Monte Carlo Methods in High Energy Physics IV

Peter Uwer

Memory consumption Vegas

Full decomposition of the integration volume $[0,1]^d$ into n sub-intervalls for each variable would mean:

$$n^d \times 8Byte = \frac{10^8 \times 8}{1024^3} GByte = 74 GByte$$

one double per cell n=10,d=10

→ this is not done for obvious reasons

(would correspond to multi-dimensional array: $w[n][n][n]\ldots[n]$)

Due to factorization assumption:

$$p(x_1, x_2, \ldots, x_d) \to p_1(x_1) p_2(x_2) \ldots p_n(x_n)$$

variables are binned independent from each other: w[n][d]

$$n \times d \times 8Byte = 800 Byte$$

n=10,d=10

Memory consumption vegas

uwer on pepnote01: /home/uwer/src/fortran/vegas/moch

```
uwer@pepnote01:moch>more vegas.F
#define FLUSHIT
#ifndef MAXD
#define MAXD 16
#endif
#ifndef LOG
#define LOG 6
#endif
C#define KNUTH

***#define MAXD 44
      BLOCK DATA
      implicit none
      double precision S1,S2,S3,S4,S5
      COMMON/RESULT/S1,S2,S3,S4,S5
      double precision CALLS,TI,TSI
      COMMON/BVEG4/CALLS,TI,TSI
C     Make the common block static so that it can be used from C
C     Actually I am not really sure if that is needed...
      save/result/
      save/BVEG4/
      end
C   20/02/90 002201221  MEMBER NAME  VEGAS    (HEAVYQ)      FVS
      SUBROUTINE VEGAS(FXN,ACC,NDIM,NCALL,ITMX,NPRN,IGRAPH)
      implicit none
*      IMPLICIT DOUBLE PRECISION ( A-H,O-Z )
      integer i,j,k,ipr,iaj1,iaj,nd,ndm,npg,ng
      integer ndim,ncall,ndmx, itmx,nprn,igraph,mds
      double precision acc, avgi,chi2a,alph,one
      double precision XO,XN,XND,XJAC,WGT,RELWGT,ti2
      double precision sd, dr,dxg,dv2g,f,f2,fb,f2b
      double precision rc
      integer NDO,IT,NXI(50,MAXD)
      double precision SI,SI2,SWGT,SCHI,XI(50,MAXD),SCALLS
     + D(50,MAXD),DI(50,MAXD)
      COMMON/BVEG2/NDO,IT,SI,SI2,SWGT,SCHI,XI,SCALLS
     + ,D,DI,NXI
      double precision CALLS,TI,TSI
      COMMON/BVEG4/CALLS,TI,TSI

      integer IA(MAXD),KG(MAXD)
      double precision XIN(50),R(50),DX(MAXD),DT(MAXD)
      double precision XL(MAXD),XU(MAXD),QRAN(MAXD),X(MAXD)
```

Code from the stone age…

3x4KByte (d=10), would even fit into first level cache of modern CPU's

Contents

The problem:

Cross sections

Top-quark pair production with an additional jet at the Tevatron

$p_T^{\rm cut}$ [GeV]	cross section [pb] LO	cross section [pb] NLO	charge asymmetry [%] LO	charge asymmetry [%] NLO
20	$1.583(2)^{+0.96}_{-0.55}$	$1.791(1)^{+0.16}_{-0.31}$	$-7.69(4)^{+0.10}_{-0.085}$	$-1.77(5)^{+0.58}_{-0.30}$
30	$0.984(1)^{+0.60}_{-0.34}$	$1.1194(8)^{+0.11}_{-0.20}$	$-8.29(5)^{+0.12}_{-0.085}$	$-2.27(4)^{+0.31}_{-0.51}$
40	$0.6632(8)^{+0.41}_{-0.23}$	$0.7504(5)^{+0.072}_{-0.14}$	$-8.72(5)^{+0.13}_{-0.10}$	$-2.73(4)^{+0.35}_{-0.49}$
50	$0.4670(6)^{+0.29}_{-0.17}$	$0.5244(4)^{+0.049}_{-0.096}$	$-8.96(5)^{+0.14}_{-0.11}$	$-3.05(4)^{+0.49}_{-0.39}$

