Herwig++ Decays and Spin Correlations

Peter Richardson IPPP, Durham

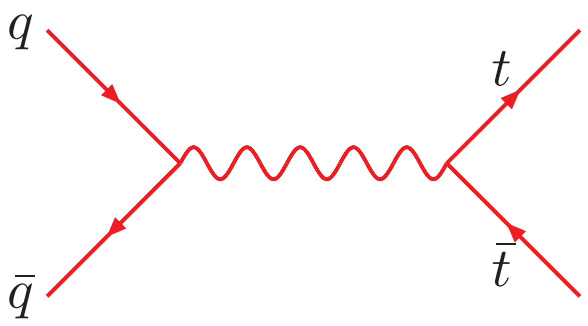
- Introduction
- Review of Algorithm
- What's done
- Perturbative Physics
- Hadron Decays
- Where next

Introduction

- The whole structure of the decays is based on including spin correlations.
- Instead of presenting the full algorithm I will illustrate it with an example.
- Consider the processes $q\bar{q} \rightarrow t\bar{t}$ followed by semi-leptonic decay of the top quarks.
- First we generate the momenta of the top quarks in $q\bar{q} \rightarrow t\bar{t}$ using the polarized cross-section

$$\rho^1_{\kappa_1\kappa'_1}\rho^2_{\kappa_2\kappa'_2}\mathcal{M}_{\kappa_1\kappa_2;\lambda_1\ldots\lambda_n}\mathcal{M}^*_{\kappa'_1\kappa'_2;\lambda_1\ldots\lambda_n}$$

Top pair Production



$$|\mathcal{M}|^2 = \rho^1_{\kappa_1\kappa'_1}\rho^2_{\kappa_2\kappa'_2}\mathcal{M}^{q\bar{q}\to t\bar{t}}_{\kappa_1\kappa_2;\lambda_t\lambda_{\bar{t}}}\mathcal{M}^{q\bar{q}\to t\bar{t}}_{\kappa'_1\kappa'_2;\lambda_t\lambda_{\bar{t}}}$$

Spin Correlations

- One of the outgoing quarks is selected at random and a spin density matrix calculated.

$$\rho_{\lambda_j\lambda'_j} = \frac{1}{N_\rho}\rho^1_{\kappa_1\kappa'_1}\rho^2_{\kappa_2\kappa'_2}\prod_{i\neq j} D^i_{\lambda_i\lambda'_i} \mathcal{M}_{\kappa_1\kappa_2;\lambda_1\ldots\lambda_j\ldots\lambda_n}\mathcal{M}^*_{\kappa'_1\kappa'_2;\lambda'_1\ldots\lambda'_j\ldots\lambda'_n}$$

  In the initial step we take $D^i_{\lambda_i\lambda'_i} = \delta_{\lambda_i\lambda'_i}$.
  The normalization is chosen so that $\mathrm{Tr}\rho = 1$.

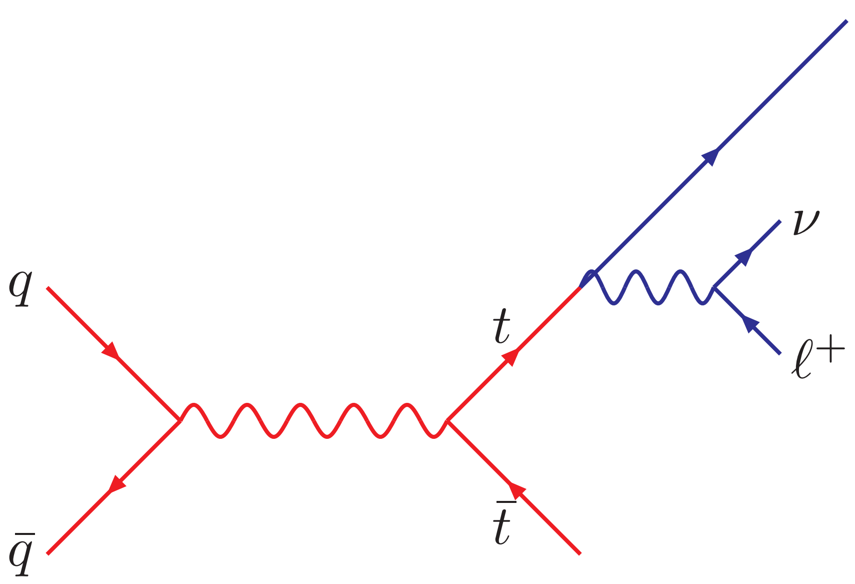
- This normalization is therefore the same as the matrix element used in the previous step.

- The decay mode of the particle is selected using the branching ratios.

- The momenta of the particles produced in the n-body decay are generated using

$$\rho_{\lambda_0\lambda'_0}\mathcal{M}_{\lambda_0;\lambda_1\ldots\lambda_n}\mathcal{M}^*_{\lambda'_0;\lambda_1\ldots\lambda_n}$$

Top pair Production



$$|\mathcal{M}|^2 = \rho^1_{\kappa_1\kappa'_1}\rho^2_{\kappa_2\kappa'_2}\mathcal{M}^{q\bar{q}\to t\bar{t}}_{\kappa_1\kappa_2;\lambda_t\lambda_{\bar{t}}}\mathcal{M}^{*q\bar{q}\to t\bar{t}}_{\kappa'_1\kappa'_2;\lambda'_t\lambda_{\bar{t}}}$$
$$\mathcal{M}^{t\to b\ell\nu}_{\lambda_t}\mathcal{M}^{*t\to b\ell\nu}_{\lambda'_t}$$

Spin Correlations

- In principle we could now continue down the decay chain, but here all the particles produced are stable.

