Fixed Target Generator
(Embedded in `gen_amp_v2`)

K. Saldaña
M. R. Shepherd
J. Sikes

December 5, 2024

Introduction

This software was written to correctly generate phase space in the case that a beam in the $\hat{z}$ direction collides with a fixed target and produces n final state particles. The final state of such a reaction should populate n-body phase space. At beam energies relevant for GlueX it is typical to consider a t-channel exchange picture where particles are produced at either the "upper (beam) vertex" or the "lower (target) vertex". If we consider u particles produced at the upper vertex with a total invariant mass of M_u and ℓ particles coming from the lower vertex with a mass of M_ℓ, we want a generator that allows us to preferentially populate regions of M_ℓ, M_u, and momentum transfer t while providing weights that allow the user to recover proper n-body ($n = \ell + u$) phase space.

The algorithm is implemented in a class called `FixedTargetGenerator` that is part of the `halld_sim` repository. This class is then used in the context of an executable called `gen_amp_v2`, which is written to function in a similar way as the familiar `gen_amp` used for GlueX MC. If you are looking for quick guidance on how to run `gen_amp_v2`, see the next section.

Quick Start to `gen_amp_v2`

`gen_amp_v2` will use the `FixedTargetGenerator` class to generate n-body phase space and enable the use of `AmpTools` to generate pseudodata with arbitrary physics content. For the latter reason, it is beneficial to build the structure of `gen_amp_v2` on the `AmpTools` config file to define the desired reaction, the structure of the config file is explained in the AmpTools User Guide. The generator takes in several required and optional arguments that specify the kinematics and output format of the reaction. The reaction to generate is specified by the `reaction` declaration in the `AmpTools` config file. For the purposes of generation, it is required that only one reaction is specified. The generated particles will be written to the output file in the same order as they are specified in the `reaction` command.

The default behavior of `gen_amp_v2` is to generate events with `FixedTargetGenerator` such that the weighted distribution respects phase space even if Breit-Wigner functions or t-dependence is introduced through the options. By default an accept/reject algorithm is performed on the product of the weight from `FixedTargetGenerator` and the intensity calculated by the `AmpTools` config file. Events are written out with unit weight. By matching the functions in `FixedTargetGenerator` roughly to the `AmpTools` intensity shapes then generation efficiency can be enhanced.

It may be desirable to use `FixedTargetGenerator` to build in a t-dependence that does not necessarily appear in the `AmpTools` intensity specification. In that case the t reweighting bit (the least significant bit of the reweight mask) should be turned off using `-mask 1 1 0`. Then this t distribution will persist through the accept/reject stage. It is this mode of operation that most similar to the behavior of the older version of `gen_amp` where a default t slope was built into the generator.

In the case that the `-f` option is provided. The accept/reject algorithm will be skipped and the events written will be those provided by `FixedTargetGenerator` with weights as specified by `FixedTargetGenerator` according to the setting of the reweighting mask. By default the weights will recover phase space shapes. In the case that no Breit-Wigner distributions or t-dependence is specified then the sample is just n-body phase space with unit weights.

Required Arguments

- **-ac** <file>
  Specify the AmpTools config to be used.
- **-o** <file>
  Name of the output ROOT file.
- **-uv** <string>
  Indices of all upper vertex particles defined in the config file.
- **-lv** <string>
  Indices of all lower vertex particles defined in the config file.

The addition of inputting the upper/lower vertex indices allows the freedom to place the particles defined in the config file in any order you desire. The indices will be used to fill the upper/lower mass vectors to be used for generating the reaction PS.

Optional Arguments

- **-bc** <file>
  Specify the beam config file to be used. A local beam config file will be generated if this argument is not called upon.
- **-fsroot**
  This will output the ROOT file in the FSROOT format.
- **-hd** <name>
  This will save an HDDM file after generation with name <name>.
- **-f**
  This option skips the `AmpTools` accept/reject algorithm and returns the output from `FixedTargetGenerator`
- **-d**
  Outputs only diagnostic plots ROOT file.
- **-n** <number>
  Number of events to generate (Default is 10,000)
- **-r** <run number>
  Run number assigned to generated events.
- **-s** <number>
  Seed used for random number initialization.
- **-lvRange** <min> <max>
  Set the allowed mass range (GeV) in the lower vertex. Need to specify the minimum and maximum.
- **-uvRange** <min> <max>
  Set the allowed mass range (GeV) in the upper vertex. Need to specify the minimum and maximum.
- **-tRange** <min> <max>
  Set the allowed momentum transfer range $(\text{GeV})^2$. Need to specify the minimum and maximum.
- **-t** <tSlope> <fraction>
  Calls `addMomentumTransfer` to define the momentum transfer slope (t). Need to specify t $(\text{GeV})^2$ and scale.

- **-uvBW** <mass> <width> <fraction>
  Calls `addUpperVtxBW` and enables concentration of events around upper vertex Breit-Wigner (BW). Need to specify the BW mass (GeV), width (GeV), and scale in order for the program to run.

