

Test cases for XCLASS package

Test cases for **MAGIX** function:

1. **test_MAGIX_demo** (start with "python test_MAGIX_demo.py"):
The script checks the results of MAGIX function using the external model program "DrudeLorentzConv", which depends on eight parameters. In this test case we fit four parameters using the Levenberg-Marquardt algorithm with max. 20 iterations, 8 cores and a variation of 1.e-3 to achieve an useful description of the synthetic data "TwoOscillators_RefFit_R.dat". The data stored in file "TwoOscillators_RefFit_R.dat" are created with the external model program "DrudeLorentzConv", which describes the reflectance of a crystal surface for different frequencies. Although "DrudeLorentzConv" is not an astronomic model program it can be used to test the handling of external model programs by MAGIX. The test script starts the MAGIX function, creates a plot showing the modeled spectrum together with the synthetic data "TwoOscillators_RefFit_R.dat" and compares the modeled spectrum with a previous calculated spectrum. A warning is printed to the screen, if both spectra are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).
2. **test_MAGIX_gDL** (start with "python test_MAGIX_gDL.py"):
The script checks the results of MAGIX function using the external model program "Generalized_Drude-Lorentz", which depends on 41 parameters. In this test case we fit 11 parameters using the Levenberg-Marquardt algorithm with max. 10 iterations, 12 cores and a variation of 1.e-9 to achieve an useful description of the four data sets "Reflectance_Polarization_00_010K.dat", "Reflectance_Polarization_30_010K.dat", "Reflectance_Polarization_90_010K.dat", and "Reflectance_010K_240_Rac_105Grad.dat". These data sets describe the measured reflectances of a crystal for different frequencies and polarization angles, i.e. this test case fits 2D data sets for different frequency ranges, simultaneously. The test script starts the MAGIX function, creates a plot showing the modeled spectra together with the four data sets and compares the modeled spectra with previous calculated spectra. A warning is printed to the screen, if both spectra are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).
3. **Further MAGIX test/example cases:** In addition to the two test cases mentioned above the XCLASS package contains further test/example cases for MAGIX located in "path-to-XCLASS-package/programs/MAGIX/run/examples/". This subdirectory contains example applications for each algorithm included in MAGIX. Additionally, the subdirectory "algorithm-chain/" describes the application of a so-called algorithm chain, i.e. a combination of several algorithms. The first algorithm (here Bees algorithm) is used to explore the parameter space. The two best results, i.e. the parameter vectors with the lowest χ^2 values, are used as initial guess for the second (here Levenberg-Marquardt) algorithm to improve the results of the Bees algorithm. Finally, the Error Estimation algorithm is used to determine the errors of the results of the aforementioned Levenberg-Marquardt algorithm applications.

Test cases for **myXCLASS** function:

1. **test_ArH+_check** (start with "python test_myXCLASS_ArH+_check.py"):
The script checks the result of the myXCLASS function for ArH+ between 617200.0 - 618000.0 MHz. Here, we use two molecules, "Ar-36-H+;v=0;" (with 15 absorption components) and "H2CNH;v=0;" (with one absorption component). Additionally, the molfit

file contains the description of absorption components which include the definition of a source size (which is not used in the myXCLASS function) and some do not contain the definition of a source size. For backward compatibility it is important that the myXCLASS function is able to handle both cases. No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. The test script starts the myXCLASS function, creates a plot showing the modeled spectrum together with the observational data "SgrB2m__ArH+_Data.dat" and compares the modeled spectrum with a previous calculated spectrum. A warning is printed to the screen, if both spectra are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).