- When all the particles in a decay have been developed a decay matrix is calculated

$$D_{\lambda_0\lambda'_0} = \frac{1}{N_D}\mathcal{M}_{\lambda_0;\lambda_1\ldots\lambda_n}\mathcal{M}^*_{\lambda'_0;\lambda'_1\ldots\lambda'_n}\prod_{i=1,n} D^i_{\lambda_i\lambda'_i}$$

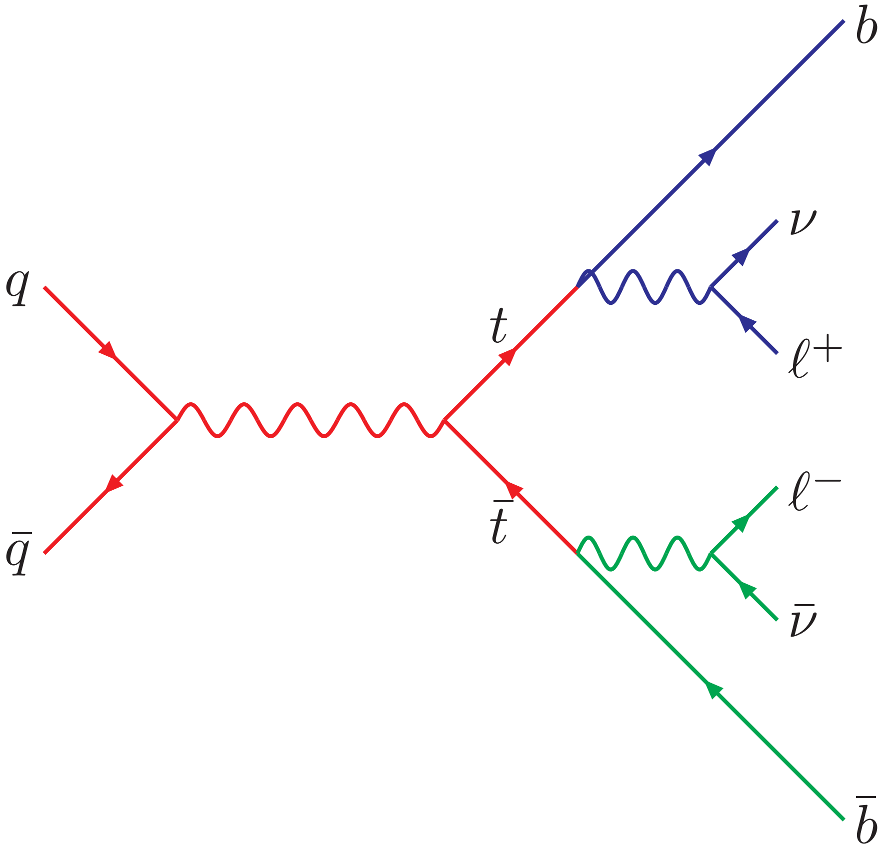
  The normalization is chosen so that $\mathrm{Tr}D = 1$.

- We then select another particle from the hard process. The whole procedure is repeated for this particle using the calculated decay matrices for those particles which have been decayed rather than the identity.

- In this case this means we decay the second quark.

- As before the normalization of the spin density matrix cancels the matrix element used for the previous step.

Top pair Production



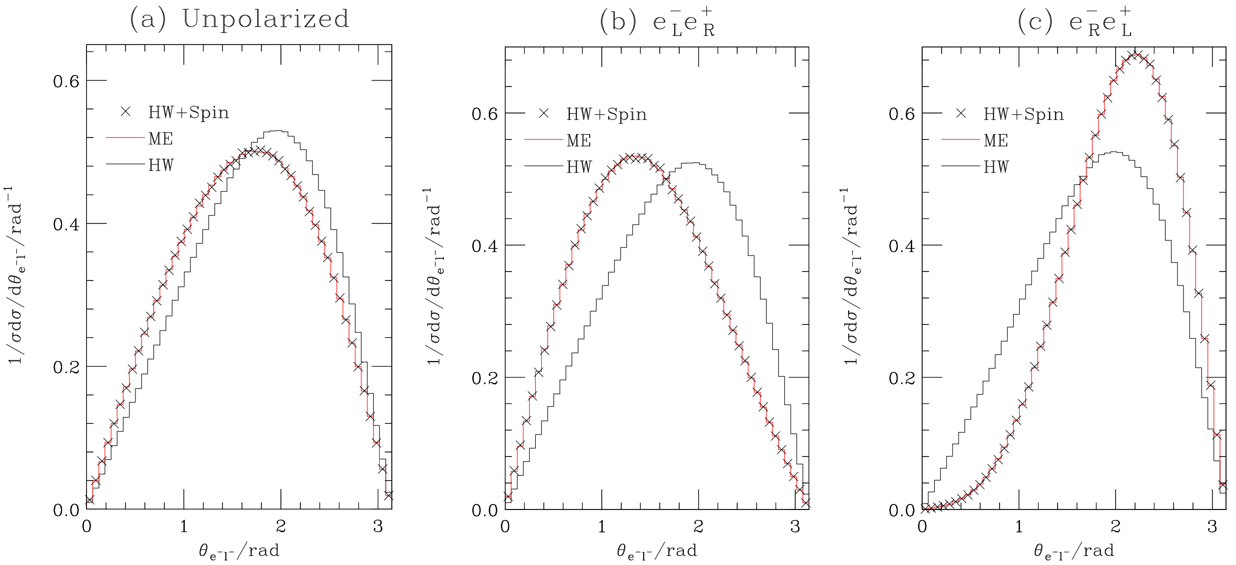
$$
\begin{aligned}
|\mathcal{M}|^2 \quad = \quad & \rho^1_{\kappa_1\kappa'_1}\rho^2_{\kappa_2\kappa'_2}\mathcal{M}^{q\bar{q}\to t\bar{t}}_{\kappa_1\kappa_2;\lambda_t\lambda_{\bar{t}}}\mathcal{M}^{*q\bar{q}\to t\bar{t}}_{\kappa'_1\kappa'_2;\lambda'_t\lambda'_{\bar{t}}} \\
& \mathcal{M}^{t\to b\ell\nu}_{\lambda_t}\mathcal{M}^{*t\to b\ell\nu}_{\lambda'_t} \\
& \mathcal{M}^{\bar{t}\to\bar{b}\ell\nu}_{\lambda_{\bar{t}}}\mathcal{M}^{*\bar{t}\to\bar{b}\ell\nu}_{\lambda'_{\bar{t}}}
\end{aligned}
$$

