Prepared for submission to JCAP

Hazma: A Python Toolkit for Studying Indirect Detection of Sub-GeV Dark Matter

Adam Coogan[a1] **Logan Morrison**[b,c] **Stefano Profumo**[b,c]

[a]GRAPPA, Institute of Physics, University of Amsterdam, 1098 XH Amsterdam, The Netherlands
[b]Department of Physics, 1156 High St., University of California Santa Cruz, Santa Cruz, CA 95064, USA
[c]Santa Cruz Institute for Particle Physics, 1156 High St., Santa Cruz, CA 95064, USA

E-mail: a.m.coogan@uva.nl, loanmorr@ucsc.edu, profumo@ucsc.edu

Abstract. With several proposed MeV gamma-ray telescopes on the horizon, it is of paramount importance to perform accurate calculations of gamma-ray spectra expected from sub-GeV dark matter annihilation and decay. We present `hazma`, a python package for reliably computing these spectra, determining the resulting constraints from existing gamma-ray data, and prospects for upcoming telescopes. For high-level analyses, `hazma` comes with several built-in dark matter models where the interactions between dark matter and hadrons have been determined in detail using chiral perturbation theory. Additionally, `hazma` provides tools for computing spectra from individual final states with arbitrary numbers of light leptons and mesons, and for analyzing custom dark matter models. `hazma` can also produce electron and positron spectra from dark matter annihilation, enabling precise derivation of constraints from the cosmic microwave background.

Contents

1 Introduction

The search for particle debris from dark matter (DM) annihilation or decay has thus far largely centered on the GeV-TeV scale, for a variety of reasons. First, if the DM shares electroweak interactions with the Standard Model, as in the weakly-interacting massive particle (WIMP) scenario, then its mass is expected to be in excess of a few GeV, a fact known as the Lee-Weinberg limit [1][1]. Second, the general expectation for the scale of new physics based on the "small hierarchy problem" is that new physics, and thus new massive particles possibly including the particle making up the cosmological DM, should appear around the electroweak scale. Finally, the GeV scale is testable with an array of currently-operating gamma-ray and cosmic-ray observatories, including, but not limited to, the Fermi Large Area Telescope (LAT) [3], Cherenkov Telescope Arrays such as MAGIC [4] and HESS [5], and the Alpha-Magnetic Spectrometer (AMS-02) [6].

On the theory front, the calculation of the detailed expected *particle spectrum* of the debris resulting from DM annihilation or decay has thus focused on the GeV-TeV regime. State-of-the-art codes utilize simulations describing the results of high-energy collisions of elementary particles yielding jets and leptons, which in turn decay and produce stable final-state particles. Many such codes, such as DarkSUSY [7], micrOMEGAs [8] and PPPC4DM [9] utilize tabulated results from PYTHIA [10], one of the most widely-used programs for performing these simulations. Such results are reliable at center-of-mass energy scales at or above roughly 5 GeV [10], but not at lower energies, where, for instance, strongly-interacting particles form hadronic bound states and are no longer described by parton showers, fragmentation and decay. It is well known that the resulting spectra of gamma rays, electrons and positrons and antiprotons, are dramatically different in that case.

For a variety of reasons it is now quite timely to offer the community a reliable computational package that provides the spectra of particles resulting from lighter, sub-GeV DM annihilation or decay. First and foremost, MeV astronomy will soon be revolutionized with a new generation of telescopes such e-ASTROGAM [11, 12] and others, including concept telescopes such as the Advanced Energetic Pair Telescope (AdEPT) [13], the PAir-productioN Gamma-ray Unit (PANGU) [14], and the Gamma-Ray Imaging, Polarimetry and Spectroscopy ("GRIPS") [15]. Second, the persistent absence of any conclusive astrophysical signal from DM in the GeV-TeV range has furthered theoretical and phenomenological interest in the mass range below the GeV, providing additional motivation to investigate the details of DM decay or annihilation processes. Lastly, at present no code exists that allows users to readily study gamma-ray and cosmic-ray production from DM particles annihilating dominantly into hadronic bound states.

With these motivations, we here introduce a Python toolkit, `hazma`[2], that computes spectra of gamma rays and cosmic-ray electrons and positrons from the decay of muons and pions, calculates constraints from gamma-ray observations and the cosmic microwave background, and allows users to compute composite spectra for selected built-in models of DM-parton interactions.

[1]Note that exceptions exist to the Lee-Weinberg limit, see e.g. [2]

[2]Hazma is a small rounded Pokemon, with light green spikes running down its back and tail, making it appear somewhat dinosaurian. Its body resembles a yellow hazardous materials suit, with a face resembling a respirator or gas mask, and a zipper-like marking running down its stomach. It has two stubby legs, the feet of which are green. Hazma is one of the few stable Nuclear types, the others being Nucleon and Urayne. Its leaded skin makes it immune to nuclear radiation. In the aftermath of a nuclear accident, groups of Hazma will appear and feed on the radioactive gas, eventually cleaning the air of the area over time. [16]

From a field theoretic standpoint, the description of the interactions of fundamental fields with hadrons is performed in the context of chiral perturbation theory (ChPT, see e.g. Ref. [17] for a review). A full account of mapping a fundamental, parton-level Lagrangian onto its ChPT counterpart will be given elsewhere [18]; here, however, we do provide selected examples of how models where the DM interacts with a *mediator* of specified spin and parity produces ChPT vertices.

As for any effective theory, ChPT possesses a certain range of validity which depends upon the size of some dimensionless parameter, here the ratio of the meson momentum to a scale $\Lambda_{\text{ChPT}} \equiv 4\pi f_\pi \sim 1$ GeV. Below we will describe the range of dark matter and mediator masses for which our EFT framework can be reliably used to compute annihilation cross sections and mediator decay rates. The mass ranges dictate which combination of mesons we include in the computational package we hereby present.

In the light DM mass limit, we also found that the standard approach for studying radiative emission from leptonic final states is problematic. In short, utilizing the Altarelli-Parisi splitting function to calculate the final state radiation spectrum assumes that radiating particle's center-of-mass frame energy is much larger than its mass. For $\mathcal{O}(100\text{ MeV})$ dark matter annihilating into muons, this is not the case. As a result we compute the *exact* spectrum for a few model cases. We also provide spectra for the final state radiation off of charged pions, and account for radiative decays of all relevant particles (e.g. $\pi^+ \to \mu^+\nu\gamma$, $\pi^+ \to e^+\nu\gamma$ etc).

A general issue with light, sub-GeV DM models is that constraints from perturbations to the cosmic microwave background (CMB) are generically very strong large if the DM freezes out as a thermal relic. Of course there exist a broad variety of workarounds and caveats (see e.g. [19]), but any light DM model is prone to CMB constraints. `hazma` implements such constraints by including functionality for computing electron and positron spectra from dark matter annihilation.

The remainder of this paper is structured as follows. Sec. (2) offers a high-level overview of the `hazma` code and introduces the `Theory` class, the main user-facing component of `hazma`. Section (3) describes the effective field theory framework used to study sub-GeV dark matter, provides details of the scalar mediator (sec. (3.1) and vector mediator (sec. (3.2)) models, and describes the particle physics outputs of `hazma`, including cross section, decay widths, and branching fractions. Section (4) explains how to calculate gamma-ray spectra from individual annihilation final states, while Sec. (5) combines the latter with the scalar- and vector-mediator models to obtain the *overall* gamma-ray spectra from DM annihilation. Section (6) describes the calculation of the positron spectra, and, finally, Sec. (7) and Sec. (8) describe, respectively, gamma-ray and CMB limits. Section (9) concludes. Examples of how to use `hazma` are woven throughout the text. Appendix (A) describes the installation process for `hazma`, App. (B) review the basics of using `hazma`, and App. (C) gives examples of more advanced applications of the code, such as incorporating new models.

Our code is available on GitHub at https://github.com/LoganAMorrison/Hazma, and the manual is located at https://hazma.readthedocs.io/. The icons 📘 and 📄 provide the exact versions of the jupyter notebook and python script used to make each figure, which are paired with `jupytext`. The code snippets appearing throughout this work are collected in a jupyter notebook: 📘. Finally, an archived version of `hazma` is available on Zenodo 🏷.

Conventions: throughout this paper, unless otherwise noted, the units used are MeV for energies, masses and decay widths and cm^3/s for the thermally-averaged DM self-annihilation cross section $\langle \sigma v \rangle$. The `numpy` package [20] is referred to in some of the code snippets as `np`. Lines in code blocks beginning with `>>>` indicate the python command prompt. We have sometimes rounded or formatted the output from `hazma` to make the code snippets more readable.

2 Structure of `hazma`

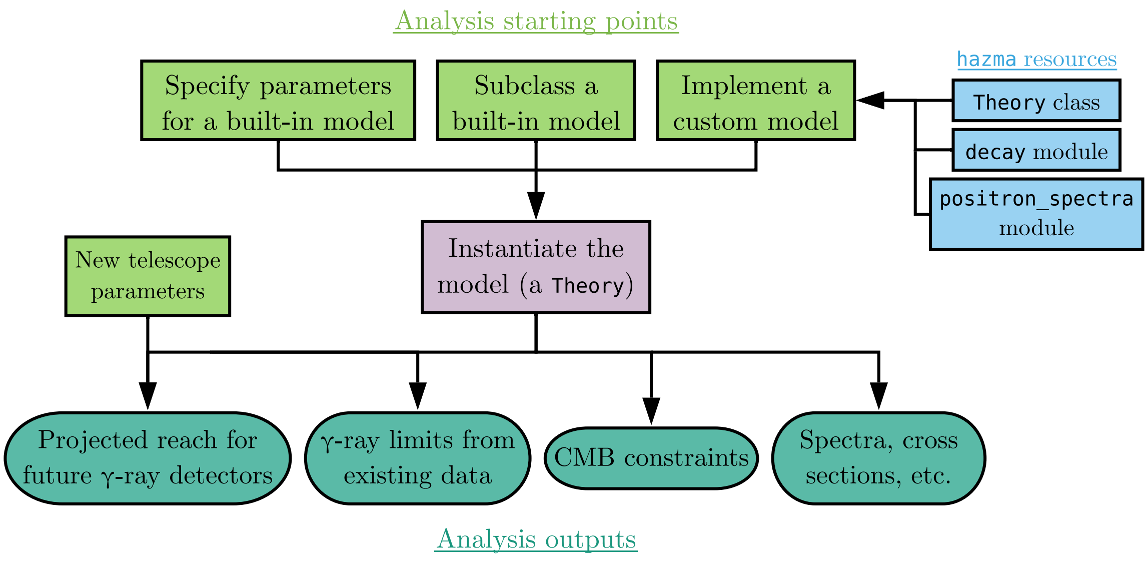
Analysis starting points
Specify parameters for a built-in model
Subclass a built-in model
Implement a custom model
hazma resources
Theory class
decay module
positron_spectra module
New telescope parameters
Instantiate the model (a Theory)
Projected reach for future γ-ray detectors
γ-ray limits from existing data
CMB constraints
Spectra, cross sections, etc.
Analysis outputs

Figure 1. **Overview of the `hazma` workflow, showing different starting points for analyzing sub-GeV dark matter models and possible outputs.** The light green boxes indicate how the user can customize the analysis. The blue boxes to the right show the components of `hazma` relevant to constructing custom models. After a model has been instantiated (purple box), various functions can be called to compute the outputs in the dark green boxes. The user can provide a custom set of gamma-ray detector parameters to assess its discovery potential for the model being studied (small green box).

In this section, we describe the structure of the `hazma` codebase and the intended workflow. The general workflow for a user of `hazma` is shown in Fig. (1), with light green boxes indicating possible user inputs, blue boxes showing some useful `hazma` resources, and dark green boxes showing possible analysis outputs.

The user has several options for tapping into the resources provided by `hazma`. The easiest is to use one of the built-in models, where a user only needs to specify the parameters of the model (see below and Sec. (3.1) and Sec. (3.2) for details on the built-in models and their corresponding parameters). Alternatively, if the user is working with a model which is a specific version one of the included models (e.g., where the mediator's couplings to Standard Model particles are interrelated), they can define their own subclass of that model. By using inheritence, one retains all of the functionality of the built-in model (such as functions for computing final state radiation spectra, cross sections, mediator decay widths, etc.) while supplying the user with a simpler, more specialized interface to the underlying models. For a detailed explanations of how to set model parameters and make subclasses of built-in models, see App. (B).

Another option is for the user to define their own model. To do this, they need to define a class which contains functions for the gamma-ray and positron spectra, as well as the annihilation cross sections and branching fractions. In App. (C) we provide a detailed example of how to do this for a toy model.

After choosing a model, the user instantiates it and gains access to methods for computing indirect detection constraints and particle physics quantities.

Finally, `hazma` also contains lower-level functions to compute gamma-ray spectra for decays and final-state radiation from particular final states containing arbitrary numbers of particles given a user-provided matrix element.

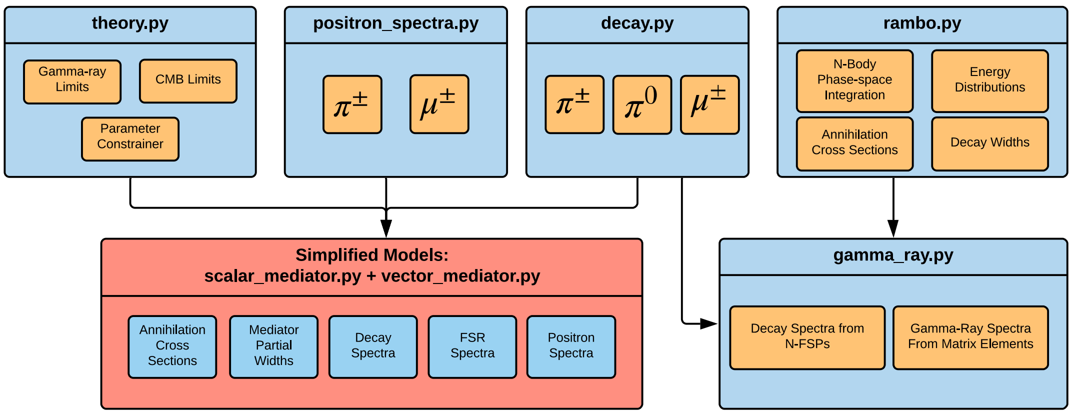

Figure 2. **Structure of `hazma`.** At the core of `hazma` are modules for computing gamma-ray and positron spectra: `decay`, `positron_spectra` and `gamma_ray`, as well modules for computing limits and constraining models using MeV gamma-ray telescopes: `theory` and a module for computing N-body phase-space integrals: `rambo`. `hazma` comes with pre-defined models which are located in `scalar_mediator` and `vector_mediator`, with all of the decay, FSR, and positron spectra, branching fractions, cross sections and mediator decay widths pre-computed.

Fig. (2) displays the modules contained in `hazma` as well as the general structure of the code, which are explained in detail below.

<u>`theory`</u> The primary goal of `hazma` is to provide users with a simple interface for setting indirect detection constraints on sub-GeV dark matter. This is done using the `Theory` abstract base class, which every dark matter model in `hazma` should inherit from. At a high level, it contains three methods for determining these constraints: one that uses existing gamma-ray data (`Theory.binned_limit`), one for computing the discovery reach for planned gamma-ray detectors (`Theory.unbinned_limit`), and one for deriving CMB constraints (`Theory.cmb_limit`). The class also provides methods for computing particle physics quantities such as annihilation cross sections (with `annihilation_cross_sections`), mediator decay widths (`partial_widths`), the continuum and monochromatic gamma-ray spectra from DM annihilation (`total_spectrum`, `gamma_ray_lines`), and the same for positron spectra (`total_positron_spectrum`, `positron_lines`), and more.

Custom models must implement functions for computing gamma-ray spectra, positron spectra and branching fractions to gain use of the methods in `Theory`. Appendix (C) shows how to implement a simple custom model.

`decay` This contains high-performance functions for computing the decay spectra from $\pi^\pm, \pi^0$ and $\mu^\pm$. Details of how these are computed are found in Sec. (4.3). The functions in this module allow the user to compute the decay spectra for particles decaying with arbitrary lab-frame energies. This requires computing the decay spectra in the rest frame of the parent-particle and performing a Lorentz boost, amounting to performing a change-of-variables along with a convolution integral. To achieve high computational performance, we perform all integrations in `c` using `cython` and build extension modules to interface with python.

`positron_spectra` This module computes the electron/positron spectra from decays of $\pi^\pm$ and $\mu^\pm$, which are critical inputs for constraining dark matter models using CMB observations. See Sec. (6) for details on how these spectra are computed. As in the `decay` module, the `positron_spectra` module allows users to compute the electron/positron spectra for arbitrary energies of the parent-particle. The procedure for computing the spectra for arbitrary parent-particle energies is identical to the procedure used for `decay`.

`rambo` While matrix elements for a given particle physics process are often easy to compute (by hand or using tools such as FeynArts [21] or FeynCalc [22, 23]), the calculations of differential or total cross sections become quite difficult as the number of final state particles increases due to the complicated phase-space integrations. The `rambo` module implements a multi-body phase-space integrator based on the RAMBO algorithm [24]. `rambo` contains low-level functions for generating phase-space points as well as high-level functions for computing energy spectra, cross-sections and decay widths, all implemented in `cython` for computational efficiency. For details see App. (D).

`gamma_ray` The functions `gamma_ray_decay` and `gamma_ray_fsr` make use of the multi-particle phase-space integrator `rambo` to compute gamma-ray spectra for two different situations.

The `gamma_ray_decay` function computes the gamma-ray spectra from the *decays* of any number of final state particles with arbitrary center-of-mass energies. The functions accepts a list of the final state particles, the center-of-mass energy and optionally the matrix element of the process (if the user doesn't supply the matrix element, it is assumed to be one.) For example, it can compute the decay spectra from $X \to \pi^\pm\pi^0\mu^\pm\mu^\pm$, where X is some initial state. It is important to note `gamma_ray_decay` does not include final state radiation.

The `gamma_ray_fsr` function allows the user to compute the *final state radiation* spectrum for a given radiative process with a single final state photon. The user supplies a list of initial state and final state particle masses, the center-of-mass energy of the annihilation event, the matrix element for the radiative process and the non-radiative matrix element, which is used to normalize the spectrum. `gamma_ray_fsr` produces a set of gamma-ray energies and the corresponding spectra. For more details on `gamma_ray`, see App. (E).

`hazma` also ships with various particle physics models of sub-GeV DM. These models are located in the modules `scalar_mediator` and `vector_mediator`, and contain all the relevant annihilation cross sections, branching fractions, decay spectra, FSR spectra and positron spectra. They can be used by instantiating the appropriate class (for example, the `HiggsPortal` class from `scalar_mediator` for the Higgs-portal model.) The user only needs to specify the parameters of the model. The particle physics frameworks used to construct these models, the specialized subclasses of these models included with `hazma`, and accessing functions to compute their annihilation cross sections and other particle physics quantities is the topic of the following section.

3 Particle physics framework

Each of the models distributed with `hazma` contain two BSM particles: a dark matter particle χ and a mediator M that interacts with the DM as well as Standard Model particles. The Lagrangian can be expressed as

$$\mathcal{L} = \mathcal{L}_{\text{SM}} + \mathcal{L}_{\text{DM}} + \mathcal{L}_{\text{M}} + \mathcal{L}_{\text{Int}(M)}, \tag{3.1}$$

which consists of the SM Lagrangian, the free Lagrangians for the Dark Matter (DM) and mediator (M), and the mediator's interactions with the DM and SM fields, collectively included in the term $\mathcal{L}_{\text{Int}(M)}$. For the models currently available in `hazma` the DM is taken to be a Dirac fermion, so that

$$\mathcal{L}_{\text{DM}} = \bar{\chi}(i\partial\!\!\!/ - m_\chi)\chi. \tag{3.2}$$

Both the dark matter and the mediator are taken to be uncharged under the Standard Model gauge group. The Lagrangian is defined in terms of the microscopic degrees of freedom of the Standard Model (quarks, leptons and gauge bosons). However, at the energy scale of interest for self-annihilations of non-relativistic MeV dark matter, quarks and gluons are not the correct strongly-interacting degrees of freedom. Instead, the microscopic Lagrangian must be matched onto the effective Lagrangian for pions and other mesons using the techniques of chiral perturbation theory (ChPT) [17, 25–28]. The models currently implemented in `hazma` utilize leading-order ChPT.

Before describing these models, we review the range of validity of ChPT, which is limited since it is an effective theory. ChPT is organized as an expansion in a small parameter, the meson momentum squared p^2 divided by the squared mass scale associated with loop diagrams, $\Lambda_{\text{ChPT}} \sim 4\pi f_\pi \approx 1.2$ GeV, where $f_\pi = 92.2$ MeV is the pion decay constant. Derivatives of the meson fields and factors of the meson masses are counted as $\mathcal{O}(p)$. The expansion parameter p^2 is used in two ways. First, the chiral Lagrangian is organized as a sum of terms of decreasing importance

$$\mathcal{L}_{\text{ChPT}} = \mathcal{L}^{(2)} + \mathcal{L}^{(4)} + \mathcal{L}^{(6)} + \ldots, \tag{3.3}$$

where the subscript indicates the number of derivatives or powers of meson masses. Second, the expansion parameter can be used to assess the relative size of an individual Feynman diagram's contribution in the calculation of a given observable. This is quantified using the *chiral dimension*, which is lower for more important diagrams. Generally the tree-level diagrams from $\mathcal{L}^{(2)}$ have the lowest chiral dimension and provide the largest contributions to observables, followed by tree-level diagrams from $\mathcal{L}^{(4)}$ and one-loop diagrams from $\mathcal{L}^{(2)}$, and so on. As $p^2 \to \Lambda^2_{\text{ChPT}}$, this scheme for organizing calculations in terms of the chiral dimension breaks down and ChPT becomes unreliable.