- **-lvBW** <mass> <width> <fraction>
  Calls `addLowerVtxBW` and enables concentration of events around lower vertex Breit-Wigner (BW). Need to specify the BW mass (GeV), width (GeV), and probability or fraction of events. This option can be repeated multiple times.

- **-mask** <upper vertex mass> <lower vertex mass> <t distribution>
  Changes mask value in `FixedTargetGenerator::setReweightMask`, giving option to remove the phase-space weight of upper vertex mass, lower vertex mass, and momentum transfer t respectively. Each of the three numbers should be either 0 or 1. (Internally they are combined to a 3-bit mask.) Setting 1 includes weights in the to recover phase space in the specified distribution. Default: 1 1 1 (`setReweightMask(7)`) This function only matters when options `-t`, `-lvBW`, or `-uvBW` are used.

The next optional arguments are used to change the local beam config file input parameters.

- **-m** <energy>
  Set energy of electron beam (GeV).

- **-p** <peak>
  Set the coherent peak value of the photon beam (GeV).

- **-a** <min E beam>
  Set the minimum photon beam energy (GeV).

- **-b** <max E beam>
  Set the maximum photon beam energy.

`FixedTargetGenerator`: Algorithm Description

The remainder of this document describes the algorithms used in the implementation of the `FixedTargetGenerator`. At the end of the section, a variety of use cases are discussed.

Generating Masses

A key feature of many multibody phase space generators is that they work recursively on the idea that n-body phase space can be generated by embedding an $n-1$ body phase space generation inside of the n-body problem. This algorithm breaks down in the case that one wants to control the mass distribution of a subset of ℓ or u particles inside of the larger n-body problem. This issue and how to address it is discussed in Ref. [1]. For the general problem where 2 or more particles from both the upper and lower vertices we can write the n-body phase space as

$$\frac{d^{n-2}R_n}{dM_\ell dM_u (dM_{\ell;2}\ldots dM_{\ell;\ell-1})(dM_{u;2}\ldots dM_{u;u-1})} \propto R_2(\sqrt{s}; M_\ell, M_u)\left[\prod_{i=2}^{\ell} p_{\mathrm{cm};\ell;i}\right]\left[\prod_{i=2}^{u} p_{\mathrm{cm};u;i}\right]. \tag{1}$$

where we have used $M_\ell; i$ and $p_{\mathrm{cm};\ell;i}$ to index the center-of-mass momenta and intermediate invariant mass at the i^{th} decay step of the lower vertex and similarly for the upper vertex.

We can properly populate phase space by starting with a sample that is drawn uniform in all the masses in the denominator of the left hand side of Eq. 1, *i.e.*, $dN/dM_\ell = \text{const.}$, etc., and then either weighting by the RHS or doing an accept/reject algorithm on the RHS.

We should draw the masses from a range that is defined independently of any of the other masses and keep only events that land within the kinematic boundary. We can write loose limits that are suitable for drawing the set of invariant masses.

$$\sum_{j=1}^{\ell} m_j^\ell < M_\ell < \sqrt{s} - \sum_{j=1}^{u} m_j^u \tag{2}$$

$$\sum_{j=1}^{i} m_j^\ell < M_{\ell;i} < \sqrt{s} - \sum_{j=1}^{u} m_j^u - \sum_{j=i+1}^{\ell} m_j^\ell \tag{3}$$

Here we have used $m_j^{\ell(u)}$ to refer to the mass of the j^{th} particle at the lower (upper) vertex. The limits on M_u and $M_{u;i}$ can be likewise be obtained by changing $\ell \to u$ in the above equations. One a full set of invariant masses are generated then they must be checked to ensure they land within the kinematic limits. If they do not, an entire new set of masses must be chosen.

In practice the `FixedTargetGenerator` allows the user to seed either or both the M_ℓ or M_u mass distributions with Breit-Wigner functions at the stage the M_ℓ and M_u are drawn. The generator can then be configured either to provide a weight for the events that allows the user to recover n-body phase space or these weights can be discarded if the users desires events to be distributed according to Breit-Wigner distributions.

Once the values of M_ℓ and M_u are chosen then the subsequent generation of the upper and lower vertex four-vectors is performed in an algorithm similar to the Raubold-Lynch method with modified kinematic limits as described in Ref. [1].

Generating t Dependence

Another key feature of t-channel production at GlueX energies is the dependence of the differential cross section on the four-momentum transfer t to the target. Empirically this is often modeled as

$$\frac{d\sigma}{dt} \propto e^{-\alpha t}, \tag{4}$$

where α is some constant of $\mathcal{O}(1)$ GeV^{-2}. A detailed model of the momentum transfer dependence is the domain of Regge phenomenology. However, this empirical observation has practical effects on Monte Carlo generation as it is a significant deviation from n-body phase space. It is desirable to be able to generate events that preferentially populate the low t region but then be able to reweight the events to recover true n-body phase space. Or in some cases generate events with unit weight whose only deviation from n-body phase space is that described by Eq. 4.

