obscura

Release v1.0.0

Timon Emken

Jun 17, 2022

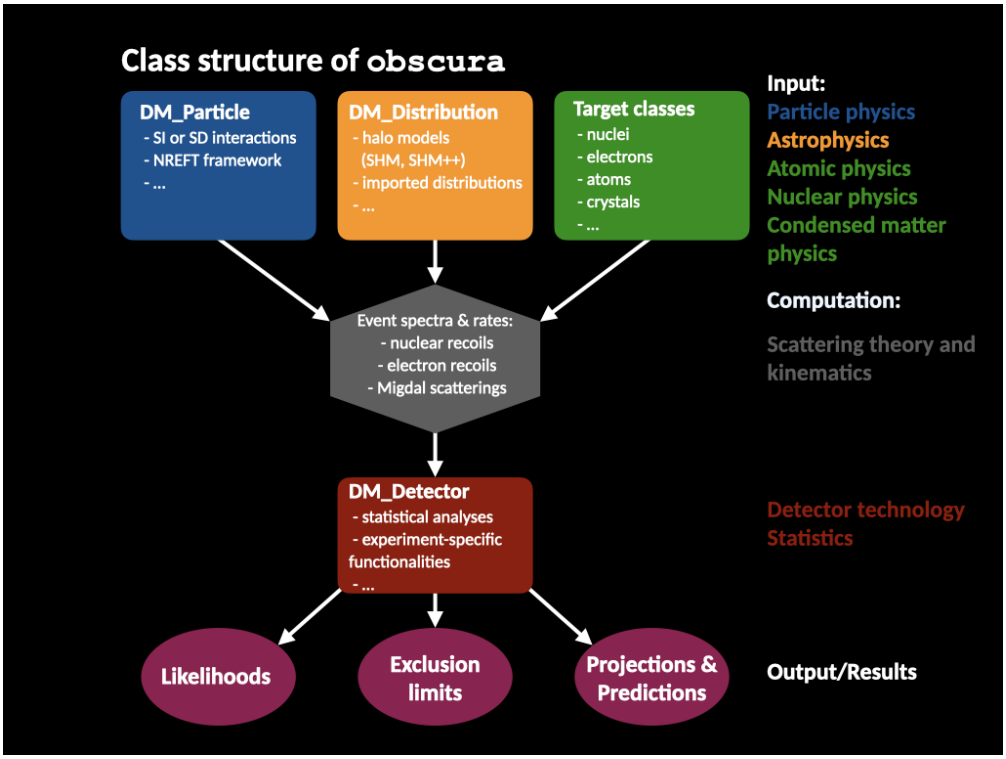
CONTENTS:

A modular C++ tool and library for dark matter direct detection computations for both nuclear and electron recoil experiments.

The purpose of this documentation or manual is to provide insight into the polymorphic class structure of *obscura* and how it can be applied in different contexts. It should also serve as a guide and describe the usage of *obscura* via code examples.

The documentation does not contain a review of the physics implemented in the library. For more physics details, we refer to e.g. chapter 3 of [Emken2019] or [Nobile2021].

If you want to contribute to *obscura*, please check out the contribution guidelines.



CHAPTER

ONE

1. GETTING STARTED

1.1 Installation

Before building *obscura*, there are a few libraries that need to be installed.

1.1.1 Dependencies

1. boost

To install *boost* on a Mac, we can use homebrew

```
brew install boost
```

On Linux machines, run:

```
sudo apt-get update && sudo apt-get install -yq libboost-all-dev
```

2. libconfig

To install *boost* on a Mac, we can use homebrew

```
brew install libconfig
```

On Linux machines, you can build *libconfig* via:

```
wget https://hyperrealm.github.io/libconfig/dist/libconfig-1.7.2.tar.gz
tar -xvzf libconfig-1.7.2.tar.gz
pushd libconfig-1.7.2
./configure
make
sudo make install
popd
```

3. libphysica

libphysica does not need to be installed. It will be downloaded and compiled during the CMake build.

1.1.2 Download & Build

The *obscura* source code can be downloaded by cloning this git repository:

```
git clone https://github.com/temken/obscura.git
cd obscura
```

The code is compiled and the executable and library is built by CMake. To build run the following commands from the repository's root folder.:

```
cmake -E make_directory build
cd build
cmake -DCMAKE_BUILD_TYPE=Release -DCODE_COVERAGE=OFF ..
cmake --build . --config Release
cmake --install .
```

If everything worked well, the executable and library file are created as:

```
bin/obscura
lib/libobscura.a
```

By default, *obscura* will be built as a static library. It is also possible to build it as shared library by adding the following option to the configuration step.:

```
cmake -DBUILD_SHARED_LIBS=ON -DCMAKE_BUILD_TYPE=Release -DCODE_COVERAGE=OFF ..
```

In that case, the library file after installation is:

```
lib/libobscura.so
```

1.2 Using *obscura* as a tool

Obscura can be used as a tool and builds an executable which can be run from */bin/* via:

```
./obscura config.cfg
```

As can be seen in the /src/main.cpp file, this script computes direct detection limits and saves them in the */results/* folder. The specifications of the exclusion limits (DM physics and halo model, statistics, experiment, mass range,...) are defined in a configuration file, in this case *config.cfg*. For the handling of configuration files, *obscura* relies on libconfig.

1.2.1 The configuration file

The configuration file contains all input parameters necessary to define the various *obscura* models.

Warning: The import of these parameters via libconfig is very case-sensitive. A float parameter has to be set to e.g. *1.0*, and **not** just *1*.

```
//obscura - Configuration File

//ID
    ID              =       "test";

//Dark matter particle
    DM_mass                 =       0.1;            // in GeV
    DM_spin                 =       0.5;
    DM_fraction             =       1.0;            // the DM particle's fractional␣
↪abundance (set to 1.0 for 100%)
    DM_light                =       false;          // Options: true or false. low mass␣
↪mode

    DM_interaction          =       "SI";           // Options: "SI" or "SD"

    DM_isospin_conserved            =       true;           // only relevant for SI and␣
↪SD
    DM_relative_couplings           =       (1.0, 0.0); //relation between proton␣
↪(left) and neutron (right) couplings.
                                                                                        ␣
↪    //only relevant if 'DM_isospin_conserved' is false.
    DM_cross_section_nucleon        =       1.0e-36;        //in cm^2
    DM_cross_section_electron       =       1.0e-36;        //in cm^2 (only relevant␣
↪for SI and SD)
    DM_form_factor          =       "Contact";      // Options: "Contact", "Electric-
↪Dipole", "Long-Range", "General"
                                                                                        ␣
↪    //(only relevant for SI)
    DM_mediator_mass        =       0.0;            // in MeV (only relevant if 'DM_form_
↪factor' is "General")

//Dark matter distribution
    DM_distribution         =       "SHM";          //Options: "SHM", "SHM++", "File"
    DM_local_density        =       0.4;            //in GeV / cm^3

    //Options for "SHM" and "SHM++"
            SHM_v0          =       220.0;                          //in km/sec
            SHM_vObserver   =       (0.0, 232.0, 0.0);      //in km/sec
            SHM_vEscape     =       544.0;                          //in km/sec
    //Options for "SHM++"
            SHMpp_eta       =       0.2;
            SHMpp_beta      =       0.9;
    //Options for "File" (The file has to be a 2-column table of format v[km/sec] ::␣
↪f(v) [sec/km])
```