- This is the narrow width matrix element for the process.

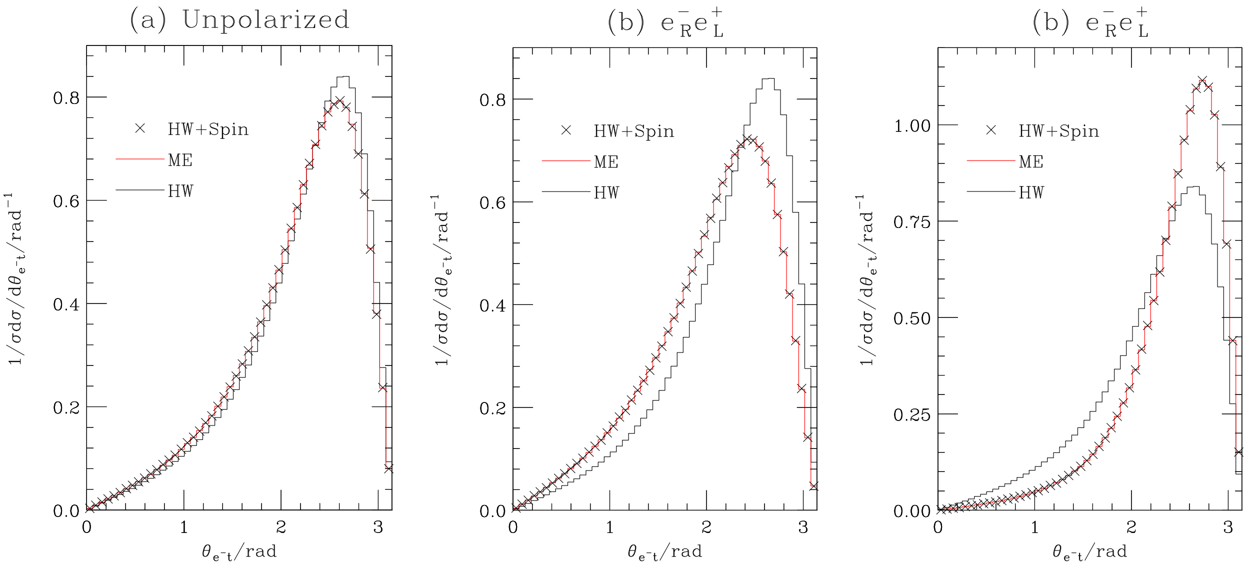
Top pair production in e^+e^- collisions

- Study the semi-leptonic decays of both top quarks and look at the correlations.

- Between the lepton produced in the antitop decay and the beam.



- Between the lepton and the top quark.



Spin Correlations in C++

- In FORTRAN we only did this for the decays of fundamental particles.

- In C++ I wanted a better, more general structure.

- I also want the computer to to more of the work, *i.e.* less calculation by hand.

- In order to do this we need to be able to
  - Numerical compute basis states, *i.e.* spinors, polarization vectors etc.
  - Perform operations, like boosts or dot products, with them
  - Combine them easily to calculate matrix elements.

- I decided to base most of this on the same ideas as HELAS.

- This gives us everything we need and was faster, at least in FORTRAN.

- Use C++ to give additional structure to make it easier to use.

Wave Functions

- A number of container classes were designed to store things
  - LorentzPolarizationVector for ϵ^μ.
  - LorentzSpinor for u and v spinors.
  - LorentzSpinorBar for $\bar{u}$ and $\bar{v}$ spinors.
  - LorentzTensor for $\epsilon^{\mu\nu}$
  - RhoDMatrix for ρ and D matrices
- Provides storage of the components, boosts etc.
- This code is now transferred to ThePEG.
- Classes were designed to calculate the wavefunction given the momentum, particle type.
  - ScalarWaveFunction trivial for scalars
  - SpinorBarWaveFunction for $\bar{u}$ and $\bar{v}$ spinors.
  - SpinorWaveFunction for u and v spinors.
  - TensorWaveFunction for $\epsilon^{\mu\nu}$
  - VectorWaveFunction for ϵ^μ.
- All based on WaveFunctionBase for storage of momentum and particle data.
- May also needed classes for spin-$3/2$ in the future.

