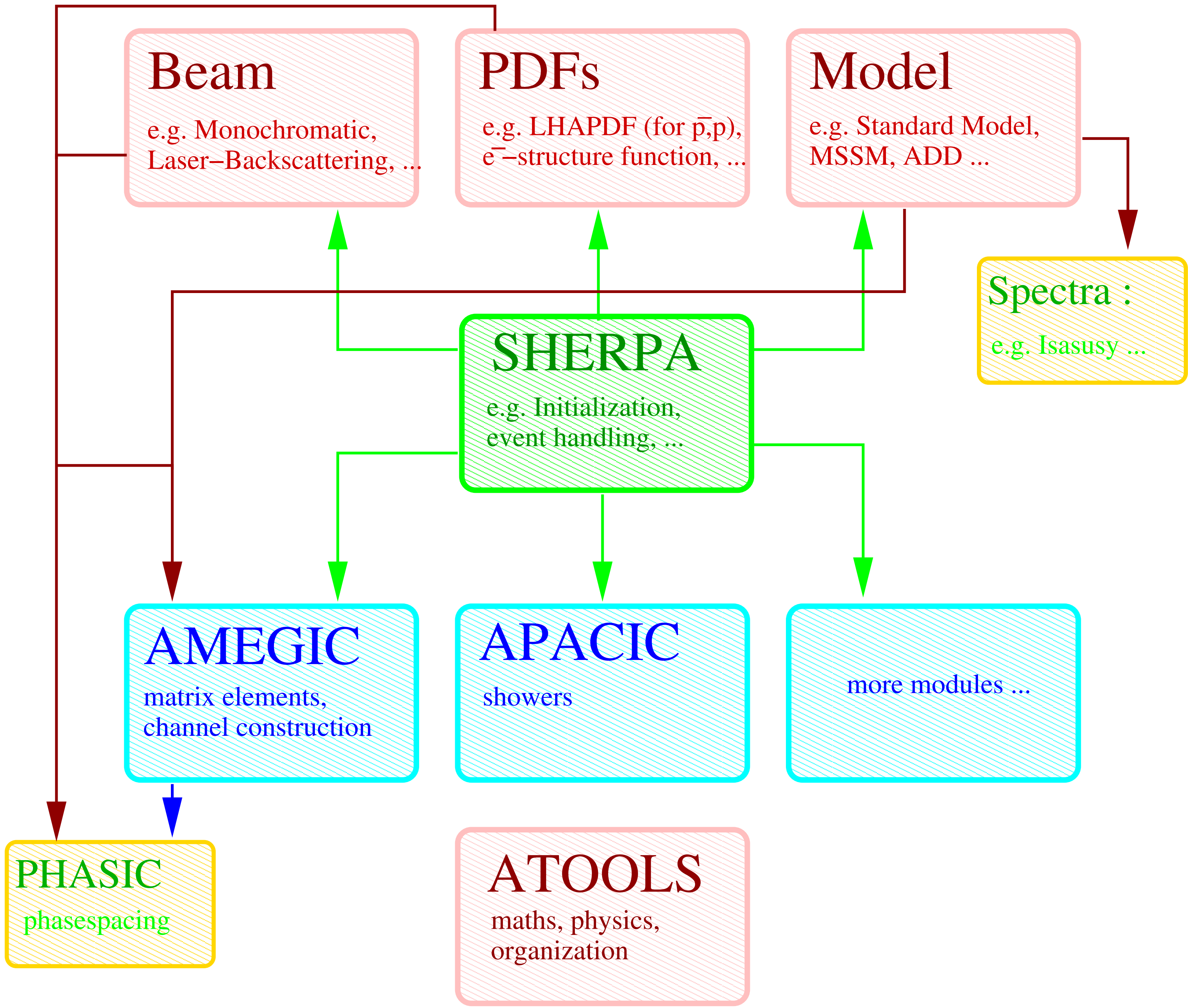
Introduction to AMEGIC++

(Frank Krauss, CERN TH)