(continues on next page)

(continued from previous page)

```
            file_path  = "DM_Speed_PDF.txt";

//Dark matter detection experiment
    DD_experiment   =       "Electron recoil";      //Options for nuclear recoils:
↪"Nuclear recoil", "DAMIC_N_2011", "XENON1T_N_2017", "CRESST-II","CRESST-III", "CRESST-
↪surface"
                                                    //Options for electron recoils:
↪"Semiconductor","protoSENSEI@MINOS","protoSENSEI@surface", "SENSEI@MINOS", "CDMS-HVeV_
↪2018", "CDMS-HVeV_2020", "Electron recoil", "XENON10_S2", "XENON100_S2", "XENON1T_S2",
↪"DarkSide-50_S2"

    //Options for user-defined experiments ("Nuclear recoil", "Electron recoil", and
↪"Semiconductor")
      //General
      DD_exposure           =       1.0;    //in kg years
      DD_efficiency                 =       1.0;    //flat efficiency
      DD_observed_events    =       0;              //observed signal events
      DD_expected_background        =       0.0;    //expected background events

      //Specific options for "Nuclear recoil"
      DD_targets_nuclear    =       (
                                    (4.0, 8),
                                    (1.0, 20),
                                    (1.0, 74)
                                );                                  // Nuclear targets defined␣
↪by atom ratio/abundances and Z
      DD_threshold_nuclear    =     4.0;    //in keV
      DD_Emax_nuclear         =     40.0;   //in keV
      DD_energy_resolution    =     0.0;    //in keV

      //Specific options for "Electron recoil" and "Semiconductor:
      DD_target_electron    =       "Xe";   //Options for "Electron recoil":        "Xe
↪", "Ar"
                                                          //Options for
↪"Semiconductor":  "Si", "Ge"
      DD_threshold_electron =       4;              //In number of electrons or␣
↪electron hole pairs.

//Computation of exclusion limits
    constraints_certainty   =       0.95;   //Certainty level
    constraints_mass_min    =       0.02;   //in GeV
    constraints_mass_max    =       1.0;    //in GeV
    constraints_masses      =       10;
```

1.3 Using *obscura* as a library

If we want to use *obscura* functions in an external code, we can do so and import it as a library. We recommend to do this inside your CMake build, where *obscura* can be downloaded, built, included, and linked automatically during the build of your code.

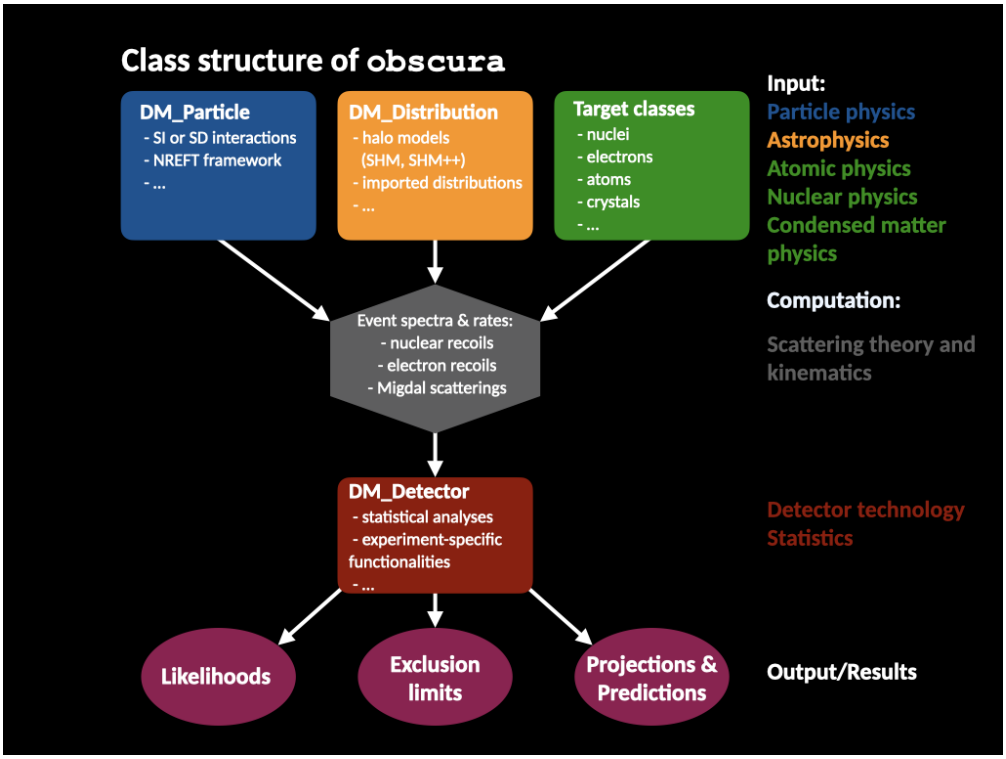
As an instructional example this repository contains a C++ project template built with CMake that imports and uses the *obscura* library.

CHAPTER

TWO

2. THE MODULAR STRUCTURE OF *OBSCURA*

The computation of e.g. the electron recoil spectrum probed in direct detection experiments combines inputs from various fields of physics. We need to specify the assumed *particle physics* of the DM particle. The properties of the DM halo of the Milky way is an important *astrophysics* input. For the description of the target particles, and how they react to a kick from an incoming DM particle, we need to include knowledge of *atomic*, *nuclear*, and *condensed matter physics*. In order to make predictions, we furthermore need to define the *detection* experiments specifications. Finally, the result of such an experiment needs to be interpreted using *statistics*.



This high level of modularity in this type of calculation needs to be reflected in the code's polymorphic structure. The goal of *obscura* is to provide for each of the different inputs one generic interface or abstract base class, that comprises the general required functionalities, without specifying the detailed implementations further. These depend on a multitude of assumptions which can change in different projects, for different users, etc.

If the base classes are defined properly, it is also possible and straight-forward to

1. extend *obscura* by implementing further derived classes overriding the virtual functions of the base class.
2. design research software that is agnostic to the detailed implementation and thereby very generally applicable to a variety of scenarios. As long as our scientific functions are formulated in terms of these base functions, they will be able to handle any new implementation that comes in the form of derived classes.

The three most important abstract base classes of *obscura* are

1. `DM_Particle`
2. `DM_Distribution`
3. `DM_Detector`

We will discuss the interface each of these classes provide in more detail. But first we take a look at the detection targets in direct DM search experiments, namely nuclei, bound electrons in atoms, and bound electrons in crystals.

CHAPTER

THREE

3. THE TARGET CLASSES

The basic hope of direct detection experiments is that the DM particles from the galactic halo occasionally collide with ordinary particles, see e.g. [Nobile2021]. The original target of direct DM searches were nuclear recoils. Later on also electron targets gained more and more attention in the context of sub-GeV dark matter [Essig2012].

