dmdd Documentation

Release 0.1

Vera Gluscevic, Sam McDermott

June 15, 2015

Contents

A python package that enables simple simulation and Bayesian posterior analysis of nuclear-recoil data from dark matter direct detection experiments for a wide variety of theories of dark matter-nucleon interactions. The code is being actively developed on GitHub; please feel free to contribute pull requests or raise issues.

`dmdd` has the following features:

- Calculation of the nuclear-recoil rates for various non-standard momentum-, velocity-, and spin-dependent scattering models.
- Calculation of the appropriate nuclear response functions triggered by the chosen scattering model.
- Inclusion of natural abundances of isotopes for a variety of target elements: Xe, Ge, Ar, F, I, Na.
- Simple simulation of data (where data is a list of nuclear recoil energies, including Poisson noise) under different models.
- Bayesian analysis (parameter estimation and model selection) of data using `MultiNest`.

All rate and response functions directly implement the calculations of Anand et al. (2013) and Fitzpatrick et al. (2013) (for non-relativistic operators, in `rate_genNR` and `rate_NR`), and Gresham & Zurek (2014) (for UV-motivated scattering models in `rate_UV`). Simulations follow the prescription from Gluscevic & Peter (2014) and Gluscevic et al. (2015).

CHAPTER 1

Dependencies

All of the package dependencies (listed below) are contained within the Anaconda python distribution, except for `MultiNest` and `PyMultinest`.

For simulations, you will need:

- basic python scientific packages (`numpy`, `scipy`, `matplotlib`)
- `cython`

To do posterior analysis, you will also need:

- `MultiNest`
- `PyMultiNest`

To install these two, follow the instructions here.

CHAPTER 2

Installation

Install `dmdd` either using pip:

```
pip install dmdd
```

or by cloning the repository:

```
git clone https://github.com/veragluscevic/dmdd.git
cd dmdd
python setup.py install
```

CHAPTER 3

Usage

For a quick tour of usage, check out the tutorial notebook; for more complete documentation, read the docs; and for the most important formulas and definitions regarding the `rate_NR` and `rate_genNR` modules, see also here.

CHAPTER 4

Basic Usage

Here is a quick example of basic usage:

```
from dmdd import UV_Model, Experiment, MultinestRun

model1 = UV_Model('SI_Higgs', ['mass', 'sigma_si'], fixed_params={'fnfp_si': 1})
model2 = UV_Model('SD_fu', ['mass','sigma_sd'], fixed_params={'fnfp_sd': -1.1})

xe = Experiment('Xe', 'xenon', 5, 40, 1000, eff.efficiency_Xe)

run = MultinestRun('sim', [xe,ge], model1,{'mass':50.,'sigma_si':70.},
               model2, prior_ranges={'mass':(1,1000), 'sigma_sd':(0.001,1000)})

run.fit()
run.visualize()
```

CHAPTER 5

Attribution

This package was originally developed for Gluscevic et al (2015). If you use this code in your research, please cite this ASCL reference [pending], and the following publications: Gluscevic et al (2015), Anand et al. (2013), Fitzpatrick et al. (2013), and Gresham & Zurek (2014).

CHAPTER 6

API Documentation

6.1 Nuclear Recoil Rates

`dmdd` has three modules that let you calculate the differential rate dR/dE_R, the total rate $R(E_R)$, and the log(likelihood) $\ln \mathcal{L}$ of nuclear-recoil events within three theoretical frameworks:

1. `rate_UV`: rates for a variety of UV-complete theories (from Gresham & Zurek (2014) and Gluscevic et al. (2015)). This takes form factors from formUV.pyx
2. `rate_genNR`: rates for all non-relativistic scattering operators, automatically including interference terms (from Fitzpatrick et al. (2013)). This takes form factors from formgenNR.pyx
3. `rate_NR`: rates for individual nuclear responses compatible with the EFT, **not** automatically including interference terms (from Fitzpatrick et al., 2013). This takes form factors from formNR.pyx

For a specified target element, the natural abundance of its isotopes (with their specific response functions) is taken into account.

Each module is written in cython for fast rate calculations.

