# FDC Project

Bin Gong

Collaborated with J.X. Wang and Y.D. Wang

Institute of High Energy Physics, CAS

June 17, 2016

# Outline

# Brief Introduction

- FDC = Feynman Diagram Calculation
- Purpose: automatic calculation of physical processes
- First developed by Prof. J.X. Wang since 1993
- First version of FDC has been presented at AIHENP93.
- Written in REDUCE and RLISP to generate Fortran Code
- Including some additional parts for certain physical research e.g. FDC-PWA (Partial Wave Analysis application for experimental study)

# REDUCE

- a general-purpose Computer Algebra System geared towards applications in physics.
- written in Portable Standard LISP
- something like Mathematica, FORM and Maple etc.
- open-sourced and freed now (since December 2008)
- user-level language: RLISP
- two modes: algebraic and symbolic

# a simple example of reduce

```
twain@Twains-MacBook:~$ reduce
Loading image file: /Users/twain/reduce-algebra/scripts/../pslbuild/x86_64-mac_unknown_version-darwin1
Reduce (Free PSL version), 12-Dec-2015 ...

1: vector p1,p2,p3,p4;

2: a:=g(l,p1,p2,p3,p4);

a := p1.p2*p3.p4 - p1.p3*p2.p4 + p1.p4*p2.p3

3: share a;

4: symbolic;

nil

5* reval(a);

(plus (times (cons p1 p2) (cons p3 p4)) (minus (times (cons p1 p3) (cons p2 p4))
) (times (cons p1 p4) (cons p2 p3)))

6* bye;

Quitting
twain@Twains-MacBook:~$
```

$\mathrm{Tr}(\hat{p}_1\hat{p}_2\hat{p}_3\hat{p}_4)$ is calculated here, shown in both algebraic and symbolic modes