In *obscura* each target type is represented by a class, that can e.g. be passed to the cross section functions of the `DM_Particle` class, see *4. The DM_Particle classes*.

3.1 Nuclear targets

For nuclear recoil experiments, we define two target classes, `Isotope` and `Nucleus` that are declared in /include/obscura/Target_Nucleus.hpp.

3.1.1 The `Isotope` class

A nuclear isotope is characterized by the number Z of protons and A of nucleons (protons *and* neutrons), its mass, spin, and average spin contribution for protons and neutrons as required e.g. in the context of spin-dependent nuclear interactions.

3.1.2 The `Nucleus` class

In addition to `Isotope`, we also define a `Nucleus` class which mainly consists of a number of isotopes with given relative abundances.

Construction of nuclear targets

There are different ways to construct instances of `Isotope` and `Nucleus`.

Example: Assume we are interested in oxygen as a target, either the isotope O-16 or the element of various isotopes.

```
#include "obscura/Target_Nucleus.hpp"

// ...

// We can define O-16 via the constructor.
Isotope oxygen_16(8,16);
```

This instance of an oxygen isotope however has no knowledge of e.g. its spin or relative abundance in nature.

For this purpose, *obscura* contains a nuclear data set, see /data/Nuclear_Data.txt ([Bednyakov2005] [Klos2013]), which can be accessed through the following function defined in *Target_Nucleus.hpp*.

```
extern Isotope Get_Isotope(unsigned int Z, unsigned int A);
extern Nucleus Get_Nucleus(unsigned int Z);
extern Nucleus Get_Nucleus(std::string name);
```

Using these functions, we can construct isotopes and nuclei simply as

```
#include "obscura/Target_Nucleus.hpp"

// ...

Isotope oxygen_16 = Get_Isotope(8,16);
Nucleus oxygen = Get_Nucleus(8);
Nucleus oxygen_alternative = Get_Nucleus("O");
```

The last two lines construct an instance of the `Nucleus` class containing all isotopes of oxygen including their relative abundance, spin, and average spin contribution of protons and neutrons.

3.2 Electron targets in atoms

For sub-GeV DM searches, an important target are electrons bound in atoms [Essig2012]. To take into account the fact that electrons are bound states, we need to evaluate the *ionization form factor* or *atomic response function* for each electronic orbital [Catena2019].

The target classes for atomic electrons are declared in /include/obscura/Target_Atom.hpp.

3.2.1 The `Atomic_Electron` class

The first target class in this context is `Atomic_Electron`.

By constructing an instance of this class, the tabulated ionization form factor is imported from /data/Form_Factors_Ionization/.

3.2.2 The `Atom` class

Having target classes for nuclei and bound electrons, we can combine them into a single atomic target, consisting of a nucleus and a number of bound electrons.

3.2.3 Included ionization form factors

At this point, *obscura* comes with the ionization form factors of

- Xenon (5p, 5s, 4d, 4p, 4s)
- Argon (3p, 3s, 2p, 2s, 1s)

The tables can be found under /data/Form_Factors_Ionization/. They have been tabulated using the DarkARC code as described in detail in [Catena2019].

The easiest way to access the ionization form factors is by constructing an instance of `Atom`, as seen in this example.

```cpp
#include "libphysica/Natural_Units.hpp"

#include "obscura/Target_Atom.hpp"

using namespace libphysica::natural_units;
// ...

Atom xenon("Xe");
Atom argon("Ar");

// For example, to access the ionization form factor of xenon's 5s (quantum numbers n=5, l=0) orbital for a given momentum transfer q and energy E_e:
int n = 5; int l = 0;
double q = 0.5 * keV;
double E_e = 10.0 * eV;

std::cout << xenon.Electron(n, l).Ionization_Form_Factor(q, E_e) << std::endl;
```

3.3 Electron targets in crystals

One of the most important targets for sub-GeV DM detectors are crystals, such as e.g. semiconductors [Essig2016]. The electronic properties of the target material is encapsulated in the crystal form factor which is tabulated and can be found in /data/Semiconductors/. The included crystals are

- Silicon semiconductors
- Germanium semiconductors

The tables have been generated using QEdark, a module of Quantum ESPRESSO.

Also for crystals, *obscura* contains a target class `Crystal` declared in /include/obscura/Target_Crystal.hpp.

The crystal form factor, similarly to the ionization form factors, are imported by the class constructor. Here is an example of how to access the crystal form factor.

```cpp
#include "libphysica/Natural_Units.hpp"

#include "obscura/Target_Crystal.hpp"

using namespace libphysica::natural_units;

// ...

Crystal silicon("Si");
Crystal germanium("Ge");

double q = 0.5 * keV;
double E_e = 10.0 * eV;

std::cout << silicon.Crystal_Form_Factor(q, E_e) << std::endl;
std::cout << germanium.Crystal_Form_Factor(q, E_e) << std::endl;
```

CHAPTER

FOUR

4. THE DM_PARTICLE CLASSES

The DM_Particle class and its derived classes are responsible for the particle physics aspects of direct detection. In particular, an instance of DM_Particle entails the particle properties of a DM candidate particle, such as its mass, spin, and its differential and total interaction cross sections with nuclei or electrons. The base class's functions provide an interface that is sufficient for the calculation of e.g. event rates at direct DM search experiments. Furthermore, it contains a number of functions regarding scattering angles, their distributions and sampling, which might not be relevant for direct detection, but can be used in the context of e.g. MC simulations.

4.1 The interface / base class

The abstract base class is defined in /include/obscura/DM_Particle.hpp and all the member functions and parameters can be seen there.

The most important (virtual) functions for direct detection specific calculations are the differential cross sections.

```
//Differential cross sections for nuclear targets
virtual double dSigma_dq2_Nucleus(double q, const Isotope& target, double vDM, double␣
↪param = -1.0) const { return 0.0; };
double dSigma_dER_Nucleus(double ER, const Isotope& target, double vDM, double param = -
↪1.0) const;
double d2Sigma_dER_dEe_Migdal(double ER, double Ee, double vDM, const Isotope& isotope,␣
↪Atomic_Electron& shell) const;

// Differential cross section for electron targets
virtual double dSigma_dq2_Electron(double q, double vDM, double param = -1.0) const {␣
↪return 0.0; };
virtual double d2Sigma_dq2_dEe_Ionization(double q, double Ee, double vDM, Atomic_
↪Electron& shell) const { return 0.0; };
virtual double d2Sigma_dq2_dEe_Crystal(double q, double Ee, double vDM, Crystal&␣
↪crystal) const { return 0.0; };
```

We point out that here we have to pass instances of the target classes discussed in the previous section (i.e. nuclear isotopes, atomic electrons, and electrons in crystals). Also included is a simple implementation of Migdal scatterings with atomic targets based on [Essig2020].

The most standard DM candidate considered in the direct detection literature is a WIMP with SI or SD interactions. *obscura* contains derived classes for each of these scenarios, which are declared in /include/obscura/DM_Particle_Standard.hpp.