6.1.1 rate_UV

rate_UV.dRdQ()

This is the main differential nuclear-recoil rate function. Its output (in units of cts/keV/kg/s) is computed for any one of 18 different scattering scenarios (involving 9 different UV operators), by setting the appropriate `sigma_*` parameter to a non-zero value.

param Q Nuclear recoil energies [keV]

type Q `ndarray`

param mass Dark-matter particle mass [GeV]

param sigma_* Various scattering cross sections [cm^2] off a proton. The value passed will be multiplied with a normalization factor given in `dmdd.PAR_NORMS`. See explanation of suffixes below; the default `PAR_NORMS` are also listed.

param fnfp_* Dimensionless ratio of neutron to proton coupling

param v_lag,v_rms,v_esc Lag, RMS, and escape velocities [km/s]. Note that `v_rms` is 3/2x the standard RMS of a Maxwellian velocity distribution; that is, the default `v_rms` value = 220 km/s.

param rho_x Dark matter energy density.

param element Target nucleus element name (all lower case).

type element `str`

Suffix	Meaning	norm	norm (massless)
_si	spin-independent	1e-47	1e-48
_sd	spin-dependent	1e-42	1e-43
_anapole	anapole	1e-40	1e-45
_magdip	magnetic dipole	1e-38	1e-39
_elecdip	electric dipole	1e-44	1e-45
_LS	$L \cdot S$ generating	1e-44	1e-42
_f1	pseudoscalar-scalar (DM-SM)	1e-47	1e-48
_f2	scalar-pseudoscalar (DM-SM)	1e-42	1e-43
_f3	pseudoscalar-pseudoscalar	1e-41	1e-42

In all cases, the mediator can turn "massless" by appending _massless.

rate_UV.R()

Theoretical total integrated recoil-energy rate.

Integrates `dRdQ()` between `Qmin` and `Qmax` using trapezoidal integration over 100 points.

param efficiency_function Recoil-detection efficiency as a function of nuclear recoil energy.

type efficiency_function `function`

param mass Dark-matter particle mass [GeV]

param sigma_* Various scattering cross sections [in cm^2] off a proton. See `dRdQ()` for details.

param fnfp_* Dimensionless ratio of neutron to proton coupling

param v_lag,v_rms,v_esc Lag, RMS, and escape velocities [km/s]. Note that `v_rms` is 3/2x the standard RMS of a Maxwellian velocity distribution; that is, the default `v_rms` value = 220 km/s.

param rho_x Dark matter energy density.

param element Target nucleus element name (all lower case).

type element `str`

param Qmin,Qmax Nuclear-recoil energy window of experiment.

For reference on other paremeters, see `dRdQ()`.

rate_UV.loglikelihood()

Log-likelihood of array of recoil energies

param Er Array of energies in keV. It can have as many entries as desired.

type Er `np.ndarray`

param efficiency Fractional efficiency as a function of energy. Efficiencies available:

dmdd.eff.efficiency_Ar dmdd.eff.efficiency_Ge dmdd.eff.efficiency_I dmdd.eff.efficiency_KIMS dmdd.eff.efficiency_Xe

right now, all of these are set to be constant and equal to 1.

type efficiency `function`

param Qmin,Qmax Minimum/maximum energy in keVNR of interest, e.g. detector threshold default 2., cutoff default 30.

param mass Dark matter mass in GeV. Default to 50. (optional)

type mass `float`

param sigma_* Various scattering cross sections [in cm^2] off a proton. See `dRdQ()` for details.

param fnfp_* Dimensionless ratio of neutron to proton coupling

param v_lag Velocity of the solar system with respect to the Milky Way in km/s. Default to 220. (optional)

type v_lag `float`

param v_rms 1.5 * (velocity dispersion in km/s) of the local DM distribution. Default to 220. (optional)

type v_rms `float`

param v_esc Escape velocity in km/s of a DM particle in the galactic rest frame. Default to 544. (optional)

type v_esc `float`

param element Name of the detector element. Choice of: 'argon', 'fluorine', 'germanium', 'helium', 'iodine', 'sodium', or 'xenon' Default to 'xenon' (optional)

type element `str`

param rho_x Local DM density in GeV/cm^3. Default to 0.3 (optional)

type rho_x `float`

For reference on other parameters, see `dRdQ()`.

