Outline

DalitzPlot

This Julia package is designed for high-energy physics applications, originally in visualizing and analyzing particle decays. It consists of the following subpackages:

- `Xs` : Provides tools for calculation cross section and creating Dalitz plots, which are essential for visualizing three-body - decays of particles. Users can specify amplitudes to generate these plots.
- `GEN` : Stands for General Event Generation. This subpackage is used for generating events, allowing users to simulate particle interactions and decays.
- `QFT` : Short for Quantum Field Theory. This subpackage includes functions for calculating spinor or polarized vectors with momentum and spin, gamma matrices, and other related calculations.
- `qBSE` : Refers to the Quasipotential Bethe-Salpeter Equation. This subpackage provides tools for solving the Bethe-Salpeter equation, which is used in the study of bound states in quantum field theory.

Note: This package is designed for phenomenological studies on the theoretical side. For experimental data analysis, please refer to other tools.

Installation

To install the "DalitzPlot" package, you can follow the standard Julia package manager procedure. Open Julia and use the following commands:

Using the Julia REPL

```
Pkg.add("DalitzPlot")
```

Alternatively, if you want to install it directly from the GitHub repository:

Using the Julia REPL:

```
import Pkg
Pkg.add(url="https://github.com/junhe1979/DalitzPlot.jl")
```

These commands will install the "DalitzPlot" package and allow you to use it in your Julia environment.

Usage

After installation, the package can be used as:

```
using DalitzPlot
```

To use the subpackages:

```
using DalitzPlot.Xs
using DalitzPlot.GEV
using DalitzPlot.QFT
using DalitzPlot.qBSE
```

Xs Package: for cross section and Dalitz plot

The cross section, denoted by $d\sigma$, can be expressed in terms of amplitudes, $\mathcal{M}$, as follows:

$$d\sigma = F|\mathcal{M}|^2\frac{1}{S}d\Phi = (2\pi)^{4-3n}F|\mathcal{M}|^2\frac{1}{S}dR$$

Here, $dR = (2\pi)^{3n-4}d\Phi = \prod_i \frac{d^3k_i}{2E_i}\delta^4(\sum_i k_i - P)$ represents the Lorentz-invariant phase space for n particles, and it is generated using the Monte-Carlo method described in Ref. [F. James, CERN 68-12].

The flux factor F for the cross section is given by: $F = \frac{1}{2E2E'v_{12}} = \frac{1}{4[(p_1\cdot p_2)^2-m_1^2m_2^2]^{1/2}}\frac{|p_1\cdot p_2|}{p_1^0p_2^0}$.
In the laboratory or center of mass frame, the relation $\vec{p}_1^2\vec{p}_2^2 = (\vec{p}_1\cdot\vec{p}_2)^2$ is utilized. In the laboratory frame, the term $\frac{|p_1\cdot p_2|}{p_1^0p_2^0}$ simplifies to 1.
Additionally, if a boson or zero-mass spinor particle is replaced with a non-zero mass spinor particle, the factor $1/2$ is replaced with the mass of the particle, m.
The total symmetry factor S is given by $\prod_i m_i!$ if there are m_i identical particles.

For decay width, the flux factor is modified to $F = \frac{1}{2E}$.

Define amplitudes with factors for the calculation

Users are required to supply amplitudes with factors $(2\pi)^{4-3n}F|\mathcal{M}|^2\frac{1}{S}$ within the function, named `amp` here.

We can take it as 1.

```
amp(tecm, kf, ch, para)=1.
```

Define more intricate amplitudes for a 2->3 process.

This function, named `amp`, calculates amplitudes with factors for a 2->3 process. The input parameters are:

- `tecm`: Total energy in the center-of-mass frame.

- `kf` : Final momenta generated.
- `ch` : Information about the process (to be defined below).
- `para` : Additional parameters.