4.2 Spin-Independent (SI) interactions

The differential cross section for SI nuclear interactions is given by

$$\frac{\mathrm{d}\sigma_N^{\mathrm{SI}}}{\mathrm{d}E_R} = \frac{m_N}{2\pi v_\chi^2}\left[f_p Z + f_n(A-Z)\right]^2 \left|F_N^{\mathrm{SI}}\left(E_R\right)\right|^2 .$$

For details, we refere to e.g. chapter 3.4 of [Emken2019].

The class `DM_Particle_SI` is derived from `DM_Particle` and evaluates the cross sections of SI interactions with nuclei (and electrons).

The following example demonstrates how to

- construct an instance of `DM_Particle_SI` that describes a DM particle of 10 GeV mass.
- set the SI proton cross section to $\sigma_p = 10^{-40}\mathrm{cm}^2$, the electron cross section of $\sigma_e = 10^{-36}\mathrm{cm}^2$.
- to evaluate the differential and total scattering cross section with argon nuclei.

```
#include "libphysica/Natural_Units.hpp"

#include "obscura/DM_Particle_Standard.hpp"

using namespace libphysica::natural_units;

// ...

// Declare the DM particle
obscura::DM_Particle_SI dm(10.0 * GeV);
dm.Set_Sigma_Proton(1.0e-40 * cm * cm);
dm.Set_Sigma_Electron(1.0e-36 * cm * cm);

// Define the target
obscura::Isotope argon = obscura::Get_Isotope(18, 40);

// Evaluate cross sections
double E_R = 1.0 * keV;
double v_DM = 300.0 * km/sec;
double diff_cross_section = dm.dSigma_dER_Nucleus(E_R, argon, v_DM);
double tot_cross_section = dm.Sigma_Total_Nucleus(argon, v_DM);

// Convert to other units
std::cout <<In_Units(diff_cross_section, cm * cm / keV)<<std::endl;
std::cout <<In_Units(tot_cross_section, cm * cm)<<std::endl;
```

4.3 Spin-Dependent (SD) interactions

The differential cross section for SD nuclear interactions is given by

$$\frac{\mathrm{d}\sigma_N^{\mathrm{SD}}}{\mathrm{d}E_R} = \frac{2m_N}{\pi v_\chi^2}\frac{J+1}{J}\left(f_p\langle S_p\rangle + f_n\langle S_N\rangle\right)^2 F_N^{\mathrm{SD}}(E_R)^2\Big|$$

Similarly to `DM_Particle_SI`, we also define a `DM_Particle_SD` class, which evaluates this cross section for nuclear targets with spin $S \neq 0$.

CHAPTER

FIVE

5. THE DM_DISTRIBUTION CLASSES

In order to make predictions for direct detection experiments, the statistical properties of the incoming DM flux need to be specified. In particular, we need to know how many DM particles pass through the detector and with what energy. In other words, we need to know the DM particle flux, or alternatively the local DM density and the energy distribution.

5.1 The interface / base class

The class `DM_Distribution`, that is declared in /include/obscura/DM_Distribution.hpp, is an abstract base class or interface that defines all the functions we require to characterize a distribution and flux of DM particles.

Most importantly, the class provides interfaces to probability density functions (PDFs) for the DM particles' velocity or speed, their local energy density, differential particle flux, etc.

5.2 The standard halo model (SHM)

The conventional assumptions on the halo DM particles' properties is the Standard Halo Model (SHM). The SHM describes the galactic DM by a truncated Maxwell-Boltzmann distribution. It is characterized by the following 4 parameters (for details see e.g. chapter 3.2 of [Emken2019]

$$\rho_\chi, v_0, v_{\rm esc}, \mathbf{v}_{\rm obs}$$

In /include/obscura/DM_Halo_Models.hpp we define the `Standard_Halo_Model` class which is an implemenation of this model. It is a derived class of `DM_Distribution`.

We can construct the SHM model by the default constructor, which assumes default values for the 4 parameters.

```
#include "obscura/DM_Halo_Models.hpp"

// ...

obscura::Standard_Halo_Model shm;
```

Or we define the parameters explicitly.

```
#include "libphysica/Natural_Units.hpp"

#include "obscura/DM_Halo_Models.hpp"

using namespace libphysica::natural_units;
```

(continues on next page)

(continued from previous page)

```
// ...
double rho = 0.4 * GeV / cm / cm / cm;
double v_0 = 230.0 * km / sec;
double v_esc = 600 * km / sec;
double v_obs = 232.0 * km / sec;
obscura::Standard_Halo_Model shm(rho, v_0, v_obs, v_esc);
```

5.3 The SHM++

As a second example for a DM halo model, *obscura* also implements the SHM++ as proposed in [Evans2019].

Since it extends the SHM, the corresponding class `SHM_Plus_Plus` is a derived class of `Standard_Halo_Model` which is in turn derived from `DM_Distribution`. The class is also declared in /include/obscura/DM_Halo_Models.hpp.

This halo model can be constructed and used essentially identically to the SHM.

5.4 Imported DM distributions

It is also possible to import a DM distribution from a file. This is the purpose of the `Imported_DM_Distribution` class, another derived class of `DM_Distribution` which can be found in /include/obscura/DM_Distribution.hpp.

As input file, we need a two-column table of the DM speed PDF using the format (v[km/sec] :: f(v) [sec/km]). Additionally we need to specify the local DM density.

Here is an example of using this class assuming a tabulated speed pdf given in the file *DM_Speed_PDF.txt*.

```
#include "libphysica/Natural_Units.hpp"

#include "obscura/DM_Distribution.hpp"

using namespace libphysica::natural_units;

// ...
double rho = 0.4 * GeV / cm / cm / cm;
obscura::Imported_DM_Distribution dm_distribution(rho, "DM_Speed_PDF.txt");
```

CHAPTER

SIX

6. THE DM_DETECTOR CLASSES

The details of a direct detection experiment are summarized in the DM_Detector class declared in /include/obscura/Direct_Detection.hpp. In particular, it responsible for:

1. The statistical methods to compute likelihoods and exclusion limits. Since these are independent of the type of experiment, this functionality is part of the base class DM_Detector. As of now, *obscura* implements the following statistical analyses. 1. Poisson statistics 2. Binned Poisson statistics 3. Maximum gap following [Yellin2002].
2. The detector details, such as detection efficiencies, energy resolution, target particles, etc. These can be very specific and are implemented in classes derived from DM_Detector, e.g. DM_Detector_Nucleus.

We provide a number of examples of how to construct different instances of derived classes of DM_Detector.

6.1 Nuclear recoil experiments

For experiments looking for DM induced nuclear recoils, *obscura* contains the DM_Detector_Nucleus class that is declared in /include/obscura/Direct_Detection_Nucleus.hpp.