Assessing the precise range of validity of leading-order ChPT is complicated. A simple estimate suggests that for processes with meson energies of e.g. 500 MeV the leading-order observables receive $\sim 500\ \text{MeV}/\Lambda_{\text{ChPT}})^2 \sim 20\%$ corrections from next-to-leading-order (NLO) contributions [29]. In reality, the magnitude of NLO corrections is larger for annihilation final states with the same quantum numbers as resonances such as the ρ [30] and $f_0(500)$ [31], since the final state particles can rescatter. While resonances can be accounted for with unitarization techniques [31, 32] or as explicit resonance fields [33, 34], we instead use leading-order ChPT, restricting to DM masses below 250 MeV in the case of annihilation into Standard-Model particles and mediator masses below 500 MeV for analysis of their decays to avoid resonances. An important consequence is that annihilation into kaons and heavier mesons is not considered herein since that would require DM masses far above this range. User-defined models (see App. (C.2)) need not use leading-order ChPT, and this is an interesting topic for future work.

While the preceding discussion is quite general, we now specialize to the two models that come with `hazma`: `scalar_mediator`, which contains a real scalar mediator S, and `vector_mediator`, where the mediator is a vector V. These are subclasses of the abstract class `Theory`, and thus implement a variety of functions for computing physical quantities. In the following two subsections, we present $\mathcal{L}_{\text{Int}(S)}$ and $\mathcal{L}_{\text{Int}(V)}$ at the level of quarks and gluons as well as the Lagrangians obtained by performing the ChPT matching.[3] Snippets are provided to demonstrate how to construct each model and change its parameters. The third subsection shows how to access various particle physics quantities in `hazma`.

3.1 Scalar Mediator

The free Lagrangian for a real scalar is

$$\mathcal{L}_S = \frac{1}{2}(\partial_\mu S)(\partial^\mu S) - \frac{1}{2}m_S^2 S^2, \tag{3.4}$$

where m_S is the scalar's mass. The interactions with the light fundamental SM degrees of freedom read

$$\begin{aligned}\mathcal{L}_{\text{Int}(S)} = -S\left(g_{S\chi} + g_{Sf}\sum_f \frac{y_f}{\sqrt{2}}\bar{f}f\right) \\ + \frac{S}{\Lambda}\left(g_{SG}\frac{\alpha_{\text{EM}}}{4\pi}F_{\mu\nu}F^{\mu\nu} + g_{SF}\frac{\alpha_s}{4\pi}G^a_{\mu\nu}G^{a\mu\nu}\right).\end{aligned} \tag{3.5}$$

The sum runs over fermions with mass below the GeV scale ($f = e, \mu, u, d, s$). Note that the coupling g_{Sf} is outside the sum. The Yukawa couplings are defined to be $y_f = \sqrt{2}m_f/v_h$, with the Higgs vacuum expectation value (vev) defined as $v_h = 246$ GeV. The parameter Λ is the mass scale at which S acquires (non-renormalizable) interactions with the photon and gluon.

After performing the matching onto the chiral Lagrangian and expanding to leading

[3]The interaction Lagrangians, matching procedure and a review of the chiral Lagrangian are explained in detail in a forthcoming companion paper [18].

order in the pion fields, the resulting interaction Lagrangian is

$$\begin{aligned}\mathcal{L}_{\mathrm{Int}(S)} = & \frac{2g_{SG}}{9\Lambda} S\left[(\partial_\mu\pi^0)(\partial^\mu\pi^0) + 2(\partial_\mu\pi^+)(\partial^\mu\pi^-)\right] \\ & + \frac{4ieg_{SG}}{9\Lambda} S A^\mu \left[\pi^-(\partial_\mu\pi^+) - \pi^+(\partial_\mu\pi^-)\right] \\ & - \frac{B(m_u+m_d)}{6}\left(\frac{3g_{Sf}}{v_h} + \frac{2g_{SG}}{3\Lambda}\right) S\left[(\pi^0)^2 + 2\pi^+\pi^-\right] \\ & + \frac{B(m_u+m_d)g_{SG}}{81\Lambda}\left(\frac{2g_{SG}}{\Lambda} - \frac{9g_{Sf}}{v_h}\right) S^2\left[(\pi^0)^2 + 2\pi^+\pi^-\right] \\ & + \frac{4e^2 g_{SF}}{9\Lambda} S\pi^+\pi^- A_\mu A^\mu \\ & - g_{S\chi} S\bar{\chi}\chi - g_{Sf} S \sum_{\ell=e,\mu} \frac{y_\ell}{\sqrt{2}} \bar{\ell}\ell. \end{aligned} \tag{3.6}$$

where $B = m_{\pi^\pm}^2/(m_u + m_d) \approx 2800$ MeV [28].

The parameters for the scalar model are attributes of the `scalar_mediator` class. Their names in `hazma` are

$$(m_\chi, m_S, g_{S\chi}, g_{Sf}, g_{SG}, g_{SF}, \Lambda) \leftrightarrow (\texttt{mx}, \texttt{ms}, \texttt{gsxx}, \texttt{gsff}, \texttt{gsGG}, \texttt{gsFF}, \texttt{lam}).$$

The following snippet shows how to instantiate `scalar_mediator`, change the value of a parameter, and print its new value:

```
>>> from hazma.scalar_mediator import ScalarMediator
>>> sm = ScalarMediator(mx=150., ms=1e3, gsxx=1., gsff=0.1,
...                     gsGG=0.1, gsFF=0.1, lam=2e5)
>>> sm.gsff
0.1
>>> sm.gsff = 0.5
>>> sm.gsff
0.5
```

In addition to the general `scalar_mediator` model, `hazma` also comes with two subclasses of `scalar_mediator`: a Higgs-portal (`HiggsPortal`) model and heavy quark model (`HeavyQuark`.) The Higgs-portal model assumes the scalar mediator interacts with the Higgs through gauge-invariant interactions resulting in the scalar mediator mixing with the Higgs. After diagonalizing the scalar-mediator/Higgs mass matrix, the scalar mediator and Higgs are replaced with:

$$h \to h\cos\theta - S\sin\theta, \qquad S \to h\sin\theta + S\cos\theta \tag{3.7}$$

where θ is the scalar-mediator/Higgs mixing angle. This replacement results in the following cutoff scale and couplings of the scalar mediator with the Standard Model fermions, gluons and photon, the later of which require integrating out the τ, c, b, t, W and Z [35]:

$$g_{Sf} = \sin\theta, \qquad g_{SG} = 3\sin\theta, \qquad g_{SF} = -\frac{5}{6}\sin\theta, \qquad \Lambda = v_h. \tag{3.8}$$

where $v_h = 246$ GeV is the Higgs vacuum expectation value. The parameters for the `HiggsPortal` class are:

$$(m_\chi, m_S, g_{S\chi}, \sin\theta) \leftrightarrow (\texttt{mx}, \texttt{ms}, \texttt{gsxx}, \texttt{stheta}). \tag{3.9}$$

The heavy-quark model assumes the existence a new heavy, colored and charged fermion which enters the Lagrangian as:

$$\mathcal{L} \supset i\bar{Q}\not{D}Q - m_Q\bar{Q}Q - \frac{m_Q}{v_h}h\bar{Q}Q - g_{SQ}S\bar{Q}Q \tag{3.10}$$

with $D_\mu = \partial_\mu - ieQ_QA_\mu - ig_sG^a_\mu\lambda^a/2$ where Q_Q is the charge of the heavy-quark. Integrating out the heavy quark induces a cutoff scale and effective couplings of the scalar mediator with gluons and photons:

$$g_{SG} = g_{SQ}, \qquad g_{SF} = 2Q_Q^2 g_{SQ}, \qquad \Lambda = m_Q. \tag{3.11}$$

Note that integrating out the heavy-quark also induced effective couplings to the SM fermions, but at two-loop order. We do not include these interactions. The parameters for the `HeavyQuark` class are:

$$(m_\chi, m_S, g_{S\chi}, g_{SQ}, m_Q, Q_Q) \leftrightarrow (\texttt{mx}, \texttt{ms}, \texttt{gsxx}, \texttt{gsQ}, \texttt{mQ}, \texttt{QQ}). \tag{3.12}$$

Both of these models can be imported and used in a similar fashion to the generic `ScalarMediator` class. The following snippet displays how to initialize these subclasses:

```
>>> from hazma.scalar_mediator import HiggsPortal, HeavyQuark
>>> hp = HiggsPortal(mx=150., ms=1e3, gsxx=1., stheta=1e-3)
>>> hq = HeavyQuark(mx=150., ms=1e3, gsxx=1., gsQ=1.0, mQ=1e6, QQ=1.)
```

3.2 Vector Mediator

For the vector mediator the free part of the Lagrangian is

$$\mathcal{L}_V = -\frac{1}{4}V_{\mu\nu}V^{\mu\nu} + \frac{1}{2}m_V^2V_\mu V^\mu, \tag{3.13}$$

where m_V is the mass of the vector. The interactions considered are

$$\mathcal{L}_{\text{Int}(V)} = V_\mu\left(g_{V\chi}\bar{\chi}\gamma^\mu\chi + \sum_f g_{Vf}\bar{f}\gamma^\mu f\right) - \frac{\epsilon}{2}V^{\mu\nu}F_{\mu\nu}. \tag{3.14}$$

The sum again runs over the light fermions ($f = e, \mu, u, d, s$), and V may have different couplings to each of these. The last term is a kinetic mixing between the photon and V, which we eliminated by transforming the photon $A_\mu \to A_\mu - \epsilon V_\mu$. Upon this field redefinition V acquires an ϵ-suppressed interaction with the SM fermions, which is captured by changing the fermion couplings

$$g_{Vf} \to g_{Vf} - \epsilon eQ_f, \tag{3.15}$$

where Q_f is the electric charge of the fermion f and $e > 0$ is the electron's charge.

Matching onto the chiral Lagrangian and isolating the terms contributing at leading order to the quantities computed in `hazma` gives

$$\begin{aligned}\mathcal{L}_{\text{Int}(V)} = &-i(g_{Vu}-g_{Vd})V^\mu\left(\pi^+\partial_\mu\pi^- - \pi^-\partial_\mu\pi^+\right)\\ &+(g_{Vu}-g_{Vd})^2 V_\mu V^\mu \pi^+\pi^-\\ &+2e(Q_u-Q_d)(g_{Vu}-g_{Vd})A_\mu V^\mu\pi^+\pi^-\\ &+V_\mu\left(g_{Ve}\bar{e}\gamma^\mu e + g_{V\mu}\bar{\mu}\gamma^\mu\mu\right)\\ &+\frac{1}{8\pi^2 f_\pi}\epsilon^{\mu\nu\rho\sigma}(\partial_\mu\pi^0)\\ &\quad\times\{e(2g_{Vu}+g_{Vd})\left[(\partial_\nu A_\rho)V_\sigma+(\partial_\nu V_\rho)A_\sigma\right]\\ &\qquad\qquad +3(g_{Vu}^2-g_{Vd}^2)(\partial_\nu V_\rho)V_\sigma\}\,.\end{aligned} \tag{3.16}$$

The contributions in the last three lines come from matching onto the anomalous terms in the chiral Lagrangian [17, 26, 36, 37], which is explained in detail in our companion paper [18]. While these terms come from the NLO chiral Lagrangian, the resulting final states contribute significantly to the gamma-ray spectra, so we include them here. In particular, in the case where V couples only to quarks, the *only* final state accessible below the two-pion threshold is $\pi^0\gamma$. In contrast, other terms from the NLO chiral Lagrangian only lead to corrections for scattering rates into final states accessible at tree-level, and are thus not included.

The correspondence between the microscopic parameters for the vector model and attributes of the base `vector_mediator` class is[4]

$$\begin{aligned}&(m_\chi, m_V, g_{V\chi}, g_{Vu}, g_{Vd}, g_{Vs}, g_{Ve}, g_{V\mu})\\ &\qquad\leftrightarrow(\texttt{mx}, \texttt{mv}, \texttt{gvxx}, \texttt{gvuu}, \texttt{gvdd}, \texttt{gvss}, \texttt{gvee}, \texttt{gvmumu}).\end{aligned}$$

The following snippet shows how to instantiate `vector_mediator`:

```
>>> from hazma.vector_mediator import VectorMediator
>>> vm = VectorMediator(mx=150., mv=1e3, gvxx=1., gvuu=0.1,
...                      gvdd=0.2, gvss=0.3, gvee=0.4,
...                      gvmumu=0.5)
```

For convenience, a subclass called `KineticMixing` is also provided to handle the important case where V couples to the SM purely through the kinetic mixing term ($\epsilon \neq 0$ and $g_{Vf}=0$ for all f before redefining the couplings). The parameters for this subclass are

$$(m_\chi, m_V, g_{V\chi}, \epsilon) \leftrightarrow (\texttt{mx}, \texttt{mv}, \texttt{gvxx}, \texttt{eps}).$$

While the underlying parameters `gvuu`, ..., `gvmumu` can be accessed by instances of `KineticMixing`, they cannot be set directly since they are fully determined by `eps`:

```
>>> from hazma.vector_mediator import KineticMixing
>>> km = KineticMixing(mx=150., mv=1e3, gvxx=1., eps=0.1)
>>> km.gvuu
-0.020187846690459792  # = -0.1 * 2/3 * sqrt(4 pi / 137)
>>> km.gvuu = 0.1
AttributeError: Cannot set gvuu
```

[4]Note that `gvss` is not currently used since `vector_mediator` is based on leading-order ChPT.

3.3 Computing Particle physics quantities

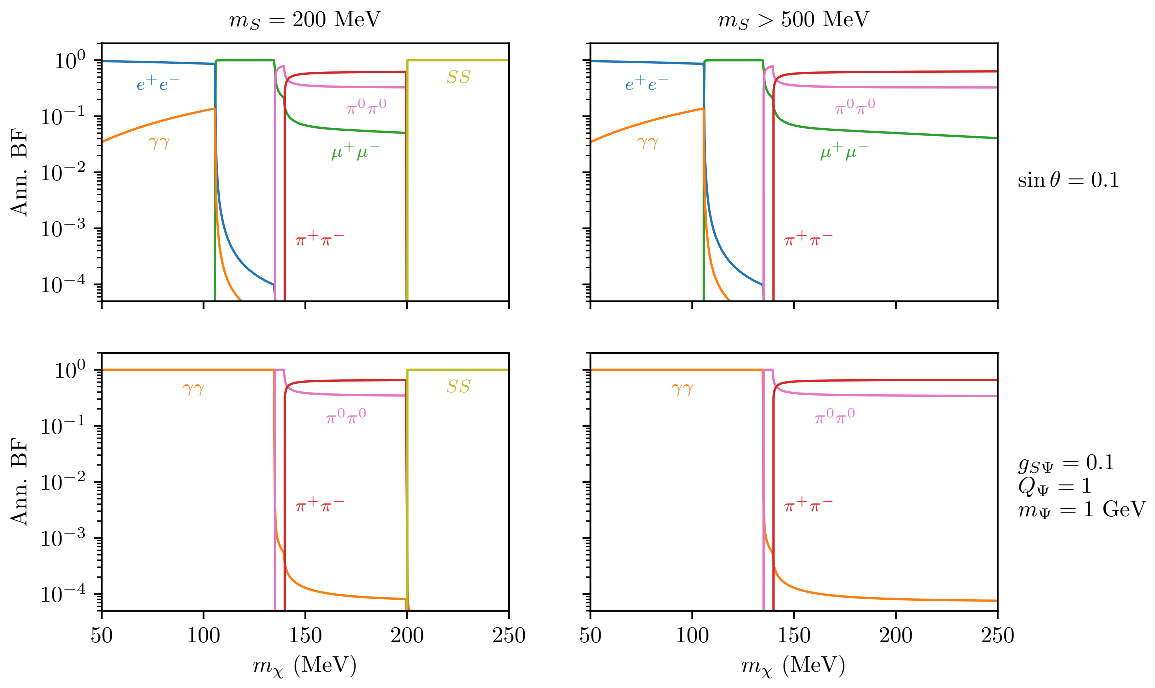

Figure 3. **Annihilation branching fractions for the scalar mediator model.** The coupling $g_{S\chi\chi}$ between the DM and mediator is set to one. The rows correspond to the Higgs portal and heavy quark UV completions and the columns are for a light (left) and heavy (right) mediator.

Every model in hazma is required to implement functions listing the available annihilation final states and providing corresponding functions for computing annihilation cross sections. For example, for the models described above we have:

```
>>> ScalarMediator.list_annihilation_final_states()
['mu mu', 'e e', 'g g', 'pi0 pi0', 'pi pi', 's s']
>>> VectorMediator.list_annihilation_final_states()
['mu mu', 'e e', 'pi pi', 'pi0 g', 'pi0 v', 'v v']
```

These lists *exclude* final states where the annihilation is extremely suppressed by couplings or phase-space factors (e.g., $\pi^+\pi^-\gamma\gamma$, $S\pi^0\pi^0$, $S\pi^+\pi^-$ for the scalar model). They also exclude final states arising from radiative processes (such as $e^+e^-\gamma$). Depending on the couplings' values and the center of mass energy of the DM, some of these final state may be inaccessible.

The corresponding annihilation cross sections and branching fractions can be accessed using the annihilation_cross_sections function:

```
>>> from hazma.scalar_mediator import ScalarMediator
>>> sm = ScalarMediator(mx=180., ms=190., gsxx=1., gsff=0.1,
...                     gsGG=0.1, gsFF=0.1, lam=2e5)
>>> e_cm = 400.
>>> sm.annihilation_cross_sections(e_cm)  # MeV^-2
```

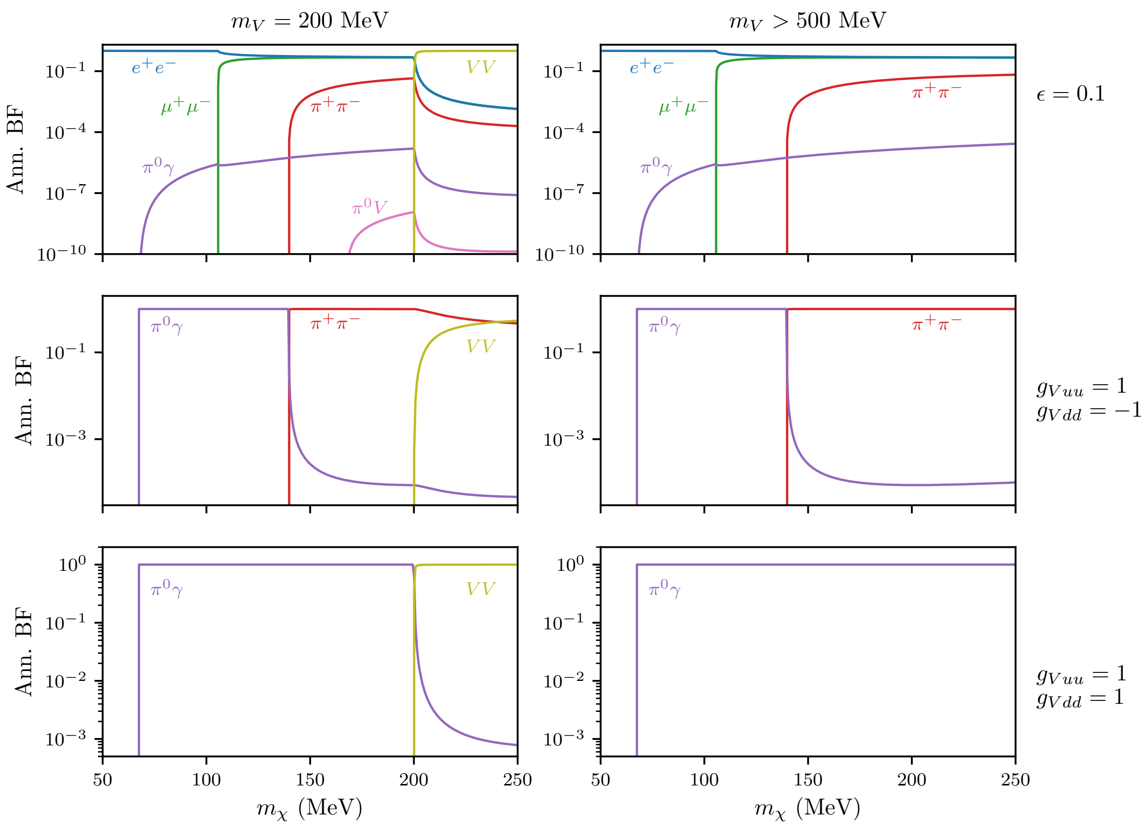

Figure 4. **Annihilation branching fractions for the vector mediator model.** The coupling $g_{S\chi\chi}$ between the DM and mediator is set to one. The rows correspond to a kinetically mixed mediator and a mediator with couplings only to quarks, and the columns are for a light (left) and heavy (right) mediator.

```
{'mu mu': 1.0186780542223538e-16,
 'e e': 3.892649866478948e-21,
 'g g': 4.8762938612605775e-21,
 'pi0 pi0': 1.5950084474945102e-16,
 'pi pi': 3.0338694778562897e-16,
 's s': 3.578493120517737e-07,
 'total': 3.5784931261653806e-07}
```

Note that in this example $m_\chi < m_S$, but the center of mass energy is large enough to allow the process $\bar{\chi}\chi \to SS$. The function `Theory.annihilation_branching_fractions()` is invoked the same way and returns a `dict` of branching fractions for annihilation into each final state X:

$$\mathrm{BF}(\bar{\chi}\chi \to X) = \frac{\sigma_{\bar{\chi}\chi\to X}}{\sum_Y \sigma_{\bar{\chi}\chi\to Y}} \tag{3.17}$$

Fig. (3) shows the annihilation branching fractions for the scalar model. The different rows exhibit results for the Higgs portal and heavy quark UV completions discussed above, which will be referred to throughout this work. In the first case, the branching fraction is largest into whichever final state is closest to threshold since S couples to Standard Model states roughly proportionally to their Yukawas. The branching fraction into $\gamma\gamma$ is small for non-relativistic DM annihilations due to the derivative coupling between S and the photon field. For the heavy quark coupling pattern only hadronic final states are accessible since the mediator couplings exclusively to photons and gluons.