A discussion of the relevant kinematics can be found for example in Martin and Spearman [2] Eq. 4.81(b). For a two-body scattering process $a + b \to c + d$ we have

$$t = m_a^2 + m_c^2 - 2E_aE_c + 2qq'\cos\theta, \tag{5}$$

where the energies and momenta before q and after q' the collision are evaluated in the center of momentum (CM) frame. Here we reproduce the notation of Martin and Spearman. For our problem, $E_a = E_{\text{beam}}$, $m_c = M_u$, ..., and

$$q' = \frac{\lambda^{\frac{1}{2}}(s, M_\ell^2, M_u^2)}{2\sqrt{s}} \tag{6}$$

$$q = \frac{\lambda^{\frac{1}{2}}(s, M_{\text{beam}}^2, M_{\text{target}}^2)}{2\sqrt{s}}. \tag{7}$$

The angle θ is the scattering angle in the CM frame. Phase space is distributed uniformly in $\cos\theta$ over the range $1 \leq \cos\theta \leq 1$. To recover the phase space distribution we therefore need to consider the relationship between an interval dt and $d\cos\theta$. In particular, we can write for fixed values M_ℓ and M_u the following relation between the distributions in t and $\cos\theta$.

$$\begin{aligned}\frac{dN}{d\cos\theta} &= \frac{dt}{d\cos\theta}\left(\frac{dN}{dt}\right) \\ &= 2qq'\left(\frac{dN}{dt}\right) \\ &= \frac{\lambda^{\frac{1}{2}}(s, M_\ell^2, M_u^2)\lambda^{\frac{1}{2}}(s, M_{\rm beam}^2, M_{\rm target}^2)}{2s}\left(\frac{dN}{dt}\right). \end{aligned} \tag{8}$$

These equations then provide a description of how to recover phase space from events that are generated according to some prescribed t-dependence (dN/dt) like that shown in Eq. 4. One weights each event by the inverse of the RHS of Eq. 8. This returns $dN/d\cos\theta$ to a constant (uniform) distribution.

In practice the `FixedTargetGenerator` has an algorithm for generating $\cos\theta$, which implies a value of t through Eq. 5. In the the case that the user has not specified a t dependence, then $\cos\theta$ is simply generated uniformly, consistent with phase space. The generation can be seeded with a sum of one or more exponential functions of the form of Eq. 4. This combined function specifies dN/dt. Values of t are drawn from this distribution (over the full range $|t| > 0$) and a value for $\cos\theta$ is determined bin inverting Eq. 5. This value is then given a weight as prescribed by the function on the RHS of Eq. 8. Since t has a physical range that is related to masses M_u and M_ℓ, which are independently generated, this procedure will occasionally return unphysical values of $\cos\theta$. How these are managed is described in the next section. The result of this is that it is not possible to purely preserve phase space distributions in M_u and M_ℓ while simultaneously generating the exact prescribed distribution in t. The values of M_u and M_ℓ affect the minimum value of $|t|$ and hence variance in these values blur or smooth out the low $|t|$ edge of the distribution. This should cause no practical problems in the context of actual use cases.

Putting the Pieces Together

The goal is to provide the user with the option of specifying distributions for M_ℓ, M_u, and t, while at the same time providing the user with a weight that allows recovery of true n-body phase space. For the masses at the two vertices, the user can seed the generation with one or more simple relativistic Breit-Wigner distributions. For the t dependence one or more functions of the form in Eq. 4 can be used. *These functions are not meant to be a library of possible dynamics, they are only intended to be used to accumulate events in the correct regions of phase space. They are chosen because they are invertable distributions that can easily be generated with perfect efficiency. When paired with an additional software package for calculating the true intensity, like AmpTools, they enhance the efficiency of generating arbitrarily complex angular distributions.*

In all cases phase space is constructed from uniform distributions in M_ℓ, M_u, and t so user-defined upper and lower limits can be placed on these variables without any loss of generality or bias. The general procedure is to independently a set of three numbers M_ℓ, M_u, and $\cos\theta$, where "independently" means that values of one do not influence the ranges used in drawing the others. If these numbers are not drawn uniformly but instead according to some user-specified distribution, then a weight is calculated such that the user can recover all or part of the phase space distribution (see further discussion below). One then checks that the three numbers are physical: $M_\ell + M_u > E_{\rm cm}$ and $|\cos\theta < 1|$. If any are unphysical, then all are regenerated.

Once values of M_ℓ and M_u are obtained then a candidate set for the four-vectors of both the upper and lower vertex decays are generated. As described above and in Ref. [1] this uses the Raubold-Lynch method with modified kinematic limits. The modification to the limits may sometimes result in an unphysical set of four-vectors for the upper and lower vertices. Specifically the generated total invariant mass at say the upper vertex may exceed M_u. If this happens *both* sets of four-vectors are discarded and the generation is repeated *for the same values of M_u and $M\ell$*. Finally, for a physical set of upper and lower vertex four-vectors the

density of phase space is computed from the RHS of Eq. 1, and an accept/reject method is implemented such that the events are uniform in the phase space of of the decays of M_ℓ and M_u.