Users are expected to customize the amplitudes within this function according to their specific requirements.

```
function amp(tecm, kf, ch, para)
    # get kf as momenta in the center−of−mass ,
    #k1,k2,k3=getkf(kf)
    #get kf as momenta in laboratory frame
    k1, k2, k3 = getkf(para.p, kf, ch)

    # Incoming particle momentum
    # Center−of−mass frame: p1 = [p 0.0 0.0 E1]
    #p1, p2 = pcm(tecm, ch.mi)
    # Laboratory frame
    p1, p2 = plab(para.p, ch.mi)

    #flux
    #flux factor for cross section in Laboratory frame
    fac = 1 / (4 * para.p * ch.mi[2] * (2 * pi)^5)

    k12 = k1 + k2
    s12 = cdot(k12, k12)
    m = 3.2
    A = 1e9 / (s12 − m^2 + im * m * 0.1)

    total = abs2(A) * fac* 0.389379e−3

    return total
```

Define the masses of initial and final particles

The mass of initial and final particles is specified in a NamedTuple (named `ch` here) with fields `mi` and `mf`.
Particle names can also be provided for PlotD as `namei` and `namef`.

The function for amplitudes with factors is saved as `amp`.

Example usage:

```
ch = (mi=[mass_i_1, mass_i_2], mf=[mass_f_1, mass_f_2, mass_f_3], namei=["p^i_{1}", "p^i_
```

Make sure to replace `mass_i_1`, `mass_i_2`, `mass_f_1`, `mass_f_2`, and `mass_f_3` with the actual masses of the particles (1.0, 1.0, 1.0, 2.0, 3.0 here).

Define the momentum or total energy

Momentum in the Laboratory frame and transfer it to the total energy in the center-of-mass frame.

Example usage:

```
p_lab = 20.0
tecm = pcm(p_lab, ch.mi)
```

Calculate

Calculate the cross section and related spectra using the GENEV function.

The function `Xsection` takes the momentum of the incoming particle in the Laboratory frame (`p_lab`), the information about the particles (`ch`), the axes representing invariant masses (`axes`), the total number of events (`nevtot`), the number of bins (`Nbin`), and additional parameters (`para`). The function uses the plab2pcm function to transform the momentum from the Laboratory frame to the center-of-mass frame.