Fig. (4) collects the branching fractions for the vector model, assuming it kinetically mixes with the photon (top row) or couples only to quarks (middle and bottom rows). The coupling-dependence of the vector's cross sections can be understood without performing detailed calculations. Since the vector model can be recast as a U(1) gauge theory (by using the Stuckelberg mechanism to generate the mass term and decoupling the Stuckelberg field), the vector's couplings to pions are the sums of its couplings to their constituents: $g_{V\pi^\pm\pi^\pm} = g_{Vuu} - g_{Vdd}$, $g_{V\pi^0\pi^0} = 0$. The cross section $\sigma_{\bar{\chi}\chi\to\pi^+\pi^-}$ is thus proportional to $(g_{Vuu} - g_{Vdd})^2$, explaining why that final state has a branching fraction of zero in the bottom row of the figure. The vector's coupling to $\pi^0\gamma$ and $\pi^0 V$ come from the anomalous part of the chiral Lagrangian. Since the anomaly is exact at one loop, the relevant diagrams in the former case can be evaluated with a proton running in the loop, making it clear that $\sigma_{\bar{\chi}\chi\to\pi^0\gamma} \propto (2g_{Vuu} + g_{Vdd})^2$. This cross section is suppressed due to the loop factors, but provides an important contribution to the gamma-ray spectrum. For the coupling choices used for the plots in this work, the spectrum contribution from the $\pi^0 V$ final state is either negligible or identically zero.

Finally, the mediator decay partial widths are easily obtained with the `partial_widths` function:

```
>>> from hazma.scalar_mediator import ScalarMediator
>>> sm = ScalarMediator(mx=120., ms=280., gsxx=1., gsff=0.1,
...                     gsGG=0.1, gsFF=0.1, lam=2e5)
>>> sigmas = sm.partial_widths()
>>> sigmas  # MeV
{'g g': 7.363085017295395e-14,
 'pi0 pi0': 6.523373994152967e-09,
 'pi pi': 1.8664869506864194e-09,
 'x x': 0.380609030857039,
 'e e': 1.1995401279596814e-13,
 'mu mu': 1.4482244706501055e-09,
 'total': 0.38060904069531803}
```

4 Building Blocks of MeV Gamma-Ray Spectra

The total gamma-ray spectrum for dark matter annihilation consists of three parts:

$$\left.\frac{dN}{dE_\gamma}\right|_{\bar{\chi}\chi} = \left.\frac{dN}{dE_\gamma}\right|_{\bar{\chi}\chi,\text{line}} + \left.\frac{dN}{dE_\gamma}\right|_{\bar{\chi}\chi,\text{FSR}} + \left.\frac{dN}{dE_\gamma}\right|_{\bar{\chi}\chi,\text{dec.}} . \tag{4.1}$$

The first term accounts for photons from final states containing one or more monochromatic gamma rays, which result in spectral lines. Photons can also be produced through final state radiation (FSR) from annihilation into electromagnetically charged final states, giving rise to the second term. FSR generically produces spectra that scale as $dN/dE_\gamma \propto E_\gamma^{-1}$ at low energies, in accordance with Low's theorem [38, 39], which could be distinguished from the softer $dN/dE_\gamma \propto E_\gamma^{-\Gamma}$, with $\Gamma \sim 2$ for typical astrophysical background. The third term is the spectrum of photons produced through the radiative decays of final state particles. Decays can produce dramatic spectral features such as the neutral pion "box" as well as E_γ^{-1} spectra.

The calculations required to account for these contributions are described in detail in this section, beginning with the trivial case of spectral lines and the associated `hazma` functions. The model-dependent FSR spectra are explained in detail since we compute them exactly for leptonic and hadronic final states. With regard to radiative decays, we present a comprehensive overview of the contributions to the $\pi^\pm$ decay spectrum, which is currently absent from the literature and thus has not been considered in prior work on sub-GeV DM. Code snippets for computing different decay spectra are included throughout.

The section concludes by showing how to compute FSR and radiative decay spectra in `hazma` for the built-in models. A key feature of `hazma` is that the spectrum functions can be employed to analyze user-defined models, as demonstrated in a detailed example in App. (C.2).

4.1 Monochromatic Gamma Rays

The only final states containing monochromatic photons that are relevant for this work are $\pi^0\gamma$ and $\gamma\gamma$:

$$\left.\frac{dN}{dE_\gamma}\right|_{\bar{\chi}\chi,\mathrm{line}}(E_\gamma) = \mathrm{Br}(\bar{\chi}\chi \to \pi^0\gamma)\ \delta(E_{\pi^0\gamma} - E_\gamma) + 2\,\mathrm{Br}(\bar{\chi}\chi \to \gamma\gamma)\ \delta(E_{\mathrm{CM}}/2 - E_\gamma), \quad (4.2)$$

where the energy of the line from the $\pi^0\gamma$ final state is $E_{\pi^0\gamma} \equiv (E_{\mathrm{CM}}^2 - m_{\pi^0}^2)/(2E_{\mathrm{CM}})$ and E_{CM} is the center of mass energy. This contribution to the DM annihilation spectrum is thus obtained by computing the branching fractions appearing in this expression.

The `Theory.gamma_ray_lines()` method returns a `dict` with information about monochromatic gamma-ray lines produced in dark matter annihilations. For example, the scalar mediator model has a 2γ line and the vector model has a line from the $\pi^0\gamma$ final state. Assuming heavy quark-type couplings for the former and quark-only couplings for the later, the line energies and branching fractions are found to be

```
>>> from hazma.scalar_mediator import HeavyQuark
>>> sm = HeavyQuark(mx=140., ms=1e3, gsxx=1., gsQ=0.1, mQ=1e3, QQ=0.1)
>>> sm.gamma_ray_lines(e_cm=300.)
{'g g': {'energy': 150.0, 'bf': 1.285653242415087e-08}}
>>> from hazma.vector_mediator import QuarksOnly
>>> vm = QuarksOnly(mx=140., mv=1e3, gvxx=1., gvuu=0.1, gvdd=0.1, gvss=0.)
>>> vm.gamma_ray_lines(e_cm=300.)
{'pi0 g': {'energy': 119.63552908740002, 'bf': 1.0}}
```

4.2 Final State Radiation

Dark matter annihilating into charged SM particles generically produces photons via final state radiation (FSR), which is accounted for by computing

$$\left.\frac{dN}{dE_\gamma}\right|_{\bar{\chi}\chi,\text{FSR}} = \frac{1}{\sigma_{\bar{\chi}\chi}} \sum_A \frac{d\sigma_{\bar{\chi}\chi\to A\gamma}}{dE_\gamma}(E_\gamma) \tag{4.3}$$

$$\approx \sum_A \left[\frac{\sigma_{\bar{\chi}\chi\to A}}{\sigma_{\bar{\chi}\chi}}\right] \left[\frac{1}{\sigma_{\bar{\chi}\chi\to A}} \frac{d\sigma_{\bar{\chi}\chi\to A\gamma}}{dE_\gamma}(E_\gamma)\right] \tag{4.4}$$

$$\equiv \sum_A \text{Br}(\bar{\chi}\chi \to A) \left.\frac{dN}{dE_\gamma}\right|_{\bar{\chi}\chi\to A}(E_\gamma), \tag{4.5}$$

where in the second line we assumed $\sigma_{\bar{\chi}\chi\to A\gamma} \ll \sigma_{\bar{\chi}\chi\to A}$ and the sum is over final states containing electromagnetically-charged particles. The required branching fractions were computed in Sec. (3.3).

The bulk of previous work on indirect detection of sub-GeV DM applies the Altarelli-Parisi (AP) splitting function to compute the FSR spectrum for the e^+e^- and $\mu^+\mu^-$ final states [40–44]: [5]

$$\left.\frac{dN}{dE_\gamma}\right|_{\bar{\chi}\chi\to\bar{\ell}\ell} = \frac{2\alpha}{\pi Q} \cdot \frac{1}{x}\left[1 + (1-x)^2\right] \cdot \left[-1 + \log\left(\frac{(1-x)}{\mu_\ell^2}\right)\right], \tag{4.6}$$

where $\mu_\ell = m_\ell/Q$, $x = 2E_\gamma/Q$ and Q is the center of mass energy. However, the AP approximation insufficient for our analysis. First, it was derived under the assumption that $m_\ell \ll Q$. This does not hold for DM in the Milky Way halo annihilating non-relativistically into $\mu^+\mu^-$, since in this case $Q \sim m_\mu \sim \mathcal{O}(100\text{ MeV})$. As a result of this assumption, the AP spectrum cuts off for photon energies larger than $(Q^2 - em_\ell^2)/(2Q^2)$, which is different from the true kinematic threshold $(Q^2 - m_\ell^2)/(2Q^2)$. The other key shortcoming is that the AP approximation only applies when radiating particles are fermions, and is thus inapplicable to FSR from pions.

In this section we instead exactly compute the model-dependent FSR spectra for the e^+e^- and $\mu^+\mu^-$ final states, as well as the $\pi^+\pi^-$ final state, which has not been done before. The resulting spectra for FSR from leptons are (See chapter 20 of [46] for details of how these

[5]Note that the widely-used DarkSUSY [45] package and the PPPC4DMID [9] also use the AP approximation, making them inapplicable at the energies we consider.

calculations can be performed)

$$\frac{dN}{dE_\gamma}\bigg|_{\bar{\chi}\chi\to S^*\to\bar{\ell}\ell} = \frac{2\alpha}{E_\gamma\pi(1-4\mu_\ell)^{3/2}}\Bigg[-(1-4\mu_\ell^2)(1-x)\sqrt{1-\frac{4\mu_\ell^2}{1-x}} + \left[2-x(2-x)-4\mu_\ell^2(3-2x-4\mu_\ell^2)\right]\tanh^{-1}\left[\sqrt{1-\frac{4\mu_\ell^2}{1-x}}\right]\Bigg] \tag{4.7}$$

$$\frac{dN}{dE_\gamma}\bigg|_{\bar{\chi}\chi\to V^*\to\bar{\ell}\ell} = \frac{2\alpha}{E_\gamma\pi(1-4\mu_\ell^2)^{1/2}(1+2\mu_\ell^2)}\Bigg[\left[2-x(2-x)+4\mu_\ell^2(1-x)\right]\sqrt{1-\frac{4\mu_\ell^2}{1-x}} -2\left[2-x(2-x)-4\mu_\ell^2(x+2\mu_\ell^2)\right]\tanh^{-1}\left[\sqrt{1-\frac{4\mu_\ell^2}{1-x}}\right]\Bigg]. \tag{4.8}$$

In the limit $m_f \ll Q \leftrightarrow \mu_\ell \ll 1$, these expressions reduce to the AP approximation (Eqn. (4.6)), with an additional term linear in x for the scalar. Additionally the spectra obey Low's theorem at low energies, providing another test of our results.

The FSR spectrum for the $\pi^+\pi^-$ final state can be computed using the interaction Lagrangians derived in our companion paper [18] and reproduced in Eqn. (3.6) and (3.16). The final expressions for the spectra are

$$\frac{dN}{dE_\gamma}\bigg|_{\bar{\chi}\chi\to S^*\to\pi^+\pi^-} = \frac{2\alpha}{E_\gamma\pi(1-4\mu_{\pi^+}^2)^{1/2}}\Bigg[(1-x)\sqrt{1-\frac{4\mu_{\pi^+}^2}{1-x}} -2(1-2\mu_{\pi^+}^2-x)\tanh^{-1}\left[\sqrt{1-\frac{4\mu_{\pi^+}^2}{1-x}}\right]\Bigg] \tag{4.9}$$

$$\frac{dN}{dE_\gamma}\bigg|_{\bar{\chi}\chi\to V^*\to\pi^+\pi^-} = \frac{2\alpha}{E_\gamma\pi(1-4\mu_{\pi^+}^2)^{3/2}}\Bigg[-[1-4\mu_{\pi^+}^2(1-x)-x(1+x)]\sqrt{1-\frac{4\mu_{\pi^+}^2}{1-x}} +2(1-4\mu_{\pi^+}^2)[1+x-2\mu_{\pi^+}^2]\tanh^{-1}\left[\sqrt{1-\frac{4\mu_{\pi^+}^2}{1-x}}\right]\Bigg]. \tag{4.10}$$

These spectra obey with Low's theorem at low photon energies.

In Fig. (5) we plot the spectra for FSR in the scalar and vector models as well as the AP spectrum (blue, red and yellow curves respectively). As can be seen from the above equations, the specific values of the couplings, mediator masses and DM mass do not impact the shape of the spectrum since they cancel when dividing by $\sigma_{\bar{\chi}\chi\to\bar{\ell}\ell}$. As expected, the AP approximation performs poorly near the di-muon threshold (upper left panel). The photon energy at which the spectrum cuts off is an order of magnitude too large, and while the normalization is within a factor of two for the scalar model's spectrum, it is an order of magnitude larger than the vector model's one. The situation improves at larger center of mass energies, and just above the $\pi\pi$ threshold the spectra all agree to within a factor of two. The pion FSR spectra are shown in Fig. (6).

We demonstrate how to compute these spectra in `hazma` at the end of this section.

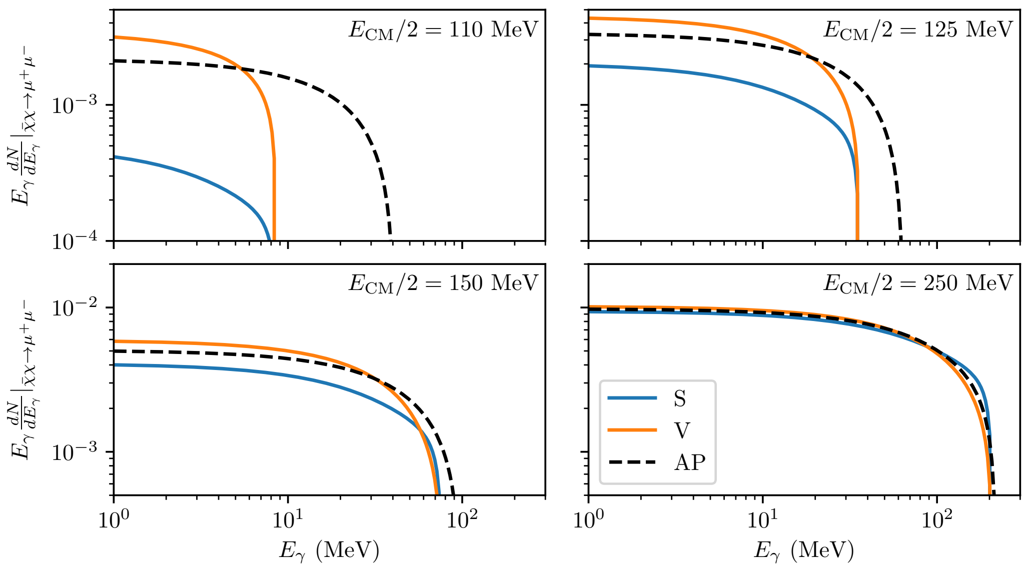

Figure 5. **Final state radiation for dark matter annihilating into $\mu^+\mu^-$.** The curves correspond to the scalar (blue curve) and vector (orange curve) models, with the indicated center of mass energies. The Altarelli-Parisi spectrum from Eqn. (4.6) is also shown (dashed black curve), illustrating the limiting behavior of the spectra as $m_\ell \ll Q$.

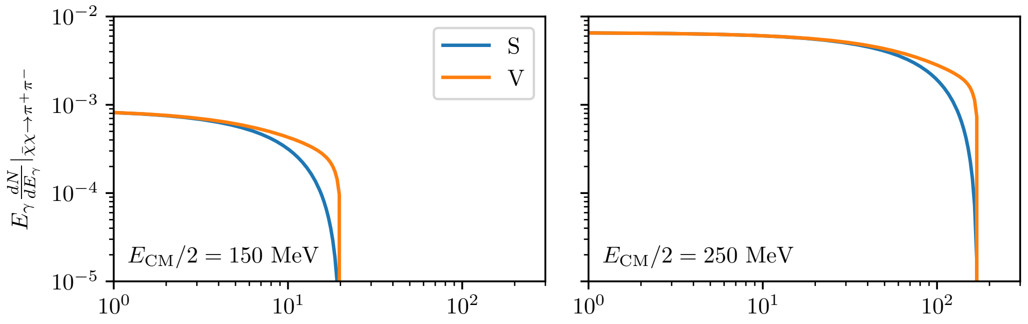

Figure 6. **Final state radiation for dark matter annihilating into $\pi^+\pi^-$.** The blue curve corresponds to the scalar-mediator model and the orange to the vector-mediator case. The panels are labeled with the annihilation's center of mass energy.

4.3 Radiative Decay Spectra

Photons are produced when dark matter annihilates into a state A that subsequently undergoes radiative decay $A \to B\gamma$:

$$\frac{dN}{dE_\gamma}\bigg|_{\bar{\chi}\chi,\text{dec.}}(E_\gamma) = \sum_A \text{Br}(\bar{\chi}\chi \to A)\, \frac{dN}{dE_\gamma}\bigg|_A (E_\gamma). \tag{4.11}$$

In addition to collecting well-known expressions for the neutral pion and muon radiative decay spectra, we carefully account for the three dominant contributions to the charge pion's radiative decay spectrum. Functions for computing radiative decay spectra are contained in the `hazma.decay` module, and described below. These can be utilized to compute the decay contribution to the annihilation spectrum for arbitrary final states.

4.3.1 Decay Spectra in Different Frames

Before computing the radiative decay spectra for the $\mu^\pm$, π^0 and $\pi^\pm$, we review how to boost a spectrum dN/dE_R from the parent particle's rest frame to obtain the "lab frame" spectrum dN/dE_L, where the particle has boost $\gamma = E/m$ along the z-axis. Let the photon phase space coordinates be $(E_R, \cos\theta_R)$ in the rest frame and $(E_L, \cos\theta_L)$ in the lab frame, where θ_R and θ_L are the angles with respect to the z axis and we have assumed azimuthal symmetry. Since the number of photons in a small patch of phase space does not depend on which coordinates we choose, we have

$$\begin{aligned} dN &= f(E_R, \cos\theta_R)\ dE_R\ d\cos\theta_R && (4.12)\\ &= \begin{vmatrix} \frac{dE_R}{dE_L} & \frac{dE_R}{d\cos\theta_L} \\ \frac{d\cos\theta_R}{dE_L} & \frac{d\cos\theta_R}{d\cos\theta_L} \end{vmatrix} f(E_R(E_L, \cos\theta_L), \cos\theta_R(E_L, \cos\theta_L)) dE_L\ d\cos\theta_L, && (4.13) \end{aligned}$$

where we recognize the partial phase space density as $f(E, \cos\theta) \equiv dN/(dE\ d\cos\theta)$ and the term with the two vertical lines is the Jacobian factor for the change of variables. If $f(E_R, \cos\theta_R)$ is independent of $c\theta_R$ (which is the case for all the spectra we will consider), we can write:

$$f(E_R, c\theta_R) = \frac{1}{2}\frac{dN}{dE_R}. \tag{4.14}$$

The variables $E_1, E_2, c\theta_1$ and $c\theta_2$ are related via a Lorentz boost. The relationships are

$$E_R = \gamma E_L(1 - \beta c\theta_L), \qquad \cos\theta_R = \frac{\beta - \cos\theta_L}{1 - \beta\cos\theta_L}, \tag{4.15}$$

where $\beta = \sqrt{1 - 1/\gamma^2}$ is the particle's velocity in natural units. The spectrum in the lab frame is then obtained by integrating Eqn. (4.12) over $\cos\theta_L$:

$$\frac{dN}{dE_L} = \int \cos\theta_L \frac{1}{2\gamma(\beta\cos\theta_L - 1)}\frac{dN}{dE_R}. \tag{4.16}$$

4.3.2 Neutral Pions

The dominant decay mode for neutral pions is $\pi^0 \to \gamma\gamma$ with a branching fraction of about 99%. Due to this decay modes' large branching fraction and the fact that it has two photons in the final state, we ignore the π^0's other decay modes. In the pion's rest frame, the gamma-ray spectrum is trivial:

$$\left.\frac{dN}{dE_\gamma}\right|_{\pi^0}(E_{\pi^0} = m_{\pi^0}) = 2 \times \delta\left(E_R - \frac{m_\pi}{2}\right) \tag{4.17}$$

where the factor of 2 comes from the fact that there are two photons in the final state.

Applying Eqn. (4.16) gives that the gamma ray spectrum in the laboratory frame is

$$\frac{dN}{dE_\gamma}\bigg|_{\pi^0}(E_{\pi^0}) = \int \cos\theta_L \frac{1}{2\gamma(\beta\cos\theta_L - 1)} 2\times\delta\left(E_R - \frac{m_{\pi^0}}{2}\right) \tag{4.18}$$

$$= \frac{2}{\gamma\beta m_{\pi^0}}\left[\theta(E_\gamma - E_-) - \theta(E_\gamma - E_+)\right] \tag{4.19}$$

with $E_\pm = m_{\pi^0}/2\gamma(1\mp\beta)$, which is the characteristic box spectrum centered at $m_{\pi^0}/2$.

The boosted spectrum can be computed in hazma as follows:

```
>>> from hazma.decay import neutral_pion as dnde_pi0
>>> e_gams = np.array([100., 125., 150])  # photon energies
>>> e_pi0 = 180.                          # pion energy
>>> dnde_pi0(e_gams, e_pi0)
array([0.0165965, 0.0165965, 0.       ])
```

4.3.3 Muons

In the muon's rest frame, the radiative decay spectrum is given by [47]

$$\frac{dN}{dE_\gamma}\bigg|_{\mu^\pm}(E_\mu = m_\mu) = \frac{\alpha(1-x)\left(12\left(3-2x(1-x)^2\right)\log\left(\frac{1-x}{r}\right) + x(1-x)(46-55x) + 102\right)}{36\pi E_\gamma}, \tag{4.20}$$

where $r \equiv (m_e/m_\mu)^2$, $x \equiv 2E_\gamma/m_\mu$ and the kinematic bounds on E_γ translate into $0 \le x \le (1+r)m_\mu/2$. To obtain the spectrum in the lab frame we substitute the above expression into Eqn. (4.16) and evaluate the integral numerically.