If the distributions of M_ℓ, M_u, or t were modified by user specified functions, then the generated events will need to be reweighted to recover correct n-body phase space. In principle, up to three weights are relevant: (1) the weight that returns the distribution of drawn M_u values to being flat (2) a similar weight for M_ℓ and (3) the weight that returns $\cos\theta$ to a uniform distribution. By default the weight returned with the generated event will be the product of all three. This means that by default if the user modifies any of the generated distributions of M_ℓ, M_u, or t then the weighted events generated should still correctly populate n-body phase space. Which of the three potential weights to include in the product is encoded by a three-bit number where from least to most significant the bits switch on the momentum transfer weight, the upper vertex mass weight, and the lower vertex mass weight, respectively. A variety of use cases and examples can be found later.

At the end, the final list of four-vectors are generated and boosted to the laboratory frame. These, along with the configured weight, are provided by the user in the form of a `Kinematics` object. The default ordering of the objects in the `Kinematics` class is as follows: beam, lower vertex particles, and upper vertex particles. The ordering of the particles from each vertex is given by the order in which their masses are specified when configuring the generator.

Using the Software

This section of the document is intended for developers that want to embed the `FixedTargetGenerator` in special executables. The `gen_amp_v2` program discussed at the start of the document likely provides an adequate interface for many GlueX applications.

The `FixedTargetGenerator` is a class inside of the `halld_sim` repository. It depends on two additional classes: `DecayChannelGenerator` and `BreitWignerGenerator`. These classes are used to generate indices according to some probability, like branching fractions of a decay, and also to generate relativistic Breit-Wigner distributions with high efficiency. The `FixedTargetGenerator` produces events in the `Kinematics` data structure that is specified by `AmpTools`. Other than this, there are no additional dependencies.

Generator Setup

The class provides a variety of constructors and member functions for seting up the reaction:

```
FixedTargetGenerator();
FixedTargetGenerator( double photonBeamEnergy, double targetMass,
                      const vector< double >& uvMasses,
                      const vector< double >& lvMasses );

void setSeed( unsigned int seed ){ m_randGen.SetSeed( seed ); }

// These functions setup the inital state.
void setBeamEnergy( double eBeam );
void setBeamP4( const TLorentzVector& p4 );
void setTargetMass( double targetMass );

// Use to set the masses of the stable particles produced
// at the upper and lower verticies
void setUpperVtxMasses( const vector< double >& uvMasses );
void setLowerVtxMasses( const vector< double >& lvMasses );
```

The algorithm depends on separating the final state particles into those coming from the upper and lower vertices. All that is necessary to specify are the masses of the particles at each vertex. For some applications it may be necessary to change the beam, target, or final state particles on an event-by-event basis. The

additional member functions permit this functionality. It is not necessary to destroy and recreate the object to do such operations. The most common operation for GlueX applications is resetting the beam energy for every event.

Adjusting the Distributions

A collection of functions are provided to adjust the generated distributions of M_u, M_ℓ, and t. Functions allow the user to set ranges from which these variables can be drawn.

```
// specify a range of mass to generate -- otherwise the full
// kinematic limits are used
void setUpperVtxRange( double min, double max );
void setLowerVtxRange( double min, double max );

// these should be absolute values:  |t|
void setMomentumTransferRange( double min, double max );
```

In addition the distribution in M_u or M_ℓ can be seeded by one or more relativistic Breit-Wigner functions by making one or multiple calls to the following functions. The `fraction` argument is optional and should be the fraction of events for which a particular Breit-Wigner distribution is used. If the sum of the fractions at either the upper or lower vertices is not equal to one, then the fractions will be renormalized such that that is the case. Similar functionality is provided by the `addMomentumTransfer` function. In this case the `tSlope` is the value of α in Eq. 4.

```
// This is used to seed generation with a BW to allow the user
// to concentrante events in regions of mass where the amplitude
// might peak.
void addUpperVtxBW( double mass, double width, double fraction = 1 );
void addLowerVtxBW( double mass, double width, double fraction = 1 );

void addMomentumTransfer( double tSlope, double fraction = 1 );
```

Controlling Reweighting

If the user has specified a non-uniform distribution in M_u, M_ℓ, or t (using any of the functions in the previous section), then one can control which variables are reweighted to try to recover phase space. An `enum` provides the bitmap for the variables.

```
enum { kMomentumTransfer = 1, kUpperVtxMass = 2, kLowerVtxMass = 4 };

void setReweightMask( unsigned int mask );
```

For example, calling `setReweightMask` with an argument of 6 (= `kUpperVtxMass | kLowerVtxMass`) will generate events with a weight so that the phase space distributions in M_u and M_ℓ are recovered but the weighted events are distributed according to the t distribution specified by the user. By default the reweight mask is 7 (all bits on) and the generated weights ensure the weighted distributions are consistent with pure n-body phase space.

