CalcHEP version 2.5

A.Pukhov, *SINP MSU, Russia*

Alexander Belyaev *Southampton University, GB*

Neil Christensen *Michigan State University, US*

Tools2008

Munich, June 30 - July 4

General features of CalcHEP.

CalcHEP is a package for computation of Feynman diagrams,
integration over multi-particle phase space,
and partonic level event generation.

Main features are: CalcHEP

- *Works with generic model;*
- *Calculates symbolically squared Feynman diagrams;*
- *Has user-friendly menu driven interface with on-line help.*
- *Batch mode is based on interactive version (simulation of keyboard hits).*

WEB disposition: CalcHEP: http://theory.sinp.msu.ru/˜ pukhov/calchep.html

Version 2.5 has to appears on WEB in the end of this workshop.

BSM implemented in CalcHEP.

During development of micrOMEGAs are realized MSSM(SuSpect, Isajet, SoftSusy, SPheno), NMSSM(NMSSNtools), CPVMSSM(CPSuperH), RHNM

The following BSM realizations are known

LeptoQuarks(A. Belyaev et al, hep-ph:/0502067)

Little Higgs (A. Belyaev, et al arXiv:hep-ph/0609179)

Higgsless 3-site model (A. Belyaev, S. Chivukula, N. Christensen, E. Simmons et. al. arXiv:hep-ph/arXiv:0708.2588)

MUED5, MUED6 (see Kong WEB page `http://home.fnal.gov/~kckong/mued/`*)*

Almost all these realization were done by means of LanHEP package.

CalcHEP development.

Last years CalcHEP mainly was developed as a matrix element generator for astro-particle package micrOMEGAs.

Recently in collaboration with Alexander Belyaev and Neil Christensen CalcHEP was significantly improved for collider physics applications.

Namely it was realized a batch for multi-processes calculations with decays of outgoing particles and event mixing using parallel calculations on PC-farms. This job mainly was done by Alexander Belyaev and Neil Christensen and will be presented by Neil in the second part of this talk.

Other developments ... Models with spin 2 and spin 3/2 particles

A possibility to work with high spin particles was realized

```
                    Particles
 Full  name       | P | aP| number |2*spin| mass |width |color|aux|
electron          |e  |E  |11      |1     |0     |0     |1    |   |
s_electron        |~eL|~EL|1000011 |0     |MSeL  |0     |1    |   |
Gravitino         |~g |~g |1000039 |3     |MGr   |wGr   |1    |   |
Massive Graviton|Gh |Gh |2000039 |4     |MHGr  |whGr  |1    |   |
```

Then interaction can be presented like

```
 Vertices
A1    |A2    |A3    |A4    |>          Factor             <|>  Lorentz part
~eL   |E     |~g    |      |i/Sqrt2/Mpl                    |G(m3)*G(p1)
~EL   |~g    |e     |      |i/Sqrt2/Mpl                    |G(p1)*G(m2)
```

Processes with polarized incoming particles.

It is realized for incoming massless particles of spin 1/2 and spin 1. One have to use the % symbol before particle name to force CalcHEP to calculate diagrams symbolically using polarized density matrix.

```
 e%,E% -> 2*x
```

In numerical session the user can set numerical values for helicities.

Automatic width calculation.

Particle widths included in the model can be calculated automatically using vertices of interaction. It mainly is interesting for BSM particles whose masses unknown.

```
                    Particle
 Full   name    | P  | aP| number  |2*spin| mass  |width |color|aux|
Light  Higgs    |h   |h  |25       |0     |Mh     | !wh  |1    |   |
Heavy  higgs    |H   |H  |35       |0     |MHH    | !wHh |1    |   |
```

The exclamation mark at particle width is a instruction to CalcHEP to generate the code for `h->2*x`*, and* `H->2*x` *and compile them together with the codes of main processes.*

Code optimization in CalcHEP

One problem of CalcHEP is a huge size of codes for 2→4 processes. We have realized a option to reduce significantly size of generated code and accelerate numerical calculation.