Vertices

- The WaveFunction classes can then be combined using Vertex classes to compute matrix elements.
- All the vertices are based on the VertexBase class to store the number of external particles interacting and which particles interact.
- The classes doing the calculation for a particular combination of particle species then inherit from this, for example the FFVVertex class implements the fermion-antifermion-vector vertex.
- The vertices are expressed in terms of general couplings
$$c\left(g_L P_L + g_R P_R\right)\gamma^\mu$$
which are a virtual function.
- The Vertex class can either compute a complex number for the vertex or an off-shell WaveFunction which can be used as the input to another vertex.
- Classes implementing specific interactions, *e.g.* the Standard Model ffZ vertex then implement this functions..

$e^+e^- \to \gamma/Z^0/\text{graviton} \to f\bar{f}$ Example

```
// setup momenta and particle data for the wavefunctions
SpinorWaveFunction     ein(meMomenta()[ielectron],
                           mePartonData()[ielectron],incoming);
SpinorBarWaveFunction  pin(meMomenta()[ipositron],
                           mePartonData()[ipositron],incoming);
SpinorWaveFunction    aout(meMomenta()[ilp]       ,
                           mePartonData()[ilp]       ,outgoing);
SpinorBarWaveFunction fout(meMomenta()[ilm]       ,
                           mePartonData()[ilm]       ,outgoing);
VectorWaveFunction inter;
TensorWaveFunction interg;
// sum over helicities to get the matrix element
for(inhel1=-1;inhel1<2;inhel1=inhel1+2)
{
ein.reset( inhel1);
for(inhel2=-1;inhel2<2;inhel2=inhel2+2)
{
pin.reset(inhel2);
for(outhel1=-1;outhel1<2;outhel1=outhel1+2)
{
fout.reset(outhel1);
for(outhel2=-1;outhel2<2;outhel2=outhel2+2)
{
aout.reset(outhel2);
// Z exchange diagram
inter = _theFFZVertex->evaluate(sHat(),1,_Z0,ein,pin);
diag1 = _theFFZVertex->evaluate(sHat(),aout,fout,inter);
me[1] += real(diag1*conj(diag1));
// photon exchange diagram
inter = _theFFPVertex->evaluate(sHat(),1,_gamma,ein,pin);
diag2 = _theFFPVertex->evaluate(sHat(),aout,fout,inter);
me[2] += real(diag2*conj(diag2));
// graviton diagram
interg = _theFFTVertex->evaluate(sHat(),1,_graviton,ein,pin);
diag3  = _theFFTVertex->evaluate(sHat(),aout,fout,interg);
me[3] += real(diag3*conj(diag3));
// the full thing including interference
diag3+=diag1+diag2;
me[0] += real(diag3*conj(diag3));
}}}}
```

Vertices

- Large number of general structures implemented

  FFSVertex, SSSSVertex, SSSVertex, VSSVertex, VVSSVertex, VVSVertex, FFTVertex, FFVTVertex, SSTVertex, VVTVertex, VVVTVertex, FFVVertex, VVVVertex, VVVVVertex,

- and the vertices of the Standard Model SMFFGVertex, SMFFHVertex, SMFFPVertex, SMFFWVertex, SMFFZVertex, SMGGGGVertex, SMGGGVertex, SMWWHVertex, SMWWWVertex, SMWWWWVertex,

- and RS model

  RSModelFFGRVertex, RSModelFFVGRVertex, RSModelSSGRVertex, RSModelVVGRVertex, RSModelVVVGRVertex

Matrix Elements

- Allows matrix elements to be easily implemented.
- Should be easy to extend to SUSY etc.
- A lot can be done in terms of general spin structures.
- Can be used to implement the spin correlations for perturbative processes, which is not done yet.
- Always provides some useful features, *e.g.* wavefunction calculation, for non-perturbative decays.
- Possible to use for automated matrix element generation etc.

Correlations

DecayVertex
Storage of ME
ρ and D matrix calculation

HardVertex
Storage of ME
ρ and D matrix calculation

ShowerVertex
Storage of branching
ρ and D matrix calculation

HelicityVertex
Base Class for the storage of the Vertex
Pointer to the incoming/outgoing particles
Virtual methods to calculate the ρ and D matrices

SpinInfo
Base class for the storage of spin info
Pointers to production/decay vertex
set/get methods for ρ/D matrices
Storage of the state of the particle

Scalar, Fermion, Vector, Tensor SpinInfo
Storage of basis vectors
vectors in production and decay frame

Correlations

- The SpinInfo classes and the HelicityVertex class are now in ThePEG.
- The DecayVertex and HardVertex classes which store the matrix elements and calculate the spin density matrices remain in Herwig++.
- Nothing done yet for the shower.
- Obviously needs Decayers and MatrixElements which can make use of and supply the information.

Hadron Decays

- A large number of hadron, and tau, decays are now implemented in Herwig++.
- All of them are inherit from the DecayIntegrator class.
  - This is a class which performs multichannel integration.
  - Channels can be specified in the code or via the Repository.
  - Can optimise the weights for the channels.
  - All the parameters can be inputted via the Repository.
- In most cases tried to put as much as possible in a few base classes
- Other classes for similar decays can then be written more easily.