Example Use Cases

For the sake of discussion, let's consider the reaction $\gamma p \to a_2^- \Delta^{++}$, where $a_2^- \to \eta\pi^-$ and $\Delta^{++} \to p\pi^+$. This is a $2 \to 4$ reaction where the dynamics generates non-trivial distributions in the four-body phase space spanned by the final state of the reaction $\gamma p \to \eta\pi^+\pi^- p$. Depending on the objective there are a variety of different ways one might used the `FixedTargetGenerator` in the context of studying this reaction.

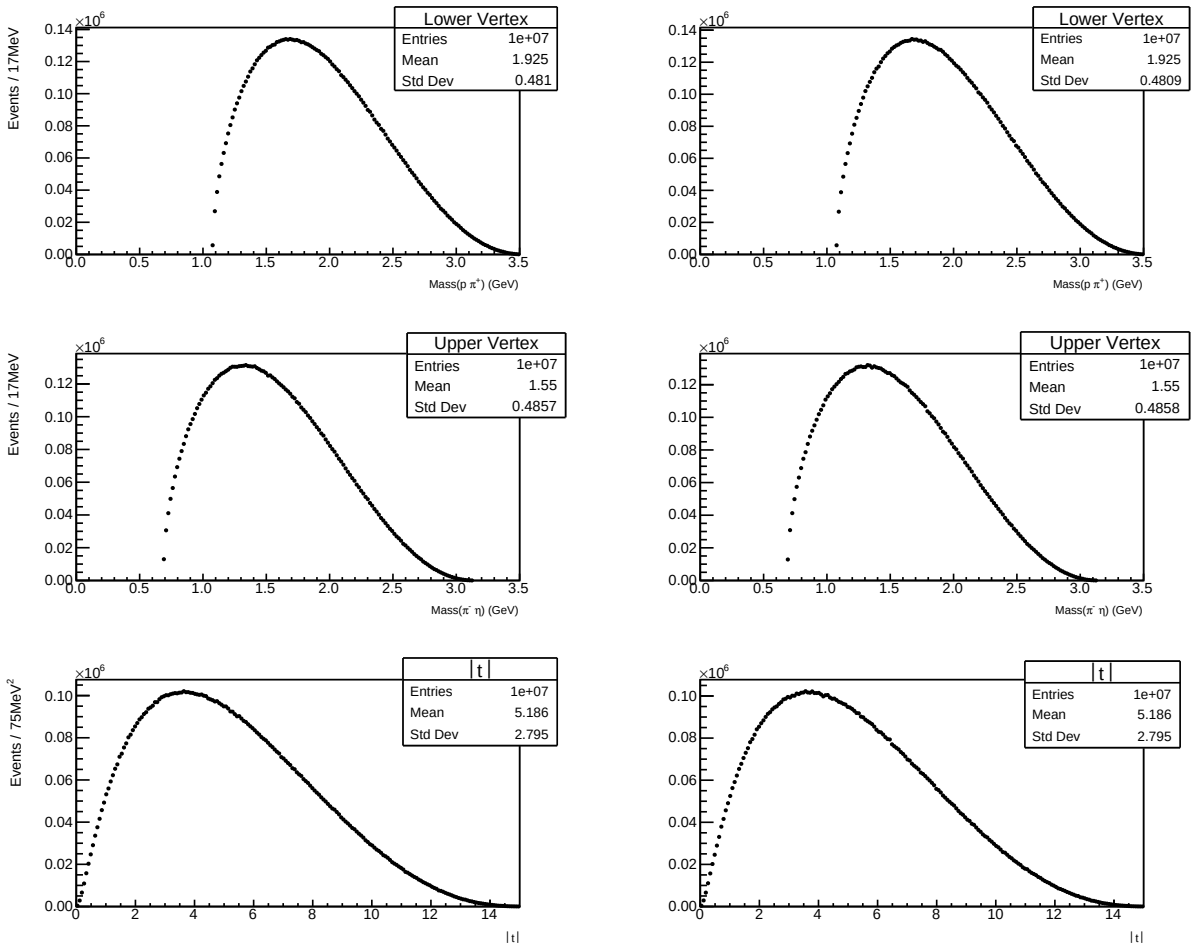

Figure 1: From top to bottom the distributions of upper vertex mass M_u, lower vertex mass M_ℓ, and momentum transfer $|t|$ for $\gamma p \to \eta\pi^-\pi^+p$ phase space space generated with `FixedTargetGenerator` (left) and `TGenPhaseSpace` (right).

Phase Space Shapes

One may be interested in the shape of phase space as a function of $M(\eta\pi^-)$ or $M(\pi^+p)$. In this case the upper vertex masses should be initialized with a vector containing M_η and M_π. The lower vertex masses should be a vector containing M_π and M_p. The target mass is that of the proton and the beam energy should be set appropriately. The `generate()` method will then generate sets of four-vectors with unit weight that are distributed according to four-body phase space and any desired variable can be plotted.

