CalcHEP for beyond Standard Model Physics.

A.Pukhov, SINP MSU, Russia

CalcHEP is a package for computation of Feynman diagrams and integration over multi-particle phase space, event generation and generation matrix element code for other programs. The main idea prescribed into CalcHEP is to make available the passing on from Lagrangians to final distributions effectively with a high level of automation. Non interactive batch calculations are supported as well.

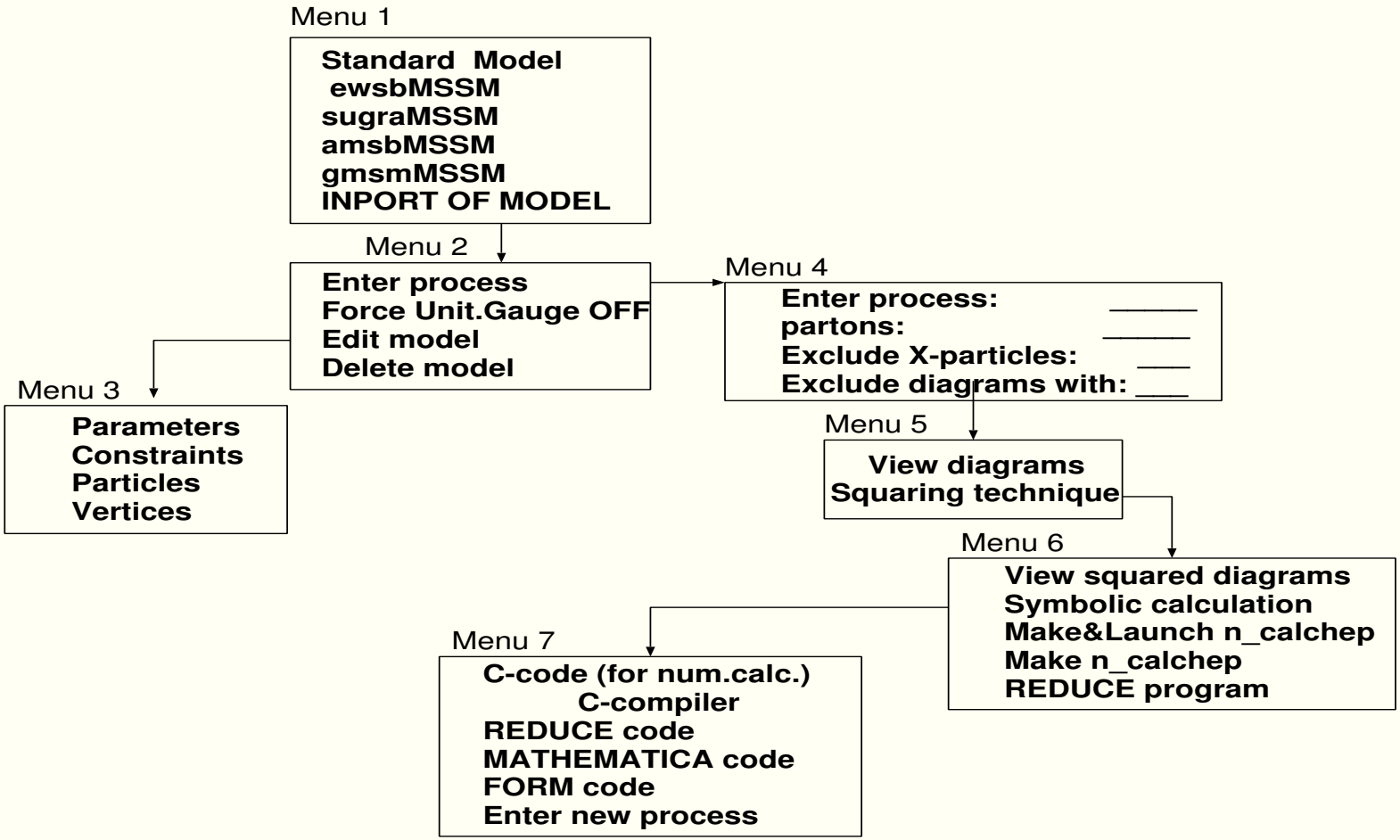
The existing of internal symbolic calculator makes CalcHEP very flexible respect to installations of new models.