Off-Shell Effects

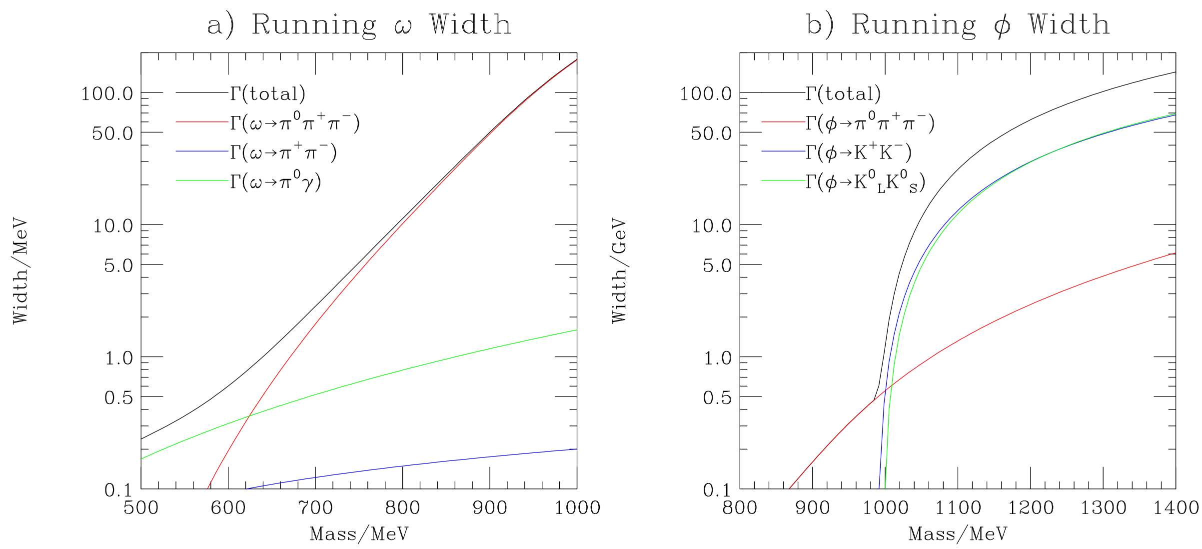
- In FORTRAN Herwig we only produced hadrons on mass-shell.
- Fundamental particles were smeared by a Breit-Wigner.
- In Herwig++, at least for the hadrons include a factor

$$\frac{1}{\pi}\int_{m^2_{\rm low}}^{m^2_{\rm max}} dm^2 \frac{m\Gamma(m)}{(m^2-M^2)^2+m^2\Gamma(m)^2}$$

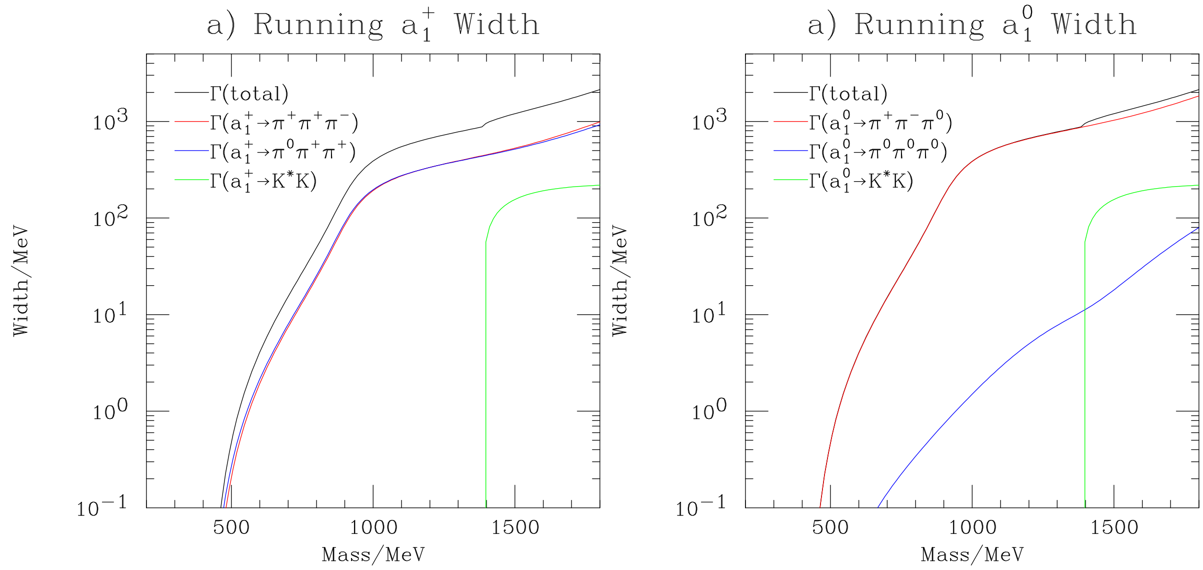
- Have a new MassGeneratorBase class, inherits from MassGenerator from ThePEG.
- Can either return this weight or unweight to generate a mass.
- Has a number of simple $1 \to 2$ matrix elements built in to calculate the running width.
- Inheriting classes can implement other matrix elements.