minimum pt of additional jet

central value

uncertainty num. intgration

shift towards μ =2m,m/2

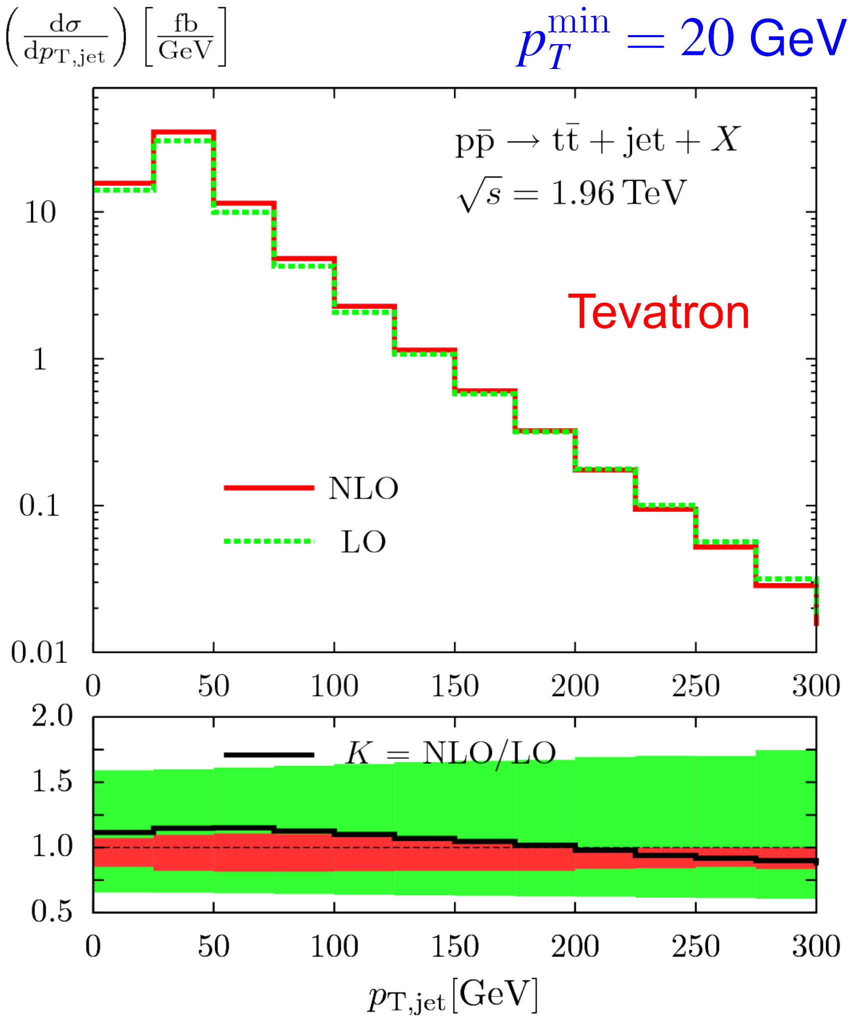
→ total cross sections including cuts and observables

→ total cross sections are difficult to measure not necessarily the best to test theory