Figure 1 illustrates that the `FixedTargetGenerator` reproduces the same shapes as generating four-body phase space with the common `TGenPhaseSpace` algorithm that is embedded in ROOT. It is helpful to compare both the means and standard deviations of the distributions. In the subseqent examples we will try to illustrate that these phase space shapes can be recovered with reweighting in a variety of scenarios.

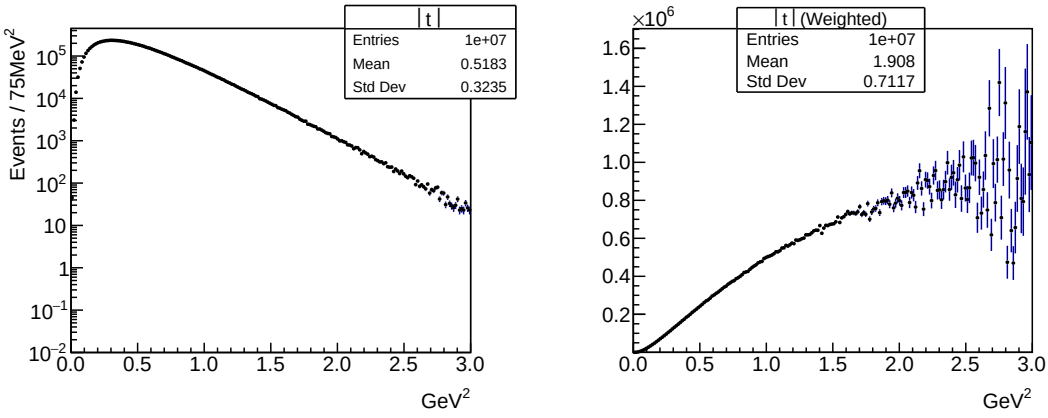

Figure 2: Example using `addMomentumTransfer` with an argument of 4. The left plot shows the distribution when the t-reweighting bit is disabled. The right plot shows the reweighted distribution that recovers phase space. Due to the limited statistics in the region $|t| > 2$ GeV^2 the mean and standard deviation are not comparable to those in Fig. 1.

Adding a t Dependence

If one defines t as the four-momentum transfer squared from the target to the Δ^{++}: $t = |p_p - p_\Delta|^2$, then one will notice that the distribution of t in phase space (as obtained by the generation exercise in the previous section) does not match that of typical GlueX data. Let's assume that the data are roughly distributed according to

$$\frac{d\sigma}{dt} \propto e^{-4|t|/\mathrm{GeV}^2}. \tag{9}$$

How one proceeds with this knowledge depends on the objective.

Matching MC to a t dependence observed in data. If one is, say, performing an amplitude analysis of the $\eta\pi^-$ or π^+p system in bins of t, then what matters for such an analysis is that the t distribution approximately matches data for the accepted MC sample that is used in the fit. To achieve this, one can call the `addMomentumTransfer` function with `tSlope = 4`. By default, the generator will then generate events where the t distribution is approximately given by Eq. 9, and these events will be given a weight such that the weighted distribution recovers phase space. If the model being fit to data has no explicit t dependence, which would be typically true if one was fitting resonances in the $\eta\pi^-$ system, then recovering phases space shape as a function t is not necessary. If the t dependence is the only modification to phase space (there are Breit-Wigners at either vertex) then the weight can be discarded or set to unity. Otherwise, calling `setReweightMask( 6 )` will disable the t weight in the overall event weight.

Fitting a t-Dependent Model to Data. Suppose that a model that includes some t dependence is now being fit to data for the reaction $\gamma p \to \eta\pi^-\Delta^{++}$. For example, such a model may include a double-Regge description of the event. These models describe a deviation from phase space and therefore if one is going to integrate such intensity distributions in the context of an amplitude analysis, one needs to start with a distribution of events that is distributed according to phase space. The problem is that very few phase space events land in the region where the model predicts a considerable intensity. Therefore, integrating this intensity with phase space MC alone is inefficient and not statistically precise. The solution is to preferentially populate the region where the intensity is large by introducing a t dependence to the MC but then doing it in a way such that true phase space can be recovered by reweighting. In this case it is useful, as