6.1.2 rate_genNR

rate_genNR.dRdQ()

Differential recoil energy spectrum in counts/keV/kg/sec. Its output (in units of cts/keV/kg/s) is computed for any combination of 28 different couplings to nucleons, by setting the appropriate `cXN_*` parameters to a non-zero value.

param Er This is a list of energies in keV. The list must be entered as a numpy array, np.array([..]). It can have as many entries as desired.

type Er `np.ndarray`

param mass Dark matter mass in GeV. Default to 50. (optional)

type mass `float`

param 28 cXN's 28 different np.arrays, all optional These are the EFT coefficients.

X can be any number from 1-15 (except 2). N must be n or p. Any cXN that is entered is a list of coefficients.

The list must be entered as a numpy array, np.array([..]).

c1N and c4N must have three entries, any of which may be zero:

-the first entry is the momentum-independent term.

-the second entry is the coefficient of the q^2/mDM^2 term.

-the third entry is the coefficient of the q^4/mDM^4 term.

c3N and c5Y-c12N must have two entries, any of which may be zero:

-the first entry is the momentum-independent term.

-the second entry is the coefficient of the q^2/mDM^2 term.

c13N-c15N must have one entry. All cXN have mass dimension negative two. The mass scale of the suppression is 500 GeV by default (may be adjusted; see below). By default all coefficients are set to zero.

param c_scale Suppression scale of all cXN coefficients in GeV. From a UV perspective, this could be mediator_mass/sqrt(couplings). Default 500. (optional)

type c_scale `float`

param v_lag Velocity of the solar system with respect to the Milky Way in km/s. Default to 220. (optional)

type v_lag `float`

param v_rms 1.5 * (velocity dispersion in km/s) of the local DM distribution. Default to 220.

param v_esc Escape velocity in km/s of a DM particle in the galactic rest frame. Default to 544.

param element Name of the detector element. Choice of: 'argon', 'fluorine', 'germanium', 'helium', 'iodine', 'sodium', or 'xenon' Default to 'xenon'

type element `str`

param rho_x Local DM density in GeV/cm^3. Default to 0.3

return dRdQ array of differential recoil energy spectrum in counts/keV/kg/sec

rate_genNR.R()

Fractional observed events in counts/kg/sec

Multiply this by an exposure in kg*sec to get a total number of observed events

param efficiency Fractional efficiency as a function of energy. Efficiencies available:

dmdd.eff.efficiency_Ar dmdd.eff.efficiency_Ge dmdd.eff.efficiency_I dmdd.eff.efficiency_KIMS dmdd.eff.efficiency_Xe

right now, all of these are set to be constant and equal to 1.

type efficiency `object`

param Qmin,Qmax Minimum/maximum energy in keVNR of interest, e.g. detector threshold default 2. and detector cutoff default 30.

param mass Dark matter mass in GeV. Default to 50. (optional)

type mass `float`

param 28 different cXN's 28 different np.arrays, all optional See `dRdQ()` for details.

param c_scale Suppression scale of all cXN coefficients in GeV. From a UV perspective, this could be mediator_mass/sqrt(couplings). Default 500. (optional)

type c_scale `float`

param v_lag Velocity of the solar system with respect to the Milky Way in km/s. Default to 220. (optional)

type v_lag `float`

param v_rms 1.5 * (velocity dispersion in km/s) of the local DM distribution. Default to 220. (optional)

type v_rms `float`

param v_esc Escape velocity in km/s of a DM particle in the galactic rest frame. Default to 544. (optional)

type v_esc `float`

param element Name of the detector element. Choice of: 'argon', 'fluorine', 'germanium', 'helium', 'iodine', 'sodium', or 'xenon' Default to 'xenon' (optional)

type element `str`

param rho_x Local DM density in GeV/cm^3. Default to 0.3 (optional)

type rho_x `float`

loglikelihood()

