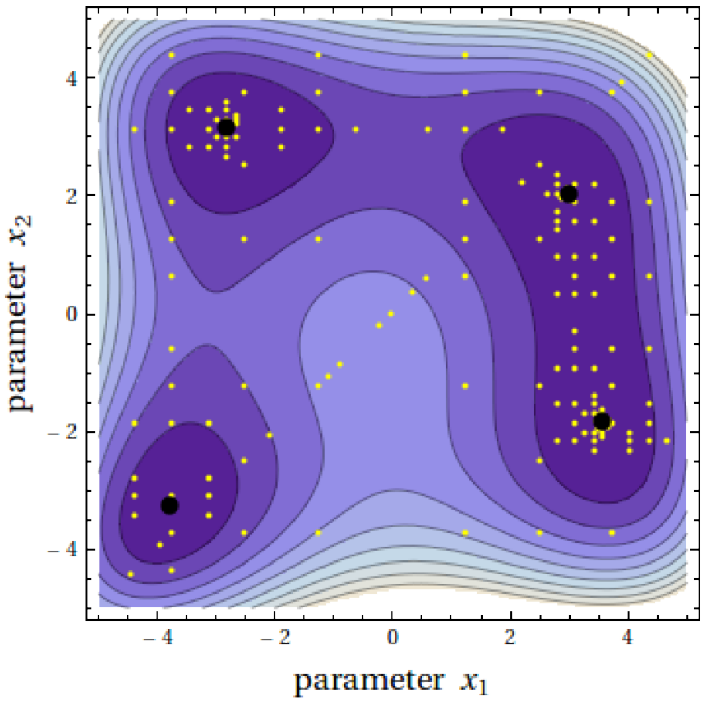
Example: Interval Nested Sampling algorithm:

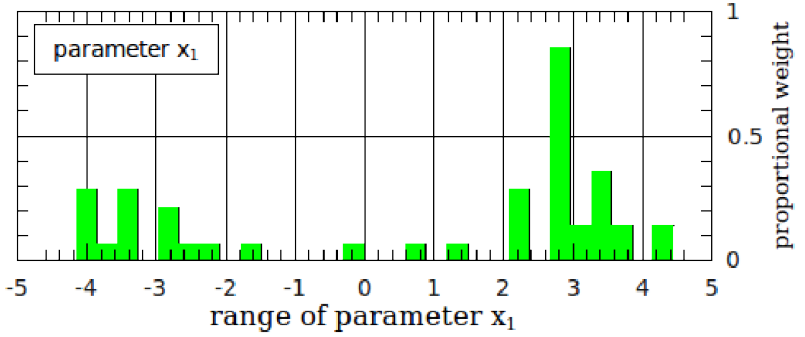
Himmelblau function

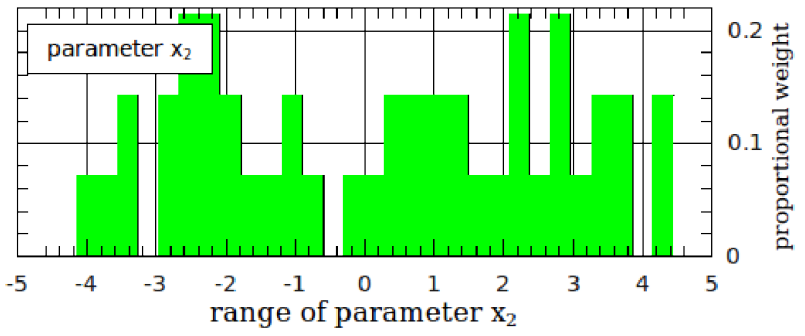
$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$

with four identical minima.