Beam
e.g. Monochromatic,
Laser–Backscattering, ...
PDFs
e.g. LHAPDF (for $\bar{p}$,p),
e^-–structure function, ...
Model
e.g. Standard Model,
MSSM, ADD ...
Spectra :
e.g. Isasusy ...
SHERPA
e.g. Initialization,
event handling, ...
AMEGIC
matrix elements,
channel construction
APACIC
showers
more modules ...
PHASIC
phasespacing
ATOOLS
maths, physics,
organization

Features of `AMEGIC++`

- Calculation of arbitrary tree-level processes in the SM, the MSSM and the ADD model using Helicity amplitudes
- Explicit polarization vectors available to cut boson props
  $\Rightarrow$ only a minimal number of helicity building blocks is required to calculate (in principle) arbitrary processes
- Multi-channel MC integration with adaptive optimization
- Completely automatic approach (a generator-generator)
  $\Rightarrow$ library files are created that have to be compiled and linked

The procedure

- Select a model framework for your calculation and specify the input
  $\Rightarrow$ can be your preferred spectrum generator

- Choose the process(es) or classes of processes to be evaluated

- All available $1 \rightarrow 2$ and $1 \rightarrow 3$ vertices are initialized

- Vertices are matched onto topologies $1 \rightarrow \mathrm{N_{jet}} + 1$
  $\Rightarrow$ conversion into amplitudes

- Feynman diagrams get translated into helicity amplitudes

- The helicity amplitudes and the associated color matrix are stored

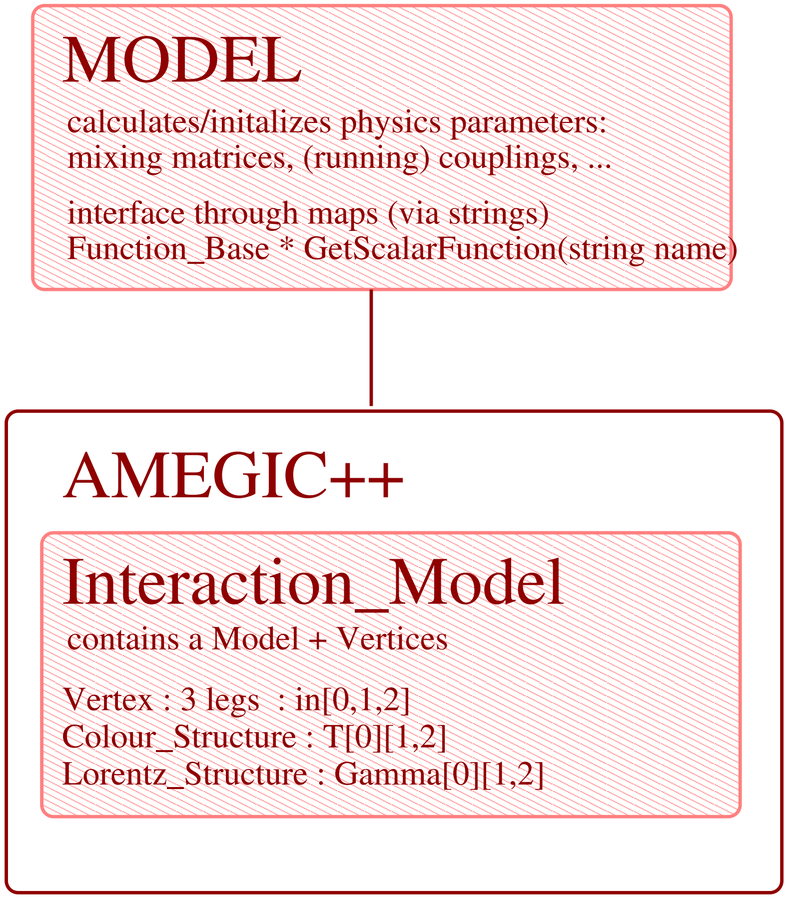
- Appropriate integration channels are produced and stored as well

$\rightarrow$ Compile the libraries, rerun and enjoy

The AMEGIC++ Model framework (C++ at work)

- The parameters: All model specific parameters are handled by classes derived from the base class Model_Base that possesses a Spectrum_Generator and stl map's of word strings to model specific parameters. These map's are filled by one of the classes Standard_Model, MSSM or ADD
- The interactions: The entire interaction vertices of a specific model are collected by classes derived from the class Interaction_Model_Base possessing a Model_Base a Vertex object and methods that organize the entire interactions, examples are Interaction_Model_SM or Interaction_Model_sQCD ...