Log-likelihood of array of recoil energies

param Er Array of energies in keV. It can have as many entries as desired.

type Er `np.ndarray`

param efficiency Fractional efficiency as a function of energy. Efficiencies available:

dmdd.eff.efficiency_Ar dmdd.eff.efficiency_Ge dmdd.eff.efficiency_I dmdd.eff.efficiency_KIMS dmdd.eff.efficiency_Xe

right now, all of these are set to be constant and equal to 1.

type efficiency `function`

param Qmin,Qmax Minimum/maximum energy in keVNR of interest, e.g. detector threshold default 2., cutoff default 30.

param mass Dark matter mass in GeV. Default to 50. (optional)

type mass `float`

param 28 cXN's 28 different np.arrays, all optional These are the EFT coefficients. See `dRdQ()` for details.

6.1.3 rate_NR

rate_NR.dRdQM()

This is the rate from the M nuclear response alone in units of cts/keV/kg/s. This is functionally equivalent to the standard spin-independent rate.

param Er An array of keV energies

type Er `ndarray`

param V0 Velocity in km/s

type V0 `float`

param v_lag Lag velocity in km/s.

type v_lag `float`

param v_esc Galactic escape velocity in km/s

type v_esc `float`

param mx Dark matter particle mass in GeV

type mx `float`

param sigp Dark-matter-proton scattering cross section normalized to give about 1 count at LUX when set to 1.

type sigp `float`

param fnfp Dimensionless ratio of neutron to proton coupling.

type fnfp `float`

param elt element name

type elt `str`

param rho_x (optional) Local dark matter density.

type rho_x `float`

rate_NR.dRdQSigPP()

This is the rate from the Sigma" response in units of cts/keV/kg/s.

Takes same parameters as `dRdQM()`

rate_NR.dRdQSigP()

This is the rate from the Sigma' response in units of cts/keV/kg/s.

Takes same parameters as `dRdQM()`

rate_NR.dRdQPhiPP()

This is the rate from the Phi" response in units of cts/keV/kg/s.

Takes same parameters as `dRdQM()`

rate_NR.dRdQDelta()

This is the rate from the Delta response in units of cts/keV/kg/s.

Takes same parameters as `dRdQM()`

rate_NR.dRdQMPhiPP()

This is the rate from the M-Phi" (interference) response in units of cts/keV/kg/s.

Takes same parameters as `dRdQM()`

rate_NR.dRdQSigPDelta()

This is the rate from the Sigma'-Delta (interference) response in units of cts/keV/kg/s.

Takes same parameters as `dRdQM()`

rate_NR.dRdQ()

differential rate for a particular EFT response. Invoking multiple responses is hazardous since some responses interfere. Use rate_genNR for more complicated EFT scenarios.

6.2 API

The following details the objects used in this package.

6.2.1 Experiment

class `dmdd.`**`Experiment`**(*name*, *element*, *Qmin*, *Qmax*, *exposure*, *efficiency_fn*, *tex_name=None*, *energy_resolution=True*)

An object representing a dark-matter direct-detection experiment.

This object packages all the information that defines a single "experiment". For statistical analysis, a list of these objects is passed to initialize an instance of a *MultinestRun* object, or to initialize an instance of a *Simulation* object. It can also be used on its own to explore the capabilities of an experiment with given characteristics. Experiments set up here can either have perfect energy resolution in a given analysis window, or no resolution (controlled by the parameter `energy_resolution`, default being `True`).

Parameters

- **`name`** (`str`) – Name of experiment.
- **`element`** (`str`) – Detector target element. Only single-element targets currently supported.
- **`Qmin,Qmax`** – Nuclear-recoil energy range of experiment [in keV].
- **`exposure`** – Total exposure of experiment [kg-years].
- **`efficiency_fn`** (`function`) – Efficiency as a function of nuclear recoil energy.
- **`tex_name`** (`str`) – Optional; provide this if you want a specific tex name on plots.
- **`energy_resolution`** (`bool`) – If `True`, then the energy of recoil events will be taken into account in likelihood analyses using this experiment; otherwise, not (e.g., for bubble-chamber experiments).