Calchep is based on method of symbolic calculation of squared diagrams. Calculated diagrams are transformed into C-code. Intermediate symbolic results can be presented in form of REDUCE of MATHEMATICA which allows a simple testing of the program.

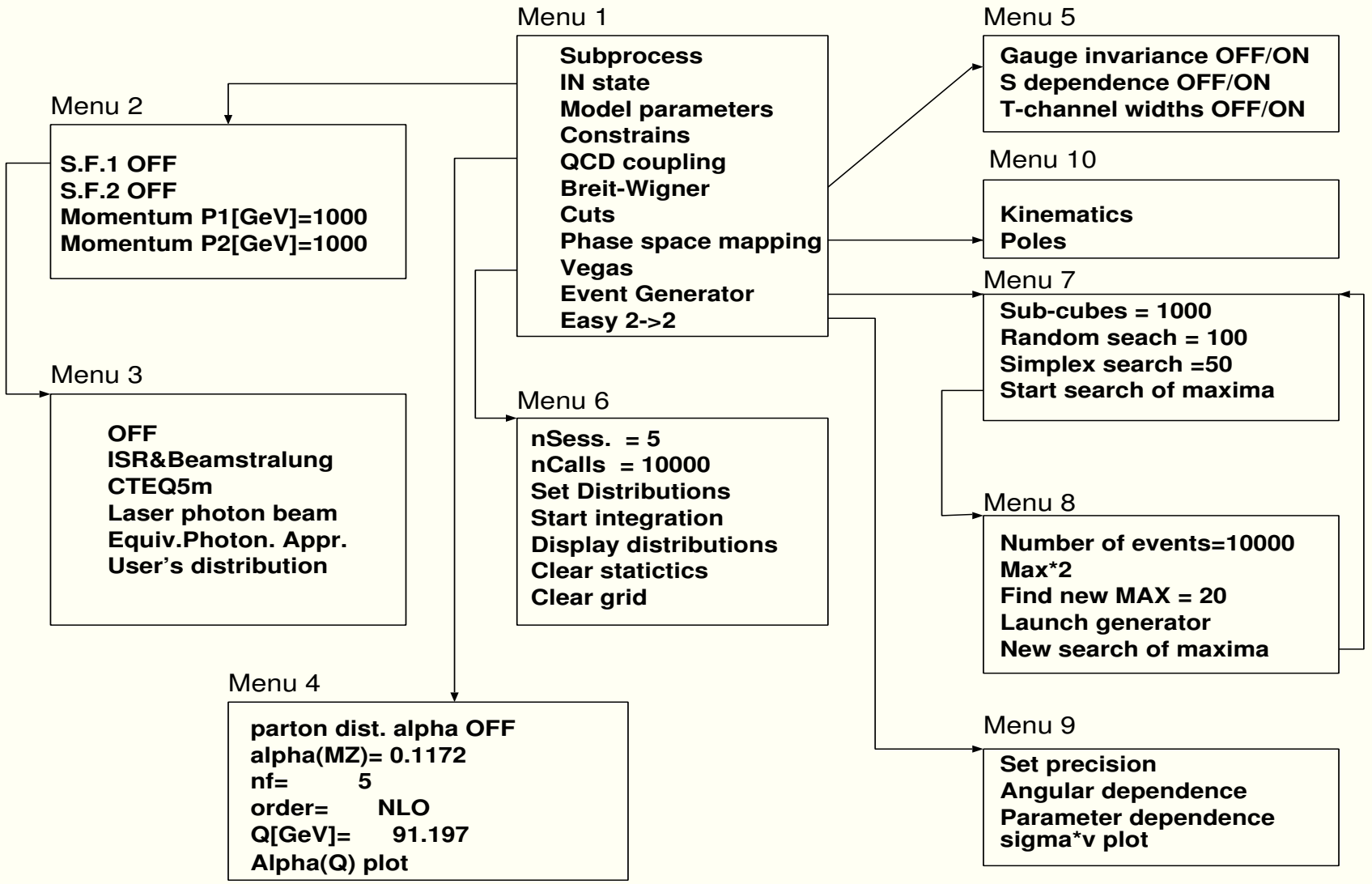
`http:theory.sinp.msu.ru/~pukhov/calchep.htmp`
`http:www.ifh.de/~pukhov/calchep.htmp`

Symbolic part

General scheme of symbolic part is presented below:



C code for matrix elements are compiled and linked with external functions. CalcHEP gives the user some standard tools to for numerical session:



Model structure.