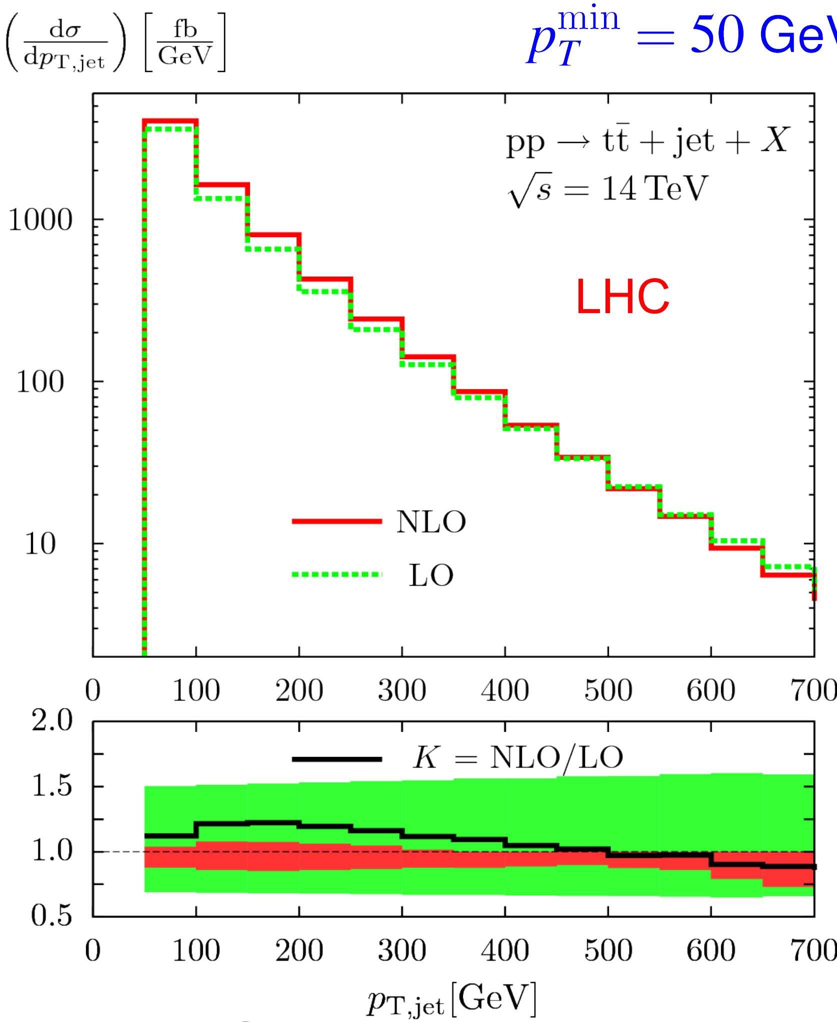
The problem

Distributions

$p_T^{min} = 20$ GeV



$p_T^{min} = 50$ GeV

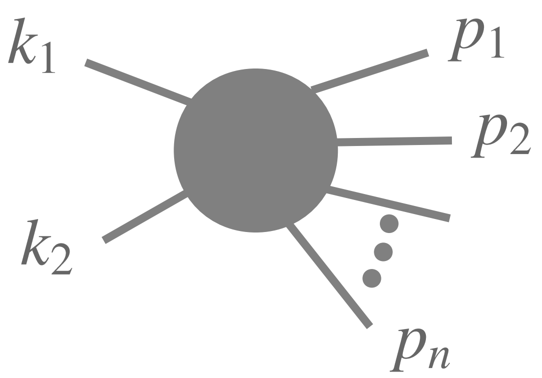


→ more sensitive probe of theory

The problem

We want to calculate

$$\sigma_{cut} = \frac{1}{2s}\int \prod_{i=1}^{n} \frac{d^3\mathbf{p}_i}{(2\pi)^3 2E_i}\delta(k_1 - k_2 - \sum_{i=1}^{n} p_i)|\mathcal{T}_{fi}|^2 \theta_{cut}(p_1,\ldots,p_n)$$



number of independent variables $3n-4$

without cuts simple for n=2,3

For n=3 with cuts i.e. Durham-Jetalgorithm, already non-trivial

$$\Theta_{\text{Durham}}(p_1,\ldots,p_n) = \prod_{i<j}\Theta\left(\frac{\max(E_i^2,E_j^2)}{s}(1-cos\theta_{ij}) - y_{cut}\right)$$

The problem

In addition we also want to calculate distributions

$$\frac{d\sigma}{dO} = \frac{1}{2s}\int\prod_{i=1}^{n}\frac{d^3\mathbf{p}_i}{(2\pi)^3 2E_i}\delta(k_1-k_2-\sum_{i=1}^{n}p_i)|\mathcal{T}_{fi}|^2\theta_{cut}(p_1,\ldots,p_n)$$
$$\delta(O-O(p_1,\ldots,p_n))$$