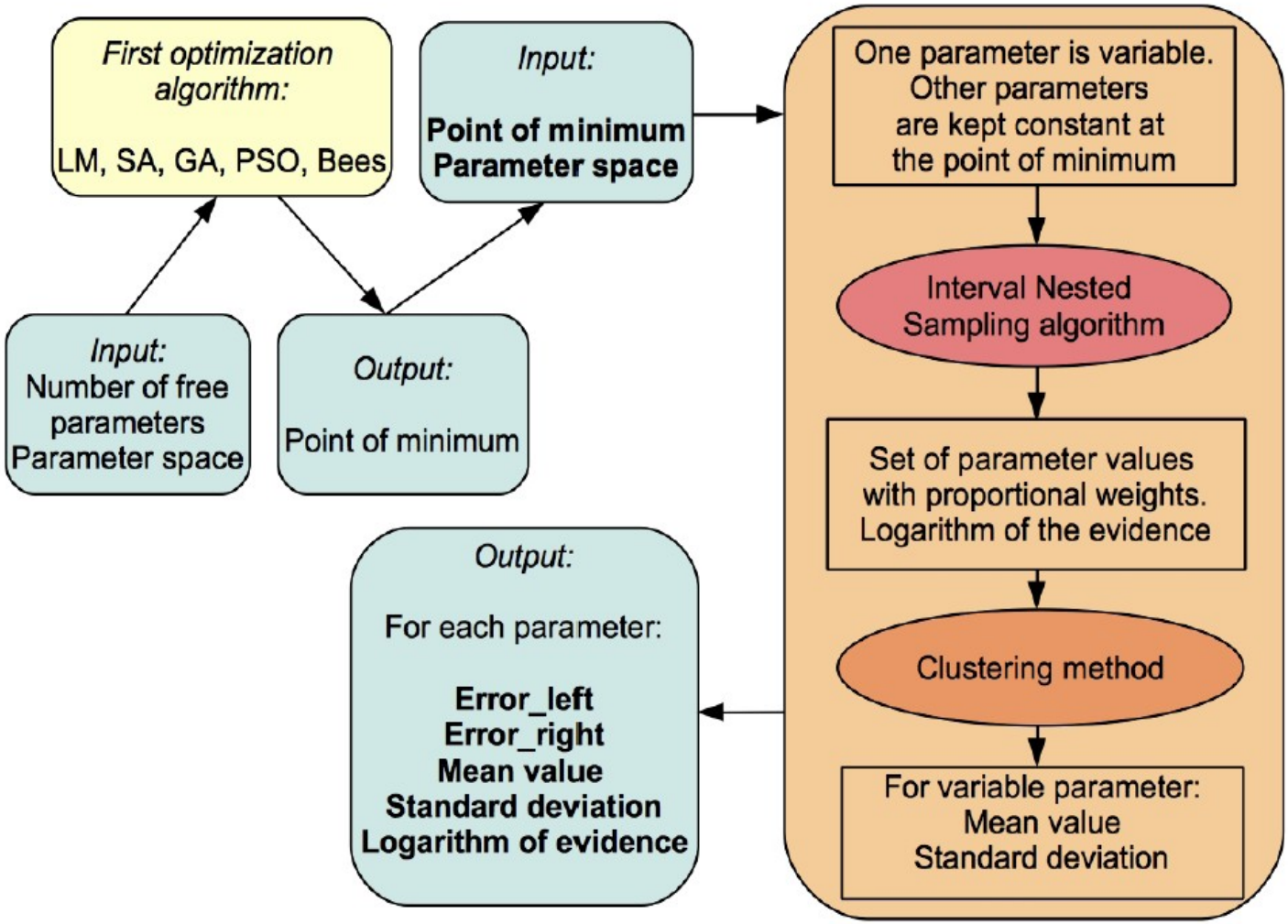






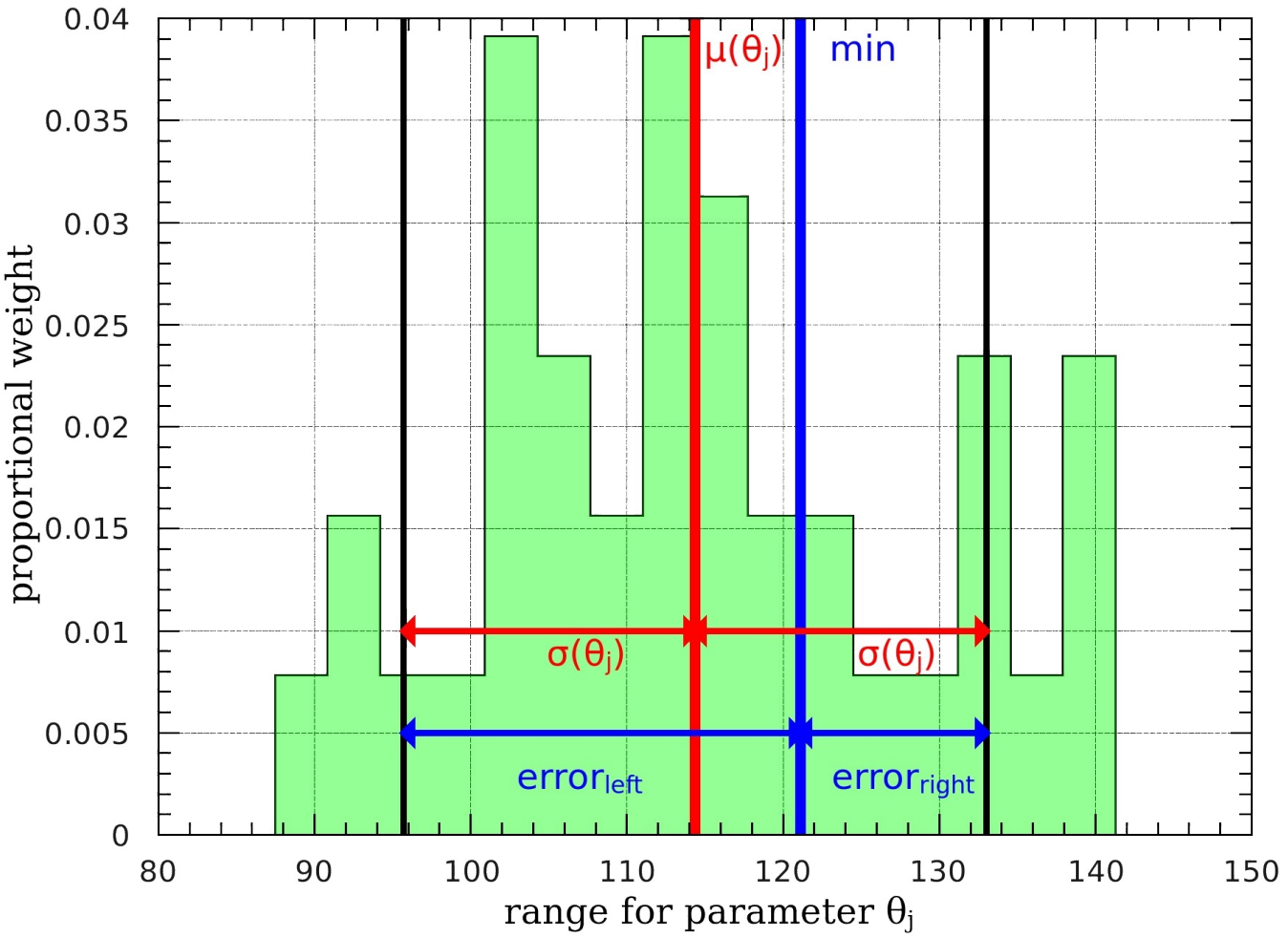
Error Estimation

Schematic diagram of error estimation module of MAGIX:



Error Estimation

Example of a histogram of distribution of parameter values after error estimation:
(Here, $\mu(\theta_j)$ indicates the mean value, $\sigma(\theta_j)$ represents the standard deviation, and "min" the value of the parameter θ_j of the best fit result).



Algorithm Chain

- each algorithm has advantages and disadvantages:

  → Combine algorithms !

- Simulated Annealing as well as the Levenberg-Marquardt algorithm requires starting values of the parameters that are to be optimized, i.e. the user has to find a good fit by hand before the application of these algorithms produces useful results.

- MAGIX includes the possibility to send the results of the optimization process performed by a certain algorithm, to another optimization procedure through some other algorithm.

- Using a so-called “algorithm chain” the user can apply one of the swarm algorithms e.g. the bees or Nested sampling algorithm to determine the starting values for the local optimization algorithms.

- An algorithm chain is essential for using the error estimation algorithm!