Once initialized this handling allows for a consistent treatment of all model parameters within the whole event generator framework


MODEL
calculates/initalizes physics parameters:
mixing matrices, (running) couplings, ...
interface through maps (via strings)
Function_Base * GetScalarFunction(string name)
AMEGIC++
Interaction_Model
contains a Model + Vertices
Vertex : 3 legs : in[0,1,2]
Colour_Structure : T[0][1,2]
Lorentz_Structure : Gamma[0][1,2]

The MSSM setup

- General Feynman rules (unitary gauge, conserved R-Parity) according to J.Rosiek, Phys.Rev.D41:3464,1990 have been implemented (some quartic sparticle interactions still missing, will soon be added)
- Interface to ISASUSY 7.67 to get the MSSM spectra
- The violated fermion number flow of Majorana fermions is cured by using appropriate Feynman rules
  - Implemented the approach of Denner *et al.*, Nucl.Phys.B387,1992
  - The relative sign of interfering Feynman graphs is determined avoiding the usage of charge-conjugation matrices
  - Instead an orientation (fermion flow) for each appearing fermion line is defined
  - Requires only a minimal number of vertices

```
//gluino - gluon - gluino
Flavour flgluino = Flavour(kf::gluino);
Flavour flgluon  = Flavour(kf::gluon);
if (flgluino.IsOn() && flgluon.IsOn()) {
   vertex[vanz].in[0]    = flgluino;
   vertex[vanz].in[1]    = flgluon;
   vertex[vanz].in[2]    = flgluino;
   kcpl0 = kcpl1 = -g3;
   vertex[vanz].cpl[0]  = vertex[vanz].cpl[1]  = kcpl0.Value();
   vertex[vanz].Str     = (kcpl0*PR+kcpl1*PL).String();
   vertex[vanz].ncf     = 1;
   vertex[vanz].Color   = new Color_Function(cf::F,0,1,2,'0','1','2');
   vertex[vanz].nlf     = 1;
   vertex[vanz].Lorentz = new Lorentz_Function(lf::Gamma);
   vertex[vanz].Lorentz->SetParticleArg(1);
   vertex[vanz].on      = 1;
   vanz++;
}
```

Helicity method :

- Basic idea : Re-write *everything* with spinors !
- Products : $\bar{u}(\lambda_1,\, p_1)u(\lambda_2,\, p_2) =$ simple functions of vector–components.
- Polarization vectors (helper $q \not\parallel p$): $\epsilon(p,\, \lambda)\frac{1}{\sqrt{4p\cdot q}}\,\bar{u}(q,\lambda)\gamma^\mu u(p,\lambda)$
- Fermion–Propagators ($m^2 = p^2$):

$$(\not{p} + \mu) \;\rightarrow \frac{1}{2}\sum_\lambda \left[u(p,\lambda)\bar{u}(p,\lambda)\left(1+\frac{\mu}{m}\right) + v(p,\lambda)\,\bar{v}(p,\lambda)\left(1-\frac{\mu}{m}\right)\right]$$

- Construct larger building blocks:

$$Z(p_1,\lambda_1;p_2,\lambda_2;p_3,\lambda_3;p_4,\lambda_4) = \bar{u}(p_1,\lambda_1)\gamma^\mu u(p_2,\lambda_2)\bar{u}(p_3,\lambda_3)\gamma_\mu u(p_4,\lambda_4)$$

- All building blocks are now complex numbers!