`NminusNbg`(*sigma_val*, *sigma_name='sigma_si'*, *fnfp_name='fnfp_si'*, *fnfp_val=None*, *mass=50.0*, *Nbackground=4*, *v_esc=540.0*, *v_lag=220.0*, *v_rms=220.0*, *rho_x=0.3*)

Expected number of events minus background

Parameters

- **sigma_val** – Scattering cross-section for interaction with proton [cm^2]
- **sigma_name** – Which sigma this corresponds to (i.e., which argument of `rate_UV.R()`)
- **fnfp_name** (`str`) – Which fnfp to use.
- **fnfp_val** – Value of fnfp (optional).
- **mass** – Dark-matter particle mass in GeV.
- **Nbackground** – Number of background events expected.
- **v_esc,v_lag,v_rms,rho_x** – Passed to `rate_UV.R()`.

VminusVesc (*mx*, *v_esc=540.0*, *v_lag=220.0*)

This function returns the value of the minimum velocity needed to produce recoil of energy Qmin, minus escape velocity in Galactic frame.

See Eq 2.3 in (Gluscevic & Peter, 2014).

Zero of this function gives minimal dark-matter particle mass mx that can be detected with this Experiment. This is usually called by *Experiment.find_min_mass()*.

Parameters

- **mx** – WIMP mass [GeV]
- **v_esc** – escape velocity in Galactic frame [km/sec]
- **v_lag** – rotational velocity of the Milky Way [km/sec]

Returns Vmin - Vesc

find_min_mass (*v_esc=540.0*, *v_lag=220.0*, *mx_guess=1.0*)

This finds the minimum dark-matter particle mass detectable with this experiment, by finding a zero of VminusVesc.

Parameters **mx_guess** – guess-value [GeV].

Other parameters documented in *Experiment.VminusVesc()*.

sigma_exclusion (*sigma_name*, *fnfp_name='fnfp_si'*, *fnfp_val=None*, *mass_max=5000*, *Nbackground=4*, *mx_guess=1.0*, *sigma_guess=10000000000.0*, *v_esc=540.0*, *v_lag=220.0*, *v_rms=220.0*, *rho_x=0.3*, *mass_spacing='log'*, *nmass_points=100*, *make_plot=False*, *ymax=None*)

Makes exclusion curve for a chosen sigma parameter.

Calculates *Experiment.sigma_limit()* for a grid of masses, and interpolates.

Parameters

- **sigma_name** (`str`) – Name of cross-section to exclude.
- **mass_spacing** – 'log' (logarithmic) or 'lin' (linear) spacing for mass grid.
- **nmass_points** – Number of points to calculate for mass grid.
- **make_plot** – Whether to make the plot. If `False`, then function will return arrays of `mass, sigma`.
- **ymax** – Set the y maximum of plot axis.

For other parameters, see *Experiment.sigma_limit()*

sigma_limit(*sigma_name='sigma_si'*, *fnfp_name='fnfp_si'*, *fnfp_val=None*, *mass=50.0*, *Nbackground=4*, *sigma_guess=10000000000.0*, *mx_guess=1.0*, *v_esc=540.0*, *v_lag=220.0*, *v_rms=220.0*, *rho_x=0.3*)

Returns value of sigma at which expected number of dark-matter induced recoil events is equal to the number of expected background events, N = Nbg, in order to get a rough projected exclusion for this experiment.

Parameters

- **sigma_guess** – Initial guess for solver.
- **mx_guess** – Initial guess for dark-matter particle mass in order to find the minimum mass detectable from experiment (*Experiment.find_min_mass()*).

For other arguments, see *Experiment.NminusNbg()*

6.2.2 Model

class dmdd.**Model**(*name*, *param_names*, *dRdQ_fn*, *loglike_fn*, *default_rate_parameters*, *tex_names=None*, *fixed_params=None*, *modelname_tex=None*)

A generic class describing a dark-matter scattering model.

This object facilitates handling of a "hypothesis" that describes the scattering interaction at hand (to be used either to simulate recoil spectra, or to fit them). There is an option to give any parameter a fixed value, which will not be varied if the model is used to fit data.