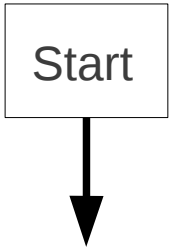
Algorithm Chain

Example of a “chain”:

Algorithm Chain

Example of a “chain”:

1st algorithm: Bees

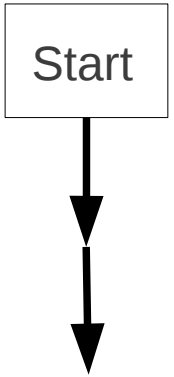


Algorithm Chain

Example of a “chain”:

1st algorithm: Bees

2nd algorithm: Levenberg-Marquardt



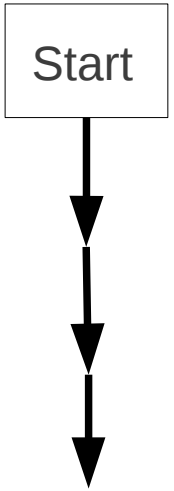
Algorithm Chain

Example of a “chain”:

1st algorithm: Bees

2nd algorithm: Levenberg-Marquardt

3rd algorithm: Error Estimation



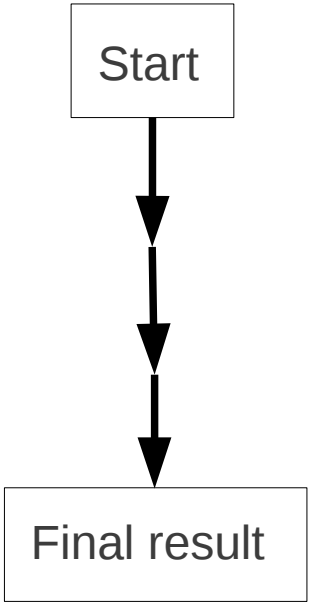
Algorithm Chain

Example of a “chain”:

1st algorithm: Bees

2nd algorithm: Levenberg-Marquardt

3rd algorithm: Error Estimation



Chain

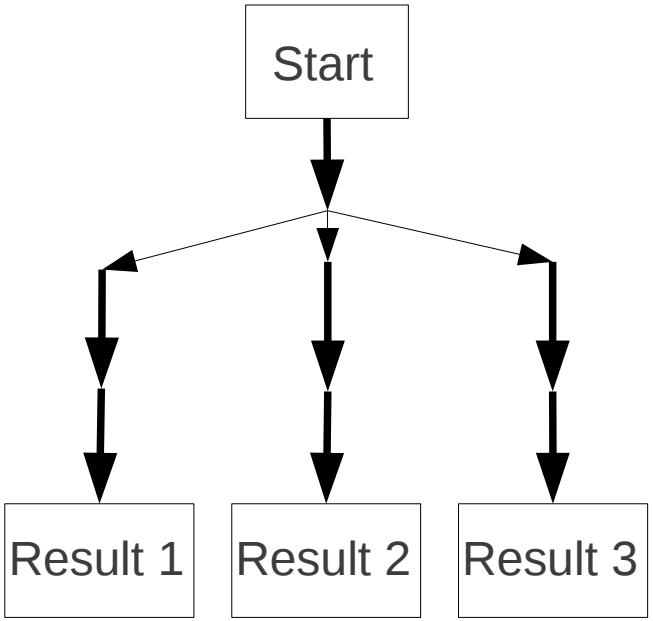
Algorithm Chain

Example of a “tree”:

1st algorithm: Bees

2nd algorithm: Levenberg-Marquardt

3rd algorithm: Error Estimation

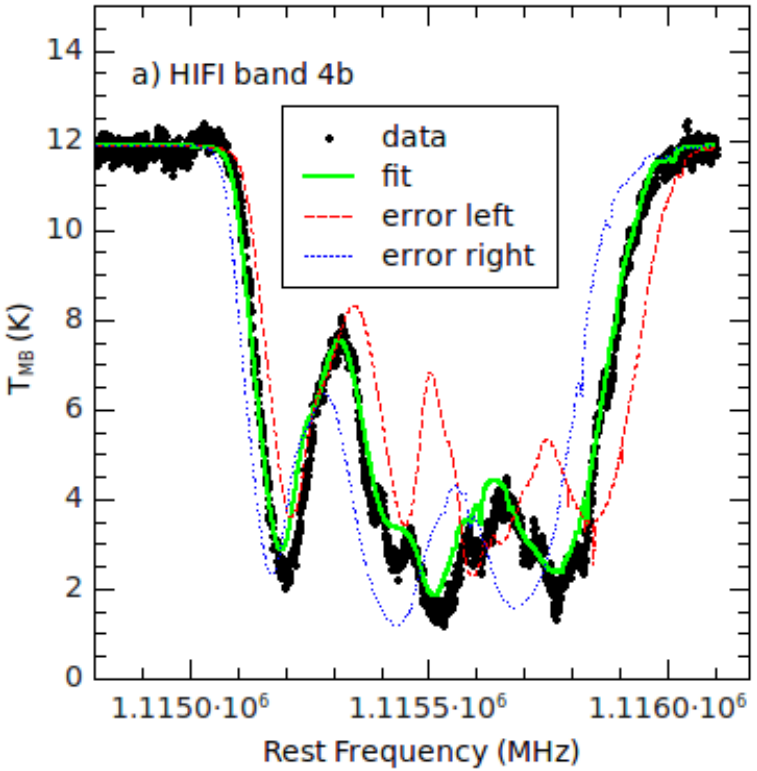


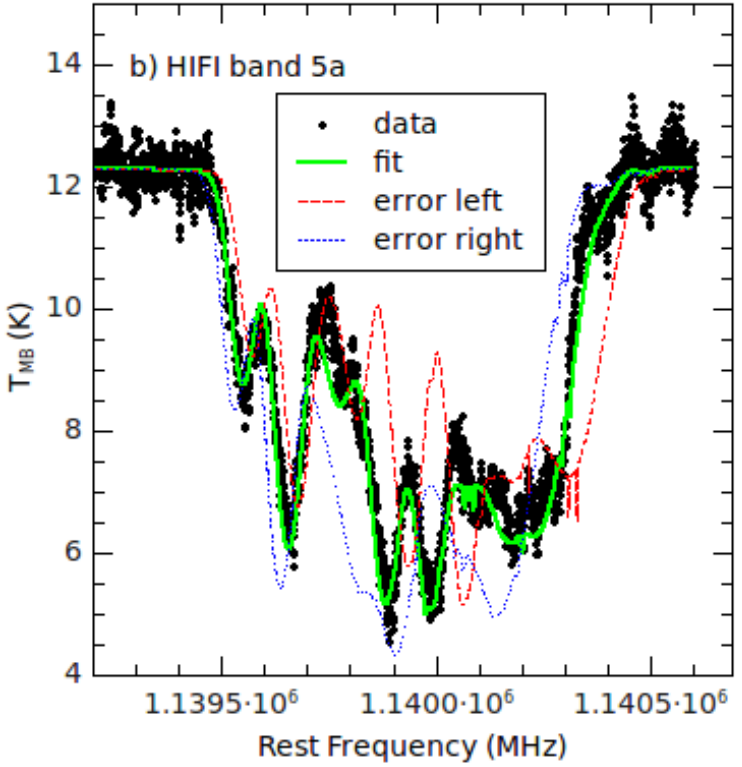
Tree

Examples

Example of an algorithm chain:

- Fit of HIFI bands a) 4b and b) 5a toward SgrB2(M) with myXCLASS using an algorithm chain consisting of the Genetic, the Simulated Annealing, and the error estimation algorithm.