Hopeless to solve this integral analytically apart from rather simple cases

→ use Monte Carlo integration

→ we need to identify the integration variables

additional argument for MC:

can easily deal with non-continous or otherwise strange integrands

Hadron colliders

If hadronic collisions are studied we have in addition two integrations over the momentum fractions of the incoming partons:

$$\sigma_{cut}^{Had.} = \int dy_1 \int dy_2 F(y_1,\mu_f) F(y_2,\mu_f) \frac{1}{2 s_{had} y_1 y_2} \int \prod_{i=1}^{n}$$

$$\frac{d^3\mathbf{p}_i}{(2\pi)^3 2E_i} \delta(y_1 k_1 - y_2 k_2 - \sum_{i=1}^{n} p_i) |\mathcal{T}_{fi}(y_1 k_1, y_2 k_2, p_1 \ldots)|^2 \theta_{cut}(p_1, \ldots, p_n)$$

→ just two additional integration variables,
MC integration does not care!

Integration variables: simple cases

2-particle final state:

ϕ, θ Azimuthal and polar angle of one particle, everything else is fixed by momentum conservation

In most cases the matrix elements do not depend on ϕ, → integrate out, only one integration variable

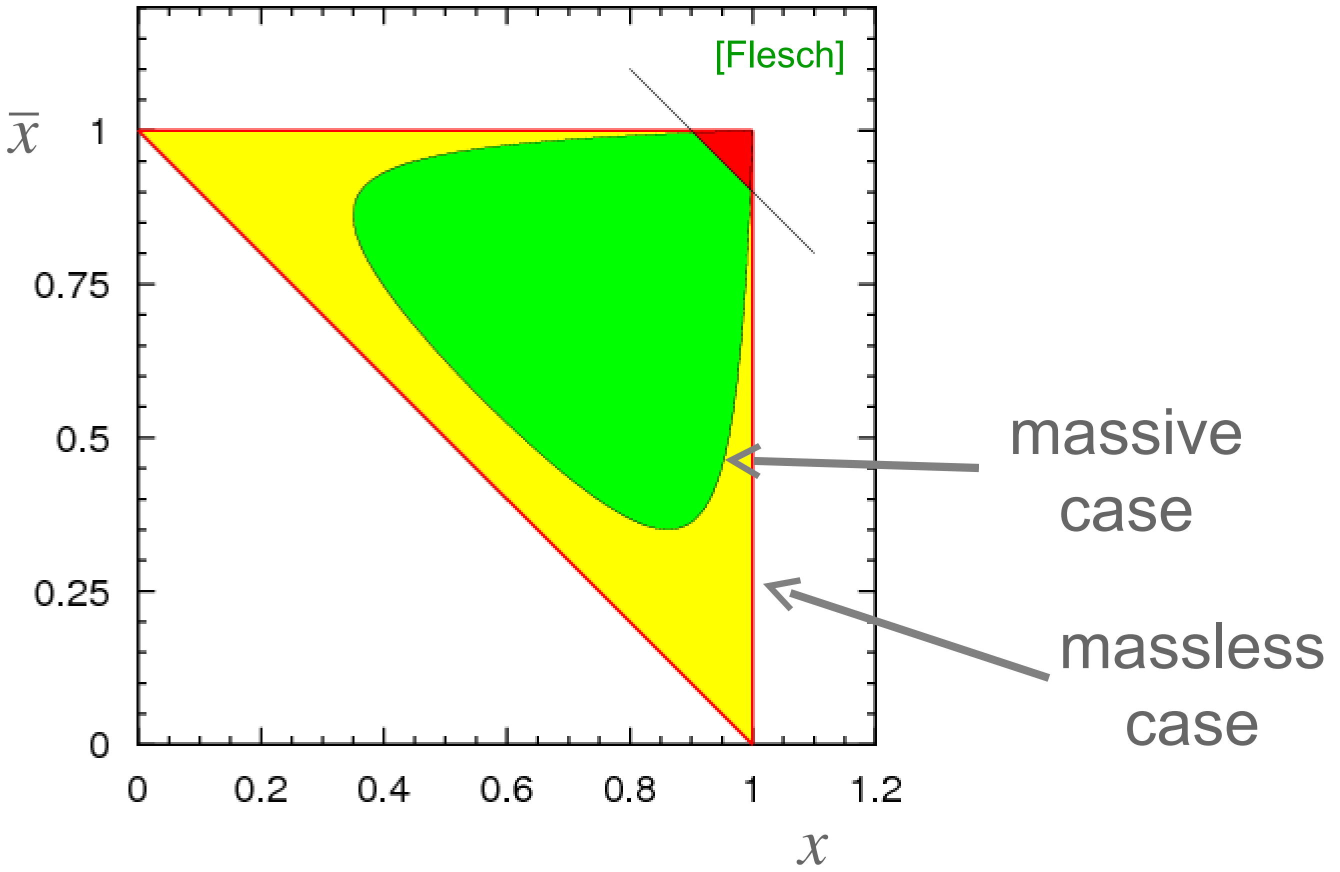
→ integration boundaries straight forward