2. **test__demo-from-manual** (start with "python test__myXCLASS__demo-from-manual.py"):
The script checks the result of the myXCLASS function for the molfit file "CH3OH__pure.molfit" (located in the demo directory for the myXCLASS function) between 580102.0 - 580546.5 MHz. Here, we use one molecule, "CH3OH;v=0;" (with one absorption component). No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. The test script starts the myXCLASS function, creates a plot showing the modeled spectrum together with the observational data "band1b.dat" and compares the modeled spectrum with a previous calculated spectrum. A warning is printed to the screen, if both spectra are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).
3. **test__different_molfit-formats** (start with "python test__myXCLASS__different_molfit-formats.py"):
The script checks the result of the myXCLASS function for SO2;v=0; between 258000.0 - 259850.0 MHz. The molfit file contains the description of 24 different components which are defined in 24 different formats. So, this test case is used to check, if all allowed formats of the molfit file are handled correctly. No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. The test script starts the myXCLASS function, creates a plot showing the modeled spectrum and compares the modeled spectrum with a previous calculated spectrum. A warning is printed to the screen, if both spectra are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).
4. **test__multiple_molecule_definitions** (start with "python test__myXCLASS__multiple_molecule_definitions.py"):
The script checks, if the myXCLASS function can handle molfit files containing multiple definitions of a molecule. Here, we define a stack of layers, where the first layer describes an emission component of "SO2;v=0;" followed by three absorption components of "SO2;v=0;". The next layer describes an absorption component of "H2CNH;v=0;". The last two components of the stack represent absorption components of "SO2;v=0;". No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. The test script starts the myXCLASS function, creates a plot showing the modeled spectrum and compares the modeled spectrum with a previous calculated spectrum. A warning is printed to the screen, if both spectra are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).
5. **test__no-iso_and_with-iso** (start with "python test__myXCLASS__no-iso_and_with-iso.py"):
The script checks, if the myXCLASS function can handle isotopologues correctly. Here, we use one molecule, "SO2;v=0;" (with one emission component and five absorption components) between 258000.0 - 259850.0 MHz. We use a phenomenological description of

the continuum and describe single dish data. The test script starts the myXCLASS function with and without using isotopologues, creates a plot showing the modeled spectra with and without using isotopologues and compares the modeled spectra with previous calculated spectra. A warning is printed to the screen, if both spectra are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).

Test cases for myXCLASSFit function:

1. **test__ArH+_fit** (start with "python test__myXCLASSFit__ArH+_fit.py"):

This test case is similar to case "test__ArH+_check" for the myXCLASS function, see above. Here, we use two molecules in the molfit file, "Ar-36-H+;v=0;" (with 15 absorption components) and "H2CNH;v=0;" (with one absorption component) to fit observational data "SgrB2m__ArH+_Data.dat" between 617200.0 - 618000.0 MHz. No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. We fit all column densities, velocity widths and offsets for all component using the Levenberg-Marquardt algorithm with max. 50 iterations, 46 cores and a variation of 1.e-3. The test script starts the myXCLASSFit function, creates a plot showing the optimized model spectrum together with the observational data and compares the modeled spectrum with a previous calculated spectrum. A warning is printed to the screen, if both spectra are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).
2. **test__demo-from-manual** (start with "python test__myXCLASSFit__demo-from-manual.py"):

The script checks the result of the myXCLASSFit function for the molfit files "CH3OH__new.molfit", "CH3OH__new__nH+kappa+beta.molfit", "CH3OH__new__Tdoff+Tdslope.molfit", "CH3OH__new__Tdoff+Tdslope+nH+kappa+beta.molfit", "CH3OH__old.molfit", "CH3OH__old__nH+kappa+beta.molfit", "CH3OH__old__Tdoff+Tdslope.molfit", "CH3OH__old__Tdoff+Tdslope+nH+kappa+beta.molfit" (located in the demo directory for the myXCLASSFit function) between 580102.0 - 580546.5 MHz. The different molfit files represents all allowed molfit file definitions, which can be used with the myXCLASSFit function. Here, we use one molecule, "CH3OH;v=0;" (with one absorption component). Source size, column density and velocity width are fitted using the Levenberg-Marquardt algorithm with max. 60 iterations, 8 cores and a variation of 1.e-3. No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. The test script starts the myXCLASSFit function using the different molfit files described above, creates plots showing the modeled spectra together with the observational data "band1b.dat" and compares the modeled spectra with previous calculated spectra. A warning is printed to the screen, if both spectra are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).
3. **test__demo-from-manual__Bees** (start with "python test__myXCLASSFit__demo-from-manual__Bees.py"):

The script starts the myXCLASSFit function for the demo molfit file "CH3OH__old.molfit" between 580102.0 - 580546.5 MHz. Source size, column density and velocity width are fitted using the Bees algorithm with max. 5 iterations, 8 cores. No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. The test script starts the myXCLASSFit function and creates a plot showing the modeled spectrum together with the observational data "band1b.dat". A comparison with previous calculated results is not possible, because the bees algorithm uses random numbers to find a good description of the data, which vary from call to call.