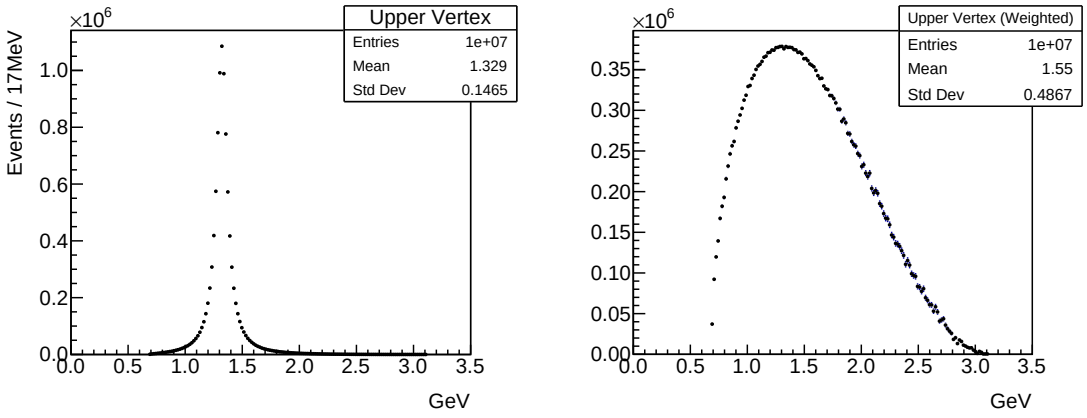

Figure 3: Adding a_2^- Breit-Wigner lineshape using `addUpperVtxBW`. The left plot shows the distribution when the upper vertex reweighting bit is turned off. The right plot shows the reweighted distribution, which matches phase space distribution illustrated in Fig. 1.

before, to call the `addMomentumTransfer` function with `tSlope = 4`. In this example the generated weights need to propagated all the way through the analysis chain in order to recover the phase space distribution at the time the amplitude analysis is performed. The generated weight should multiply any additional weight introduced in the analysis so that the weighted sample always behaves like a phase space sample.

It is important to note that in such cases the generated t distribution need not exactly match the data. The only requirement is that the reweighted t distribution match phase space. A better match of generated MC to data in this case will only enhance the statistical precision of the normalization integrals used in the amplitude analysis for a fixed size of generated MC.

Generating Events with the t-Dependent Model. If one would like to generate events with t-dependent model that is, for example, coded in an `AmpTools` intensity computation, then one should proceed in a similar was as the section above. It is advantageous to introduce a simple t-dependence in the generator that approximates the model. The generator then generates a set of four-vectors, call the set $\vec{x}_i$ for the i^{th} generated event, and a weight w_i that can be used to recover phase space. Using `AmpTools` one can then compute the intensity (probability) of observing $\vec{x}_i$: $I(\vec{x}_i)$. One then performs a standard accept/reject algorithm on the product $w_i I(\vec{x}_i)$. If the t dependence closely approximates the model then the fluctuations of $w_i I(\vec{x}_i)$, will be significantly less than the fluctuations of $I(\vec{x}_i)$ alone, and the accept/reject algorithm will be much more efficient. The resulting distribution t of the events that are accepted will be that of phase-space modified by the model and not the t-dependence that was introduced to approximate the model.

Adding a Mass Dependence

In cases like $\gamma p \to a_2^- \Delta^{++}$, it can be advantageous to introduce a dependence on M_u or M_ℓ that deviates from phase space since very few events will simultaneously populate the regions of the a_2 and the Δ^{++}. The functions `addLowerVtxBW` and `addUpperVtxBW` allow the user to introduce one or more relativistic Breit-Wigner distributions into the shape of the mass distribution. By default, weights are provided such that the reweighted events correctly populate phase space. There are similar ideas and applications for adding a mass dependence as to the t dependence. Figures 3 and 4 show examples of generating Breit-Wigner distributions and using weights to recover phase space.

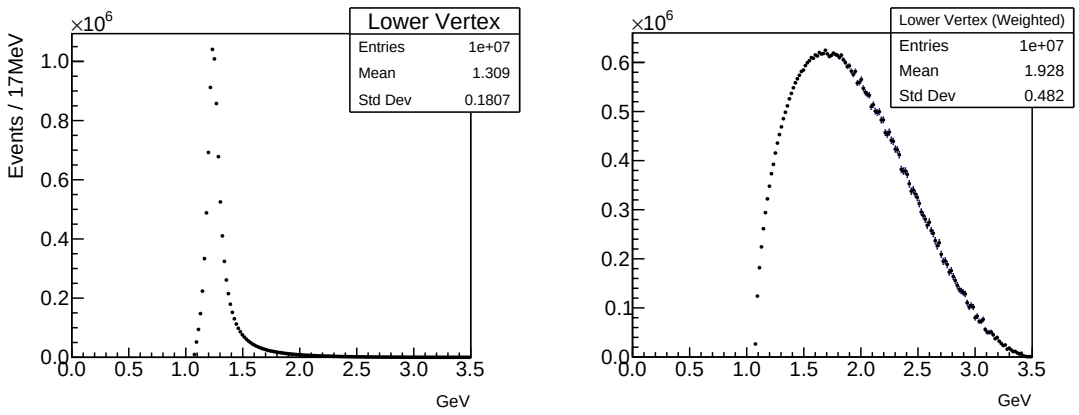

Figure 4: Adding Δ^{++} Breit-Wigner lineshape using `addLowerVtxBW`. The left plot shows the distribution when the lower vertex reweighting bit is turned off. The right plot shows the reweighted distribution, which matches phase space distribution illustrated in Fig. 1.