3-particle final state

ϕ, θ, x_1, x_2 Two angles to describe the orientation of the event plane, $x_i = \frac{2E_i}{\sqrt{s}}$ energies of two outgoing particles

→ integration boundaries not so straight forward anymore

Integration boundaries



→ for the massive case complicated boundaries

Integration variables: simple cases

4 parton final space:

$$\phi, \theta, t_{12}, t_{13}, t_{14}, t_{23}, t_{34}$$

$$t_{ij} = 2p_i \cdot p_j$$

→ need phase space boundaries, and jacobian

→ rather involved phase space boundaries,

→ search for general approach based on MC methods

Phase space integration

If we want to use a simple Monte Carlo integrator we need:

$$[0,1]^{3n-4} \longrightarrow (p_1, p_2, \ldots, p_n)$$

satisfying "on-shell"-condition and momentum conservation

$$p_i^2 = m_i^2, \quad k_1 + k_2 = \sum_i p_i$$

In addition we need the jacobian/weight of the transformation:

$$\prod_i d^3\mathbf{p}_i = \frac{\partial(p_1, \ldots, p_n)}{\partial(x_1, \ldots, x_{3n-4})} dx_1 \ldots dx_{3n-4}$$

Democratic approach to phase space

RAMBO = RA(NDOM) M(OMENTA) B(EAUTIFULLY) O(RGANIZED)

[Ellis (SD), Kleiss, Stirling]

Scetch of the derivation:

Only mass-shell condition, no momentum conservation

Consider: $R_n = \int \prod_{i=1}^{n} d^4 q_i \delta(q_i^2) f(q_i^0) \Theta(q_i^0)$ with $f(x) = \exp(-x)$

Replace q_i by (use delta-functions!):

$$p_i^0 = x(\gamma q_i^0 + \mathbf{b} q_i), \quad p_i = x(\mathbf{q}_i + \mathbf{b} q_i^0 + a(\mathbf{b}\mathbf{q}_i)\mathbf{b})$$

$$\mathbf{b} = -Q/M, \quad x = \sqrt{s}/M, \quad \gamma = Q^0/M = \sqrt{1+b^2},$$

$$a = 1/(1+\gamma), \quad Q^\mu = \sum_{i=1}^{n} q_i^\mu, M = \sqrt{Q^2}$$

→ Integration over $\mathbf{b}$ and x can done

Democratic approach to phase space

The remaining integral in p gives the ordinary phase space measure:

$$\frac{d^3\mathbf{p}}{2E_i^0}$$

$$R = c \times \int \delta(P - \sum_i p_i) \prod_i d^4 p_i \delta(p_i^2) \Theta(p_i^0)$$

constant determined from integral over $\mathbf{b}, x$

Algorithm:

1. Generate the q_i
2. Calculate the p_i from the q_i
   q_i^0 is distributed according to $x\exp(-x)$
   $$q_i^0 = -\ln(u_1 u_2), c_i = 2u_3 - 1, \phi = 2\pi u_4$$
   can also use this to map $[0,1]^{4n} \rightarrow (p_1, \ldots, p_n)$

works with minor modification also for massive momenta

RAMBO

uwer on pepnote01: /home/uwer/src/fortran/rambo

```
uwer@pepnote01:rambo>more rambo-mod.f
C      Modified version of Rambo, instead of creating the
C      random numbers in Rambo, they are passed through an
C      additional Variable RN(4,100).
C
       SUBROUTINE phpoint(N,ET,XM,RN,P,WT)
C-------------------------------------------------------
C
C                          RAMBO
C
C     RA(NDOM)  M(OMENTA)  B(EAUTIFULLY)  O(RGANIZED)
C
C     A DEMOCRATIC MULTI-PARTICLE PHASE SPACE GENERATOR
C     AUTHORS:  S.D. ELLIS,  R. KLEISS,  W.J. STIRLING
C     THIS IS VERSION 1.0 -  WRITTEN BY R. KLEISS
C
C     N  = NUMBER OF PARTICLES (>1, IN THIS VERSION <101)
C     ET = TOTAL CENTRE-OF-MASS ENERGY
C     XM = PARTICLE MASSES ( DIM=100 )
C     P  = PARTICLE MOMENTA ( DIM=(4,100) )
C     WT = WEIGHT OF THE EVENT
C
C-------------------------------------------------------
**       IMPLICIT REAL*8(A-H,O-Z)
       implicit none
       double precision xm,rn,p,q,z,r,b,p2,xm2,e,v,xmt
       double precision acc,F0,x,bq,wt,xmax,accu,g0,x2,wt2,wt3,wtm
       double precision twopi,C,Rmas,G,A,F,S,po2log,et
       integer iwarn,ibegin,iter,itmax,k,I,nm,n
       DIMENSION XM(100),RN(4,100), P(4,100),Q(4,100),Z(100),R(4),
     .    B(3),P2(100),XM2(100),E(100),V(100),IWARN(5)
       DATA ACC/1.D-14/,ITMAX/6/,IBEGIN/0/,IWARN/5*0/
C      I added the following line otherwise the variables are
C      not static, for ibegin the statement is not required (P.U.)
       SAVE IBEGIN,TWOPI,PO2LOG,Z

C
C  INITIALIZATION STEP: FACTORIALS FOR THE PHASE SPACE WEIGHT
       IF(IBEGIN.NE.0) GOTO 103
       IBEGIN=1
       TWOPI=8.*DATAN(1.D0)
```

[Kleiss, Ellis, Stirling]

from CERNLIB (?)

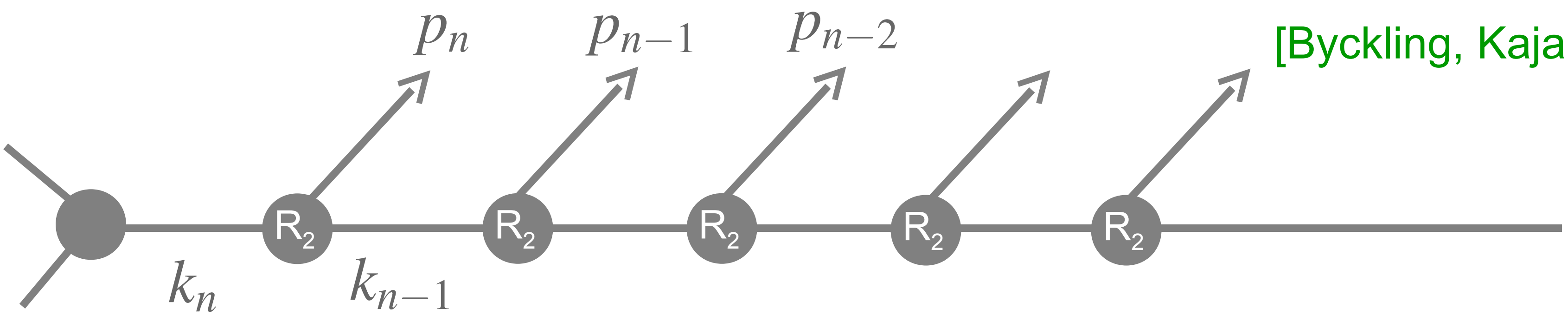
RAMBO

Comments:

- Events have uniform weight in phase space
- Useful for testing purposes
- For real integration not that useful:
  - more integration variables than actually needed
  - Due to complicated mapping vegas unable to optimize
- Useful in constructing multi channel generators

Sequential splitting

[Byckling, Kajantie]



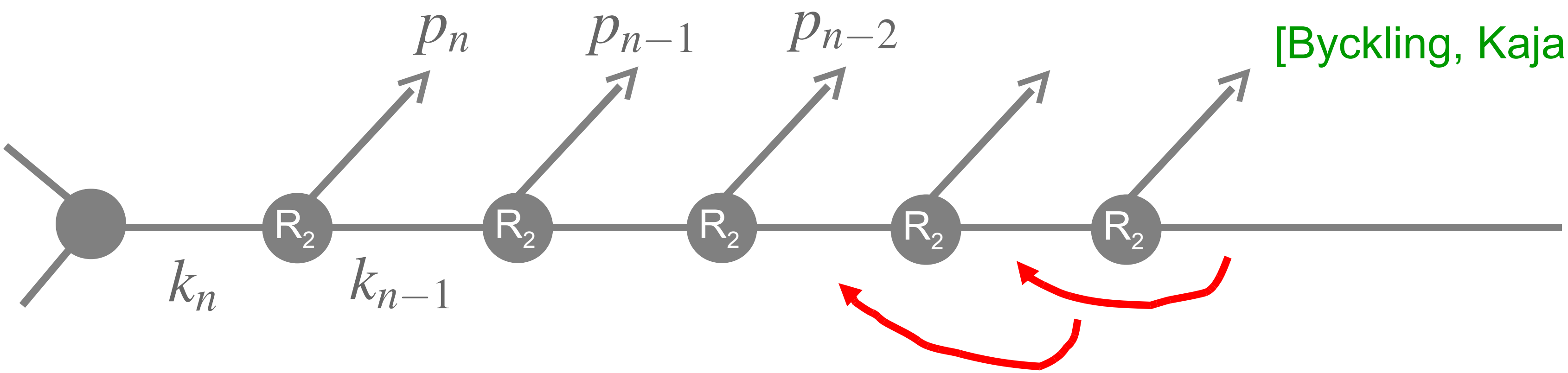
Phase space can be factorized:

$$R_n(M_n^2) = \frac{1}{2M_n}\int_{\mu_{n-1}}^{M_n-m_n} dM_{n-1}d\Omega_{n-1}\frac{1}{2}p_n \dots \int_{\mu_2}^{M_3-m_3} dM_2 d\Omega_2 \frac{1}{2}P_3 \int d\Omega_1 \frac{1}{2}P_2$$

$$M_n^2 = k_n^2, k_i = p_1 + \dots + p_i, \mu_i = m_1 + \dots + m_i$$

$$P_i = \frac{\sqrt{\lambda(M_i^2, M_{i-1}^2, m_i^2)}}{2M_i}$$

Sequential splitting



[Byckling, Kajantie]

Generate angles in the individual restframes,
generate the masses M_i

The momenta are generated in the respective rest frames

Apply boosts to all the momenta to transform them into the same (overall) rest frame (iterative procedure)

method gives mapping $[0,1]^{3n-4} \longrightarrow (p_1, \ldots, p_n)$

$$w \sim \frac{1}{2M_n} \prod_i \frac{1}{2} P_i$$

Sequential splitting

Comments:

- Some freedom in ordering
- Can also be used for direct integration
- Seems to work better than Rambo when combined with Vegas
- Can be adopted to generate soft/collinear configurations
- Possible to combine different orderings

Test of soft/collinear limits of scattering amplitudes

Note:

There is no "one size fits all" general solution to phase integration

RAMBO and sequential splitting should be taken as a starting point, very useful to get a "first" program

Multi-channel

In typical phase space integrals there are usually more problematic variables than integration variables

→ not possible to be good in all problematic variables !

→ multi-channel methods

Define different mappings optimized for specific configurations

Sample/integrate using a weighted sum over the individual mappings

$$\sum_i p_i f_i(\vec{x}, p_1, \ldots, p_n)$$

→ sampling by composition

Multi-channel

Taken to the extreme:

Generate one mapping for each Feynman integral [?]