The model is presented by four files:

Variables

Name	Value	Comment
alfEMZ	0.0078180608	MS-BAR electromagnetic alpha(MZ)
alfSMZ	0.1172	Strong coupling constant
SW	0.481	MS-BAR sin of the Weinberg angle
MZ	91.1884	Z mass
Ml	1.777	mass of tau lepton
MbMb	4.23	b-quark mass
tb	10	Tangent beta
mZero	450	scalar masses at GUT
mGrav	60000	Gravitino mass
sgn	1	sign of mu

```
            Constraints
  Name  |> Expression
 smOk   |saveSM(MbMb,Mtp,SW,alfSMZ,alfEMZ,MZ,Ml)
 ambsOk|suspectAMSB(smOk,mZero,mGrav,tb,sgn)
 *Mh    |Mh(ambsOk)
%wh     |wh(ambsOk)
```

The dummy parameters `smOk` ***and*** `ambsOk` ***are used to show dependences caused by implicit parameters.***

Star before `Mh` ***is a signal that*** `Mh` ***will be attached to C code even if does not contribute to matrix element.***

`suspectAMSB` ***is external function which has to be passed to linker.***

Particles

Full name	P	aP	number	2*spin	mass	width	color	aux
gluon	G	G	21	2	0	0	8	G
photon	A	A	22	2	0	0	1	G
Z boson	Z	Z	23	2	MZ	wZ	1	G
W boson	W+	W-	24	2	MW	wW	1	G
b-quark	b	B	5	1	Mb	0	3	
t-quark	t	T	6	1	Mt	wt	3	
Light Higgs	h	h	25	0	Mh	!wh	1	
Heavy higgs	H	H	35	0	MHH	!wHh	1	

There is a way to force automatic width calculation. "!wh" is a signal that code for `h->2*x` ***will be generated and added to the main matrix element.***

The presence of PDG number as particle attribute simplifies the interface with other packages.

```
                    Vertices
P1 |P2  |P3  |P4| Factor              | dLagrangian/dA(p1)dA(p2)dA(p3)
A  |H+  |H-  |  |-EE                  |m1.p2-m1.p3
A  |W+  |W-  |  |-EE                  |m1.p3*m2.m3-m2.p3*m1.m3- ...
B  |b   |A   |  | EE/3                |G(m3)
B  |b   |G   |  | GG                  |G(m3)
G  |G   |G.t |  |GG/Sqrt2             |m1.M3*m2.m3-m1.m3*m2.M3
T  |b   |W+  |  |EE/2/Sqrt2/SW        |G(m3)*(1-G5)
T  |b   |W+.f|  |-i*EE/2/Sqrt2/MW/SW|Mb*(1+G5)-Mt  *(1-G5)
A.C|W-.c|W+  |  |-EE                  |p1.m3
```

G.t - point-like auxiliary tensor particles for 4-gluon vertex.

A.c/A.C - Faddev Popov ghosts

W+.f - goldstone

Generation of model files by Lanhep.

Lanhep package written by Andrei Semenov

A. Semenov. Nucl.Inst.\&Meth. A393 (1997) p. 293.
A.~Semenov, preprint LAPTH-884/01, 2001 (hep-ph0208011).
http://theory.sinp.msu.ru/~semenov/lanhep.html

solves the problem of generation of CalcHEP tables. Lahep input is easy readable files. The package performs *let* ***substitutions,*** *where* ***cycles, dummy indexes convolution, work with fields multiplets, check of generated mass matrix, and recognizes appearance of mixing matrices.***

For example, in MSSM

```
%% SU(2) DD terms
let s_q1  = {~uL,~dL }, s_Q1=anti(s_q1).

let a1=g*s_Q1*tau*s_q1/2,
    a2=g*s_Q2*tau*s_q2/2,
    a3=g*s_Q3*tau*s_q3/2,
    a4=g*s_L1*tau*s_l1/2,
    a5=g*s_L2*tau*s_l2/2,
    a6=g*s_L3*tau*s_l3/2,
    a7=g*s_H1*tau*s_h1/2,
    a8=g*s_H2*tau*s_h2/2.

lterm  -(a1+a2+a3+a4+a5+a6+a7+a8) ** 2 / 2.
```