Off-Shell Effects

- Allows us to calculate the running width easily.
- ω and ϕ widths



- a_1 width



- DecayIntegrator can used this when calculating the matrix element.

Off-shell effects

- Some questions about things we may want to change in ThePEG.
  - Make the limits on the off-shell mass asymmetric.
  - Some limit on masses allowed for given decay modes to allow off-shell modes to be included.
  - Varying the branching ratio with mass.
- I probably need to understand more about what's in the DecayMode class.

Tau Decays

- Any tau decay can be written as

$$\mathcal{M} = \frac{G_F}{\sqrt{2}}\bar{u}(p_{\nu_\tau})\gamma^\mu(1-\gamma_5)u(p_\tau)J_\mu,$$

- Leptonic part implemented in the TauDecayerBase class.

- All decayers inherit from this and implement the virtual member to calculate the hadronic hadronic current, J^μ.

- Most of the important modes are now implemented
  - $\tau \to K, \pi, \rho, K^*, a_1$ in Tau1MesonDecayer,
  - $\tau \to \pi\pi, K\pi, KK$ in Tau2MesonDecayer,
  - $\tau \to \ell\nu_\ell$ in Tau2LeptonsDecayer,
  - $\tau \to \pi\pi\pi$ in Tau3PionCLEODecayer,
  - $\tau \to PPP$ in Tau3MesonDefaultDecayer,
  - $\tau \to \pi\pi\pi\pi$ in Tau4PionNovosibirskDecayer.

- Most models taken from TAUOLA.

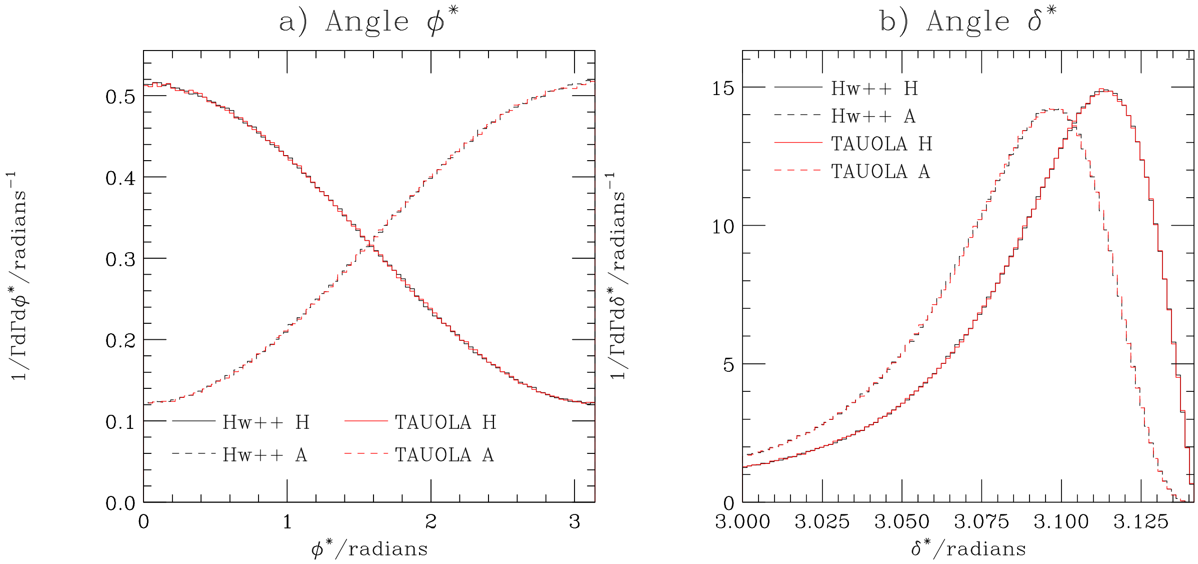
- All parameters available via the repository.

- Spin Correlations included

Tau Decays

- Some examples of tau decays.

- Correlation between decays in $H, A \rightarrow \tau^+\tau^-$ followed by $\tau^\pm \rightarrow \nu_\tau \pi^\pm$