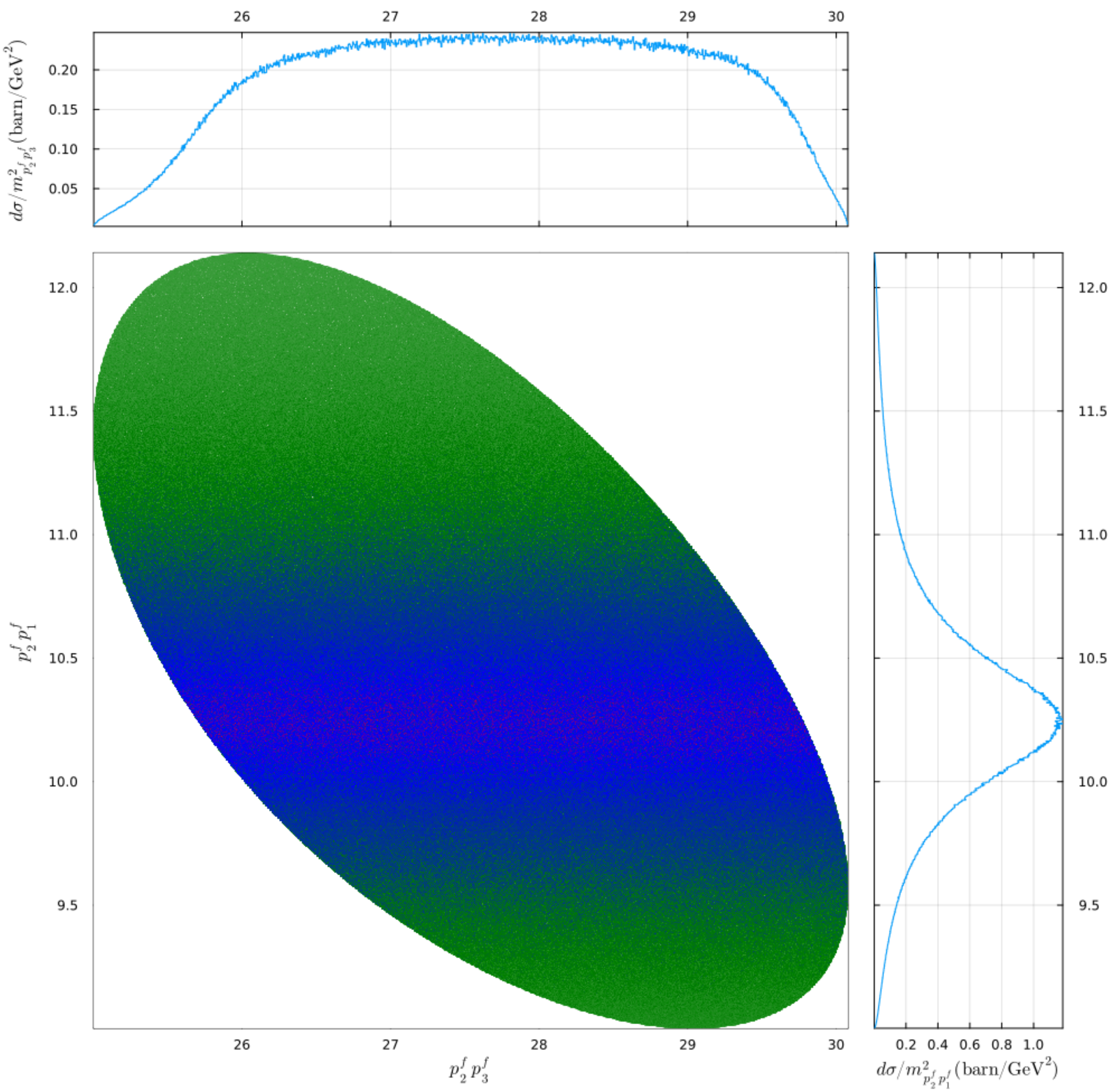
Example usage:

```
res = Xsection(plab2pcm(p_lab, ch.mi), ch, axes=[23, 21], nevtot=Int64(1e7), Nbin=500, pa
```

The results are stored in the variable `res` as a `NamedTuple`. Specifically, `res.cs0` corresponds to the total cross section, `res.cs1` represents the invariant mass spectrum, and `res.cs2` captures the data for the Dalitz plot.

Plot Dalitz Plot

```
plotD(res)
```



GEN Package: for Generating Events

The GEN package is used for generating events for cross-section calculations and Dalitz plots. The Lorentz-invariant phase space used here is defined as:

$$dR = (2\pi)^{3n-4} d\Phi = \prod_i \frac{d^3 k_i}{2E_i} \delta^4(\sum_i k_i - P)$$

for n particles. The events are generated using the Monte-Carlo method described in Ref. [F. James, CERN 68-12].

The primary function provided by this package is `GENEV`, which can be used as follows:

```
PCM, WT=GENEV(tecm,EM)
```

- Input: total momentum in center of mass frame `tecm`, and the mass of particles `EM`.
  - `tecm`: a `Float64` value representing the total momentum in the center of mass frame.
  - `EM`: a `Vector{Float64}` containing the masses of the particles.
- Output: the momenta of the particles `PCM`, and a weight `WT`.
  - `PCM`: a StaticArrays `@MArray zeros(Float64, 5, 18)` storing the momenta of the particles. Note that at most 18 particles can be considered.
  - `WT`: `Float64` value representing the weight for this event.

QFT Package: for Numerical Calculation of Feynman Rules