For example, assume we have a nuclear recoil experiment with $CaWO_4$ crystals, an energy threshold of 500 eV, and an exposure of 100 kg days. This information suffices to define a toy experiment.

```
#include "libphysica/Natural_Units.hpp"

#include "obscura/Target_Nucleus.hpp"
#include "obscura/DM_Detector_Nucleus.hpp"

using namespace libphysica::natural_units;

// ...

double exposure = 100.0 * kg * day;
std::vector<Nucleus> nuclear_targets = {obscura::Get_Nucleus(8), obscura::Get_
↪Nucleus(20), obscura::Get_Nucleus(74)};
std::vector<double> target_ratios = {4, 1, 1};
double energy_threshold = 500 * eV;
obscura::DM_Detector_Nucleus detector("Nuclear recoil experiment", exposure, nuclear_
↪targets, target_ratios);
```

6.2 Electron recoil experiments

For electron recoil experiments with atomic targets, we have to use the `DM_Detector_Ionization_ER` class that can be found in /include/obscura/Direct_Detection_ER.hpp

Here is an example of a xenon target experiment probing DM-electron interactions and DM induced ionizations. As exposure we choose 100 kg days, and we furthermore assume that only events with at least 4 ionized electrons can be detectred.

```
#include "libphysica/Natural_Units.hpp"

#include "obscura/DM_Detector_ER.hpp"

using namespace libphysica::natural_units;

// ...

double exposure = 100.0 * kg * day;
obscura::DM_Detector_Ionization_ER xenon_experiment("Electron recoil experiment",␣
↪exposure, "Xe");
argon_experiment.Use_Electron_Threshold(4);
```

Alternatively, many experiments looking for sub-GeV DM use semiconductor crystals as targets. In this case, there is another derived class, `DM_Detector_Crystal`.

Again we construct an example toy experiment. This time, we assume a silicon crystal target, choose an exposure of 10 g year, and assume that only events with at least 2 electron-hole pairs can trigger the detector.

```
#include "libphysica/Natural_Units.hpp"

#include "obscura/DM_Detector_Crystal.hpp"

using namespace libphysica::natural_units;

// ...

double exposure = 10.0 * gram * year;
obscura::DM_Detector_Crystal silicon_experiment("Crystal target experiment", exposure,
↪"Si");
silicon_experiment.Use_Q_Threshold(2);
```

CHAPTER

SEVEN

7. EXAMPLES: PUTTING IT ALL TOGETHER

7.1 Computing the recoil spectrum of SI & SD nuclear interactions

As an example for nuclear recoils that also illustrates nicely the modular structure of *obscura*, we compute the nuclear recoil spectrum $\frac{\mathrm{d}R}{\mathrm{d}E_R}$ for a 10 GeV DM particle interacting with xenon nuclei via spin-independent and spin-dependent interactions.

For the definition and details of the nuclear recoil spectrum, see e.g. chapter 3.5 of [Emken2019].

1. First we define the DM particle objects that describe SI and SD interactions

```
// 1. DM particle (SI and Sd)
obscura::DM_Particle_SI dm_SI(10.0 * GeV);
dm_SI.Set_Sigma_Proton(1.0e-40 * cm * cm);
dm_SI.Print_Summary();

obscura::DM_Particle_SD dm_SD(10.0 * GeV);
dm_SD.Set_Sigma_Proton(1.0e-40 * cm * cm);
dm_SD.Print_Summary();
```

The `Print_Summary()` function is a member of many of the classes and provides a terminal output that summarizes the object.

2. For the DM distribution we use the standard halo model with default parameters.

```
// 2. DM distribution
obscura::Standard_Halo_Model shm;
shm.Print_Summary();
```

3. As target nuclei, we choose xenon and import the nuclear data.

```
// 3. Direct detection targets
obscura::Nucleus xenon = obscura::Get_Nucleus("Xe");
xenon.Print_Summary();
```

4. With these three objects, we can compute the differential nuclear recoil spectrum for a given recoil energy E_R.

```
double E_R           = 1.0 * keV;
double dRdER_SI = obscura::dRdER_Nucleus(E_R, dm_SI, shm, xenon);
double dRdER_SD = obscura::dRdER_Nucleus(E_R, dm_SD, shm, xenon);
```

5. The results are given in natural units in powers of GeV. To convert it to another unit, we can use the unit functionality of the *libphysica* library.

```
std::cout << "SI-interactions: \tdR/dER (1 keV) = " << In_Units(dRdER_SI, 1.0 / kg /␣
↪year / keV) << " events / kg / year / keV" << std::endl;
std::cout << "SD-interactions: \tdR/dER (1 keV) = " << In_Units(dRdER_SD, 1.0 / kg /␣
↪year / keV) << " events / kg / year / keV" << std::endl;
```

```
#include <iostream>

#include "libphysica/Natural_Units.hpp"

#include "obscura/DM_Halo_Models.hpp"
#include "obscura/DM_Particle_Standard.hpp"
#include "obscura/Direct_Detection_Nucleus.hpp"
#include "obscura/Target_Nucleus.hpp"

using namespace libphysica::natural_units;

int main()
{

    // 1. DM particle (SI and Sd)
    obscura::DM_Particle_SI dm_SI(10.0 * GeV);
    dm_SI.Set_Sigma_Proton(1.0e-40 * cm * cm);
    dm_SI.Print_Summary();

    obscura::DM_Particle_SD dm_SD(10.0 * GeV);
    dm_SD.Set_Sigma_Proton(1.0e-40 * cm * cm);
    dm_SD.Print_Summary();

    // 2. DM distribution
    obscura::Standard_Halo_Model shm;
    shm.Print_Summary();

    // 3. Direct detection targets
    obscura::Nucleus xenon = obscura::Get_Nucleus("Xe");
    xenon.Print_Summary();

    // 4. Evalute the nuclear recoil spectrum
    double E_R              = 1.0 * keV;
    double dRdER_SI = obscura::dRdER_Nucleus(E_R, dm_SI, shm, xenon);
    double dRdER_SD = obscura::dRdER_Nucleus(E_R, dm_SD, shm, xenon);

    std::cout << "SI-interactions: \tdR/dER (1 keV) = " << In_Units(dRdER_SI, 1.0 / kg /
↪ year / keV) << " events / kg / year / keV" << std::endl;
    std::cout << "SD-interactions: \tdR/dER (1 keV) = " << In_Units(dRdER_SD, 1.0 / kg /
↪ year / keV) << " events / kg / year / keV" << std::endl;

    return 0;
}
```

```
----------------------------------------
DM particle summary:
        Mass:                   10 GeV
```

(continues on next page)