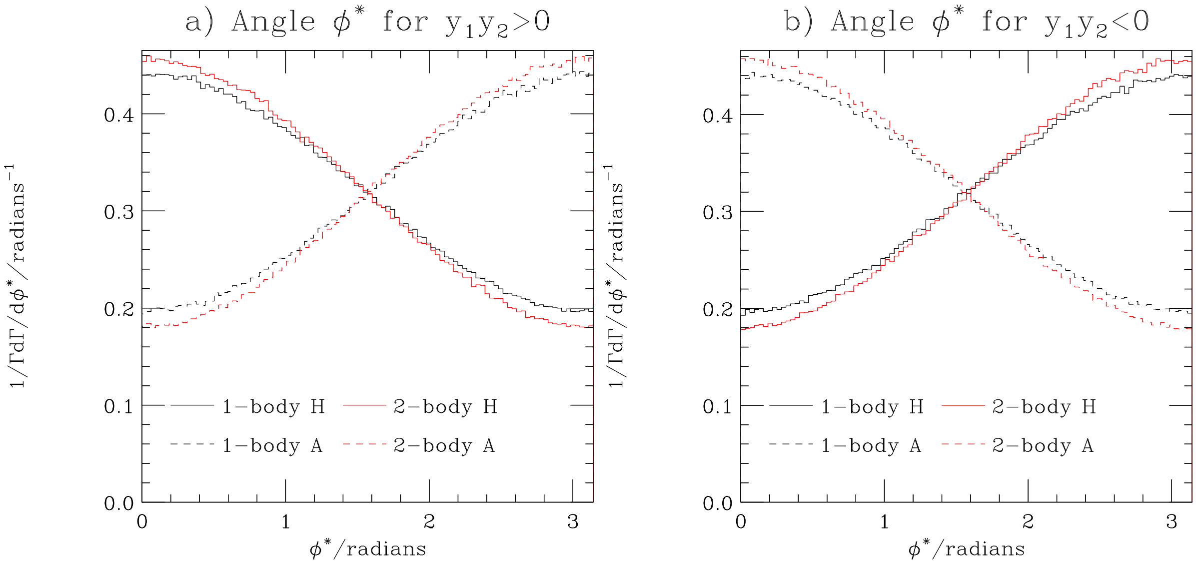


- ϕ^* is the angle between the two decay planes

- δ^* is the angle between the pions in the Higgs rest frame.

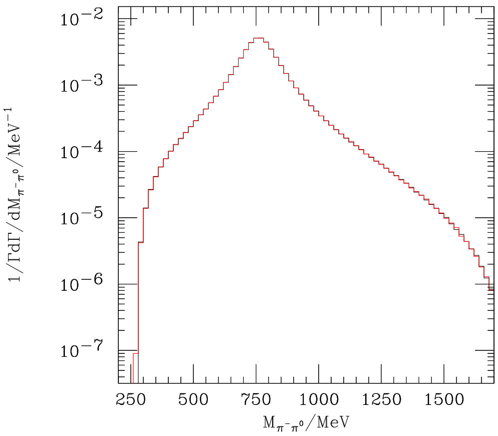
Tau Decays

- Correlation between decays in $H, A \rightarrow \tau^+\tau^-$ followed by $\tau^\pm \rightarrow \nu_\tau \rho^\pm$



- ϕ^* angle between ρ decay planes in the $\rho^+\rho^-$ rest frame.

- Difference between lines due a bug in the boosting of the polarization vectors, I think this will need to be discussed.



Vector Meson Decays

- Write the vector meson decays as

$$\mathcal{M} = \epsilon^\mu J_\mu,$$

- As with the taus the VectorMesonDecayerBase class calculates the matrix element from the polarization vector and current.

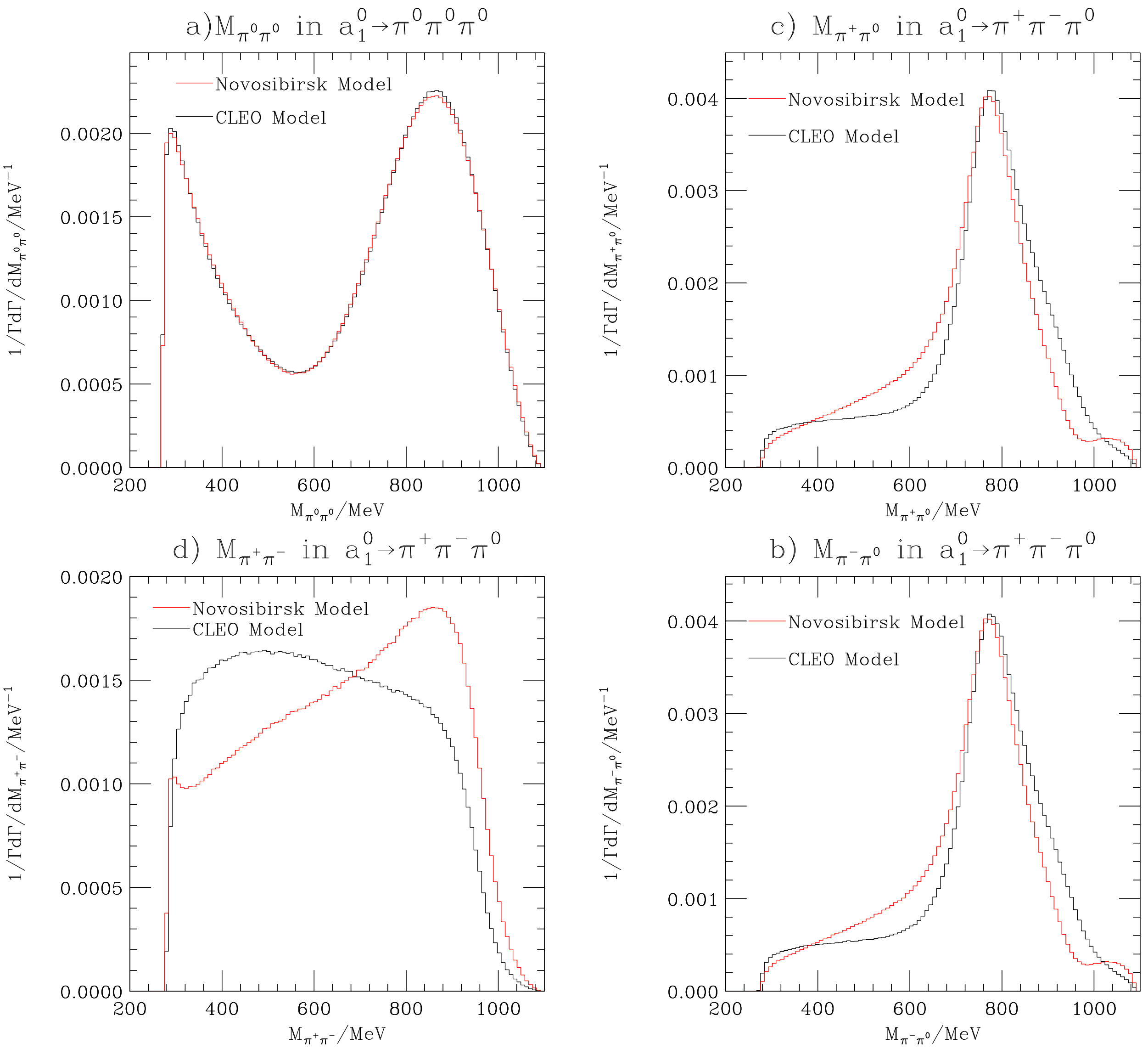
- Classes inheriting from this implement the current, J^μ
  - $V \to f\bar{f}$ in VectorMeson2FermionDecayer
  - $V \to PP$ in VectorMeson2MesonDecayer
  - $\omega, \phi \to \pi\pi\pi$ in VectorMeson3PionDecayer
  - $V \to Vf\bar{f}$ in VectorMesonPScalarFermionsDecayer
  - $a_1 \to \pi\pi\pi$ in a1ThreePionCLEODecayer & a1ThreePionDecayer