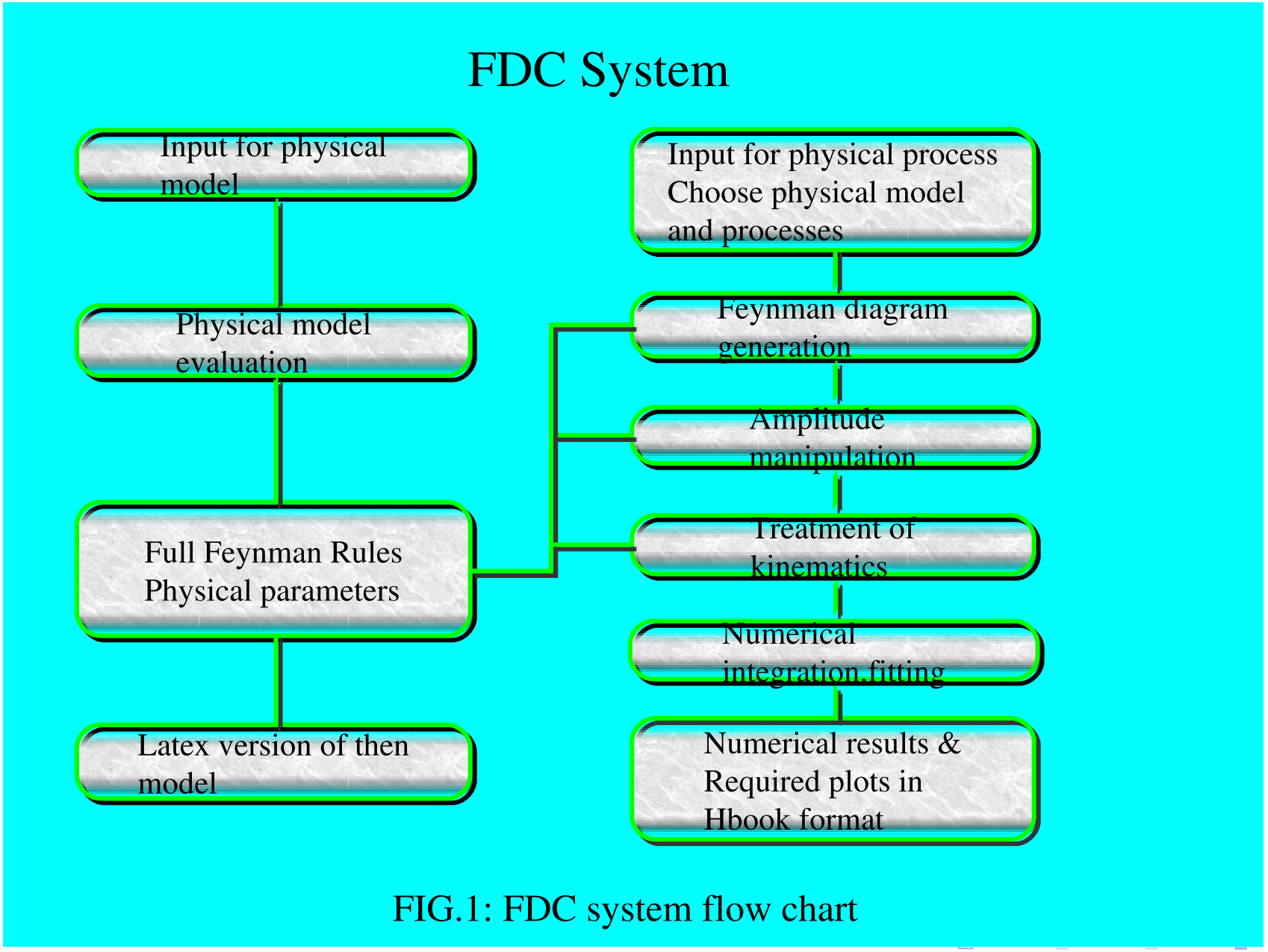


FIG.1: FDC system flow chart

# Current Models

- The Standard Model (SM) has already been constructed.
- The Minimal Supersymmetric Standard Model (MSSM) also has been constructed.
- Also compatible with phenomenological models.
- Most intermediate states and effective vertices from NRQCD has been implemented in the SM.
- QCD counter terms have been manually inserted in the SM.

# Process Calculation at Tree Level

- generate Feynman diagrams (after you choose your model and process)
- manipulate amplitude diagram by diagram (and square it according to your option)
- generate Fortran codes for amplitude or squared amplitude
- output analytic results for (squared) amplitude (LaTeX Format available)
- generate Fortran codes for phase space integration (including special treatment for peaks, BASES)
- cross section integration (run the Fortran codes above)
- parton level event generation (SPRING inside BASES)
- parton shower and hadronization (PYTHIA)

# One-loop Part of FDC

- The one-loop part of FDC is completed in 2007, and upgraded in 2011 to calculate processes involving P-wave particles
- The results are obtained analytically.
  - at the level of amplitude square, before the integration of phase space.
  - usually they are still in numerical form (Fortran codes), as in most cases, they are too complicated to read.
- A two-cutoff phase space slicing method (PSS) [Harris and Owens (2002)] is realized in FDC to deal with IR divergences in real correction processes
- The divergences are factorized in soft/collinear limit, and added to corresponding virtual correction processes.
- All the divergence are separated analytically, and then summed up to check if they are really cancelled with others.

# More on one-loop calculation

- Counter term diagrams are generated automatically (after the input of renormalization constant)
- Loop integrals are calculated analytically under dimensional regularization.
- All the divergence (both UV and IR) are separated during the calculation of amplitude analytically
- In 2007 version, Passarino-Veltman reduction method is used for tensors reduction
- In 2011, new reduction method (a kind of IBP) for loop integrals is realized.
- The cutoff independence has to be checked after summing up both real and virtual corrections.

# Work Done with FDC at one-loop level (mostly in quarkonium physics)

- Quarkonium production in $e^+e^-$ annihilation
  - $J/\psi + \eta_c$, $J/\psi + J/\psi$, $J/\psi + X$
- Quarkonium production and polarization puzzle at hadron colliders.
  - $J/\psi(\Upsilon)$: ${}^3S_1^{[1]}$, ${}^3S_1^{[8]}$, ${}^1S_0^{[8]}$, ${}^3P_J^{[8]}$
  - $\chi_{c(b)}$: ${}^3P_J^{[1]}$, ${}^3S_1^{[8]}$
- Decays
  - $\eta_b \to J/\psi + J/\psi$
- Others