(continued from previous page)

```
        Spin:                   0.5
        Low mass:               [ ]

        Interaction:            Spin-Independent (SI)

        Coupling ratio fixed:   [x]
        Isospin conservation:   [x]
        Coupling ratio:         fn/fp = 1

        Sigma_P[cm^2]:          1e-40
        Sigma_N[cm^2]:          1e-40
        Sigma_E[cm^2]:          1e-40

        Interaction type:       Contact
----------------------------------------


----------------------------------------
DM particle summary:
        Mass:                   10 GeV
        Spin:                   0.5
        Low mass:               [ ]
Interaction:            Spin-Dependent (SD)

        Coupling ratio fixed:   [x]
        Isospin conservation:   [x]
        Coupling ratio:         fn/fp = 1

        Sigma_P[cm^2]:          1e-40
        Sigma_N[cm^2]:          1e-40
        Sigma_E[cm^2]:          1e-40

----------------------------------------

Dark matter distribution - Summary
        Standard halo model (SHM)

        Local DM density[GeV/cm^3]:     0.4
        Speed domain [km/sec]:          [0,777]
        Average DM velocity [km/sec]:   (-11.1 , -232 , -7.3)
        Average DM speed [km/sec]:      330

        Speed dispersion v_0[km/sec]:   220
        Gal. escape velocity [km/sec]:  544
        Observer's velocity [km/sec]:   (11.1 , 232 , 7.3)
        Observer's speed [km/sec]:      233


Xe
Isotope Z       A       Abund.[%]       Spin    <sp>    <sn>
-----------------------------------------------------------
Xe-124  54      124     0.095           0       0       0
Xe-126  54      126     0.089           0       0       0
```

(continues on next page)

(continued from previous page)

```
Xe-128  54      128     1.91            0       0       0
Xe-129  54      129     26.4            0.5     0.01    0.329
Xe-130  54      130     4.07            0       0       0
Xe-131  54      131     21.2            1.5     -0.009  -0.272
Xe-132  54      132     26.9            0       0       0
Xe-134  54      134     10.4            0       0       0
Xe-136  54      136     8.86            0       0       0
Total:          131     99.999

SI-interactions:        dR/dER (1 keV) = 13621.8 events / kg / year / keV
SD-interactions:        dR/dER (1 keV) = 0.132525 events / kg / year / keV
```

7.2 Exclusion limits for a sub-GeV DM particle via electron recoil experiments

As a second example for an application of *obscura*, we will compute the 95% confidence level exclusion limit on the DM-electron cross section for a sub-GeV DM particle.

We assume a DM mass of 100 MeV, and two different direct detection experiments.

1. An argon based experiment with an exposure of 100 kg years and an observational threshold of at least 4 ionized electrons.
2. A semiconductor experiment with Si crystal targets, an exposure of 10 gram years, and an observational threshold of minimum 2 electron-hole pairs.

Let us set up the different objects to obtain the limits.

1. First we define the DM particle object with 100 MeV mass.

```
// 1. DM particle
obscura::DM_Particle_SI dm(100.0 * MeV);
dm.Print_Summary();
```

2. For the DM distribution we again use the standard halo model with default parameters.

```
// 2. DM distribution
obscura::Standard_Halo_Model shm;
shm.Print_Summary();
```

3. For the first experiment, we create an instance of the `DM_Detector_Ionization_ER` class and specify the desired detector properties of the toy experiment.

```
// 3. Argon target experiment
obscura::DM_Detector_Ionization_ER argon_experiment("Argon toy experiment", 100.0 * kg *␣
↪year, "Ar");
argon_experiment.Use_Electron_Threshold(4);
argon_experiment.Print_Summary();
```

4. The same for the semiconductor experiment:

```
// 4. Si target experiment
obscura::DM_Detector_Crystal silicon_experiment("Silicon toy experiment", 10.0 * gram *␣
↪year, "Si");
silicon_experiment.Use_Q_Threshold(2);
silicon_experiment.Print_Summary();
```

4. With these three objects, we can compute the limit on the DM-electron cross section.

```
// 5. Compute the 95% CL exclusion limits for m = 100.0 MeV
double limit_Ar = argon_experiment.Upper_Limit(dm, shm, 0.95);
double limit_Si = silicon_experiment.Upper_Limit(dm, shm, 0.95);
```

5. As in the previous example, the results are given in natural units in powers of GeV. We convert it to cm^2, and print the result on the terminal.

```
std::cout << "Argon experiment: \tsigma_e < " << In_Units(limit_Ar, cm * cm) << " cm^2␣
↪(95%CL)" << std::endl;
std::cout << "Silicon experiment: \tsigma_e < " << In_Units(limit_Si, cm * cm) << " cm^2␣
↪(95%CL)" << std::endl;
```

```
#include <iostream>

#include "libphysica/Natural_Units.hpp"

#include "obscura/DM_Halo_Models.hpp"
#include "obscura/DM_Particle_Standard.hpp"
#include "obscura/Direct_Detection_Crystal.hpp"
#include "obscura/Direct_Detection_ER.hpp"
#include "obscura/Target_Atom.hpp"
#include "obscura/Target_Crystal.hpp"

using namespace libphysica::natural_units;

int main()
{

    // 1. DM particle
    obscura::DM_Particle_SI dm(100.0 * MeV);
    dm.Print_Summary();

    // 2. DM distribution
    obscura::Standard_Halo_Model shm;
    shm.Print_Summary();

    // 3. Argon target experiment
    obscura::DM_Detector_Ionization_ER argon_experiment("Argon toy experiment", 100.0 *␣
↪kg * year, "Ar");
    argon_experiment.Use_Electron_Threshold(4);
    argon_experiment.Print_Summary();

    // 4. Si target experiment
    obscura::DM_Detector_Crystal silicon_experiment("Silicon toy experiment", 10.0 *␣
↪gram * year, "Si");
```

(continues on next page)

(continued from previous page)

```
    silicon_experiment.Use_Q_Threshold(2);
    silicon_experiment.Print_Summary();

    // 5. Compute the 95% CL exclusion limits for m = 100.0 MeV
    double limit_Ar = argon_experiment.Upper_Limit(dm, shm, 0.95);
    double limit_Si = silicon_experiment.Upper_Limit(dm, shm, 0.95);

    std::cout << "Argon experiment: \tsigma_e < " << In_Units(limit_Ar, cm * cm) << "␣
↪cm^2 (95%CL)" << std::endl;
    std::cout << "Silicon experiment: \tsigma_e < " << In_Units(limit_Si, cm * cm) << "␣
↪cm^2 (95%CL)" << std::endl;

    return 0;
}
```

```
----------------------------------------
DM particle summary:
        Mass:                   100 MeV
        Spin:                   0.5
        Low mass:               [ ]

        Interaction:            Spin-Independent (SI)

        Coupling ratio fixed:   [x]
        Isospin conservation:   [x]
        Coupling ratio:         fn/fp = 1

        Sigma_P[cm^2]:          1e-40
        Sigma_N[cm^2]:          1e-40
        Sigma_E[cm^2]:          1e-40

        Interaction type:       Contact
----------------------------------------
Dark matter distribution - Summary
        Standard halo model (SHM)

        Local DM density[GeV/cm^3]:     0.4
        Speed domain [km/sec]:          [0,777]
        Average DM velocity [km/sec]:   (-11.1 , -232 , -7.3)
        Average DM speed [km/sec]:      330

        Speed dispersion v_0[km/sec]:   220
        Gal. escape velocity [km/sec]:  544
        Observer's velocity [km/sec]:   (11.1 , 232 , 7.3)
        Observer's speed [km/sec]:      233


----------------------------------------
Experiment summary:     Argon toy experiment
        Target particles:       Electrons
        Exposure [kg year]:     100
        Flat efficiency [%]:    100
```