- All parameters available via the Repository.

- In many cases new modes can also be added.

a_1 Decays

- Neutral a_1 decay



Tensor Meson Decays

- As with the tau and vector mesons write the matrix element in a general form.

$$\mathcal{M} = \epsilon^{\mu\nu} T_{\mu\nu},$$

- The combination of the decay tensor and the matrix element is handled by the TensorMesonDecayerBase class.
- The virtual hadronicTensor() member is implemented in classes inheriting from this.
- A number of modes are implemented
  - $T \to PP$ in TensorMeson2PScalarDecayer
  - $T \to PV$ in TensorMesonVectorPScalarDecayer
  - $T \to VV$ in TensorMesonVectorVectorDecayer

Scalar Decays

- No point in having a base class in this case.
- A few simple modes implemented.
  - $P \to V f \bar{f}$ in PScalarVectorFermionsDecayer
  - $P \to VV$ in PScalarVectorVectorDecayer
- Some work on semi-leptonic decay modes.

Semi Leptonic Decays

- For the semipleptonic decays implement the decay in terms of form factors in a few base classes.

- Any model for the decays can then be implemented in classes inheriting from this and implementing the form factors.

- At the moment three base classes
  - $P \to P\ell\nu$ in SemiLeptonicScalarScalarDecayer
  - $P \to V\ell\nu$ in SemiLeptonicScalarTensorDecayer
  - $P \to T\ell\nu$ in SemiLeptonicScalarVectorDecayer

- and a couple of models to test them.

Baryon Decays

- Nothing done yet.
- Plan to implement a few base classes with general structures for the decays
  - $B \to B'\ell\nu$
  - $B \to B'P$
  - $B \to B'\gamma$
- Only for spin-$\frac{1}{2}$
- Need to think about implementing things for spin-$3/2$.

EvtGEN

- The experimentalists are becoming very attached to this.
- We should be able to interface to it.
- This is a two-edged sword because the experimentalists will then use this even if the things in Herwig are better.
- Also calls FORTRAN PYTHIA in a lot of cases.
- Very strong for a few B mesons but weaker for light mesons and nothing for heavier B mesons or any baryons.

Where Next

- Need to finish the implementation and documentation of what's all ready there.
- Implement something for the baryons.
- Interface to EvtGEN.
- Do something for the perturbative decays.

Photon Radiation

- In FORTRAN we only had photon radition in the shower.
- A lot of decay modes in the tables with hard photons.
- The distributions are probably pretty bad.
- Want to write something in C++ to replace the FORTRAN PHOTOS package.
- Should be able to do a better job.
- Hopefully Keith Hamilton will work on this with me.

Decay Tables

- Having started to edit the Herwig decay tables I think we need some tools to help.

- The idea would be a seperate database storing the particle data.

- This would have
  - Additional comments, *i.e.* why arbitary decisions were made.
  - Star ratings for particles and decay modes.
  - Other info?

- Could then output the input files for ThePEG or it a human readable format, *i.e.* HTML or LaTeX.

Plan for Perturbative Physics

- The Helicity code which is all ready implemented provides a good base.

- Code simple classes based on spin structures.

- Using the Vertex classes these can then be used for a large number of decays or hard processes without any further work.

Plans for BSM Physics

- All the parameters of the model should be implemented as a class inheriting from the Herwig StandardModel class.
- The Herwig StandardModel class is based on the StandardModelBase class of ThePEG but also supplies access to the Vertex classes.
- Implement new vertices for the model as well.
- This is already done for the RSModel as a simple example.
- Have a number of classes with general spin structures that can use these vertices to calculate decays and production matrix elements.
- This still needs to be done.

GRID

- We have 3 years of funding for a GRID postdoc in Durham to work mainly on Herwig++ and partly on educating U.K. phenomenologists about the GRID.

- We need to agree deliverables with PPARC, we only got there comments at the end of last week.

- Nigel and I thought this miht be a good opportunity to discuss the deliverables so we get things we want done.