| $n$(intermediate state) | ${}^3S_1^{[1]}$ | ${}^3S_1^{[8]}$ | ${}^1S_0^{[8]}$ | ${}^3P_J^{[8]}$ |
|---|---|---|---|---|
| $gg \to \langle c\bar{c}\rangle_n + g$ | 6/129 | 16/413 | 12/267 | 12/267 |
| $gq \to \langle c\bar{c}\rangle_n + q$ | - | 5/111 | 2/49 | 2/49 |
| $q\bar{q} \to \langle c\bar{c}\rangle_n + g$ | - | 5/111 | 2/49 | 2/49 |
| $gg \to \langle c\bar{c}\rangle_n + gg$ | 60 | 123 | 98 | 98 |
| $gg \to \langle c\bar{c}\rangle_n + q\bar{q}$ | 6 | 36 | 20 | 20 |
| $gq \to \langle c\bar{c}\rangle_n + gq$ | 6 | 36 | 20 | 20 |
| $q\bar{q} \to \langle c\bar{c}\rangle_n + gg$ | 6* | 36 | 20 | 20 |
| $q\bar{q} \to \langle c\bar{c}\rangle_n + q\bar{q}$ | - | 14 | 4 | 4 |
| $q\bar{q} \to \langle c\bar{c}\rangle_n + q'\bar{q}'$ | - | 7 | 2 | 2 |
| $qq \to \langle c\bar{c}\rangle_n + qq$ | - | 14 | 4 | 4 |
| $qq' \to \langle c\bar{c}\rangle_n + qq'$ | - | 7 | 2 | 2 |

Number of diagrams for subprocesses in inclusive $J/\psi(\Upsilon)$ hadroproduction at NLO

# Prerequisites

- A Unix-like system
- REDUCE
  - open source
  - available at http://reduce-algebra.com/downloading.htm
- A Fortran Compiler
- MPI environment if you want to use MPI

# Establish Environments

- environment variables
  - fdc: where you store your FDC source
  - model: where you store your models
  - PATH: you have to tell your OS where to find "reduce" and other commands provided by FDC
- bash: modify .bashrc in your home directory and add:
  - export fdc=~/fdc2.0
  - export model=~/model
  - export PATH=$PATH:/usr/local/bin:$fdc/bin:./
- csh/tcsh: modify .cshrc in your home directory and add:
  - setenv fdc ~/fdc2.0
  - setenv model ~/model
  - set path=($path /usr/local/bin $fdc/bin ./)

# Installation

- Install Reduce (usually psl version)
  - you will get a script called "redpsl" after installation
  - make a new script "reduce" with only two lines:
    redhome="path of your reduce/../pslbuild/..."
    exec $redhome/psl/bpsl -td 1000 -f $redhome/red/reduce.img
    and put it in the directory you choose before
    check /usr/local/bin/reduce in the virtual machine for this step
- Copy FDC source to the directory your chosen above, and run util/xbuild to build fdc source in the source directory.
- Compile Fortran Libraries of FDC and BASES (confirm Fortran compiler).
- Construct/obtain a model

# Particles in the given model:

| *name* | *name in FDC* | *mass* | *mass in FDC* | *width in FDC* | *charge* | *spin* | *cp* |
|---|---|---|---|---|---|---|---|
| $\nu_e$ | *nue* | 0 | 0 | 0 | 0 | 1/2 | *no* |
| $\nu_\mu$ | *numu* | 0 | 0 | 0 | 0 | 1/2 | *no* |
| $\nu_\tau$ | *nut* | 0 | 0 | 0 | 0 | 1/2 | *no* |
| $e^-$ | *ef* | $m_e$ | *fme* | *whe* | $-1$ | 1/2 | *no* |
| $\mu^-$ | *mu* | $m_\mu$ | *fmmu* | *whmu* | $-1$ | 1/2 | *no* |
| $\tau^-$ | *tau* | $m_\tau$ | *fmtau* | *whtau* | $-1$ | 1/2 | *no* |
| $u$ | *qu* | $m_u$ | *fmu* | *whu* | 2/3 | 1/2 | *no* |
| $c$ | *qc* | $m_c$ | *fmc* | *whc* | 2/3 | 1/2 | *no* |
| $t$ | *qt* | $m_t$ | *fmt* | *wht* | 2/3 | 1/2 | *no* |
| $d$ | *qd* | $m_d$ | *fmd* | *whd* | $-1/3$ | 1/2 | *no* |
| $s$ | *qs* | $m_s$ | *fms* | *whs* | $-1/3$ | 1/2 | *no* |
| $b$ | *qb* | $m_b$ | *fmb* | *whb* | $-1/3$ | 1/2 | *no* |
| $\gamma$ | *p* | 0 | 0 | 0 | 0 | 1 | *no* |
| $Z^0$ | *z* | $m_{Z^0}$ | *zm* | *wh* | 0 | 1 | *no* |
| $w^+$ | *w* | $m_w$ | *wm* | *wh* | 1 | 1 | *no* |
| $g$ | *gs* | 0 | 0 | 0 | 0 | 1 | *no* |
| $h^0$ | *h0* | $m_h$ | *hm* | *wh* | 0 | 0 | *no* |
| $g_g$ | *gsg* | 0 | 0 | 0 | 0 | 0 | *no* |