FF terms coming from superpotential:

```
lterm ( -df(superW,Ai,Aj)*Fi*Fj/2 + AddHermConj
where
 Ai=s_h1,Fi=f_h1; Ai=s_h2,Fi=f_h2; Ai=s_n, Fi=f_N;
 Ai=s_l1,Fi=f_l1; Ai=s_l2,Fi=f_l2; Ai=s_l3,Fi=f_l3;
 Ai=s_r1,Fi=f_r1; Ai=s_r2,Fi=f_r2; Ai=s_r3,Fi=f_r3;
 Ai=s_q1,Fi=f_q1; Ai=s_q2,Fi=f_q2; Ai=s_q3,Fi=f_q3;
 Ai=s_u1,Fi=f_u1; Ai=s_u2,Fi=f_u2; Ai=s_u3,Fi=f_u3;
 Ai=s_d1,Fi=f_d1; Ai=s_d2,Fi=f_d2; Ai=s_d3,Fi=f_d3)
where
 Aj=s_h1,Fj=f_h1; Aj=s_h2,Fj=f_h2; Aj=s_n, Fj=f_N;
 Aj=s_l1,Fj=f_l1; Aj=s_l2,Fj=f_l2; Aj=s_l3,Fj=f_l3;
 Aj=s_r1,Fj=f_r1; Aj=s_r2,Fj=f_r2; Aj=s_r3,Fj=f_r3;
 Aj=s_q1,Fj=f_q1; Aj=s_q2,Fj=f_q2; Aj=s_q3,Fj=f_q3;
 Aj=s_u1,Fj=f_u1; Aj=s_u2,Fj=f_u2; Aj=s_u3,Fj=f_u3;
 Aj=s_d1,Fj=f_d1; Aj=s_d2,Fj=f_d2; Aj=s_d3,Fj=f_d3.
```

A very usefull tool.

Batchs calculation

Initially CalcHEP was desined for work in interactive sessions. Later on it was recognized that batch calculations also are needed. This problem was solved by the following trick.

In the end of interactive session launched with option "+blind", for example,

```
calchep  +blind
```

program writes on the screen the sequence of pressed keys. Let we enter the process `e,E->W+,W-`*, perform symbolic calculation and write C code for matrix elements. Then output should be*

```
{{e,E->W+,W-{{[{[{{{{0
```

In principle, one can decode it,

```
{ - Enter ;      [ - Down key
```

To force CalcHEP to repeat this session one can call

```
bin/s_calchep -blind "{{e,E->W+,W-{{[{[{{{{0"
```

This command can be embedded into some sprint with `$1` *(script argument) instead of* `e,E->W+,W-`*. It gives us a tool to generate codes for different processes.*

The same trick works for numerical sessions too. It allows to organize cycles over parameters.

Thanks to authors of LHCFast for demonstrations of these facilities!

Generation of libraries.

Suppose we have to create a library of matrix elements.

The names of global functions initially generated have suffix `_ext`***. Replacing this suffix by the*** `sed` ***command we can collect in one project several codes generated separately. To simplify interface routine all global functions and variables are attached to one structure***

```
struct CalcHEP_interface
{
  int nvar, nfunc;       // number of variables and functions
  char ** varName;       // names of parameters
  double* va;            // values of parameters
  int nin, nout, nprc;   // numbers on in-partiucles, out-particle,subprocesses
  char* (*pinf)(int, int , double*,long * PDG); // information about particles
  int (*calcFunc)(void);  double * BWrange;     // calculation of function
  double (*sqme)(int , double*, int*);          // calculation SQME
  .....
} interface_ext;
```

Shared libraries, dynamic loading.