Subclassed by *UV_Model*.

Parameters

- **name** (str) – Name of the model, matching the operator(s) name. It cannot have spaces.
- **param_names** (list) – Names of the parameters.
- **dRdQ_fn** (function) – Appropriate rate function.
- **loglike_fn** (function) – Function that returns the log-likelihood of an array of event energies, given experimental and astrophysical parameters. Must take `Q, eff_fn, **kwargs` as arguments.
- **default_rate_parameters** (dict) – Default parameters to be passed to rate function.
- **tex_names** (dict) – Dictionary of LaTeX names of parameters.
- **fixed_params** (dict) – Parameters of model that are not intended to be fit for.

class dmdd.**UV_Model**(*name*, *param_names*, ***kwargs*)

Subclass of Model implementing UV-complete scattering models. Rate function and log-likelihood function are taken from the `rate_UV` module.

6.2.3 Simulation

class dmdd.**Simulation**(*name*, *experiment*, *model*, *parvals*, *path='/home/docs/.dmdd/simulations_uv/'*, *force_sim=False*, *asimov=False*, *nbins_asimov=20*, *plot_nbins=20*, *plot_theory=True*, *silent=False*)

A simulation of dark-matter direct-detection data under a given experiment and scattering model.

This object handles a single simulated data set (nuclear recoil energy spectrum). It is generaly initialized and used by the *MultinestRun* object, but can be used stand-alone.

Simulation data will only be generated if a simulation with the right parameters and name does not already exist, or if `force_sim=True` is provided upon *Simulation* initialization; if the data exist, it will just be read in. (Data is a list of nuclear recoil energies of "observed" events.) Initializing *Simulation* with given parameters for the first time will produce 3 files, located by default at `$DMDD_PATH/simulations` (or `./simulations` if `$DMDD_PATH` not defined):

•.dat file with a list of nuclear-recoil energies (keV), drawn from a

Poisson distribution with an expected number of events given by the underlying scattering model.

•.pkl file with all relevant initialization parameters for record

•.pdf plot of the simulated recoil-energy spectrum with simulated

data points (with Poisson error bars) on top of the underlying model

Parameters

- **name** (`str`) – Identifier for simulation (e.g. 'sim1')
- **experiment** (*Experiment*) – Experiment for simulation.
- **model** (*Model*) – Model under which to simulate data.
- **parvals** (`dict`) – Values of model parameters. Must contain the same parameters as `model`.
- **path** (`str`) – The path under which to store the simulations.
- **force_sim** (`bool`) – If `True`, then redo the simulations no matter what. If `False`, then the simulations will be redone if and only if the given simulation parameters don't match what has already been simulated for this simulation name.
- **asimov** – Do asimov simulations. Not currently implemented.
- **nbins_asimov** – Number of asimov bins.
- **plot_nbins** – Number of bins to bin data in for rate plot.
- **plot_theory** – Whether to plot the "true" theoretical rate curve along with the simulated data.
- **silent** – If `True`, then print messages will be suppressed.

plot_data(*plot_nbins=20*, *plot_theory=True*, *save_plot=True*, *make_plot=True*, *return_plot_items=False*)

Plot simuated data.

Parameters

- **plot_nbins** – Number of bins for plotting.
- **plot_theory** – Whether to overplot the theory rate curve on top of the data points.
- **save_plot** – Whether to save plot under `self.plotfile`.
- **make_plot** – Whether to make the plot. No reason really to ever be false unless you only want the "plot items" returned if `return_plot_items=True` is passed.
- **return_plot_items** – If `True`, then function will return lots of things.

simulate_data()

Do Poisson simulation of data according to scattering model's dR/dQ.

6.2.4 MultinestRun

class dmdd.**MultinestRun**(*sim_name*, *experiments*, *sim_model*, *param_values*, *fit_model*, *prior_ranges*, *prior='logflat'*, *sim_root='/home/docs/.dmdd/simulations_uv/'*, *chains_root='/home/docs/.dmdd/chains_uv/'*, *force_sim=False*, *asimov=False*, *nbins_asimov=20*, *n_live_points=2000*, *evidence_tolerance=0.1*, *sampling_efficiency=0.3*, *resume=False*, *basename='1-'*, *silent=False*, *empty_run=False*)

This object controls a single simulated data set and its MultiNest analysis.