# Calculation

- create a directory for the process use process_cp
- modify "process.def" and "option" files in the directory to specific
  - physical model of your process
  - incoming and outgoing particles
  - order of result
  - way to obtain squared amplitude
  - some others...
- use doall to perform the following:
  - gen_diag: generate diagrams of the process (psdraw)
  - amp: manipulate square of amplitude and output in Fortran
  - kine: generate code for phase space integration
  - make: compile the Fortran code
- run the Fortran code with int (int2, int3, int4)

# Fortran Codes

- all stored in the directory fort
- makefile
- parameter(1).f: physical parameters
- int.f: main program, need parameters in input.dat
- func.f: phase space
- amps2.f: squared amplitude
  - method 1:
    - amp???.f /ampl???.f: LO/NLO amplitude of corresponding diagram
    - ams??.f: square of LO amplitude
    - amsl??.f square of NLO amplitude with LO amplitude
  - method 2
    - amps20.f: square of LO amplitude
    - amp???.f: square of corresponding NLO diagram with LO amplitude
- xxn2(3,4).f: duplicate of code for divergences (int2-4)

Thanks for your attention!

# construction and triangulation of convex polyhedral cone

- Start from:

$$G_k = C' \int_0^1 \mathrm{d}^{N-1}\alpha \alpha^v U_k^\beta F_k^\gamma \tag{1}$$

- do transformation $\alpha_i = \mathrm{e}^{-y_i}$

$$G_k = C' \int_0^\infty \mathrm{d}^{N-1} y \mathrm{e}^{-v\cdot y} U_k^\beta F_k^\gamma \tag{2}$$

- suppose $\Delta_{bb'}$ is the domain where $\mathrm{e}^{-b(b')\cdot y}$ is maximal

$$\begin{aligned} G_k \quad = \quad & \sum_b \sum_{b'} \int_{\Delta_{bb'}} \mathrm{d}^{N-1} y \mathrm{e}^{-(v+b\beta+b'\gamma)\cdot y} \times \left[ c_b + \sum_{d\neq b} c_d \mathrm{e}^{-(d-b)\cdot y} \right]^\beta \\ & \times \left[ c_{b'} + \sum_{d'\neq b'} c_{d'} \mathrm{e}^{-(d'-b')\cdot y} \right]^\gamma \end{aligned} \tag{3}$$

- $Z_{bb'}$ is a set of vectors, $Z_{bb'} = \{v_i\}$, without zero vector.
- $\Delta_{bb'}$ is a set of all possible $y$ defined by

$$\Delta_{bb'} \equiv \{y | (y, v_i) \geq 0, \forall v_i \in Z_{bb'}\}. \tag{4}$$

- $\mathrm{C}(Z_{bb'})$ is a vector space generated by $Z_{bb'}$

$$\mathrm{C}(Z_{bb'}) \equiv \left\{ \sum_i c_i v_i | c_i \geq 0, v_i \in Z_{bb'} \right\}. \tag{5}$$

  It is also a convex polyhedral cone in $N$-dimensional Euclidean space.
- Dual cone of $\mathrm{C}(Z_{bb'})$ is defined by

$$\mathrm{C}(Z_{bb'})^{\mathrm{V}} \equiv \{y | (y, v_i) \geq 0, \forall v_i \in \mathrm{C}(Z_{bb'})\} = \Delta_{bb'}. \tag{6}$$

- $\mathrm{C}(Z_{bb'})^{\mathrm{V}}$ can also be expressed by all its edges:

$$\mathrm{C}(Z_{bb'})^{\mathrm{V}} = \left\{ \sum_i c_i u_i | c_i \geq 0, u_i \in U_{bb'} \right\}. \tag{7}$$

- finding $\Delta_{bb'} \rightarrow$ finding $U_{bb'}$

- The dual cone is constructed by finding all its edges.
- "positive" and "negative" vectors
- $\forall\ N-1$ linear independent vectors (inequalities) $\rightarrow$ a candidate, judge by $(u, v_i) \geq 0$ for all other $v_i$
- suppose there are $m$ vectors remaining after redundancy removal, total number of possible edges: $C_m^{N-1}$

B. W. Harris and J. F. Owens, Phys. Rev. **D65**, 094032 (2002).