New library of matrix element can be compiled as** shared **and loaded in runtime. This trick is widely used in micrOMEGAs package, because we don't know a'priori which co-annihilation channels are contribute to Dark Matter. Also we give micrOMEGAS users a possibility to accompany evaluation of Dart Matter with calculation of other cross sections. It looks like:

```
cc=newProcess("e,E->2*x","eE_2x");
procInfo1(cc,&ntot,&nin,&nout);
assignVal("tb",10.);
procInfo2(cc,nsub,names,masses);
cs= cs22(cc,nsub,Pcm,cosmin,cosmax,&err);
```

This scheme can be used for cross section calculation in any other project.

micrOMEGAs_2.0, Dark Matter calculation.

micrOMEGAs is a project for Dark Matter calculation based on CalcHEP models. In the beginning it was MSSM.

G.~Belanger, F.~Boudjema, A.~Pukhov, A.~Semenov, hep-ph/0405253

New version micrOMEGAs_2.0 is intended for Dark Matter calculation in any model realized in CalcHEP. It is assumed that particles can be separated on *odd* ***and*** *even* ***and that such parity is conserved. Then in general the lightest*** *odd* ***particle in stable and can be considered as a candidate of Dark Matter.***

The micrOMEGAs_2 directory contents

```
calchep\  sources\  MSSM\  NMSSM\  micro_make*    README
```

The command

```
micro_make <new_project>
```

creates template of directory for new project with all needed structure inside. The user has to add the model in CalcHEP notations, add library for external functions and specify *odd* ***particles. The list of odd particles can be presented by the user***

```
OddPrtclsSTR OddPrtcls[NODD]=
{/* pname, apname, mass, width, spin*2;,cdim */
 {"~o1","~o1","MNE1","wNE1",1,1}
,{"~o2","~o2","MNE2","wNE2",1,1}
,{"~1+","~1-","MC1","wC1",1,1}
......
}
```

Or will be constructed by default from the particles whose name is started from tilde.

Interface with Pythia.

Interface with Pythia is realized via two Les Houches accords.

`GENERIC USER PROCESS INTERFACE FOR EVENT GENERATORS, hep-ph/0109068`
`SUSY LES HOUCHES ACCORD, hep-ph/0311123`

According to the first one we fill PYTHIA COMMON blocks. Because PDG numbers are presented from the beginning in CalcHEP particle description, no interface problems appear.

On the next step we have to teach PYTHIA to decay BSM particles. It is realized via SLHA and CalcHEP option to calculate `1->2` ***widths and branchings and write down them in SLHA format. This SLHA format. Alternatively the SLHA file generated by spectrum calculator can be used.***

```
DECAY        36     2.79446481E-06   # Lightest pseudoscalar
     6.68305122E-02    2          15        -15   # BR(A_1 -> tau tau)
     5.98727072E-04    2           4         -4   # BR(A_1 -> c cbar)
     9.27202388E-01    2           5         -5   # BR(A_1 -> b bbar)
     0.00000000E+00    2           6         -6   # BR(A_1 -> t tbar)
```

Particle masses are passed via `BLOCK MASS`***. This approach was restricted by SUSY applications until Peter Scands adds to PYTHIA an option to add new particle to the particle list***

```
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)
```

After this implementation we in principle have a tool for automatic calculation from Lagrangian to event generation for BSM physics.

New project for interface with Pythia and other hardonization ge

I am going to accomany CalcHEP event generator with decay generator also based on CalcHEP. The program should automaticaly detect outgoing BSM particles and decay them on SM ones trying subsequently $X \longrightarrow 2 * x$, $X \longrightarrow 3 * x$, *and* $X \longrightarrow 4 * x$ *channels.*

The codes for decays will be generated automatically and in runtime linked to main process like it done in micrOMEGAs.

It also gives a possibility to keep correct momentum distribution for decay particles.

We need a standard way for registration of new particles in hadronization generators.

List of BSM implementations.

During development of micrOMEGAS package we have implemented

MSSM based on SuSpect/Isajet/SoftSUSY/SPHENO.

NMSSM based NMHDecay

MSSM with CP violation based on CPSuperHiggs.

There are realization of extra dimension models in notations of CalcHEP/CompHEP

G. Servant, K. Agashe ... - the LZP model.

K.Matchev ... -the UED model.

The leptoquark model was implemented together with A.Belyaev and C.Leroy, R. Mediyev (hep-ph:/0502067)

The Little Higgs model : A.Birkedal et al `hep-ph/0603077`.