In general CalcHEP compiles code for diagram which mainly is a polynomial over momenta scalar product with coefficients which are functions on model parameters. Now CalcHEP calculates one time for all diagrams all needed products of scalar products and store them in array. Coefficients at this monomial are calculated one time for session and also stored in some array.

So, the code for one diagram is extremely short now.

```
static const int jumps[30]={266,271,272,274,276,285,293,294,393,395,398
399,409,410,418,419,422,424,429,432,433,435,441,442,459,468,477,478,492
};
 for(i=0,R=0 ;i<30;i++){R+=CC[i]*DPmonom[jumps[i]];}
```

*Array **jumps** is used to bypass zero coefficients.*

After MC4BSM3 workshop ... Quadruple Precision

Me met some physical applications which need calculation with high precision. We have a switch which force high precision calculation.

Even standard gcc compiler on PC supports long double length of 10.

Old platforms like OSF1 work with long double length of 16.

Modern Intel compiler also supports real type quad length of 16.

CalCHEP_2.5 supports calculations with hight precision real types.

Universal SLHA reader.

The reader for SLHA with arbitrary BLOCK structure is created.

```
......
BLOCK NMIX  # Neutralino Mixing Matrix
  1  1     9.98499129E-01   # Zn11
  1  2    -1.54392008E-02   # Zn12
......
```

There is one command to read SLHA file:

*slhaRead(char*fileName,int mode)*

which downloads in computer memory all information and another command which returns the need item

*slhaVal(char * BlockName,double Scale, int KeyLength, ...);*

There is a possibility to check that the needed item exists: For example

if(slhaValExists("Nmix",2,1,2))Z12=slhaVal("Nmix",100.,2,1,2)); else ...

If there are several blocks with the same name but different scales, slhaVal interpolates numbers for Scale point.

Universal SLHA reader.

Work with DECAY items

```
DECAY   1000037   3.895221E+00 # second chargino(~2+) width
#   Branching  Nout  ID_1    ID_2    ....
   1.8877E-03   2    24     1000022       # Br(~2+ -> W+  ~o1)
   2.4367E-01   2    25     1000024       # Br(~2+ -> h   ~1+)
     ..........................
```

is organized in the same manner:

```
int slhaDecayExists(int pNum);
double slhaWidth(int pNum);
double slhaBranch(int pNum,int N, int * nCh);
```

The is code disposed in CalcHEP/c_sources/model_aux/SLHAread.c and can be used out of CalcHEP package.

Even Fortran interface is written.

Interface with Showering and Hadronization event Generators

This interface is based on Les Houches Accord for event generators hep-ph/0109068—.

But instead of `FORTRAN COMMON BLOCKs` the information is passed into text file according to the later conventions.

In result of work of CalcHEP event generator the `event_#.txt` files are created in different directories.

```
      SCANDIR('dirname')
```

command opens all `event_#.txt` files and reads their headers. It can be called several times for different directories for both scattering and decay events.

After that any call of `UPEVNT` will choose a subprocess according to its cross section, read an event record and substitute recursively all decays generated.

QCD color flows generated in collisions are continued in decays tails.

Interface with .... Qnumbers

The main idea of this approach was to share New Physics/SM physics between CalcHEP/PYTHA. In principle only SM particles should be passed to PYTHIA.

From the other side in the event record we keep information about all intermediate particles, in particular their life time is saved for subsequent reconstruction of space-time picture of event.

So, we have to pass to PYTHIA information about quantum numbers of non-SM particles.

It is done via QNUMBERS (P. Skands et al, hep-ph/0311123) block generated by CalcHEP automatically and include description of all non-SM particles:

```
BLOCK QNUMBERS 1000045 # ~chi_50
  1   0  # 3 times electric charge
  2   2  # number of spin states (2S+1)
  3   1  # color rep (1: singlet, 3: triplet, 8: octet)
  4   0  # Particle/Antiparticle distinction (0=own anti)
```

The Batch job

The preliminary calls of CalcHEP for cross section calculation and one channels event generations and subsequent event mixing with recursive decays can by realized by one BATCH command which will be presented by Neil Christensen now.