4. **test__demo-from-manual__Genetic** (start with "python test__myXCLASSFit__demo-from-manual__Genetic.py"):
The script starts the myXCLASSFit function for the demo molfit file "CH3OH__old.molfit" between 580102.0 - 580546.5 MHz. Source size, column density and velocity width are fitted using the Genetic algorithm with max. 50 iterations, 10 cores. No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. The test script starts the myXCLASSFit function and creates a plot showing the modeled spectrum together with the observational data "band1b.dat". A comparison with previous calculated results is not possible, because the Genetic algorithm uses random numbers to find a good description of the data, which vary from call to call.
5. **test__demo-from-manual__MCMC** (start with "python test__myXCLASSFit__demo-from-manual__MCMC.py"):
The script starts the myXCLASSFit function for the demo molfit file "CH3OH__old.molfit" between 580102.0 - 580546.5 MHz. Source size, column density and velocity width are fitted using the Markov chain Monte Carlo (MCMC) algorithm with 47 objects, 10 burn-in iterations and max. 20 iterations with 45 cores. No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. The test script starts the myXCLASSFit function and creates a plot showing the modeled spectrum together with the observational data "band1b.dat". A comparison with previous calculated results is not possible, because the MCMC algorithm uses random numbers to find a good description of the data, which vary from call to call.
6. **test__demo-from-manual__NS** (start with "python test__myXCLASSFit__demo-from-manual__NS.py"):
The script starts the myXCLASSFit function for the demo molfit file "CH3OH__old.molfit" between 580102.0 - 580546.5 MHz. Source size, column density and velocity width are fitted using the nested sampling (NS) algorithm with 100 objects, max. 20 iterations, 8 cores. No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. The test script starts the myXCLASSFit function and creates a plot showing the modeled spectrum together with the observational data "band1b.dat". A comparison with previous calculated results is not possible, because the NS algorithm uses random numbers to find a good description of the data, which vary from call to call.
7. **test__demo-from-manual__PSO** (start with "python test__myXCLASSFit__demo-from-manual__PSO.py"):
The script starts the myXCLASSFit function for the demo molfit file "CH3OH__old.molfit" between 580102.0 - 580546.5 MHz. Source size, column density and velocity width are fitted using the particle swarm optimization (PSO) algorithm with max. 10 iterations, 8 cores. No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. The test script starts the myXCLASSFit function and creates a plot showing the modeled spectrum together with the observational data "band1b.dat". A comparison with previous calculated results is not possible, because the PSO algorithm uses random numbers to find a good description of the data, which vary from call to call.
8. **test__demo-from-manual__SA** (start with "python test__myXCLASSFit__demo-from-manual__SA.py"):
The script starts the myXCLASSFit function for the demo molfit file "CH3OH__old.molfit" between 580102.0 - 580546.5 MHz. Source size, column density and velocity width are fitted using the Simulated annealing (SA) algorithm with temperature 1, a reduction

coefficient for the temperature for each reduction cycle of 0.7, 90 reductions, one reheating phase, max. 100 iterations, and 8 cores. No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. The test script starts the myXCLASSFit function and creates a plot showing the modeled spectrum together with the observational data "band1b.dat" and compares the modeled spectrum with previous calculated spectrum. A warning is printed to the screen, if both spectra are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).