This is a "master" class of dmdd that makes use of all other objects. It takes in experimental parameters, particle-physics parameters, and astrophysical parameters, and then generates a simulation (if it doesn't already exist), and prepares to perform MultiNest analysis of simulated data. It has methods to do a MultiNest run (*MultinestRun.fit()*) and to visualize outputs (*visualize()*). *Model* used for simulation does not have to be the same as the *Model* used for fitting. Simulated spectra from multiple experiments will be analyzed jointly if MultiNest run is initialized with a list of appropriate *Experiment* objects.

The likelihod function is an argument of the fitting model (*Model* object); for UV models it is set to dmdd.rate_UV.loglikelihood(), and for models that would correspond to rate_genNR, dmdd.rate_genNR.loglikelihood() should be used. Both likelihood functions include the Poisson factor, and (if energy_resolution=True of the *Experiment* at hand) the factors that evaluate probability of each individual event (i.e. each recoil-energy measurement), given the fitting scattering model.

MultiNest-related files produced by this object will go to a directory, under $DMDD_MAIN_PATH, with the name defined by the parameters passed. This directory name will be accessible via self.chainspath after the object is initialized.

Parameters

- **sim_name** (str) – The name of the simulation (e.g. 'sim1')
- **experiments** (list) – A list of *Experiment* objects, or a single such object.
- **sim_model** (*Model*) – The true underlying model for the simulations (name cannot have spaces).
- **param_values** (dict) – The values of the parameters for sim_model.
- **fit_model** (*Model*) – The model for MultiNest to fit to the data. Does not have to be the same as sim_model, but can be. Its name cannot have spaces.
- **prior_ranges** (dict) – Dictionary of prior ranges for parameters of fit_model. e.g. {'mass':(1,1000), 'sigma_si':(0.1,1e4), etc....}
- **prior** (str) – either 'logflat' or 'flat'
- **sim_root** (str) – The path under which to store the simulations.
- **chains_root** (str) – The path under which to store the Multinest chains.
- **force_sim** (bool) – If *True*, then redo the simulations no matter what. If *False*, then the simulations will be redone if and only if the given simulation parameters don't match what has already been simulated for this sim_name.
- **asimov** – Do asimov simulations. Not currently implemented.
- **nbins_asimov** – Number of asimov bins.
- **n_live_points,evidence_tolerance,sampling_efficiency,resume,basename** – Parameters to pass to MultiNest, defined in the PyMultiNest documentation.
- **silent** – If True, then print messages will be suppressed.
- **empty_run** – if True, then simulations are not initialized.

fit (*force_run=False*)
Runs MultiNest; parameters set by object initialization.

Parameters **force_run** – If `True`, then fit will re-run; by default, it will not, unless the simulation data has changed, or chains don't exist.

flat_prior (*cube*, *ndim*, *nparams*)
Flat prior, passed to MultiNest.

Converts unit cube into correct parameter values based on flat prior within range defined by `self.prior_ranges`.

get_evidence ()
Returns evidence from stats file produced by MultiNest.

global_bestfit ()
Returns maximum a posteriori values for parameters.

logflat_prior (*cube*, *ndim*, *nparams*)
Logflat prior, passed to MultiNest.

Converts unit cube into correct parameter values based on log-flat prior within range defined by `self.prior_ranges`.

loglikelihood_total (*cube*, *ndim*, *nparams*)
Log-likelihood function used by MultiNest.

Parameters **ndim, nparams** (*cube,*) – Params required by MulitNest.

return_chains_loglike ()
Returns MultiNest chains and equal-weighted posteriors.

visualize (***kwargs*)
Makes plots of data for each experiment with theoretical and best-fit models.

Also makes 2-d posteriors for each fitted parameter vs. every other. These plots get saved to `self.chainspath`.

Parameters ****kwargs** – Keyword arguments passed to `dmdd.dmdd_plot.plot_2d_posterior()`.

d