The lab-frame decay spectrum can be compute for arbitrary muon energies using:

```
>>> from hazma.decay import muon as dnde_mu
>>> e_gams = np.array([1., 10., 100.])  # photon energies
>>> e_mu = 130.                         # muon energy
>>> dnde_mu(e_gams, e_mu)
array([1.76076858e-02, 1.34063877e-03, 4.64775301e-08])
```

4.3.4 Charged Pions

The dominant pion decay is $\pi^+ \to \ell^+\nu_\ell$, where $\ell = e, \mu$ and ℓ is produced approximately on shell, thanks to the narrow width approximation when $\ell = \mu$. Contributions to the pion's radiative decay spectrum come from initial state radiation from the photon or FSR from the lepton, as well as emission from virtual hadronic states (the "structure-dependent" component). An expression for $d^2\Gamma_{\pi^+\to\ell^+\nu_\gamma}/dxdy$ is given in Eqn. (4) of [48], where $x \equiv 2E_\gamma/m_{\pi^+}$ and $y \equiv 2E_\ell/m_{\pi^+}$. After changing to the Mandelstam variables

$$s \equiv (p_\pi - p_\gamma)^2 = m_{\pi^+}^2(1-x),$$
$$t \equiv (p_\pi - p_\ell)^2 = m_{\pi^+}^2\left(1 - y + \frac{m_\ell^2}{m_{\pi^+}^2}\right),$$

integrating over $0 \leq t \leq (m_{\pi^+}^2 - s)(s - m_\ell^2)/s$, and dividing by the total π^+ decay width, we obtain an analytic expression for this channel's contribution to the π^+ decay spectrum:

$$\mathrm{Br}(\pi^+ \to \ell^+\nu_\ell) \cdot \left.\frac{dN}{dE_\gamma}\right|_{\pi^+\to\ell^+\nu_\ell} (E_{\pi^+} = m_{\pi^+}) = \frac{2}{m_{\pi^+}\Gamma_{\pi^+}} \cdot \frac{\alpha(f(x) + g(x))}{24\pi f_\pi^2 (r-1)^2(x-1)^2 rx}, \tag{4.21}$$

where $r \equiv m_\ell^2/m_{\pi^+}^2$ and

$$\begin{aligned} f(x) = (r + x - 1) \big[&m_{\pi^+}^2 x^4 \left(A^2 + V^2\right)\left(r^2 - rx + r - 2(x-1)^2\right) \\ &-12 f_\pi m_{\pi^+} r (x-1) x^2 (A(r - 2x + 1) + Vx) \\ &-12 f_\pi^2 r (x-1)\left(4r(x-1) + (x-2)^2\right)\big], \end{aligned} \tag{4.22}$$

$$\begin{aligned} g(x) = 12 f_\pi r (x-1)^2 \log\left(\frac{r}{1-x}\right) \big[&m_{\pi^+} x^2 (A(x - 2r) - Vx) + \\ &f_\pi \left(2r^2 - 2rx - x^2 + 2x - 2\right)\big]. \end{aligned} \tag{4.23}$$

When $\ell = \mu$, the muon's subsequent radiative decay also contributes significantly to the pion's decay spectrum. Since we can take the muon to be on shell by the narrow width approximation, this decay path's contribution is simply the product of the branching fraction for $\pi^+ \to \mu^+\nu_\mu$ and the muon decay spectrum Eqn. (4.20) evaluated at the muon's energy. Our final expression for the charged pion decay spectrum is thus

$$\begin{aligned} \left.\frac{dN}{dE_\gamma}\right|_{\pi^+} (E_{\pi^+} = m_{\pi^+}) = &\sum_{\ell = e,\mu} \mathrm{Br}(\pi^+ \to \ell^+\nu_\ell) \cdot \left.\frac{dN}{dE_\gamma}\right|_{\pi^+\to\ell^+\nu_\ell} (E_{\pi^+} = m_{\pi^+}) \\ &+ \mathrm{Br}(\pi^+ \to \mu^+\nu_\mu) \cdot \left.\frac{dN}{dE_\gamma}\right|_{\mu^\pm} \left(E_\mu = \frac{m_{\pi^+}^2 - m_\mu^2}{2m_{\pi^+}}\right). \end{aligned} \tag{4.24}$$

This rest frame spectrum can be substituted into Eqn. (4.16) and numerically integrated to obtain the total charged pion radiative decay spectrum.

Fig. (7) shows the contributions of each term and the total spectrum in the pion's rest frame. The muon's radiative decay is the most important component due to its large branching fraction. There is also more phase space available for the photon in comparison with $\pi^+ \to \mu^+\nu_\mu\gamma$ since the other final state particles are massless: the former spectrum cuts off at ~ 69 MeV and the latter at $(m_{\pi^+}^2 - m_\mu^2)/(2m_{\pi^+}) \approx 27$ MeV. The contribution from $\pi^+ \to e^+\nu\gamma$ is smaller than the others due to helicity suppression.

The charged pion decay spectrum is included in `hazma.decay`:

```
>>> from hazma.decay import charged_pion as dnde_pi
>>> e_gams = np.array([1., 10., 100.])  # photon energies
>>> e_pi = 150.                         # pion energy
>>> dnde_pi(e_gams, e_pi)
array([2.54329145e-02, 1.70431824e-03, 2.71309637e-08])
```

The individual contributions can also be computed by setting the `mode` argument. The orange, blue and green curves are obtained using `mode="munu"`, `"munug"` and `"enug"` respectively.

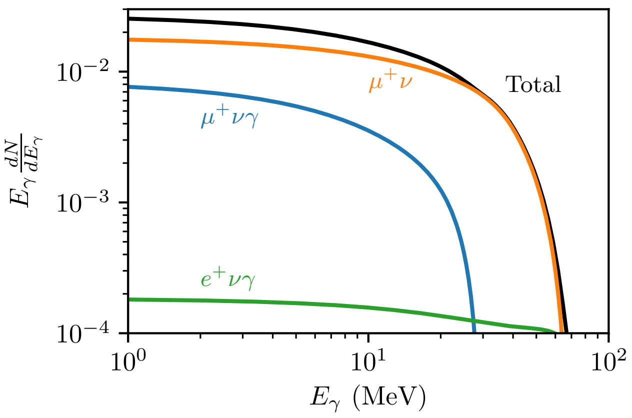

Figure 7. **Contributions to the charged pion radiative decay spectrum in the pion's rest frame.** The decay $\pi^+ \to \mu^+\nu_\mu$ followed by radiative muon decay is plotted in orange. The other curves show the spectra for $\pi^+ \to \ell^+\nu_\ell$, where the photon is produced via initial/final state radiation from the lepton or emission from virtual hadronic states.

4.4 Continuum Gamma-Ray Spectra in `hazma`

Models in `hazma` provide a few methods for computing gamma-ray spectra at different levels of detail. The `total_spectrum` method gives the total *continuum* gamma-ray spectrum at specified photon energies and fixed center-of-mass energy:

```
>>> from hazma.scalar_mediator import HiggsPortal
>>> e_cm = 305.                              # DM center of mass energy
>>> e_gams = np.array([1., 10., 100.])  # photon energies
>>> hp = HiggsPortal(mx=150., ms=1e3, gsxx=0.7, stheta=0.1)
>>> hp.total_spectrum(e_gams, e_cm)
array([0.03292422, 0.00223235, 0.01948197])
```

Underlying this are methods for computing the gamma-ray spectra for individual final states. These are accessible through the `spectra` method, which also accepts a list of photon energies and a center of mass energies. It returns the total continuum spectrum and the contribution from each final state (ie, dN/dE_γ multiplied by the final state's branching fraction):

```
>>> hp.spectra(e_gams, e_cm)
{'mu mu': array([2.60063906e-03, 2.04199179e-04, 6.55397055e-07]),
 'e e': array([2.23372139e-07, 2.09526852e-08, 1.18533468e-09]),
 'pi0 pi0': array([0.        , 0.        , 0.0194812]),
 'pi pi': array([3.03233536e-02, 2.02813250e-03, 1.18432883e-07]),
 's s': array([0., 0., 0.]),
 'total': array([0.03292422, 0.00223235, 0.01948197])}
```

The `spectra` and `total_spectrum` methods are provided by the `Theory` class, and are driven by the method `spectrum_funcs` which classes inheriting from `Theory` are required to implement. This returns a `dict` whose keys are strings corresponding to annihilation final states and whose values are methods returning the continuum gamma-ray spectrum for annihilations into that final state (ie, omitting the branching fraction factor).

The models built into hazma provide more fine-grained methods for studying different channels' spectra. Each of these has a name corresponding to the final state and takes an additional string argument specifying the spectrum type ("decay", "fsr" or "all", the default). The return value depends on the spectrum type argument. In the first case, the returned spectrum only accounts for the final state particles' radiative decays, which are model-independent. In the second case, the method returns the radiative decay spectrum, which is model-dependent. In the third, default case, the method returns the total continuum spectrum.

For example, in the scalar mediator model, the annihilation final states e^+e^-, $\mu^+\mu^-$, $\pi^0\pi^0$, $\pi^+\pi^-$ and SS contribute to the continuum annihilation spectrum. The following snippet shows how to call the spectrum methods for the instance of HiggsPortal created above. The methods all follow the same naming conventions and return dN/dE_γ in MeV^{-1}:

```
>>> from hazma.scalar_mediator import HiggsPortal
>>> e_cm = 305.                              # DM center of mass energy
>>> e_gams = np.array([1., 10., 100.])  # photon energies
>>> hp = HiggsPortal(mx=150., ms=1e3, gsxx=0.7, stheta=0.1)
>>> hp.dnde_ee(e_gams, e_cm, spectrum_type="fsr")
>>> array([0.05435176, 0.00509829, 0.00028842])
>>> hp.dnde_ee(e_gams, e_cm, spectrum_type="decay")
>>> array([0., 0., 0.])  # electrons don't decay
>>> hp.dnde_pipi(e_gams, e_cm, spectrum_type="fsr")
>>> array([1.01492577e-03, 5.00245201e-05, 0.00000000e+00])
>>> hp.dnde_pipi(e_gams, e_cm, spectrum_type="decay")
>>> array([5.08808459e-02, 3.42094713e-03, 2.02687537e-07])
>>> hp.dnde_pipi(e_gams, e_cm)
>>> array([5.18957717e-02, 3.47097165e-03, 2.02687537e-07])
```

There are several other useful methods. For example, the spectra() method, which all subclasses of Theory must implement, computes the total continuum spectra for all final states as well as their sum:

```
>>> hp.spectra(e_gams, e_cm)
{'total':   array([0.03292422    , 0.00223235    , 0.01948197]),
 'mu mu':   array([2.60063906e-03, 2.04199179e-04, 6.55397055e-07]),
 'e e':     array([2.23372139e-07, 2.09526852e-08, 1.18533468e-09]),
 'pi0 pi0': array([0.            , 0.            , 0.0194812]),
 'pi pi':   array([3.03233536e-02, 2.02813250e-03, 1.18432883e-07]),
 's s':     array([0.            , 0.            , 0.])}  # since ms > mx
```

5 Gamma Ray Spectra from DM annihilation

Using the ingredients described in the previous section, we now compute the photon spectra from DM annihilation for the scalar and vector models with the same couplings as in Fig. (3)

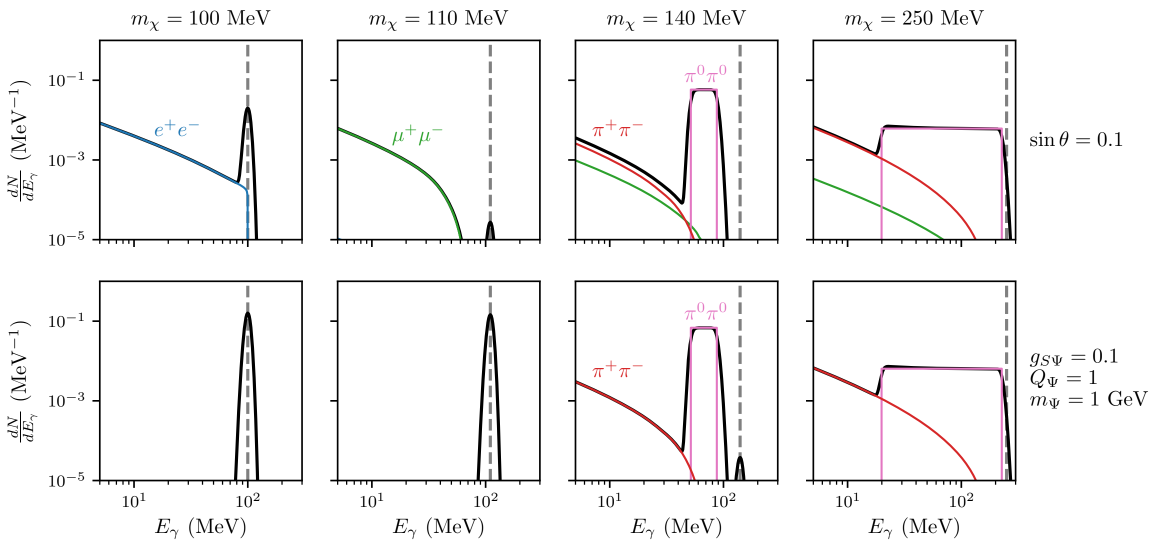

Figure 8. **Spectra from DM annihilation into Standard Model particles for scalar-mediator models.** The two-mediator final state is kinematically forbidden. The black curve is the total spectrum as seen by an instrument with 5% energy resolution. The labeled colored curves are the spectra for individual final states, and the vertical dashed line marks the photons' energies in the $\gamma\gamma$ final state. The rows correspond to Higgs portal and heavy fermion coupling patterns. The DM mass is fixed to the indicated value in each column.

and Fig. (4). In what follows, the center of mass energy is set to the non-relativistic approximation $E_{\text{CM}} = 2m_\chi\left(1 + \frac{1}{2}v_{\text{MW}}^2\right)$, where $v_{\text{MW}} = 10^{-3}c$ is the fiducial velocity dispersion for the Milky Way.

Fig. (8) displays the spectrum for the scalar model when annihilation into the mediator is kinematically prohibited ($m_S \gtrsim m_\chi$). The dark matter mass is fixed for each column and the model parameters in each row. Individual final states' spectra are indicated by the colored curves. To visualize the di-photon final state's contribution, the black curve shows the total spectrum convolved with a 5% energy resolution function (the case for COMPTEL [49]), with a vertical dashed line highlighting the photons' energies ($E_{\text{CM}}/2$).

For low DM masses, the electron and di-photon line are responsible for the spectrum. Above the muon threshold the Yukawa-suppressed electron contribution is negligible. Since the branching fractions into different pion species are identical (up to isospin-breaking corrections proportional to $m_{\pi^\pm} - m_{\pi^0}$), the charged pion spectrum is always sub-dominant to the neutral pion box for $E_{\text{CM}} > 2m_\pi$. Depending on the ratio of the microscopic coupling to photons (g_{SF}) with the couplings to gluons and fermions (g_{SG} and g_{Sf}), the neutral pion box or di-photon line are generally the most distinctive spectral features for large DM masses, which can also be seen in the figure. Note that if g_{Sff} and g_{SGG} are considered as independent parameters, they can take values such that the cross section for annihilation into pions is zero, though this scenario is very fine-tuned.

Spectra for annihilation into *mediators* are shown in Fig. (9). The panels are labeled with the DM mass and the different colored curves are the spectra computed with the scalar's mass set to the indicated values. For $m_S < 2m_e$ the mediator can only decay to two photons, so the annihilation spectrum is a box centered at $m_S/2$, just like for the neutral pion. For

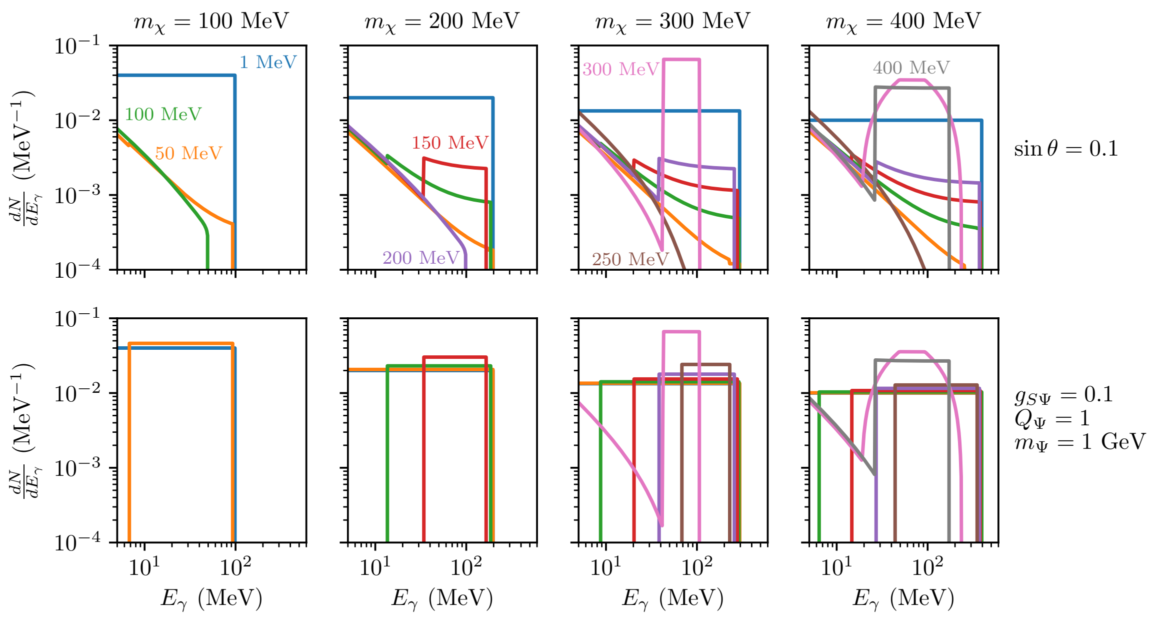

Figure 9. **Spectra from DM annihilation into mediators for scalar-mediator models.** Curves are only shown when $E_{\rm CM} \geq 2m_S$, and are labeled with the corresponding mediator mass. The spectra have not been convolved with an energy resolution function since the SS final state does not produce gamma-ray lines.

$2m_e < m_S < 2m_\mu$ the spectrum is a superposition of the box and the electron FSR spectra, and when the scalar can decay into two muons this dominates the spectrum. For $m_S > 2m_\pi$ the spectrum consists of a peak from boosting the π^0 decay spectrum and a softer component from FSR and decays from the other final states into which S decays. The peak is more rounded if the scalar is highly boosted ($m_S \ll E_{\rm CM}/2$) and boxy for $m_S \sim E_{\rm CM}/2$. The difference in the case of heavy fermion microscopic couplings is that the muon and electron final states are not present.

The components of the spectrum for DM annihilation into SM final states in the vector model are illustrated in Fig. (10). In the kinetic mixing case the spectrum consists purely of a continuum, since the branching fraction for annihilation into $\pi^0\gamma$ is highly suppressed. In contrast, if the vector couples only to quarks the spectrum is composed of a monochromatic gamma ray line and box from the π^0 decay at low energies. If not forbidden by the vector's couplings (as is the case for the bottom row), as the center-of-mass energy grows above the two-pion threshold the continuum of photons from the $\pi^+\pi^-$ final state overwhelm those from $\pi^0\gamma$.

Fig. (11) exhibits spectra from DM annihilating into VV followed by the vectors' decays for a kinetically mixed vector (top row) and couplings to quarks only. As in the case of annihilations directly into SM particles, a power law spectrum is obtained in the kinetically mixed case. For the spectrum for quark-only couplings (middle and bottom rows) is sourced exclusively by the vector's decay to $\pi^0\gamma$. This resulting spectrum is comprised of a box (from the boosting the monochromatic photon) and a rounded peak (from boosting the neutral pion's decay spectrum). The kinematic edge feature is only observable at a detector with very good energy resolution.