Matching MC to Data. In some stages of an analysis, one might only have a desire to approximate the a_2 and $\Delta_+ +$ line shape. This could be helpful, for example, in understanding efficiencies or in the case that one is doing fits but the fit functions only include angular distributions of the decays and not the line shapes. In this case one should add Breit-Wigner distributions using the nominal a_2 and Δ^{++} parameters. Likely it will also be helpful to a t-distribution similar to that of data. Then the reweight mask can be set to zero or the weights can be discarded. In this case, one will have events generated with the simple Breit-Wigner lineshapes for the a_2 and Δ^{++} and the proper t-dependence.

Fitting a Mass-Dependent Model to Data. Suppose that one wants to study the lineshape of the Δ^{++} in the context of a_2 production. In this case, one may desire to fit the data with an amplitude that includes Δ^{++} decay angles and $M(p\pi^+)$ distribution. To compute the acceptance for such a fit would be desirable to have a MC set that has the correct t dependence and approximates the shape of a_2 such that the efficiency is correct but leaves the lower vertex mass distribution according to phase space such that all of the dynamics is captured in the amplitudes that are describing the Δ^{++} decay. In this application one should add both the a_2 and Δ^{++} Breit-Wigners as well as the correct t dependence to the generator. The reweight mask should be set to only `kLowerVtxMass`. The effect of this will be concentrate generated events at low t and in the regions of the a_2 and Δ^{++}, but when one makes a plot of the weighted $M(\pi^+ p)$ mass distribution then it follows the phase space shape. This strategy then enhances the statistical precision of the integral of the amplitude, which includes the Δ^{++} line shape that peaks in the region where the events are concentrated. The weights for the events need to be propagated through the entire analysis chain when creating the generated and accepted MC sets for fitting.

Generating events with a Mass-Dependent Model. In a similar fashion to the t-dependence case discussed above one can seed the upper and lower vertex mass distributions with one or more Breit-Wigner distributions that approximate a mass-dependent model that is fully specified by an `AmpTools` config file. It then becomes much more efficient to do accept/reject on the product of the generated weight and the computed event intensity from `AmpTools`.

If one desires to introduce a t-dependence into the generated data then one can combine this approach with that described above for generating according to a t-dependent model. If a simple exponential t-dependence is sufficient, then one can add this using the `addMomentumTransfer` function, as noted above. In this case it is useful to set the reweight mask to `( kUpperVtxMass | kLowerVtxMass )`. Then the events will be

Table 1: Fraction of events remaining after making successive kinematic requirements using a phase space sample of MC and a sample of MC generated with `FixedTargetGenerator` where generation is seeded with approximate a_2 and Δ^{++} line shapes along with a t distribution.

Requirement	`FixedTargetGenerator` Fraction Remaining [%]	`TGenPhaseSpace` Fraction Remaining [%]
none	100	100
$M(\eta\pi^-)$: $\pm$ 1$\Gamma(a_2^-)$	76.4	16.4
and $M(p\pi^+)$: $\pm$ 1$\Gamma(\Delta^{++})$	62.0	1.0
and $\lvert t\rvert < 3$	62.0	0.2

reweighted to respect phase space in the upper and lower vertex mass distributions, which is appropriate for accept/reject using an intensity that includes dynamics in those variables, but there will be no reweighting of t and the events will have approximately the specified t distribution.

Generation Efficiency

A common practical problem with simulating reactions like $\gamma p \to a_2^- \Delta^{++}$ is that the intensity for this reaction peaks in a small region of four-body phase space (around the a_2 mass, around the Δ^{++} mass, and at low momentum transfer). If one is doing accept/reject on phase space MC to carve out the correct mass and angular distributions for this reaction then only a tiny fraction of generated phase space events will be accepted.

Seeding the generation with approximate distributions in the various kinematic variables but then reweighting to recover phase space can dramatically increase efficiency. Table 1 shows the number of events that pass a series of increasingly restrictive kinematic requirements that select events in the region where the intensity is significantly different from zero. Using `FixedTargetGenerator` increases the efficiency for events landing in the region of interest by a factor of three-hundred. In the limit that the approximate Breit-Wigner distributions used in `FixedTargetGenerator` match the true Breit-Wigner distributions used in the `AmpTools` accept/reject algorithm the efficiency for accept/reject increases dramatically because the product of the weight to recover phase space (which is small at the peak of the Breit-Wigner distribution) and the intensity function of `AmpTools` (which has a Breit-Wigner peak) exhibits small fluctuations. As the desired generated distributions become more narrow or the dimension of phase space grows, the increase in generation efficiency can easily rise to many orders of magnitude.

References

[1] M.R. Shepherd, "Notes on Phase Space," GlueX-doc 6399 (2024).

[2] A.D. Martin and T.D. Spearman, *Elementary Particle Theory*, Wiley (1970).