(continues on next page)

(continued from previous page)

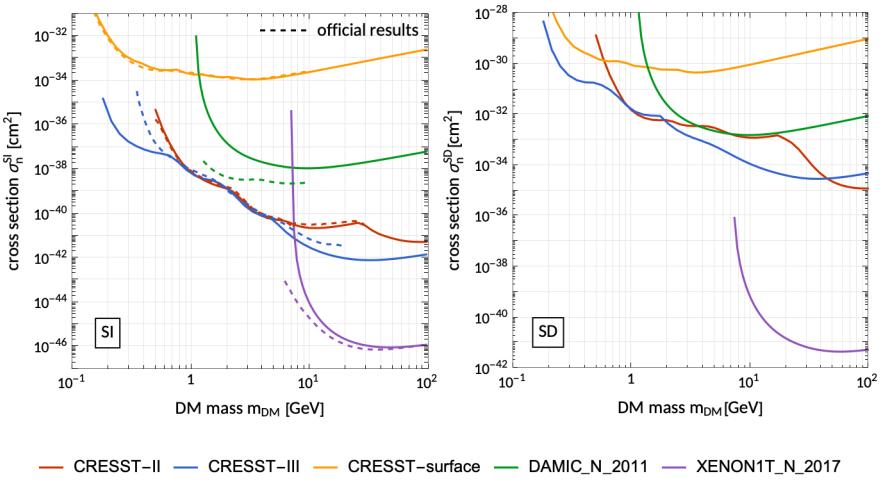
```
        Observed events:        0
        Expected background:    0
        Statistical analysis:   Poisson


        Electron recoil experiment (ionization).
        Target(s):
                        Ar      (100%)
        Electron bins:          [ ]
        PE (S2) bins:           [ ]
                Ne threshold:   4
                Ne max:         15
----------------------------------------


----------------------------------------
Experiment summary:     Silicon toy experiment
        Target particles:       Electrons
        Exposure [kg year]:     0.01
        Flat efficiency [%]:    100
        Observed events:        0
        Expected background:    0
        Statistical analysis:   Poisson


        Electron recoil experiment (semiconductor).
        Target:                 Si semiconductor
        eh pair threshold:      2
----------------------------------------

Argon experiment:       sigma_e < 1.67038e-41 cm^2 (95%CL)
Silicon experiment:     sigma_e < 1.1756e-39 cm^2 (95%CL)
```

CHAPTER

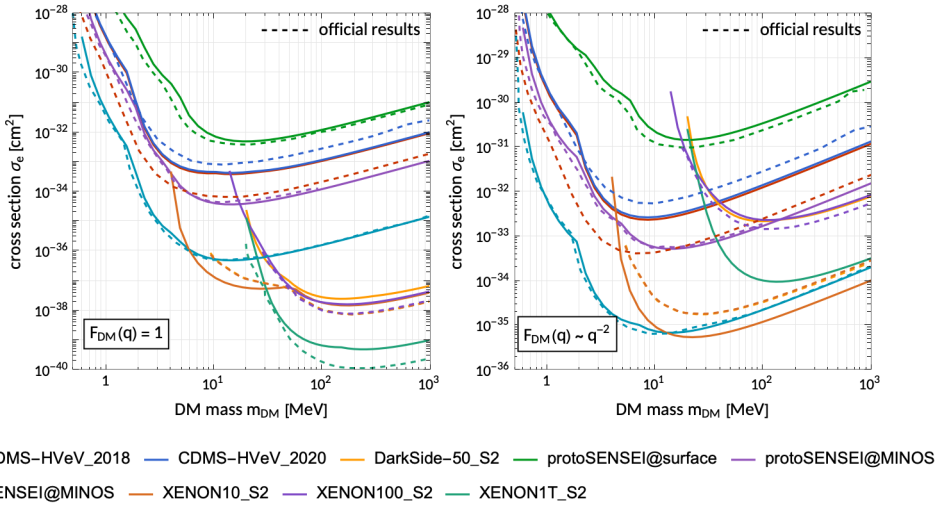
EIGHT

8. INCLUDED EXPERIMENTAL ANALYSES

DM–nucleus interactions



DM–electron interactions



The module Experiments.hpp contains a series of functions that build a number of experimental analysis as instances of the `DM_Detector` class and its derivatives.

For example, for an analysis based on the CRESST-II experiment, we can construct the class instance via

```
#include "obscura/Experiments.hpp"

// ...

DM_Detector_Nucleus detector = CRESST_II();

// ...
```

The following nuclear and electron recoil direct detection experiments are implemented in *obscura*.

8.1 Nuclear recoil experiments

8.1.1 CRESST-II

- **Results on light dark matter particles with a low-threshold CRESST-II detector**

  CRESST Collaboration (G. Angloher et al.)

- **Description of CRESST-II data**

  CRESST Collaboration (G. Angloher et al.)

8.1.2 CRESST-III

- **First results on low-mass dark matter from the CRESST-III experiment**

  CRESST Collaboration (F. Petricca et al.)

- **Description of CRESST-III data**

  CRESST Collaboration (A.H. Abdelhameed et al.)

8.1.3 CRESST-surface

- **Results on MeV-scale dark matter from a gram-scale cryogenic calorimeter operated above ground**

  CRESST Collaboration (G. Angloher et al.)

8.1.4 DAMIC_N_2012

- **Direct Search for Low Mass Dark Matter Particles with CCDs**

  DAMIC Collaboration (J. Barreto et al.)

8.1.5 XENON1T_N_2017

- **First Dark Matter Search Results from the XENON1T Experiment**

  XENON Collaboration (E. Aprile et al.)

8.2 Electron recoil experiments

8.2.1 CDMS-HVeV_2018

- **First Dark Matter Constraints from a SuperCDMS Single-Charge Sensitive Detector**

  SuperCDMS Collaboration (R. Agnese et al.)

8.2.2 CDMS-HVeV_2010

- **Constraints on low-mass, relic dark matter candidates from a surface-operated SuperCDMS single-charge sensitive detector**

  SuperCDMS Collaboration (D.W. Amaral et al.)

8.2.3 DarkSide-50_S2

- **Constraints on Sub-GeV Dark-Matter–Electron Scattering from the DarkSide-50 Experiment**

  DarkSide Collaboration (P. Agnes et al.)

8.2.4 protoSENSEI@surface

- **SENSEI: First Direct-Detection Constraints on sub-GeV Dark Matter from a Surface Run**

  SENSEI Collaboration (Michael Crisler et al.)

8.2.5 protoSENSEI@MINOS

- **SENSEI: Direct-Detection Constraints on Sub-GeV Dark Matter from a Shallow Underground Run Using a Prototype Skipper-CCD**

  SENSEI Collaboration (Orr Abramoff et al.)

8.2.6 SENSEI@MINOS

- **SENSEI: Direct-Detection Results on sub-GeV Dark Matter from a New Skipper-CCD**

  SENSEI Collaboration (Liron Barak et al.)