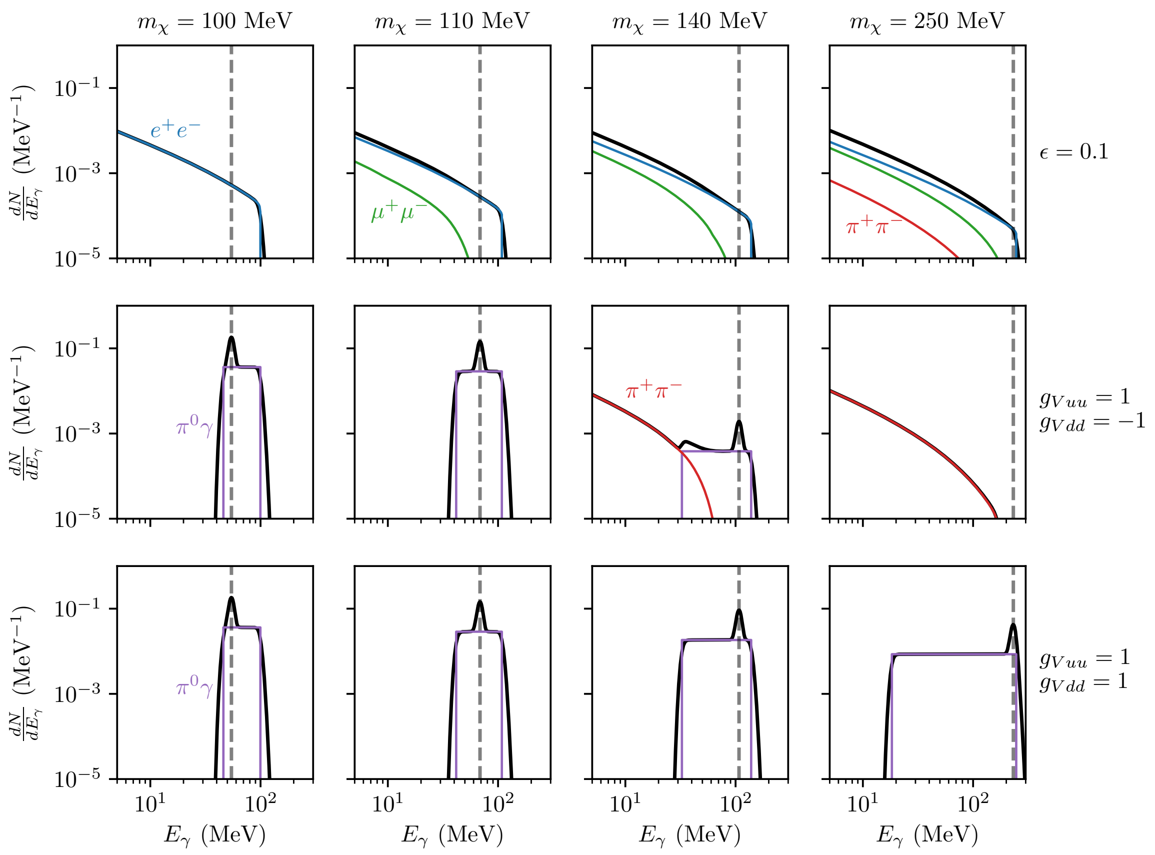

Figure 10. **Spectra from DM annihilation into Standard Model particles for vector-mediator models.** The two-mediator final state is taken to be kinematically forbidden. The rows correspond to the case where the mediator kinetically mixes with the photon and where it couples only to quarks. The vertical dashed line indicates the energy of the photon in the $\pi^0\gamma$ final state.

The colored curves in the Fig. (8)-(11) can be reproduced using the `Theory.spectra()` function described in Sec. (4.4). The total spectrum convolved with an energy resolution function can be obtained using `Theory.total_conv_spectrum_fn()`. The arguments specify the range of photon energies over which to perform the convolved spectrum, the center of mass energy of the DM annihilation, the energy resolution function, and the number of points to use when computing the convolution:

```
>>> from hazma.scalar_mediator import HiggsPortal
>>> e_cm = 305.                     # DM center of mass energy
>>> e_min, e_max = 1., 100.         # define energy range
>>> energy_res = lambda e: 0.05   # 5% energy resolution function
>>> hp = HiggsPortal(mx=150., ms=1e3, gsxx=0.7, stheta=0.1)
>>> dnde_conv = hp.total_conv_spectrum_fn(
...     e_min, e_max, e_cm, energy_res, n_pts=1000)
```

Decreasing `n_pts` from its default value of 1000 increases speed at the cost of accuracy. The

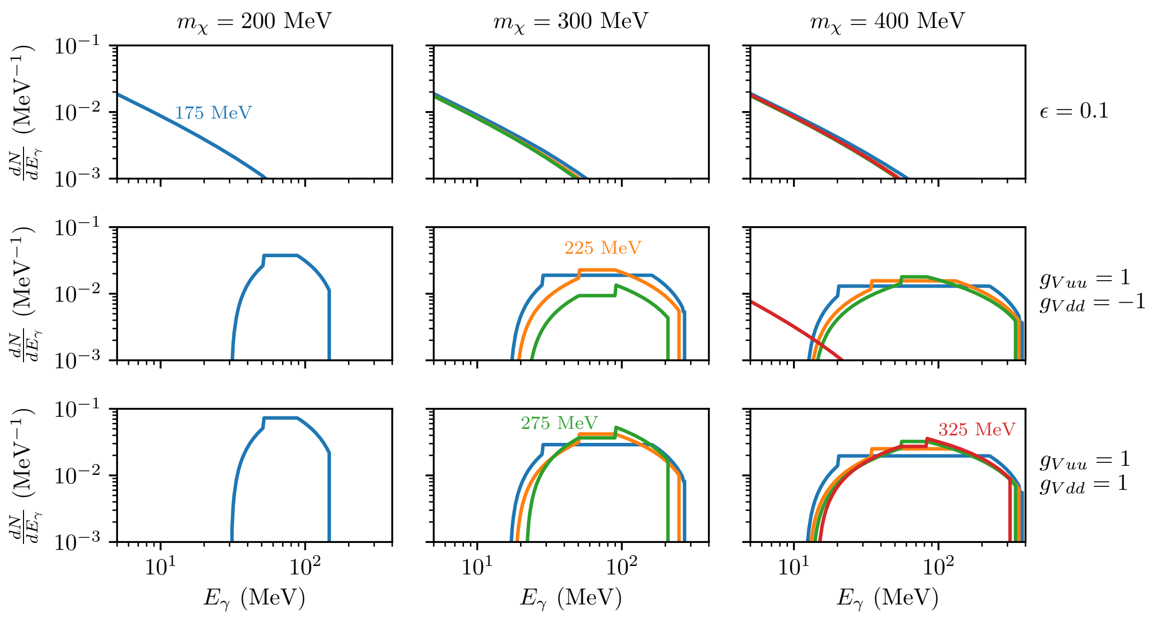

Figure 11. **Spectra from DM annihilation into mediators for vector-mediator models.** The labels and colors indicate the value of m_V for each curve.

return value (`dnde_conv` above) is an interpolator of type `InterpolatedUnivariateSpline` from the `scipy.interpolate` module. This can be used to compute the spectrum at a particular photon energy:

```
>>> dnde_conv(25.)
array(0.00048767)
```

This spectrum is can easily and efficiently be integrated over a range of photon energies between `e_min` and `e_max`:

```
>>> dnde_conv.integral(25., 85.)
0.8691133833417997
```

6 Electron and Positron Spectra from DM annihilation

Dark matter annihilation in the mass range under consideration produces relativistic electrons and positrons. Such particles could be in principle directly detected as part of the cosmic radiation, albeit at energies well below the GeV solar modulation strongly affects cosmic-ray electron spectra. Additionally, relativistic electrons and positrons can be indirectly detected through (1) effects on photons of the cosmic microwave background and (2) the secondary emission of radiation via e.g. up-scattering of background photons (inverse Compton processes), synchrotron radiation, and bremsstrahlung. The calculation of the electron-positron spectra for MeV dark matter is therefore needed to reliably compute CMB limits and the spectrum of secondary radiation.

In this section we compute the positron spectrum for MeV dark matter, which follows along the lines of the gamma-ray ones.[6] The spectrum consists of a line-like spectrum from $\bar{\chi}\chi \to e^+e^-$ and a continuum piece from the decays of unstable particles:

$$\left.\frac{dN}{dE_{e^+}}\right|_{\bar{\chi}\chi}(E_{e^+}) = \mathrm{Br}(\bar{\chi}\chi \to e^+e^-)\;\delta(E_{\mathrm{CM}}/2 - E_{e^+}) + \left.\frac{dN}{dE_{e^+}}\right|_{\bar{\chi}\chi,\mathrm{dec}}(E_{e^+}) \tag{6.1}$$

For the scalar and vector model, the decay piece receives contributions from the $\mu^+\mu^-$ and $\pi^+\pi^-$ final states.

The muon decays nearly 100% of the time through $\mu^+ \to e^+\nu_e\bar{\nu}_\mu$. In the muon's rest frame, the positron's energy spectrum is [50]

$$\frac{dN}{dE_{e^+}} = -\frac{4\sqrt{x^2-4r^2}\left(r^2(4-3x)+x(2x-3)\right)}{m_\mu} \tag{6.2}$$

where $r = m_e/m_\mu$ and $x = 2E_{e^+}/m_\mu$. The spectrum can be boosted to other frames using Eqn. (4.16). It is accessible in `hazma` using:

```
>>> from hazma.positron_spectra import muon as dnde_p_mu
>>> e_mu = 150.                          # muon energy
>>> e_p = np.array([1., 10., 100.])  # positron energies
>>> dnde_p_mu(e_p, e_mu)
array([4.86031362e-05, 4.56232320e-03, 4.45753994e-03])
```

The charged pion primarily decays through $\pi^+ \to \mu^+\nu_\mu$, which yields a positron when the muon decays. The energy spectrum is thus obtained by boosting the muon spectrum from Eqn. (6.2). An additional contribution comes from the monochromatic positron produced through $\pi^+ \to e^+\nu_e$, which has a helicity suppressed branching fraction $\mathrm{Br}(\pi^+ \to e^+\nu_e) = 1.23 \times 10^{-4}$. The total spectrum can be computed in `hazma` with

```
>>> from hazma.positron_spectra import charged_pion as dnde_p_pi
>>> e_pi = 150.                          # charged pion energy
>>> e_p = np.array([1., 10., 100.])  # positron energies
>>> dnde_p_pi(e_p, e_pi)
array([3.84163631e-05, 3.85242442e-03, 2.55578895e-05])
```

Fig. (12) exhibits representative positron annihilation spectra from the scalar and vector mediator models, for Higgs portal and kinetic mixing couplings (first and second rows) and several DM masses (different columns). Changing the couplings to prohibit annihilation into leptons has a straightforward impact on the total spectrum. Here annihilation into mediators is kinematically forbidden. Positron spectra for annihilation into mediators are shown for these models in Fig. (13). When the mediator is light the spectrum is box-shaped since the mediator can only produce electrons by decaying into them directly. At higher energies the spectrum is a superposition of the rounded spectra from mediator decays into muons and pions and the box spectrum.

[6]The electron spectrum looks identical to the positron one since our dark matter particles do not carry lepton number.

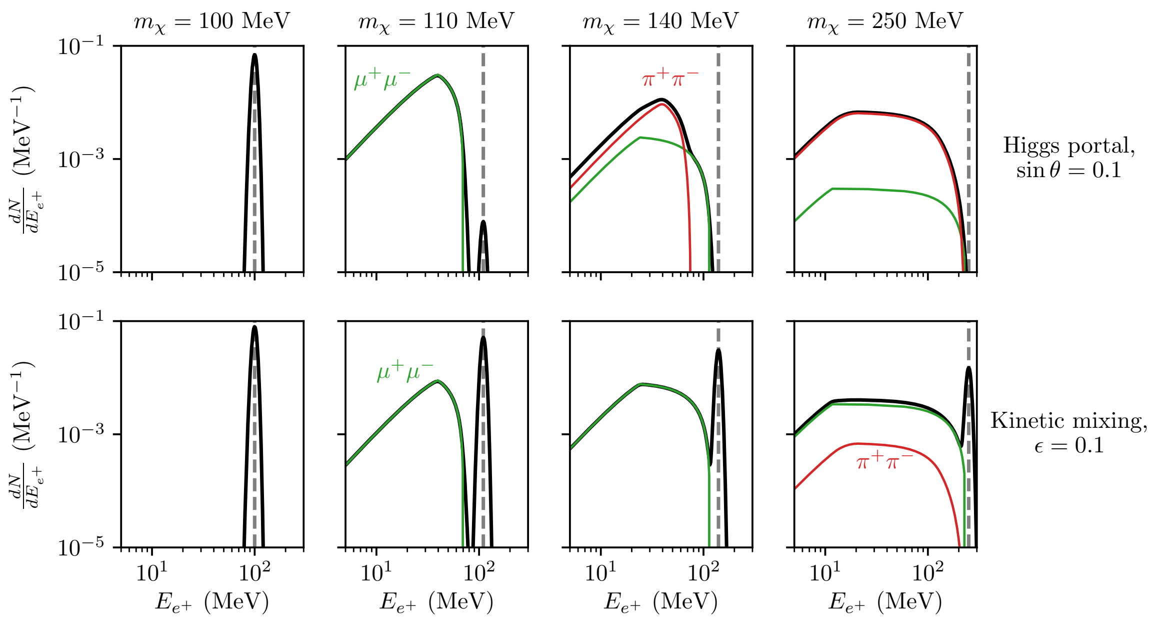

Figure 12. **Positron spectra for DM annihilating into Standard Model final states.** Results for the scalar model with Higgs portal couplings are shown in the top row and for the vector model with a kinetically-mixed mediator in the bottom. The grey vertical dashed line indicates the location of the monochromatic $\bar{\chi}\chi \to e^+e^-$ final state. The total spectrum (black curve) is convolved with a 5% energy resolution function.

The interface for computing positron spectra mirrors the gamma-ray spectrum interface. The `total_positron_spectrum` and `positron_spectra` methods in `Theory` give the total positron spectrum and individual channels' contributions (scaled by branching fraction). The `positron_lines` method gives information about final states producing monochromatic positrons:

```
>>> from hazma.scalar_mediator import HiggsPortal
>>> e_cm = 305.                          # DM center of mass energy
>>> e_ps = np.array([1., 10., 100.])    # positron energies
>>> hp = HiggsPortal(mx=150., ms=1e3, gsxx=0.7, stheta=0.1)
>>> hp.total_positron_spectrum(e_ps, e_cm)
array([2.60677406e-05, 2.58192085e-03, 4.09383294e-04])
>>> hp.positron_spectra(e_ps, e_cm)
{'mu mu': array([3.25408271e-06, 3.03854662e-04, 3.08297785e-04]),
 'pi pi': array([2.28136579e-05, 2.27806619e-03, 1.01085509e-04]),
 's s': array([0., 0., 0.]),
 'total': array([2.60677406e-05, 2.58192085e-03, 4.09383294e-04])}
>>> hp.positron_lines(e_cm)
{'e e': {'energy': 152.5, 'bf': 4.109750194098895e-06}}
```

For the built-in models, functions computing the positron spectra for annihilations into particular final states follow a similar naming scheme to the gamma-ray ones, and can be called

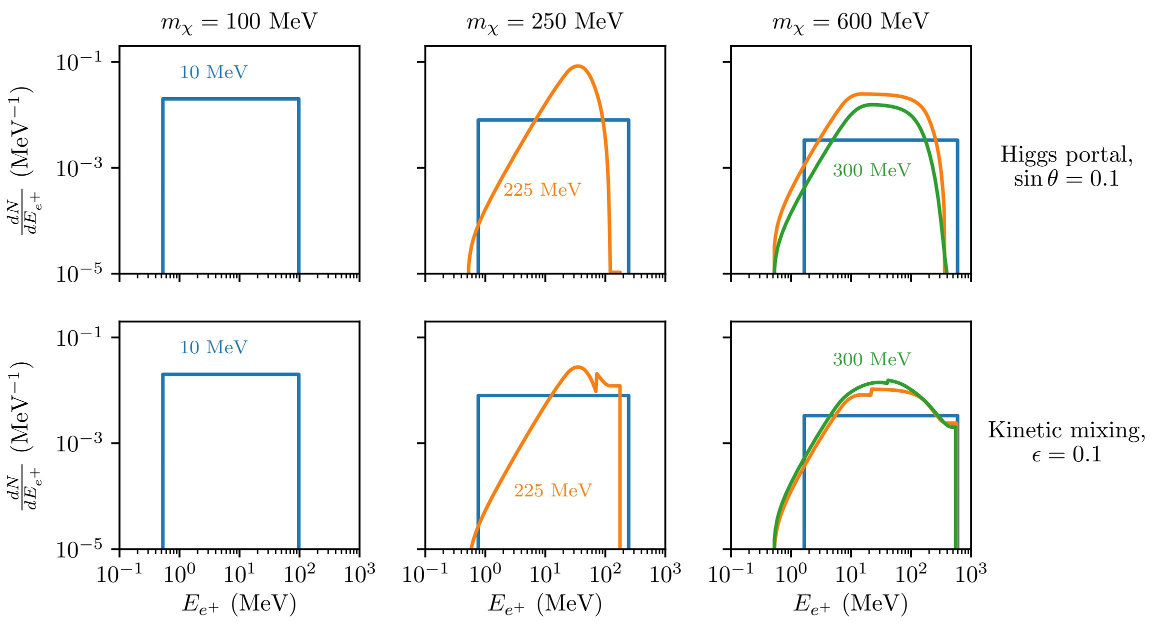

Figure 13. **Positron spectra for DM annihilating into two mediators.** The rows correspond to the Higgs portal scalar model and kinetically mixed vector model.

directly.

A method (more precisely, an `InterpolatedUnivariateSpline`) to compute the total positron spectrum convolved with an energy resolution function is obtained using:

```
>>> e_cm = 305.                        # DM center of mass energy
>>> e_p_min, e_p_max = 1., 100.   # define energy range
>>> energy_res = lambda e: 0.05
>>> dnde_p_conv = hp.total_conv_positron_spectrum_fn(
...     e_p_min, e_p_max, e_cm, energy_res)
>>> dnde_p_conv(20.)
0.008336136582135356
>>> dnde_p_conv.integral(10, 100)  # integrate spectrum
0.6369485109837103
```

7 Gamma Ray Limits

In `hazma` we include two routines to set constraints on the DM self-annihilation cross section $\langle\sigma v\rangle_{\bar{\chi}\chi}$ using current gamma-ray flux measurements and to make projections for planned detectors.

As is well-known, the primary gamma-ray flux from DM annihilating in a region of the sky subtending a solid angle $\Delta\Omega$ is

$$\left.\frac{d\Phi}{dE_\gamma}\right|_{\bar{\chi}\chi}(E) = \frac{\Delta\Omega}{4\pi}\cdot\frac{\langle\sigma v\rangle_{\bar{\chi}\chi}}{2f_\chi m_\chi^2}\cdot J\cdot\left.\frac{dN}{dE_\gamma}\right|_{\bar{\chi}\chi}(E). \tag{7.1}$$

The factor f_χ is 1 if the DM is self-conjugate and 2 otherwise (2 for the models currently included with `hazma`). The J factor counts the number of pairs of DM particles in the observing region:

$$J \equiv \frac{1}{\Delta\Omega} \int_{\Delta\Omega} d\Omega \int_{\text{LOS}} dl\, \rho(r(l,\psi))^2, \tag{7.2}$$

where $\rho(r)$ is the DM density profile. The annihilation spectrum $dN/dE_\gamma|_{\bar{\chi}\chi}$ was computed in detail in the previous section. A detector with finite energy resolution observes the smoothed flux

$$\left.\frac{d\Phi_\epsilon}{dE_\gamma}\right|_{\bar{\chi}\chi}(E) \equiv \int dE'\, R_\epsilon(E|E')\, \left.\frac{d\Phi}{dE_\gamma}\right|_{\bar{\chi}\chi}(E'), \tag{7.3}$$

where the spectral resolution function is the probability that a gamma ray with true energy E' is reconstructed with energy E, and can be approximated as a Gaussian [51]:

$$R_\epsilon(E|E') \equiv \frac{1}{\sqrt{2\pi}\,\epsilon(E')\,E'} \exp\left[-\frac{(E-E')^2}{2(\epsilon(E')\,E')^2}\right]. \tag{7.4}$$

In the limit-setting procedure for *existing* experiments, the integral of the smoothed spectrum in each energy bin is required not to exceed the observed value plus twice the upper error bar: $\Phi_\epsilon^{(i)}|_{\bar{\chi}\chi} < \Phi^{(i)}|_{\text{obs}} + 2\sigma^{(i)}$. While better limits can be obtained by assuming a background model, this introduces significant systematic uncertainties since the model is derived from precisely the observations under consideration and would thus only marginally improve the limits on $\langle\sigma v\rangle_{\bar{\chi}\chi}$ [52].

For assessing the discovery reach by *upcoming* experiments, `hazma` uses an unbinned procedure. A detector can be characterized by an effective area $A_{\text{eff}}(E)$ and an (energy-independent) observing time T_{obs}. Over the energy range $[E_{\min}, E_{\max}]$, the number of photons observed is then

$$N_\gamma|_{\bar{\chi}\chi} = \int_{E_{\min}}^{E_{\max}} dE\, T_{\text{obs}}\, A_{\text{eff}}(E)\, \left.\frac{d\Phi_\epsilon}{dE_\gamma}\right|_{\bar{\chi}\chi}(E). \tag{7.5}$$

Assuming Poisson statistics, the minimum-detectable $\langle\sigma v\rangle_{\bar{\chi}\chi}$ is obtained by requiring the signal-to-noise ratio to be significant at the $n_\sigma = 5$ level: $N_\gamma|_{\bar{\chi}\chi}/\sqrt{N_\gamma|_{\text{bg}}} = n_\sigma$. Holding the DM mass fixed, the lowest-possible value of $\langle\sigma v\rangle_{\bar{\chi}\chi}$ is found by optimizing over $E_{\min}$ and $E_{\max}$ to maximize $N_\gamma|_{\bar{\chi}\chi}/\sqrt{N_\gamma|_{\text{bg}}}$. This energy range does not completely align with spectra features. For example, in the case of the π^0 spectrum, setting $E_{\min}$ equal to the left boundary of the "box" would include too much of the $d\Phi/dE_\gamma|_{\text{bg}} \propto E_\gamma^{-2}$ spectrum.

The unbinned analysis requires a background model, two of which are included with `hazma`. The simplest was obtained by fitting a power law to COMPTEL and EGRET data from high galactic latitudes over the energy range 0.8 MeV − 10 GeV [53]. This is suitable for projecting limits on $\langle\sigma v\rangle_{\bar{\chi}\chi}$ from dwarf galaxy observations. A model derived using `GALPROP` to fit existing MeV gamma-ray data in the region $|b| < 10°$, $|l| < 30°$ is also included to assess the sensitivity to emission from the Galactic Center.