Amplitude Generation

- Problem: Factorial growth of Feynman diagrams contributing with increasing number of outgoing particles

$e^+e^- \rightarrow$	Feynman diagrams
e^+e^-	4
$e^+e^-\mu^+\mu^-$	50
$e^+e^-e^+e^-$	144
$e^+e^-e^+e^-\mu^+\mu^-$	3690
$e^+e^-e^+e^-e^+e^-$	13896

traditional method of summing and squaring inappropriate

Helicity method Limited number of building blocks (complex numbers):

- $Z(1,2;3,4) = [\bar{u}(p_1)\gamma^\mu u(p_2)]\,[\bar{u}(p_3)\gamma_\mu u(p_4)]$
- $X(1,2;3) = [\bar{u}(p_1)\not{p}_3 u(p_2)]$
- $Y(1,2) = [\bar{u}(p_1)u(p_2)]$

$\Rightarrow$ Every amplitude corresponds to a complex number

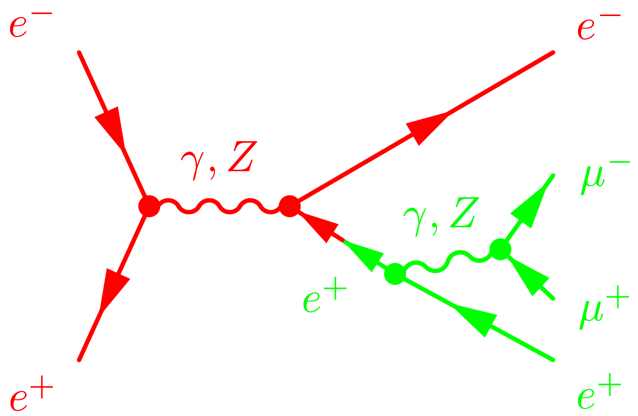
$\Rightarrow$ Summing over diagrams and multiplying with the complex conjugate results into the value of the amplitude for one helicity combination

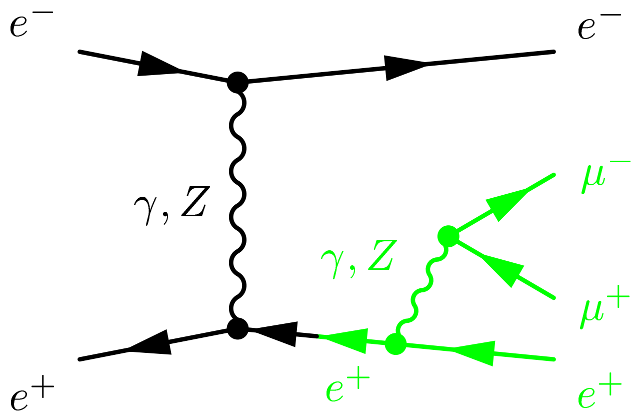
$\Rightarrow$ Summing over all helicity combinations results into the total amplitude

$\Rightarrow$ Natural treatment of polarizations

Superamplitudes

Idea: Factorize out common Helicity building blocks of different amplitudes





$$
\begin{aligned}
& Z^1_\gamma \cdot Z^2_\gamma + Z^1_\gamma \cdot Z^2_Z + Z^1_Z \cdot Z^2_\gamma + Z^1_Z \cdot Z^2_Z \\
+ \quad & Z^3_\gamma \cdot Z^2_\gamma + Z^3_\gamma \cdot Z^2_Z + Z^3_Z \cdot Z^2_\gamma + Z^3_Z \cdot Z^2_Z \\
= \quad & (Z^1_\gamma + Z^1_Z + Z^3_\gamma + Z^3_Z) \cdot (Z^2_\gamma + Z^2_Z)
\end{aligned}
$$

Superamplitudes $\Rightarrow$ Number of complex multiplications is reduced significantly
$\Rightarrow$ Results in an enormous acceleration of the program

Features of the algorithm:

- Amplitudes that differ only by boson props are combined
- s- and t-channels can be combined as well
- Works only for amplitudes with equal color structure
- Algebraic tools can be improved further to be more efficient

Some more refinements

- Explicit polarization vectors/tensors for external particles, allows to
- “cut open” internal vector/tensor bosons through insertion of suitable completeness relation.
- Colour handling: Write out word string of colour factors, simplify and store result for colour matrix (colour decomposition straightforward)
- Next targets: “Knowledge storing” for multi-channel integrators – many Jacobians appear far more than once . . .