8.2.7 XENON10_S2

- **A search for light dark matter in XENON10 data**

  XENON10 Collaboration (J. Angle et al.)

- **First Direct Detection Limits on sub-GeV Dark Matter from XENON10**

  Rouven Essig, Aaron Manalaysay, Jeremy Mardon, Peter Sorensen, Tomer Volansky

- **New Constraints and Prospects for sub-GeV Dark Matter Scattering off Electrons in Xenon**

  Rouven Essig, Tomer Volansky, Tien-Tien Yu

8.2.8 XENON100_S2

- **Low-mass dark matter search using ionization signals in XENON100**

  XENON Collaboration (E. Aprile et al.)

- **New Constraints and Prospects for sub-GeV Dark Matter Scattering off Electrons in Xenon**

  Rouven Essig, Tomer Volansky, Tien-Tien Yu

8.2.9 XENON1T_S2

- **Light Dark Matter Search with Ionization Signals in XENON1T**

  XENON Collaboration (E. Aprile et al.)

CHAPTER

NINE

CITING *OBSCURA*

9.1 How to cite

If you decide to use this code, or if you want to add a reference to it, please cite both the paper and the code.

> Emken, T., obscura: A modular C++ tool and library for the direct detection of (sub-GeV) dark matter via nuclear and electron recoils, Journal of Open Source Software, 6(68), 3725, 2021
>
> Emken, T., 2021, obscura - A C++ library for dark matter detection computations [Code] [DOI:10.5281/zenodo.4557187]

```
@article{Emken:2021uzb,
author = "Emken, Timon",
title = "{obscura: A modular C++ tool and library for the direct detection of (sub-GeV)␣
↪dark matter via nuclear and electron recoils}",
eprint = "2112.01489",
archivePrefix = "arXiv",
primaryClass = "hep-ph",
doi = "10.21105/joss.03725",
journal = "J. Open Source Softw.",
volume = "6",
pages = "3725",
year = "2021"
}
```

```
@software{obscura,
author = {Emken, Timon},
title = {{obscura - A C++ library for dark matter detection computations [Code]}},
year         = {2021},
publisher    = {Zenodo},
doi          = {DOI:10.5281/zenodo.4557187},
url          = {https://doi.org/10.5281/zenodo.4557187},
howpublished={The code can be found under \url{https://github.com/temken/obscura}.}
}
```

9.1.1 Cite a specific version

If you want to cite a specific version, please cite the respective DOI that can be found here. For example, for v1.0.1:

> Emken, T., 2021, obscura - A C++ library for dark matter detection computations [Code, v1.0.1] [DOI:10.5281/zenodo.5956877]

```
@software{obscura_1_0_1,
author = {Emken, Timon},
title = {{obscura - A C++ library for dark matter detection computations [Code, v1.0.1]}}
↪,
year         = {2021},
publisher    = {Zenodo},
version      = {v1.0.1},
doi          = {DOI:10.5281/zenodo.5956877},
url          = {https://doi.org/10.5281/zenodo.5956877},
howpublished={The code can be found under \url{https://github.com/temken/obscura}.}
}
```

9.2 Research and research software using *obscura*

The library *obscura* has been applied to obtain the scientific results of the following papers

2. **Solar constraints on captured electrophilic dark matter**, by D. Bose et al.

1. **Solar reflection of light dark matter with heavy mediators**, by Timon Emken

Here is a list of research software using *obscura*:

1. Emken, T., 2021, Dark Matter Simulation Code for Underground Scatterings - Sun Edition (DaMaSCUS-SUN) Astrophysics Source Code Library, record [ascl:2102.018], [DOI:10.5281/zenodo.4559874]

CHAPTER

TEN

RELEASE HISTORY

- 10.11.2021: Release of version 1.0.0
- 23.02.2021: Release of version 0.1.0

CHAPTER

ELEVEN

LICENSE

MIT License

CHAPTER

TWELVE

CONTACT & SUPPORT

The author of *obscura* is Timon Emken.

For questions, support, bug reports, or other suggestions, please contact timon.emken@fysik.su.se or open an issue on github.

CHAPTER

THIRTEEN

REFERENCES

For the interpretation of past and future direct searches for DM particles, it is important to be able to provide accurate predictions for event rates and spectra under a variety of possible and viable assumptions in a computationally efficient way. While there exists a few tools to compute DM induced nuclear recoil spectra, such as DDCalc or WimPyDD, *obscura* is not limited to nuclear targets. Instead its main focus lies on sub-GeV DM searches probing electron recoils which typically requires methods from atomic and condensed matter physics, see e.g. [Essig2012] or [Catena2019]. In the context of sub-GeV DM searches, new ideas such as target materials or detection techniques are being proposed regularly, and the theoretical modelling of these are getting improved continuosly. At the same time, currently running experiments continue to publish their results and analyses, setting increasingly strict bounds on the DM parameter space. In such a dynamic field, *obscura* can be an invaluable tool due to its high level of adaptability and facilitate and accelerate the development of new, reliable research software for the preparation of a DM discovery in the hopefully near future.

BIBLIOGRAPHY

[Catena2019] R. Catena et al., *Atomic responses to general dark matter-electron interactions*, Phys.Rev.Res. 2 (2020) 3, 033195, [arXiv:1912.08204].

[Bednyakov2005] V.A. Bednyakov, *Nuclear spin structure in dark matter search: The Zero momentum transfer limit*, Phys.Part.Nucl. 36 (2005) 131-152, [arXiv:0406218].

[Emken2019] T. Emken, *Dark Matter in the Earth and the Sun - Simulating Underground Scatterings for the Direct Detection of Low-Mass Dark Matter*, PhD thesis 2019, [arXiv:1906.07541].

[Essig2012] R. Essig et al. , *Direct Detection of Sub-GeV Dark Matter*, Phys.Rev.D 85 (2012) 076007 , [arXiv:1108.5383].

[Essig2016] R. Essig et al. , *Direct Detection of sub-GeV Dark Matter with Semiconductor Targets*, JHEP 05 (2016) 046 , [arXiv:1509.01598].

[Essig2020] R. Essig et al. , *Relation between the Migdal Effect and Dark Matter-Electron Scattering in Isolated Atoms and Semiconductors*, Phys.Rev.Lett. 124 (2020) 2, 021801 , [arXiv:1908.10881].

[Evans2019] N.W. Evans et al., *Refinement of the standard halo model for dark matter searches in light of the Gaia Sausage*, Phys.Rev.D 99 (2019) 2, 023012, [arXiv:1810.11468].

[Klos2013] P. Klos et al., *Large-scale nuclear structure calculations for spin-dependent WIMP scattering with chiral effective field theory currents*, Phys.Rev.D 88 (2013) 8, 083516, [arXiv:1304.7684].

[Nobile2021] E. Del Nobile, *Appendiciario – A hands-on manual on the theory of direct Dark Matter detection*, [arXiv:2104.12785].

[Yellin2002] S. Yellin, *Finding an upper limit in the presence of unknown background*, Phys.Rev.D 66 (2002) 032005, [arXiv:0203002].