These two limit-setting procedures can currently be applied in `hazma` to data from three existing experiments (COMPTEL, EGRET and Fermi-LAT) and one proposed detector (e-ASTROGAM). The energy ranges, approximate effective areas and energy resolutions for these are supplied in Tab. (1). Each gamma-ray data set is paired with a target region, defined

Detector	Energy range	$A_{\rm eff}$ (cm^2)	ϵ
COMPTEL [49]	0.7 – 27 MeV	50	5%
EGRET [54]	27 MeV – 8.6 GeV	10 – 10^3	18%
Fermi-LAT [55]	150 MeV – 95 GeV	10^3 – 10^4	7.5%
e-ASTROGAM [56]	0.3 MeV – 3 GeV	10^2 – 10^3	$\lesssim$2%, E_γ<10 MeV 20–30%, $E_\gamma \geq$10 MeV

Table 1. **Figures of merit for the detectors used in** **`hazma`.** Note that the effective area is only used in `hazma` for projecting limits for e-ASTROGAM.

Detector	Region	$\Delta\Omega$ (sr)	J (MeV^2 cm^{-5} sr^{-1})
COMPTEL [49]	$\|b\| < 20°,\ \|l\| < 60°$	1.433	3.725×10^{28}
EGRET [57]	$20° < \|b\| < 60°,\ \|l\| < 180°$	6.585	3.79×10^{27}
Fermi-LAT [58]	$8° < \|b\| < 90°,\ \|l\| < 180°$	10.817	4.698×10^{27}
e-ASTROGAM [56]	$\|b\| < 10°,\ \|l\| < 10°$	0.121	1.795×10^{29}

Table 2. **Target regions and data references (first column) for detectors included in** **`hazma`.** The J-factors for COMPTEL, EGRET and Fermi-LAT are taken from [59] and the value for e-ASTROGAM is from ref. [9]. All assume an NFW profile for the galactic DM distribution [9].

by the galactic coordinate ranges, solid angles and J-factors (assuming an NFW distribution for the galactic DM halo) in Tab. (2). These are packaged into `FluxMeasurements` objects in `hazma`.

The limit-setting procedures are methods in the `Theory` class. For example, the following snippet imports a `FluxMeasurement` object containing information about EGRET observations and computes the $\langle\sigma v\rangle_{\bar{\chi}\chi}$ limits for the Higgs portal model at three DM masses in cm^3/s:

```
>>> from hazma.scalar_mediator import ScalarMediator
>>> from hazma.gamma_ray_parameters import egret_diffuse, TargetParams
>>> egret_diffuse.target  # target region information
TargetParams(J=3.79e+27, dOmega=6.584844306798711)
>>> sm = ScalarMediator(mx=150., ms=1e4, gsxx=1., gsff=0.1, gsGG=0.5,
...                     gsFF=0, lam=1e5)
>>> sm.binned_limit(egret_diffuse)
7.841784676562726e-27  # cm^3 / s
```

It is important to note that `hazma` only uses the model parameters (m_S, $g_{S\chi\chi}$, g_{Sff}, g_{SGG}, g_{SFF} and Λ) to compute the *shape* of the spectrum and treats $\langle\sigma v\rangle_{\bar{\chi}\chi}$ as an independent parameter (c.f. 7.1). To relate the model parameters to $\langle\sigma v\rangle_{\bar{\chi}\chi}$, the user can employ the method `Theory.annihilation_cross_sections()`. The `FluxMeasurement` objects for COMPTEL and Fermi-LAT observations are named `comptel_diffuse` and `fermi_diffuse` and can be imported similarly. App. (C.1) describes how to create new flux measurement objects.

Computing unbinned limits requires specifying several quantities:

- An effective area function returning $A_{\rm eff}(E)$ in cm^2;
- An energy resolution function returning $\epsilon(E')$ in %;

- An observing time in seconds;
- An object of type TargetParams containing the J-factor and $\Delta\Omega$ for the target region;
- A BackgroundModel object, which has a method dPhi_dEdOmega() returning the differential gamma-ray flux flux per solid angle for the background model, and an attribute e_range specifying the range of E_γ values for which the model is valid.

Using the same instance of scalar_mediator and DM masses as above, we can compute the projected limits for e-ASTROGAM using the $10^\circ \times 10^\circ$ window around the Galactic Center from Tab. (2) as the target:

```
>>> from hazma.gamma_ray_parameters import A_eff_e_astrogam
>>> from hazma.gamma_ray_parameters import energy_res_e_astrogam
>>> from hazma.gamma_ray_parameters import T_obs_e_astrogam
>>> from hazma.gamma_ray_parameters import gc_target, gc_bg_model
>>> gc_target.J                                  # target J-factor
1.795e+29
>>> gc_target.dOmega                             # target extent
0.12122929761504078
>>> sm.unbinned_limit(A_eff_e_astrogam,          # effective area
...                   energy_res_e_astrogam,     # energy resolution
...                   T_obs_e_astrogam,          # observing time
...                   gc_target,                 # target region
...                   gc_bg_model)               # background model
5.63534935653953e-30  # cm^3 / s
```

Modifying these ingredients to study other planned gamma-ray detectors is described in App. (C.1.

We do not currently use secondary radiation to set limits on our models. Such limits are highly dependent on (for example) the choice of observational target, the magnetic field intensity and spatial structure [60, 61]; Ref. ([40]) calculates the secondary emission for select MeV dark matter annihilation final states. Since hazma can already compute the relevant input spectra for determining secondary spectra, we plan to incorporate this into future versions of the code.

8 Cosmic Microwave Background limits

Dark matter annihilations at recombination inject ionizing particles into the photon-baryon plasma. The resulting changes in the residual ionization fraction and baryon temperature modify the CMB temperature and polarization power spectra, particularly at small scales, which puts a constraint on $\langle\sigma v\rangle_{\bar{\chi}\chi,\mathrm{CMB}}$, the DM self-annihilation cross section at recombination [62–67]. The size of the effect of the changes depend on the energy per unit volume per unit time, which is related to the effective parameter

$$p_{\mathrm{ann}} \equiv f_{\mathrm{eff}}^{\chi} \frac{\langle\sigma v\rangle_{\bar{\chi}\chi,\mathrm{CMB}}}{m_\chi}. \tag{8.1}$$

The current 95% upper limit on p_{ann} from Planck is 3.5×10^{-31} cm^3 s^{-1} MeV^{-1} [67]. f_{eff}^{χ} is the fraction of energy per DM annihilation imparted to the plasma. This depends on how many electrons/positrons and photons are produced per DM annihilation (ie, the spectra computed earlier in this work) and how efficiently these particles re-ionize the CMB (quantified by the factors f_{eff}^{γ} and $f_{\text{eff}}^{e^+e^-}$) [66]:

$$f_{\text{eff}}^{\chi} = \frac{1}{E_{\text{CM}}} \int_0^{E_{\text{CM}}/2} dE \left(2 f_{\text{eff}}^{e^+e^-} E \left. \frac{dN}{dE_{e^+}} \right|_{\bar{\chi}\chi} + f_{\text{eff}}^{\gamma} E \left. \frac{dN}{dE_{\gamma}} \right|_{\bar{\chi}\chi} \right). \tag{8.2}$$

If the DM self-annihilation cross section is velocity dependent (as is the case for the scalar mediator model, where the annihilation is p-wave), the DM velocity $v_{\chi,\text{CMB}}$ at recombination is required to relate $\langle\sigma v\rangle_{\bar{\chi}\chi,\text{CMB}}$ to the present-day value of $\langle\sigma v\rangle_{\bar{\chi}\chi}$ in indirect detection targets. This velocity is also required for computing the center of mass energy in DM self-annihilation events at recombination. The velocity depends on the kinetic decoupling temperature, T_{KD}. Before kinetic decoupling DM initially is coupled to the plasma so that $v_{\chi,\text{CMB}} = \sqrt{3T_\gamma/m_\chi}$, but afterwards it cools more quickly ($T_\chi \propto z^2$). The DM velocity is thus [52]

$$v_{\chi,\text{CMB}} = \sqrt{\frac{3T_\chi}{m_\chi}} \approx 2 \times 10^{-4} \left(\frac{T_\gamma}{1\text{ eV}}\right) \left(\frac{1\text{ MeV}}{m_\chi}\right) \left(\frac{10^{-4}}{x_{\text{kd}}}\right)^{1/2}, \tag{8.3}$$

where $x_{\text{kd}} \equiv T_{\text{KD}}/m_\chi$ generally lies in the range $10^{-6} - 10^{-4}$. Since this velocity is much smaller than the characteristic velocity in the Milky Way, the CMB constraints on $\langle\sigma v\rangle_{\bar{\chi}\chi}$ for models with p-wave suppression are in general much weaker than the gamma-ray ones. The CMB constraints in the s-wave case are generally much stronger, but can be evaded in models where the DM is partially produced through the decay of a heavier dark-sector degree of freedom after recombination [19].

The quantity f_{eff}^{χ} as well as the separate $e^{\pm}$ and γ terms in Eqn. (8.2) can be computed for a given kinetic decoupling temperature:

```
>>> from hazma.scalar_mediator import ScalarMediator
>>> sm = ScalarMediator(mx=150., ms=1e4, gsxx=1., gsff=0.1, gsGG=0.5,
...                     gsFF=0, lam=1e5)
>>> x_kd = 1e-4
>>> sm.f_eff(x_kd), sm.f_eff_ep(x_kd), sm.f_eff_g(x_kd)
(0.4373205896743133, 0.1657568724934657, 0.2715637171808476)
```

Fig. (14) shows this factor as a function of DM and mediator mass for the scalar and vector models with the couplings we have been using as benchmarks. There are sharp variations in f_{eff}^{χ} arising from SM or two-mediator final states becoming accessible as the DM mass changes. Deriving the kinetic decoupling temperature for particular models rather than leaving it as a free parameter is left for future work (or the user).

The `Theory.cmb_limit()` function computes the constraint on $\langle\sigma v\rangle_{\bar{\chi}\chi,\text{CMB}}$ given a value of x_{kd} and upper limit on p_{ann}:

```
>>> from hazma.cmb import p_ann_planck_temp_pol as p_ann
>>> p_ann
3.5e-31  # cm^3 / MeV / s
>>> x_kd = 1e-4
>>> sm = HiggsPortal(mx=150., ms=1e4, gsxx=1., stheta=0.01)
```

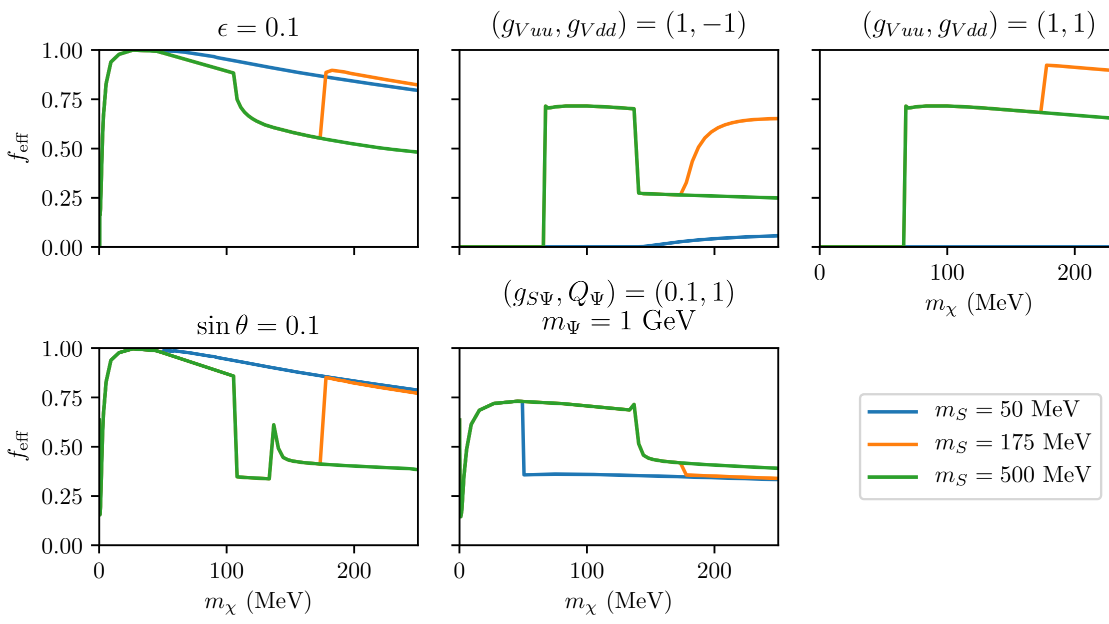

Figure 14. **Efficiency of energy injection into the CMB by dark matter annihilation.** The top panels are for the vector model with kinetic mixing and two possible values for mediator couplings to quarks only. The bottom row shows $f_{\rm eff}^\chi$ for the scalar mediator with Higgs portal and heavy quark-type couplings. The DM-mediator coupling is fixed to one in all panels. Note that the curves for different mediator masses have some overlap.

```
>>> sm.cmb_limit(x_kd, p_ann)
1.2209926980668773e-28  # cm^3 / s
```

As with the gamma-ray limits described in the previous section, the parameters of the theory (here m_S, $g_{S\chi\chi}$ and s_θ) determine the *shapes* of the photon and $e^\pm$ spectra, and `Theory.cmb_limit()` treats $\langle\sigma v\rangle_{\bar\chi\chi,\rm CMB}$ as a free parameter. The user can find which model parameters reproduce this cross section using `Theory.annihilation_cross_sections()`.

9 Conclusion

We have introduced `hazma`, a code for the calculation of gamma-ray and electron-positron spectra as well as constraints and projected reach for future telescopes for sub-GeV dark matter annihilation or decay. `hazma` is timely both because of forthcoming new observational capabilities at sub-GeV gamma-ray energies, and because of renewed interest in dark matter models beyond the WIMP paradigm and at lower-than-usually-considered masses.

`hazma` allows users to compute both spectra of gamma rays, electrons and positrons for individual annihilation or decay final states containing arbitrarily many particles, as well as for user-implemented or a set of built-in dark matter models. In the latter case, the code computes the relevant branching fractions and the ensuing decay chains leading to photons, electrons and positrons, and calculates model-dependent emission processes such as e.g. internal bremsstrahlung. The interactions between the mediator particles in these models and the final state mesons is implemented using chiral perturbation theory techniques.

`hazma` contains several updates and refinements over how gamma-ray spectra have been calculated in the recent literature. These include several radiative decay spectra, such as from charged pion decay or from muon decay, final state radiation calculations going beyond the Altarelli-Parisi splitting function approximation for the built-in models, and the ability to include arbitrarily many mesons and leptons in the final states.

We presented numerous snippets of code, and all code used to produce the figures shown herein manuscript is available through clickable links. We also showed a few advanced usage features, such as adding new gamma-ray experiments for deriving constraints from existing data or assessing projected reach for planned detectors, and implementing user-defined models.

Acknowledgments

LM and SP are partly supported by the U.S. Department of Energy grant number de-sc0010107. We acknowledge with gratitude early collaboration on this project with Francesco D'Eramo. Francesco D'Eramo was the first to put forward the idea of using chiral perturbation theory in the context of accurately calculating the annihilation and decay products of MeV dark matter. He was also responsible for several of the early calculations, part of the writing of the text, and for overall educating us all on numerous topics in chiral perturbation theory. We thank Michael Peskin for important input and feedback, and Regina Caputo for discussions about MeV gamma-ray observatories. Finally, we thank Eulogio Oset and Jose Antonio Oller for clarifying some points about unitarized chiral perturbation theory.

A Installation

`hazma` was developed with python3.6, and it is recommended to use python3. While `hazma` should work correctly with python2.7, this version of python will soon lose support. Before installing `hazma`, the user needs to install several well-established python packages: `cython`, `scipy`, `numpy`, and `scikit-image`. Theses are easily installed by using PyPI. If the user has PyPI installed on their system, then these packages can be installed using

```
pip install cython, scipy, numpy, scikit-image, matplotlib
```

`hazma` can be installed in the same way, using:

```
pip install hazma
```

This will download a tarball from the PyPI repository, compile all the c-code and install `hazma` on the system. Alternatively, the user can install `hazma` by downloading the package from `https://github.com/LoganAMorrison/Hazma.git`. Once downloaded, navigate to the package directory using the command line and run either

```
pip install .
```

or

```
python setup.py install
```

Note that since hazma makes extensive usage of the package cython, the user will need to have a c and c++ compiler installed on their system (for example gcc and g++ on unix-like systems or Microsoft Visual Studios 2015 or later on Windows). For more information, see the cython installation guide: https://cython.readthedocs.io/en/latest/src/quickstart/install.html

B Basic Usage

B.1 Using the Built-In Models

Throughout the text, we gave usage examples of how to use the models built into hazma. Here we give a compact review of how to use these models in hazma. All the models built into hazma have identical interfaces. The only difference in the interfaces is the parameters which need to be specified for the particular model being used. Thus, we will only show the usage of one of the models. The others can be used in an identical fashion. The example we will use is the KineticMixing model. To create a KineticMixing model, we use:

```
# Import the model
>>> from hazma.vector_mediator import KineticMixing
# Specify the parameters
>>> params = {'mx': 250.0, 'mv': 1e6, 'gvxx': 1.0, 'eps': 1e-3}
# Create KineticMixing object
>>> km = KineticMixing(**params)
```

Here we have created a model with a dark matter fermion of mass 250 MeV, a vector mediator which mixes with the standard model with a mass of 1 TeV. We set the coupling of the vector mediator to the dark matter, $g_{V\chi} = 1$ and set the kinetic mixing parameter $\epsilon = 10^{-3}$. To list all of the available final states for which the dark matter can annihilate into, we use:

```
>>> km.list_annihilation_final_states()
['mu mu', 'e e', 'pi pi', 'pi0 g', 'pi0 v', 'v v']
```

This tells us that we can potentially annihilate through: $\chi\bar{\chi} \to \mu^+\mu^-, e^+e^-, \pi^+\pi^-, \pi^0\gamma, \pi^0 V$ or $V^\mu V^\mu$. However, which of these final states is actually available depends on the center of mass energy. We can see this fact by looking at the annihilation cross sections or branching fractions, which can be computed using:

```
>>> cme = 2.0 * km.mx * (1.0 + 0.5 * 1e-6)
>>> km.annihilation_cross_sections(cme)
{'mu mu': 8.948397750213 93e-25,
 'e e': 9.064036692829845e-25,
 'pi pi': 1.2940469635262499e-25,
 'pi0 g': 5.206158864833925e-29,
 'pi0 v': 0.0,
 'v v': 0.0,
 'total': 1.9307002022456507e-24}
>>> km.annihilation_branching_fractions(cme)
{'mu mu': 0.46347940191883763,
 'e e': 0.4694688839980031,
 'pi pi': 0.06702474894968717,
 'pi0 g': 2.6965133472190545e-05,
 'pi0 v': 0.0,
 'v v': 0.0}
```

Here we have chosen a realistic center of mass energy for dark matter in our galaxy, which as a velocity dispersion of $\delta v \sim 10^{-3}$. We can that the $V^{\mu}V^{\mu}$ final state is unavailable, as it should be since the vector mediator mass is too heavy. In this theory, the vector mediator can decay. If we would like to know the decay width and the partial widths, we can use:

```
>>> km.partial_widths()
{'pi pi': 0.0018242846671063036,
 'pi0 g': 2.1037425397685694,
 'x x': 79577.47154594581,
 'e e': 0.007297139521307641,
 'mu mu': 0.007297139521307641,
 'total': 79579.5917070493}
```

If we would like to know the gamma-ray spectrum from dark matter annihilations, we can use

```
# Specify photon energies to compute spectrum at
>>> photon_energies = np.array([cme/4])
>>> km.spectra(photon_energies, cme)
{'mu mu': array([2.94759389e-05]),
 'e e': array([0.000013171]),
 'pi pi': array([2.20142244e-06]),
 'pi0 g': array([2.29931655e-07]),
 'pi0 v': array([0.]),
 'v v': array([0.]),
 'total': array([0.00016362])}
```

Note that we only used a single photon energy because of display purposes, but in general the user can specify any number of photon energies. If the user would like access to the underlying spectrum functions, they can use:

```
>>> spec_funs = km.spectrum_functions()
>>> spec_funs['mu mu'](photon_energies, cme)
[6.35970849e-05]
>>> mumu_bf = km.annihilation_branching_fractions(cme)['mu mu']
>>> mumu_bf * spec_funs['mu mu'](photon_energies, cme)
[2.94759389e-05]
```

Notice that the direct call to the spectrum function for $\chi\bar{\chi} \to \mu^+\mu^-$ doesn't given the same result as `km.spectra(photon_energies, cme)['mu mu']`. This is because the branching fractions are not applied for the `spec_funs = km.spectrum_functions()`. If the user doesn't care about the underlying components of the gamma-ray spectra, the can simply call

```
>>> km.total_spectrum(photon_energies, cme)
array([0.00016362])
```

to get the total gamma-ray spectrum. The reader may have caught the fact that there is a gamma-ray line in the spectrum for $\chi\bar{\chi} \to \pi^0\gamma$. To get the location of this line, the user can use:

```
>>> km.gamma_ray_lines(cme)
{'pi0 g': {'energy': 231.78145156177675, 'bf': 2.6965133472190545e-05}}
```

This tells us the process which produces the line, the location of the line and the branching fraction for the process. We do not include the line in the total spectrum since the line produces a Dirac-delta function. In order to get a realistic spectrum including the line, we need to convolve the gamma-ray spectrum with an energy resolution. This can be achieved using:

```
>>> min_photon_energy = 1e-3
>>> max_photon_energy = cme
>>> energy_resolution = lambda photon_energy : 1.0
>>> number_points = 1000
>>> spec = km.total_conv_spectrum_fn(min_photon_energy, max_photon_energy,
...                                  cme, energy_resolution, number_points)
>>> spec(cme / 4)  # compute the spectrum at a photon energy of `cme/4`
array(0.001718)
```

The `km.total_conv_spectrum_fn` computes and returns an interpolating function of the convolved function. An important thing to note here is that the `km.total_conv_spectrum_fn` takes in a *function* for the energy resolution. This allows the user to define the energy resolution to depend on the specific photon energy, which is often the case for gamma-ray telescopes.