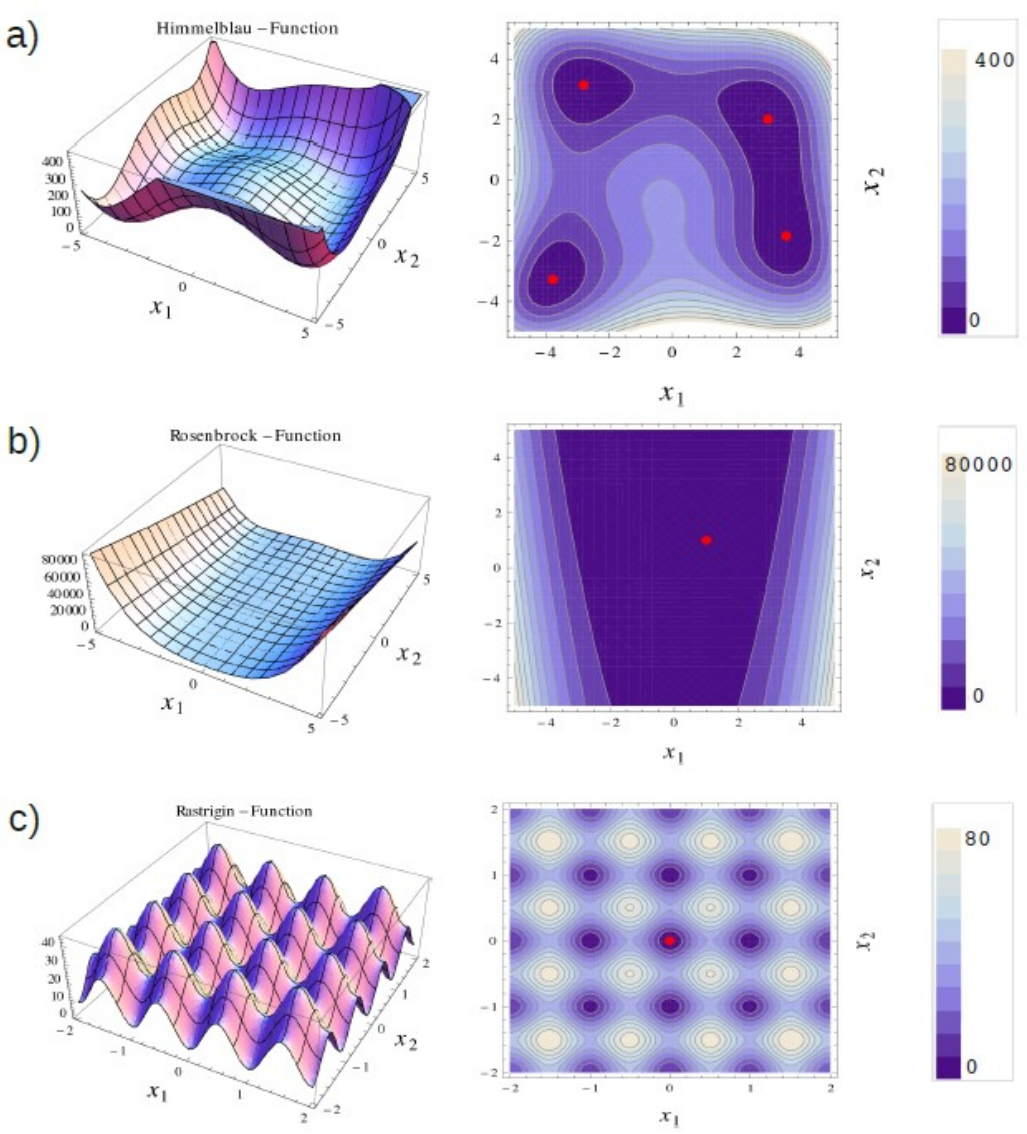




comparison

Comparison of algorithms:

Test function:



comparison

Comparison of algorithms:

Total cost (in function evaluations) for each test function for each global optimization algorithm:

algorithm	Rastrigin function $\chi^2_{limit} = 1$	Rosenbrock function $\chi^2_{limit} = 4.e\text{-}3$	Himmelblau function $\chi^2_{limit} = 5.e\text{-}4$
Bees	1220	14491	101664
PSO	1317	535	770
Genetic	241	533	1626
NS	4230	5080	8720
INS	20	1144	168

Summary

- MAGIX is a very helpful tool for modelling physical and chemical data using an arbitrary external model program.
- It's a highly flexible toolbox where in principle any theoretical external model program can be plugged in.
- MAGIX is able to explore the landscape of the χ^2 function without the knowledge of starting values and calculates probabilities for the occurrence of minima.
- MAGIX can find multiple minima and give informations about confidence intervals for the parameters
- MAGIX package contains the possibility to combine algorithms in a so-called algorithm chain and make use of the advantages of the different algorithms included in the package.
- If the external model program fulfils certain requirements MAGIX can run in a parallel mode to speed up the computation.

For further informations see:

T. Möller *et al.,* "Modeling and Analysis Generic Interface for eXternal numerical codes (MAGIX)", A&A (2012)

Summary

- MAGIX is a very helpful tool for modelling physical and chemical data using an arbitrary external model program.
- It's a highly flexible toolbox where in principle any theoretical external model program can be plugged in.
- MAGIX is able to explore the landscape of the χ^2 function without the knowledge of starting values and calculates probabilities for the occurrence of minima.
- MAGIX can find multiple minima and give informations about confidence intervals for the parameters
- MAGIX package contains the possibility to combine algorithms in a so-called algorithm chain and make use of the advantages of the different algorithms included in the package.
- If the external model program fulfils certain requirements MAGIX can run in a parallel mode to speed up the computation.

For further informations see:

T. Möller *et al.*, "Modeling and Analysis Generic Interface for eXternal numerical codes (MAGIX)", A&A (2012)