9. **test__demo-from-manual__Genetic+LM+Error** (start with "python test__demo-from-manual__Genetic+LM+Error.py"):

The script starts the myXCLASSFit function for the demo molfit file "CH3OH__old.molfit" between 580102.0 - 580546.5 MHz. Source size, column density and velocity width are fitted using an algorithm chain consisting of the Genetic (max. ten iterations, 4 cores), the Levenberg-Marquardt (max. 50 iterations, 28 cores) and the error estimation (calculating the 2sigma errors by using the MCMC method, without using previous results/function calls, with 20 iterations within the burn-in phase, and max. 100 iterations with 47 cores) algorithm. No isotopologues are used. We use a phenomenological description of the continuum and describe single dish data. The test script starts the myXCLASSFit function and creates a plot showing the modeled spectrum together with the observational data "band1b.dat". A comparison with previous calculated results is not possible, because the Genetic algorithm uses random numbers to find a good description of the data, which vary from call to call.

10. **test__myXCLASS+myXCLASSFit__Compare** (start with "python test__myXCLASS+myXCLASSFit__Compare.py"):

The script checks, if the myXCLASS and myXCLASSFit functions produce the same spectrum for the same molfit file. We use a phenomenological description of the continuum, do not use isotopologues, and describe single dish data. The test script starts the myXCLASS and the myXCLASSFit function (with only one iteration where all parameters are kept constant), creates a plot showing both modeled spectra together and compares the modeled spectra. A warning is printed to the screen, if both spectra are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).

11. **test__dust-continuum-fit** (start with "python test__myXCLASSFit__dust-continuum-fit.py"):

The script checks the result of the myXCLASSFit function for the molfit files "SO2__dust-continuum-fit__no-background-defined.molfit", "SO2__dust-continuum-fit__fit-all-dust-parameters.molfit", "SO2__dust-continuum-fit__T_dOff.molfit". Here, two interferometric data files "spw01.dat" between 240140.25 – 242014.75 MHz and "spw23.dat" between 256140.25 - 258014.75 MHz are fitted simultaneously using the Levenberg-Marquardt algorithm with max. 50 iterations, 46 cores, and a variation of 1.e-3. No isotopologues are used. In the first application of the myXCLASSFit function we use a phenomenological description of the continuum and use the molfit file "SO2__dust-continuum-fit__no-background-defined.molfit". In the second run we do not use a phenomenological description of the continuum, define the parameters describing the dust opacity globally, and fit the dust temperature offset for each component. In the last run, we neither use a phenomenological description of the continuum nor a global definition of the dust opacity. Here, we fit all parameters describing the dust opacities and the dust temperature offset and slope for each component. After finishing the fits the script creates plots showing the modeled spectra together with the observational data sets "spw01.dat" and "spw23.dat" and compares the modeled spectra with previous calculated spectra. A warning is printed to the

screen, if both spectra are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).

Test cases for **myXCLASSMapFit** function:

1. **test_myXCLASSMapFit____demo** (start with "python test_myXCLASSMapFit____demo.py"):
The script checks the result of the myXCLASSMapFit function for the first example described in the manual. Here, we use the molfit file "CH3OH__new.molfit" and fit a region of 4 x 5 pixel within the data cube "Orion.methanol.cbc.contsub.image.fits" between 229763.161321 - 229743.627877 MHz. Source size, column density, velocity width and offset and dust temperature offset are fitted using the Levenberg-Marquardt algorithm with max. 15 iterations, 8 cores and a variation of 1.e-3. No isotopologues are used. We use a phenomenological description of the continuum and describe interferometric data. The test script starts the myXCLASSMapFit function and compares the produced fits images and cubes with previous calculated results. A warning is printed to the screen, if both results are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).

Test cases for **LineIdentification** function:

1. **test_LineIdentification____demo** (start with "python test_LineIdentification____demo.py"):
The script checks the result of the LineIdentification function for a synthetic created spectrum. Here, the number of possible molecules is reduced to speed up the identification process for testing. The test run uses a cluster of four nodes, i.e. the script performs 4 single molecule fits simultaneously and requires $4 \times 5 = 20$ core in total. After finishing the LineID process the result is compared with a reference run. A warning is printed to the screen, if both results are not identical (within a tolerance of 1.e-3, i.e. $\text{abs}(\text{spectrum_new} - \text{spectrum_prev}) > 1.e-3$).