Next we present the positron spectra. These have an identical interface to the gamma-ray spectra, so we only show how to call the functions and we suppress the output

```
>>> from hazma.parameters import electron_mass as me
>>> positron_energies = np.logspace(np.log10(me), np.log10(cme), num=100)
>>> km.positron_spectra(positron_energies, cme)
>>> km.positron_lines(cme)
>>> km.total_positron_spectrum(positron_energies, cme)
>>> dnde_pos = km.total_conv_positron_spectrum_fn(min(positron_energies),
...                                               max(positron_energies),
...                                               cme,
...                                               energy_resolution,
...                                               number_points)
```

The last thing that we would like to demonstrate is how to compute limits. In order to compute the limits on the annihilation cross section of a model from a gamma-ray telescope, say EGRET, we can use:

```
>>> from hazma.gamma_ray_parameters import egret_diffuse
# Choose DM masses from half the electron mass to 250 MeV
>>> mxs = np.linspace(me/2., 250., num=100)
# Compute limits from e-ASTROGAM
>>> limits = np.zeros(len(mxs), dtype=float)
>>> for i, mx in enumerate(mxs):
...     km.mx = mx
...     limits[i] = km.binned_limit(egret_diffuse)
```

Similarly, if we would like to set constraints using e-ASTROGAM, one can use:

```
>>> # Import target and background model for the e-ASTROGAM telescope
>>> from hazma.gamma_ray_parameters import gc_target, gc_bg_model
>>> # Import telescope parameters
>>> from hazma.gamma_ray_parameters import A_eff_e_astrogam
>>> from hazma.gamma_ray_parameters import energy_res_e_astrogam
>>> from hazma.gamma_ray_parameters import T_obs_e_astrogam
>>> # Choose DM masses from half the electron mass to 250 MeV
>>> mxs = np.linspace(me/2., 250., num=100)
>>> # Compute limits
>>> limits = np.zeros(len(mxs), dtype=float)
>>> for i, mx in enumerate(mxs):
...     km.mx = mx
...     limits[i] = km.unbinned_limit(A_eff_e_astrogam,
...                                   energy_res_e_astrogam,
...                                   T_obs_e_astrogam,
...                                   gc_target,
...                                   gc_bg_model)
```

B.2 Subclassing the Built-In Models

As mentioned in Sec. (2), the user may not be interested in the generic models built into `hazma`, but instead a more specialized model. In this case, it makes sense for the user to subclass one of the models (i.e. create a class which inherits from one of the models.) As and example, we illustrate how to do this with the Higgs-portal model (of course this model is already built into `hazma`, but it works nicely as an example.) Recall that the full set of parameters for the scalar mediator model are:

1. m_χ: dark matter mass,
2. m_S: scalar mediator mass,
3. $g_{S\chi}$: coupling of scalar mediator to dark matter,
4. g_{Sf}: coupling of scalar mediator to standard model fermions,
5. g_{SG}: effective coupling of scalar mediator to gluons,
6. g_{SF}: effective coupling of scalar mediator to photons and

7. Λ: cut-off scale for the effective interactions.

In the case of the Higgs-portal model, the scalar mediator talks to the standard model only through the Higgs boson, i.e. it mixes with the Higgs. Therefore, the scalar mediator inherits its interactions with the standard model fermions, gluons and photon through the Higgs. In Sec. (3.1), we gave the relationships between the couplings $g_{Sf}, g_{SG}, g_{SF}, \Lambda$ and $\sin\theta, v_h$, where $\sin\theta$ is the mixing angle for the scalar and Higgs and v_h is the Higgs vacuum expectation value. These relationships are:

$$g_{Sf} = \sin\theta, \qquad g_{SG} = 3\sin\theta, \qquad g_{SF} = -\frac{5}{6}\sin\theta, \qquad \Lambda = v_h. \tag{B.1}$$

Therefore, the Higgs-portal model doesn't need to know about all of the couplings of the generic scalar mediator model. It only needs to know about $m_\chi, m_S, g_{S\chi}$ and $\sin\theta$. Below, we construct a class which subclasses the scalar mediator class to implement the Higgs-portal model.

```python
from hazma.scalar_mediator import ScalarMediator
from hazma.parameters import vh

class HiggsPortal(ScalarMediator):
    def __init__(self, mx, ms, gsxx, stheta):
        self._lam = vh
        self._stheta = stheta
        super(HiggsPortal, self).__init__(mx, ms, gsxx, stheta, 3.*stheta,
                                          -5.*stheta/6., vh)

    @property
    def stheta(self):
        return self._stheta

    @stheta.setter
    def stheta(self, stheta):
        self._stheta = stheta
        self.gsff = stheta
        self.gsGG = 3. * stheta
        self.gsFF = - 5. * stheta / 6.

    # Hide underlying properties' setters
    @ScalarMediator.gsff.setter
    def gsff(self, gsff):
        raise AttributeError("Cannot set gsff")

    @ScalarMediator.gsGG.setter
    def gsGG(self, gsGG):
        raise AttributeError("Cannot set gsGG")

    @ScalarMediator.gsFF.setter
    def gsFF(self, gsFF):
        raise AttributeError("Cannot set gsFF")
```

There are a couple things to note about our above implementation. First, our model only takes in $m_\chi, m_S, g_{S\chi}$ and $\sin\theta$, as desired. But the underlying model, i.e. the `ScalarMediator` model only knows about $m_\chi, m_S, g_{S\chi}, g_{Sf}, g_{SG}, g_{SF}$ and Λ. So if we update $\sin\theta$, we additionally need to update the underlying parameters, g_{Sf}, g_{SG}, g_{SF} and Λ. The easiest way to do this is using getters and setters by defining $\sin\theta$ to be a `property` through the `@property` decorator. Then every time we update $\sin\theta$, we can also update the underlying parameters. The second thing to note is that we want to make sure we don't accidentally change the underlying parameters directly, since in this model, they are only defined through $\sin\theta$. We an ensure that we cannot change the underlying parameters directly by overriding the getters and setters for `gsff`, `gsGG` and `gsGG` and raising an error if we try to change them. This isn't strictly necessary (as long as the user is careful), but can help avoid confusing behavior.

C Advanced Usage

C.1 Adding New Gamma-Ray Experiments

Currently `hazma` only includes information for producing projected unbinned limits with e-ASTROGAM, using the dwarf Draco or inner $10° \times 10°$ region of the Milky Way as a target. Adding new detectors and target regions is straightforward. As described in Sec. (7), a detector is characterized by the effective area $A_{\text{eff}}(E)$, the energy resolution $\epsilon(E)$ and observation time T_{obs}. In `hazma`, the first two can be any callables (functions) and the third must be a float. The region of interest is defined by a `TargetParams` object, which can be instantiated with:

```
>>> from hazma.gamma_ray_parameters import TargetParams
>>> tp = TargetParams(J=1e29, dOmega=0.1)
```

The background model should be packaged in an object of type `BackgroundModel`. This lightweight class has a function `dPhi_dEdOmega()` for computing the differential photon flux per solid angle (in MeV^{-1} sr) and an attribute `e_range` specifying the energy range over which the model is valid (in MeV). New background models are defined by passing these two the initializer:

```
>>> from hazma.background_model import BackgroundModel
>>> bg = BackgroundModel(e_range=[0.5, 1e4],
...                      dPhi_dEdOmega=lambda e: 2.7e-3 / e**2)
```

As explained in Sec. (7), gamma-ray observation information from Fermi-LAT, EGRET and COMPTEL is included with `hazma`, and other observations can be added using the container class `FluxMeasurement`. The initializer requires:

- The name of a CSV file containing gamma-ray observations. The file's columns must contain:

    1. Lower bin edge (MeV)
    2. Upper bin edge (MeV)

3. $E^n d^2\Phi/dE\,d\Omega$ (in MeV^{n-1} cm^{-2} s^{-1} sr^{-1})
4. Upper error bar (in MeV^{n-1} cm^{-2} s^{-1} sr^{-1})
5. Lower error bar (in MeV^{n-1} cm^{-2} s^{-1} sr^{-1})

Note that the error bar values are their y-coordinates, not their relative distances from the central flux.

- The detector's energy resolution function.
- A `TargetParams` object for the target region.

For example, a CSV file `obs.csv` containing observations

```
150.0,275.0,0.0040,0.0043,0.0038
650.0,900.0,0.0035,0.0043,0.003
```

with $n = 2$ for an instrument with energy resolution $\epsilon(E) = 0.05$ observing the target region `tp` defined above can be loaded using[7]

```
>>> from hazma.flux_measurement import FluxMeasurement
>>> obs = FluxMeasurement("obs.dat", lambda e: 0.05, tp)
```

The attributes of the `FluxMeasurement` store all of the provide information, with the E^n prefactor removed from the flux and error bars, and the errors converted from the positions of the error bars to their sizes. These are used internally by the `Theory.binned_limit()` method, and can be accessed as follows:

```
>>> obs.e_lows, obs.e_highs
(array([150., 650.]), array([275., 900.]))
>>> obs.target
<hazma.gamma_ray_parameters.TargetParams at 0x1c1bbbafd0>
>>> obs.fluxes
array([8.85813149e-08, 5.82726327e-09])
>>> obs.upper_errors
array([6.64359862e-09, 1.33194589e-09])
>>> obs.lower_errors
array([4.42906574e-09, 8.32466181e-10])
>>> obs.energy_res(10.)
0.05
```

C.2 User-Defined Models

In this subsection, we demonstrate how to implement new models in Hazma. A notebook containing all the code in this appendix can be downloaded from GitHub. The model we will consider is an effective field theory with a Dirac fermion DM particle which talks to

[7]If the CSV containing the observations uses a different power of E than $n = 2$, this can be specified using the `power` keyword argument to the initializer for `FluxMeasurement`.

neutral and charged pions through gauge-invariant dimension-5 operators. The Lagrangian for this model is:

$$\mathcal{L} \supset \frac{c_1}{\Lambda}\overline{\chi}\chi\pi^+\pi^- + \frac{c_2}{\Lambda}\overline{\chi}\chi\pi^0\pi^0 \tag{C.1}$$

where c_1, c_2 are dimensionless Wilson coefficients and Λ is the cut-off scale of the theory. In order to implement this model in Hazma, we need to compute the annihilation cross sections and the FSR spectra. The annihilation channels for this model are simply $\bar{\chi}\chi \to \pi^0\pi^0$ and $\bar{\chi}\chi \to \pi^+\pi^-$. The computations for the cross sections are straight forward and yield:

$$\sigma(\bar{\chi}\chi \to \pi^+\pi^-) = \frac{c_1^2\sqrt{1-4\mu_\pi^2}\sqrt{1-4\mu_\chi^2}}{32\pi\Lambda^2} \tag{C.2}$$

$$\sigma(\bar{\chi}\chi \to \pi^0\pi^0) = \frac{c_2^2\sqrt{1-4\mu_{\pi^0}^2}\sqrt{1-4\mu_\chi^2}}{8\pi\Lambda^2} \tag{C.3}$$

where Q is the center of mass energy, $\mu_\chi = m_\chi/Q$, $\mu_\pi = m_{\pi^\pm}/Q$ and $\mu_{\pi^0} = m_{\pi^0}/Q$. In addition to the cross sections, we need the FSR spectrum for $\overline{\chi}\chi \to \pi^+\pi^-\gamma$. This is:

$$\frac{dN(\bar{\chi}\chi \to \pi^+\pi^-\gamma)}{dE_\gamma} = \frac{\alpha\left(2f(x) - 2\left(1-x-2\mu_\pi^2\right)\log\left(\frac{1-x-f(x)}{1-x+f(x)}\right)\right)}{\pi\sqrt{1-4\mu_\pi^2}x} \tag{C.4}$$

where

$$f(x) = \sqrt{1-x}\sqrt{1-x-4\mu_\pi^2} \tag{C.5}$$

We are now ready to set up the Hazma model. For `hazma` to work properly, we will need to define the following functions in our model:

- `annihilation_cross_section_funcs()`: A function returning a dictionary of the annihilation cross sections functions, each of which take a center of mass energy.
- `spectrum_funcs()`: A function returning a dictionary of functions which take photon energies and a center of mass energy and return the gamma-ray spectrum contribution from each final state.
- `gamma_ray_lines(e_cm)`: A function returning a dictionary of the gamma-ray lines for a given center of mass energy.
- `positron_spectrum_funcs()`: Like `spectrum_funcs()`, but for positron spectra.
- `positron_lines(e_cm)`: A function returning a dictionary of the electron/positron lines for a center of mass energy.

In the interests of compartmentalization, we find it easiest place each of these components in its own class (a mixin in python terminology) which are then combined into a master class representing our model. Before we begin writing the classes, we need to import a few helper functions and constants from `hazma`:

```python
import numpy as np # NumPy is heavily used
import matplotlib.pyplot as plt # Plotting utilities
# neutral and charged pion masses
from hazma.parameters import neutral_pion_mass as mpi0
from hazma.parameters import charged_pion_mass as mpi
from hazma.parameters import qe # Electric charge
# Positron spectra for neutral and charged pions
from hazma.positron_spectra import charged_pion as pspec_charged_pion
# Deay spectra for neutral and charged pions
from hazma.decay import neutral_pion, charged_pion
# The `Theory` class which we will ultimately inherit from
from hazma.theory import Theory
```

Now, we implement a cross section class:

```python
class HazmaExampleCrossSection:
    def sigma_xx_to_pipi(self, Q):
        mupi = mpi / Q
        mux = self.mx / Q

        if Q > 2 * self.mx and Q > 2 * mpi:
            sigma = (self.c1**2 * np.sqrt(1 - 4 * mupi**2) *
                     np.sqrt(1 - 4 * mux**2)**2 /
                     (32.0 * self.lam**2 * np.pi))
        else:
            sigma = 0.0

        return sigma

    def sigma_xx_to_pi0pi0(self, Q):
        mupi0 = mpi0 / Q
        mux = self.mx / Q

        if Q > 2 * self.mx and Q > 2 * mpi0:
            sigma = (self.c2**2 * np.sqrt(1 - 4 * mux**2) *
                     np.sqrt(1 - 4 * mupi0**2) /
                     (8.0 * self.lam**2 * np.pi))
        else:
            sigma = 0.0

        return sigma

    def annihilation_cross_section_funcs(self):
        return {'pi0 pi0': self.sigma_xx_to_pi0pi0,
                'pi pi': self.sigma_xx_to_pipi}
```

The key function is `annihilation_cross_section_funcs`, which is **required** to be implemented. Next, we implement the spectrum functions which will produce the FSR and decay spectra:

```python
class HazmaExampleSpectra:
    def dnde_pi0pi0(self, e_gams, e_cm):
        # Decay spectra for the neutral pions
        return 2.0 * neutral_pion(e_gams, e_cm / 2.0)

    def __dnde_xx_to_pipig(self, e_gam, Q):
        # Unvectorized function for computing FSR spectrum
        mupi = mpi / Q
        mux = self.mx / Q
        x = 2.0 * e_gam / Q
        if 0.0 < x and x < 1. - 4. * mupi**2:
            dnde = (
                qe ** 2
                * (
                    2 * np.sqrt(1 - x) * np.sqrt(1 - 4 * mupi ** 2 - x)
                    + (-1 + 2 * mupi ** 2 + x)
                    * np.log(
                        (-1 + np.sqrt(1 - x) *
                         np.sqrt(1 - 4 * mupi ** 2 - x) + x) ** 2
                        / (1 + np.sqrt(1 - x) *
                           np.sqrt(1 - 4 * mupi ** 2 - x) - x) ** 2
                    )
                )
            ) / (Q * 2.0 * np.sqrt(1 - 4 * mupi ** 2) * np.pi ** 2 * x)
        else:
            dnde = 0

        return dnde

    def dnde_pipi(self, e_gams, e_cm):
        # Sum the FSR and decay spectra for the charged pions
        return (np.vectorize(self.__dnde_xx_to_pipig)(e_gams, e_cm) +
                2. * charged_pion(e_gams, e_cm / 2.0))

    def spectrum_funcs(self):
        return {'pi0 pi0':  self.dnde_pi0pi0,
                'pi pi':  self.dnde_pipi}

    def gamma_ray_lines(self, e_cm):
        return {}
```

The required functions here are `spetrum_funcs` and `gamma_ray_lines`. Note the the second `__dnde_xx_to_pipig` is an unvectorized helper function, which is not to be used directly. Next we implement the positron spectra:

```
class HazmaExamplePositronSpectra:
    def dnde_pos_pipi(self, e_ps, e_cm):
        return pspec_charged_pion(e_ps, e_cm / 2.)

    def positron_spectrum_funcs(self):
        return {"pi pi": self.dnde_pos_pipi}

    def positron_lines(self, e_cm):
        return {}
```

Lastly, we group all of these classes into a master class and we're done:

```
class HazmaExample(HazmaExampleCrossSection,
                   HazmaExamplePositronSpectra,
                   HazmaExampleSpectra,
                   Theory):
    # Model parameters are DM mass: mx,
    # Wilson coefficients: c1, c2 and
    # cutoff scale: lam
    def __init__(self, mx, c1, c2, lam):
        self.mx = mx
        self.c1 = c1
        self.c2 = c2
        self.lam = lam

    @staticmethod
    def list_annihilation_final_states():
        return ['pi pi', 'pi0 pi0']
```

Now we can easily compute gamma-ray spectra, positron spectra and limit on our new model from gamma-ray telescopes. To implement our new model with $m_\chi = 200$ MeV, $c_1 = c_2 = 1$ and $\Lambda = 100$ GeV, we can use:

```
>>> model = HazmaExample(200.0, 1.0, 1.0, 100e3)
```

To compute a gamma-ray spectrum:

```
# Photon energies from 1 keV to 1 GeV
>>> e_gams = np.logspace(-3.0, 3.0, num=150)
# Assume the DM is moving with a velocity of 10^-3
>>> vdm = 1e-3
# Compute CM energy assuming the above velocity
>>> Q = 2.0 * model.mx * (1 + 0.5 * vdm**2)
# Compute spectra
>>> spectra = model.spectra(e_gams, Q)
```

Then we can plot the spectra using:

```
>>> plt.figure(dpi=100)
>>> for key, val in spectra.items():
...     plt.plot(e_gams, val, label=key)
>>> plt.xlabel(r'$E_{\gamma} (\mathrm{MeV})$', fontsize=16)
>>> plt.ylabel(r'$\frac{dN}{dE_{\gamma}} (\mathrm{MeV}^{-1})$', fontsize=16)
>>> plt.xscale('log')
>>> plt.yscale('log')
>>> plt.legend()
```

Additionally, we can compute limits on the thermally-averaged annihilation cross section of our model for various DM masses using

```
# Import target and background model for the E-Astrogam telescope
>>> from hazma.gamma_ray_parameters import gc_target, gc_bg_model
# Choose DM masses from half the pion mass to 250 MeV
>>> mxs = np.linspace(mpi/2., 250., num=100)
# Compute limits from E-Astrogam
>>> limits = np.zeros(len(mxs), dtype=float)
>>> for i, mx in enumerate(mxs):
...     model.mx = mx
...     limits[i] = model.unbinned_limit(target_params=gc_target,
...                                      bg_model=gc_bg_model)
```

D Multi-particle phase space integration: `rambo`

The `rambo` module implements the RAMBO algorithm, developed by Kleiss, Stirling and Ellis [24], for generating phase-space points using a Monte Carlo procedure. We advise the interested reader to consult the original paper for details on how this algorithm works. We will only described how the algorithm works at a high-level. The RAMBO algorithm works by first democratically generations random, independent, isotropic, massless four-momenta, q_i^μ, with i running from 1 to N, N being the number of final state particles. The momenta q_i^μ are generated with energies distributed according to $q_i^0 \exp(-q_i^0)dq_i^0$. These four momenta are then transformed into four-momenta p_i^μ by performing a particular Lorentz boost such that $\sum_i p_i^\mu = P^\mu$ where P^μ is the total four-momenta of the given process in the center-of-mass frame (i.e. $P^0 = E_{\rm CM}$ and $\vec{P} = \vec{0}$.) At this stage, a corresponding weight $W_0(p_i)$ is computed to give the probability of the event with the particular four-momenta p_i^μ. Lastly, the four-momenta p_i^μ are transformed into four-momenta k_i^μ such that the k_i^μ have the correct masses (i.e. $k_i^\mu k_{i,\mu} = m_i^2$) and a new weight $W(k_i)$ is computed. This completes the algorithm. Below we give pseudo-code for the algorithm:

1. Generate the q_i^μ's with energies distributed according to $q_i^0 \exp(-q_i^0)dq_i^0$ using:

$$q_i^0 = -\log(\rho_i^{(3)}\rho_i^{(4)}), \quad q_i^x = q_i^0\sqrt{1-c_i^2}\cos\phi_i, \quad q_i^y = q_i^0\sqrt{1-c_i^2}\sin\phi_i, \quad q_i^z = q_i^0 c_i. \tag{D.1}$$

   where $c_i = 2\rho_i^{(1)} - 1$, $\phi_i = 2\pi\rho_i^{(2)}$ and $\rho_i^{(1,2,3,4)}$ are uniform random numbers on $(0,1)$. See [24] for a proof that these are indeed distributed according to $q_i^0 \exp(-q_i^0)dq_i^0$.

2. Transform the q_i^μ's into p_i^μ's such that the $\sum_i p_i^\mu = P^\mu$ with $P^0 = E_{\rm CM}$ and $\vec{P} = \vec{0}$. This is done using:

$$p_i^0 = x(\gamma q_i^0 + \vec{b}\cdot\vec{q}_i), \qquad \vec{p}_i = x(\vec{q}_i + \vec{b}q_i^0 + a(\vec{b}\cdot\vec{q}_i)\vec{b}). \tag{D.2}$$

   where

$$\begin{aligned} &\vec{b} = -\vec{Q}/M, && x = E_{\rm CM}/M, && \gamma = Q^0/M, && a = 1/\sqrt{1+\gamma}, \\ &Q^\mu = \sum_i q_i^\mu, && M = \sqrt{Q_\mu Q^\mu} \end{aligned} \tag{D.3}$$

   Assign the following weight to the event:

$$W_0 = \left(\frac{\pi}{2}\right)^{N-1} E_{\rm CM}^{2N-4}\frac{(2\pi)^{4-3N}}{\Gamma(N)\Gamma(N-1)} \tag{D.4}$$

3. Next we transform the p_i^μ's to k_i^μ's such that $k_i^\mu k_{i,\mu} = m_i$. This is done via:

$$\vec{k}_i = \xi\vec{p}_i, \qquad k_i^0 = \sqrt{m_i + (\xi p_i^0)^2} \tag{D.5}$$

   where ξ is the solution to

$$E_{\rm CM} = \sum_{i=1}^{N}\sqrt{m_i^2 + (\xi p_i^0)^2} \tag{D.6}$$

   This equation can be solved efficiently using Newton's method with the initial guess $\xi_0 = \sqrt{1 - \left(\sum_{i=1}^N m_i/E_{\rm CM}\right)^2}$. Lastly, we re-weight the probability of the event using:

$$W = W_0 E_{\rm CM}\left(\sum_{i=1}^{N}\frac{|\vec{k}_i|}{E_{\rm CM}}\right)^{2N-3}\left(\sum_{i=1}^{N}\frac{|\vec{k}_i|^2}{k_i^0}\right)^{-1}\left(\prod_{i=1}^{N}\frac{|\vec{k}_i|}{k_i^0}\right) \tag{D.7}$$

   We note that this fixes a typo from the original paper, where the overall factor of $E_{\rm CM}$ was missing. Additionally, we note that the weights clearly don't take into account the matrix element of the particle physics process. The matrix elements are included by simply multiplying the weights by the corresponding matrix element.

This completes the pseudo-code for the RAMBO algorithm.

In **hazma**, the **rambo** module can be used to perform various tasks. At the highest level, the function `compute_annihilation_cross_section` (`compute_decay_width`) is provided for computing cross-sections (decay widths) for $2 \to N$ ($1 \to N$) processes. For example, consider computing the partial decay width of $\mu^- \to e^-\overline{\nu}_e\nu_\mu$. One first declares a function that takes a list of the final state particles' four-momenta and muon's four-momentum, and returns the matrix element squared for the reaction:

```
# Import the fermi constant
>>> from hazma.parameters import GF
# Import a helper function for scalar products of four-vectors
>>> from hazma.field_theory_helper_functions.common_functions import \
...     minkowski_dot as MDot
# Declare the matrix element
>>> def msqrd_mu_to_enunu(momenta):
...     pe = momenta[0] # electron four-momentum
...     pve = momenta[1] # electron-neutrino four-momentum
...     pvmu = momenta[2] # muon-neutrino four-momentum
...     pmu = sum(momenta) # muon four-momentum
        # squared matrix element
...     return 64. * GF**2 * MDot(pe, pvmu) * MDot(pmu, pve)
```

Then, the partial decay width can be computed using:

```
# Import function to compute decay width
>>> from hazma.rambo import compute_decay_width
# import masses of muon and electron
>>> from hazma.parameters import muon_mass as mmu
>>> from hazma.parameters import electron_mass as me
# Specify the masses of the electron and neutrinos
>>> fsp_masses = np.array([me, 0., 0.])
# compute the partial width
>>> partial_width = compute_decay_width(fsp_masses, mmu, num_ps_pts=50000,
...                                      mat_elem_sqrd=msqrd_mu_to_enunu)
```

Using 50000 phase-space points gives a result within 5% of the analytical value.

In addition, `rambo` includes a function for performing integrations over all variables except the energy of one of the final-state particles called `generate_energy_histogram`. This is useful for computing energy spectra for FSR from final states with large numbers of particles. The following demonstrates how it can be used:

```
>>> from hazma.rambo import generate_energy_histogram
>>> import numpy as np
>>> num_ps_pts = 100000 #number of phase-space points to use
# masses of final-state particles
>>> masses = np.array([100., 100., 0.0, 0.0])
>>> cme = 1000. # center-of-mass energy
>>> num_bins = 100 # number of energy bins to use
# computing energy histograms
>>> eng_hist = generate_energy_histogram(num_ps_pts, masses, cme,
...                                       num_bins=num_bins)
# plot the results
>>> import matplotlib as plt
>>> for i in range(len(masses)):
...     # pts[i, 0] are the energies of particle i
...     # pts[i, 1] are the probabilities
...     plt.loglog(pts[i, 0], pts[i, 1])
```

At the lowest level, `rambo` includes a function for generating phase-space points called `generate_phase_space`. We also include a function `integrate_over_phase_space` which will perform the integral

$$\int \left(\prod_{i=1}^{N} \frac{d^3\vec{p}_i}{(2\pi)^3} \frac{1}{2E_i} \right) (2\pi)^4 \delta^4 \left(P - \sum_{i=1}^{N} p_i \right) |\mathcal{M}|^2 \tag{D.8}$$

where P^μ is the total four-momentum, p_i^μ are the individual four-momenta for each of the N final-state particles and $\mathcal{M}$ is the matrix element.

E Gamma-ray Spectra from many final state particles: `gamma_ray`

Since computing gamma-ray spectra is a model-dependent process, we include in `hazma` tools for computing gamma-ray spectra from both FSR and the decay of final state-particles. The `gamma_ray` module contains two functions called `gamma_ray_decay` and `gamma_ray_fsr`. The `gamma_ray_decay` function accepts a list of the final-state particles, the center-of-mass energy, the gamma-ray energies to compute the spectrum at and optionally the matrix element squared. Currently, the final-state particles can be $\pi^0, \pi^\pm, \mu^\pm, e^\pm, K^\pm, K_L$ and K_S where K stands for kaon. We caution that when including many final-state mesons, one needs to take care to supply a matrix element squared that is valid at the reaction's center-of-mass energy (see Sec. (1) for a discussion on the validity of ChPT for large energies).

The `gamma_ray_decay` function works by first computing the energies distributions of all the final-state particles and convolving these with the decay spectra of the final-state particles. It can be used as follows:

```
>>> from hazma.gamma_ray_decay import gamma_ray_decay
>>> import numpy as np
# specify the final-state particles
>>> particles = np.array(['muon', 'charged_kaon', 'long_kaon'])
# choose the center of mass energy
>>> e_cm = 5000.
# choose list of the gamma-ray energies to compute spectra at
>>> e_gams = np.logspace(0., np.log10(e_cm), num=200, dtype=np.float64)
# compute the gamma-ray spectra assuming a constant matrix element
>>> spec = gamma_ray_decay(particles, e_cm, e_gams)
```

The `gamma_ray_fsr` function computes the gamma-ray spectrum from $X \to Y\gamma$, i.e.:

$$\frac{dN(X \to Y\gamma)}{dE_\gamma} = \frac{1}{\sigma(X \to Y)} \frac{d\sigma(X \to Y\gamma)}{dE_\gamma} \tag{E.1}$$

where X and Y are any particles excluding the photon. As input this function takes a list of the initial state particle masses (either 1 or 2 particles), the final state particle masses, the center-of-mass energy, a function for the tree-level matrix element squared (for $X \to Y$) and a function for the radiative matrix element squared ($X \to Y\gamma$). The functions for the squared matrix elements must take is a single argument which is a list of the four-momenta for the final state particles. As an example, we consider the process of two dark-matter particles annihilating into charged pions, $\chi\bar{\chi} \to \pi^+\pi^-(\gamma)$ using the model from App. (C.2). In App (C.2), we gave the analytic expressions for the gamma-ray spectra. The tree-level and radiative matrix elements squared for this process are:

$$|\mathcal{M}(\chi\bar{\chi} \to \pi^+\pi^-)|^2 = \frac{c_1^2\left(s - 4m_\chi^2\right)}{2\Lambda^2} \tag{E.2}$$

$$\begin{aligned}|\mathcal{M}(\chi\bar{\chi} \to \pi^+\pi^-\gamma)|^2 &= \frac{2c_1^2\left(4\mu_\chi^2 - 1\right)Q^2e^2}{\Lambda^2\left(t - \mu_\pi^2Q^2\right)^2\left(u - \mu_\pi^2Q^2\right)^2} \\ &\quad\times\left(\left(\mu_\pi Q(t+u) - 2\mu_\pi^3Q^3\right)^2 + s\left(t - \mu_\pi^2Q^2\right)\left(\mu_\pi^2Q^2 - u\right)\right)\end{aligned} \tag{E.3}$$

where Q is the center-of-mass energy, e is the electromagnetic coupling, $\mu_{\pi,\chi} = m_{\pi,\chi}/Q$ and

$$s = (p_{\pi,1} + p_{\pi,2})^2, \qquad t = (p_{\pi,1} + k)^2, \qquad u = (p_{\pi,2} + k)^2 \tag{E.4}$$

with $p_{\pi,1,2}$ are the four-momenta of the two final-state pions and k is the four-momenta of the final-state photon. Below, we create a class to implement functions for the tree and radiative squared matrix elements. Note that these functions take in an array of four-momenta.

```python
from hazma.field_theory_helper_functions.common_functions import \
    minkowski_dot as MDot

class Msqrd(object):
    def __init__(self, mx, c1, lam):
        self.mx = mx # DM mass
        self.c1 = c1 # effective coupling of DM to charged pions
        self.lam = lam # cut off scale for effective theory

    def tree(self, momenta):
        ppi1 = momenta[0] # first charged pion four-momentum
        ppi2 = momenta[1] # second charged pion four-momentum
        Q = ppi1[0] + ppi2[0] # center-of-mass energy
        return -((self.c1**2 * (4 * self.mx**2-Q**2)) / (2 * self.lam**2))

    def radiative(self, momenta):
        ppi1 = momenta[0] # first charged pion four-momentum
        ppi2 = momenta[1] # second charged pion four-momentum
        k = momenta[2] # photon four-momentum
        Q = ppi1[0] + ppi2[0] + k[0] # center-of-mass energy

        mux = self.mx / Q
        mupi = mpi / Q

        s = MDot(ppi1 + ppi2, ppi1 + ppi2)
        t = MDot(ppi1 + k, ppi1 + k)
        u = MDot(ppi2 + k, ppi2 + k)

        return ((2*self.c1**2*(-1 + 4*mux**2)*Q**2*qe**2 *
                  (s*(-(mupi**2*Q**2) + t)*(mupi**2*Q**2 - u) +
                   (-2*mupi**3*Q**3 + mupi*Q*(t + u))**2)) /
```

```
                    (self.lam**2*(-(mupi**2*Q**2) + t)**2*
                    (-(mupi**2*Q**2) + u)**2))
```

Next, we can compute the gamma-ray spectrum for $\chi\bar{\chi} \to \pi^+\pi^-\gamma$ using:

```
>>> from hazma.gamma_ray import gamma_ray_fsr
# specify the parameters of the model
>>> params = {'mx': 200.0, 'c1':1.0, 'lam':1e4}
# create instance of our Msqrd class
>>> msqrds = Msqrd(**params)
# specify the initial and final state masses
>>> isp_masses = np.array([msqrds.mx, msqrds.mx])
>>> fsp_masses = np.array([mpi, mpi, 0.0])
# choose the center-of-mass energy
>>> e_cm = 4.0 * msqrds.mx
# compute the gamma-ray spectrum
>>> spec = gamma_ray_fsr(isp_masses, fsp_masses, e_cm, msqrds.tree,
                         msqrds.radiative, num_ps_pts=500000, num_bins=50)
# plot the spectrum
>>> import matplotlib.pyplot as plt
>>> plt.figure(dpi=100)
>>> plt.plot(spec[0], spec[1])
>>> plt.yscale('log')
>>> plt.xscale('log')
>>> plt.ylabel(r'$dN/dE_{\gamma} \ (\mathrm{MeV}^{-1})$', fontsize=16)
>>> plt.xlabel(r'$E_{\gamma} \ (\mathrm{MeV})$', fontsize=16)
```

References

[1] B. W. Lee and S. Weinberg, Phys. Rev. Lett. **39**, 165 (1977), [,183(1977)].

[2] S. Profumo, Phys. Rev. **D78**, 023507 (2008), 0806.2150.

[3] W. B. Atwood et al. (Fermi-LAT), Astrophys. J. **697**, 1071 (2009), 0902.1089.

[4] J. Aleksić et al. (MAGIC), Astropart. Phys. **72**, 76 (2016), 1409.5594.

[5] A. Abramowski et al. (H.E.S.S.), Phys. Rev. **D90**, 112012 (2014), 1410.2589.

[6] M. Aguilar et al. (AMS), Phys. Rev. Lett. **117**, 231102 (2016).

[7] P. Gondolo, J. Edsjo, P. Ullio, L. Bergstrom, M. Schelke, and E. A. Baltz, JCAP **0407**, 008 (2004), astro-ph/0406204.

[8] G. Belanger, F. Boudjema, A. Pukhov, and A. Semenov, Comput. Phys. Commun. **185**, 960 (2014), 1305.0237.

[9] M. Cirelli, G. Corcella, A. Hektor, G. Hutsi, M. Kadastik, P. Panci, M. Raidal, F. Sala, and A. Strumia, JCAP **1103**, 051 (2011), [Erratum: JCAP1210,E01(2012)], 1012.4515.

[10] T. Sjostrand, S. Mrenna, and P. Z. Skands, Comput. Phys. Commun. **178**, 852 (2008), 0710.3820.

[11] R. Rando, A. De Angelis, and M. Mallamaci (thee-ASTROGAM), J. Phys. Conf. Ser. **1181**, 012044 (2019).

[12] M. Tavani et al. (e-ASTROGAM), JHEAp **19**, 1 (2018), 1711.01265.

[13] S. D. Hunter, P. F. Bloser, M. P. Dion, M. L. McConnell, G. A. de Nolfo, S. Son, J. M. Ryan, and F. W. Stecker, in *Space Telescopes and Instrumentation 2010: Ultraviolet to Gamma Ray* (2010), vol. 7732 of *Proceedings of the SPIE*, p. 773221.

[14] X. Wu, PoS **ICRC2015**, 964 (2016).

[15] N. Duncan et al., Proc. SPIE Int. Soc. Opt. Eng. **9905**, 99052Q (2016), 1609.08558.

[16] P. U. W. Contributors, *Hazma* (2019), URL https://pokemon-uranium.fandom.com/wiki/Hazma.

[17] S. Scherer, *Introduction to Chiral Perturbation Theory* (Springer US, Boston, MA, 2003), pp. 277–538, ISBN 978-0-306-47916-8, hep-ph/0210398.

[18] Coogan, Adam and Morrison, Logan and Profumo, Stefano (to appear).

[19] F. D'Eramo and S. Profumo, Phys. Rev. Lett. **121**, 071101 (2018), 1806.04745.

[20] T. Oliphant, *NumPy: A guide to NumPy*, USA: Trelgol Publishing (2006–), online; accessed 2019-07-23, URL http://www.numpy.org/.

[21] T. Hahn, Comput. Phys. Commun. **140**, 418 (2001), hep-ph/0012260.

[22] V. Shtabovenko, R. Mertig, and F. Orellana, Comput. Phys. Commun. **207**, 432 (2016), 1601.01167.

[23] R. Mertig, M. Böhm, and A. Denner, Computer Physics Communications **64**, 345 (1991), ISSN 0010-4655, URL http://www.sciencedirect.com/science/article/pii/001046559190130D.

[24] R. Kleiss, W. J. Stirling, and S. D. Ellis, Comput. Phys. Commun. **40**, 359 (1986).

[25] S. Weinberg, Physica A: Statistical Mechanics and its Applications **96**, 327 (1979), ISSN 0378-4371, URL http://www.sciencedirect.com/science/article/pii/0378437179902231.

[26] J. Gasser and H. Leutwyler, Annals Phys. **158**, 142 (1984).

[27] J. Gasser and H. Leutwyler, Nuclear Physics B **250**, 465 (1985), ISSN 0550-3213, URL http://www.sciencedirect.com/science/article/pii/0550321385904924.

[28] G. Ecker, Progress in Particle and Nuclear Physics **35**, 1 (1995).

[29] A. Pich, Rept. Prog. Phys. **58**, 563 (1995), hep-ph/9502366.

[30] U. G. Meissner, Rept. Prog. Phys. **56**, 903 (1993), hep-ph/9302247.

[31] J. R. Pelaez, Phys. Rept. **658**, 1 (2016), 1510.00653.

[32] T. N. Truong, Phys. Rev. Lett. **61**, 2526 (1988), URL https://link.aps.org/doi/10.1103/PhysRevLett.61.2526.

[33] G. Ecker, J. Gasser, A. Pich, and E. D. Rafael, Nuclear Physics B **321**, 311 (1989), ISSN 0550-3213, URL http://www.sciencedirect.com/science/article/pii/0550321389903465.

[34] J. Soto, P. Talavera, and J. Tarrus, Nucl. Phys. **B866**, 270 (2013), 1110.6156.

[35] W. J. Marciano, C. Zhang, and S. Willenbrock, Phys. Rev. **D85**, 013002 (2012), 1109.5304.

[36] E. Witten, Nucl. Phys. **B223**, 422 (1983).

[37] J. Wess and B. Zumino, Phys. Lett. **37B**, 95 (1971).

[38] F. E. Low, Phys. Rev. **110** (1958).

[39] T. H. Burnett and N. M. Kroll, Phys. Rev. Lett. **20** (1968).

[40] R. Bartels, D. Gaggero, and C. Weniger, Journal of Cosmology and Astroparticle Physics **2017**, 001 (2017), 1703.02546[astro-ph.HE].

[41] J. F. Beacom, N. F. Bell, and G. Bertone, Phys. Rev. Lett. **94**, 171301 (2005), astro-ph/0409403, URL https://link.aps.org/doi/10.1103/PhysRevLett.94.171301.

[42] J. Mardon, Y. Nomura, D. Stolarski, and J. Thaler, Journal of Cosmology and Astroparticle Physics **2009**, 016 (2009), 0901.2926[hep-ph], URL http://stacks.iop.org/1475-7516/2009/i=05/a=016.

[43] T. Bringmann, L. Bergstrom, and J. Edsjo, Journal of High Energy Physics **2008**, 049 (2008), 0710.3169[hep-ph], URL http://stacks.iop.org/1126-6708/2008/i=01/a=049.

[44] R. Essig, N. Sehgal, and L. E. Strigari, Phys. Rev. D **80**, 023506 (2009), 0902.4750[hep-ph].

[45] T. Bringmann, J. Edsjo, P. Gondolo, P. Ullio, and L. Bergstrom (2018), 1802.03399[hep-ph].

[46] M. D. Schwartz, *Quantum field theory and the standard model* (Cambridge University Press, 2017).

[47] Y. Kuno and Y. Okada, Rev. Mod. Phys. **73**, 151 (2001), URL https://link.aps.org/doi/10.1103/RevModPhys.73.151.

[48] D. Bryman, P. Depommier, and C. Leroy, Physics Reports **88**, 151 (1982).

[49] S. C. Kappadath, Ph.D. thesis, University of New Hampshire (1993).

[50] L. Michel, Proc. Phys. Soc. **A63**, 514 (1950), [,45(1949)].

[51] T. Bringmann, M. Doro, and M. Fornasa, Journal of Cosmology and Astroparticle Physics **2009**, 016 (2009), 0809.2269[astro-ph].

[52] R. Essig, E. Kuflik, S. D. McDermott, T. Volansky, and K. M. Zurek, JHEP **11**, 193 (2013), 1309.4091.

[53] K. K. Boddy and J. Kumar, Phys. Rev. D **92**, 023533 (2015), 1504.04024[astro-ph.CO].

[54] D. J. Thompson et al., Astrophys. J. Suppl. **86**, 629 (1993).

[55] W. B. Atwood et al., The Astrophysical Journal **697**, 1071 (2009), URL http://stacks.iop.org/0004-637X/697/i=2/a=1071.

[56] A. D. Angelis et al., Tech. Rep. (2018), astro-ph/0406254.

[57] A. W. Strong, I. V. Moskalenko, and O. Reimer, Astrophys. J. **613**, 962 (2004), astro-ph/0406254.

[58] M. Ackermann, M. Ajello, W. B. Atwood, L. Baldini, J. Ballet, G. Barbiellini, D. Bastieri, K. Bechtol, R. Bellazzini, B. Berenji, et al., The Astrophysical Journal **750**, 3 (2012), URL https://doi.org/10.1088%2F0004-637x%2F750%2F1%2F3.

[59] R. Essig, E. Kuflik, S. D. McDermott, T. Volansky, and K. M. Zurek, Journal of High Energy Physics **2013**, 193 (2013), ISSN 1029-8479, hep-ph/1309.4091, URL https://doi.org/10.1007/JHEP11(2013)193.

[60] S. Colafrancesco, S. Profumo, and P. Ullio, Astron. Astrophys. **455**, 21 (2006), astro-ph/0507575.

[61] S. Colafrancesco, S. Profumo, and P. Ullio, Phys. Rev. **D75**, 023513 (2007), astro-ph/0607073.

[62] X.-L. Chen and M. Kamionkowski, Phys. Rev. **D70**, 043502 (2004), astro-ph/0310473.

[63] N. Padmanabhan and D. P. Finkbeiner, Phys. Rev. **D72**, 023508 (2005), astro-ph/0503486.

[64] S. Galli, F. Iocco, G. Bertone, and A. Melchiorri, Phys. Rev. **D80**, 023505 (2009), 0905.0003.

[65] T. R. Slatyer, N. Padmanabhan, and D. P. Finkbeiner, Phys. Rev. **D80**, 043526 (2009), 0906.1197.

[66] T. R. Slatyer, Phys. Rev. D **93**, 023527 (2016), 1506.03811[hep-ph], URL https://link.aps.org/doi/10.1103/PhysRevD.93.023527.

[67] N. Aghanim et al. (Planck) (2018), 1807.06209.