Combine all channels as it was done for the probability distributions (sampling by composition)

Individual channels can be constructed using sequential splitting, RAMBO

Sarge

In the case of QCD tree-amplitudes where the pole structure is pretty well understood there exist dedicated algorithms

→ Sarge, an algorithm for generating QCD-antennas

[Hameren, Kleiss, Draggiotis]

Note:

In next-to-leading order calculations the situation is different:

We integrate $|\mathcal{T}| - \sum_i \text{Dipoles}_i$ [see Kouhei Hasegawa's talk]

→ behaviour of the combination very different compared to un-subtracted matrix elements

→ no general technique

Distributions

Monte Carlo integrator provides weight and configuration

Possible to calculate (discrete) distributions = histograms at the same time

i.e. $p_\perp, m_{ij} = \sqrt{(p_i + p_j)}$

MC integrator $\xrightarrow{w, (p_1, \ldots, p_n)}$ $d\sigma(p_1, \ldots, p_n), \quad O(p_1, \ldots, p_n)$

fill histogram with $d\sigma \times w$ according to the value of O

$$\rightarrow \quad \frac{d\sigma}{dO}$$

Can also be understood as integrating a vector

modern MC integration packages are usually prepared for that, see i.e. Cuba by Thomas Hahn

Steps towards a full Monte Carlo

Goal: Want to have full simulation as close as possible to nature

i.e. want to have hadronic events which are distributed as in nature

→ we need so called *un-weighted* events in difference from *weighted* ones

in ideal simulation no difference between real and simulated events

→ optimal to test the experimental analysis

If affordable (CPU time!):

Create as many MC events as you expect to observe

From weighted to un-weighted events

For un-weighted events distribution should be according to underlying theory, i.e. matrix elements, parton distribution, …

Events generated in MC integration are *weighted*:

$$w \sim \frac{\partial(p_1,\ldots,p_n)}{\partial(x_1,\ldots,x_{3n-4})}|\mathcal{T}(y_1k_1,y_2k_2,p_1,\ldots,p_n)|^2F(y_1,\mu_f)F(y_2,\mu_f)$$

If the maximum weight w_m is known we can “un-weight” events:

1. For each event generate uniform random number r between 0 and w_m
2. If w(p1,…) < r reject the event otherwise keep the event
3. Give any surviving event the weight 1

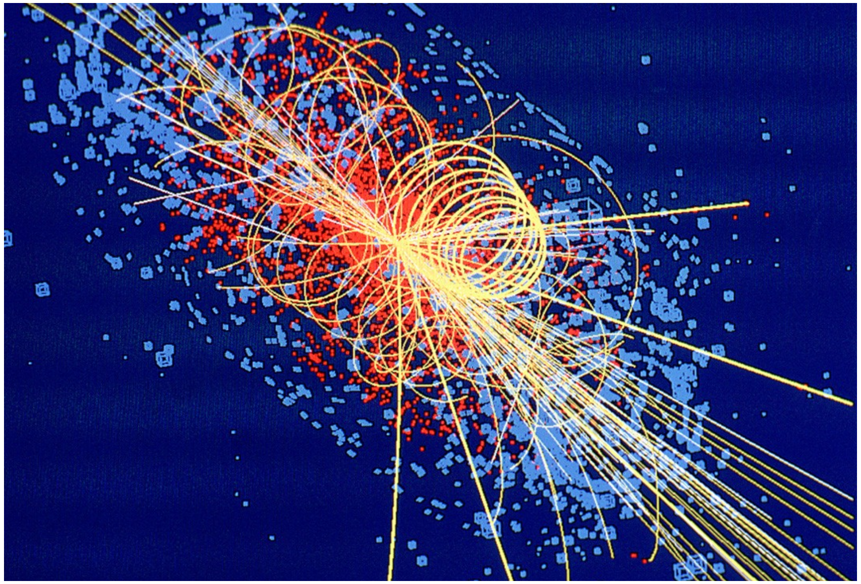
→ hit and miss algorithm

(as far as efficiency is concerned only useful if processing takes much longer then generating)

Event generators

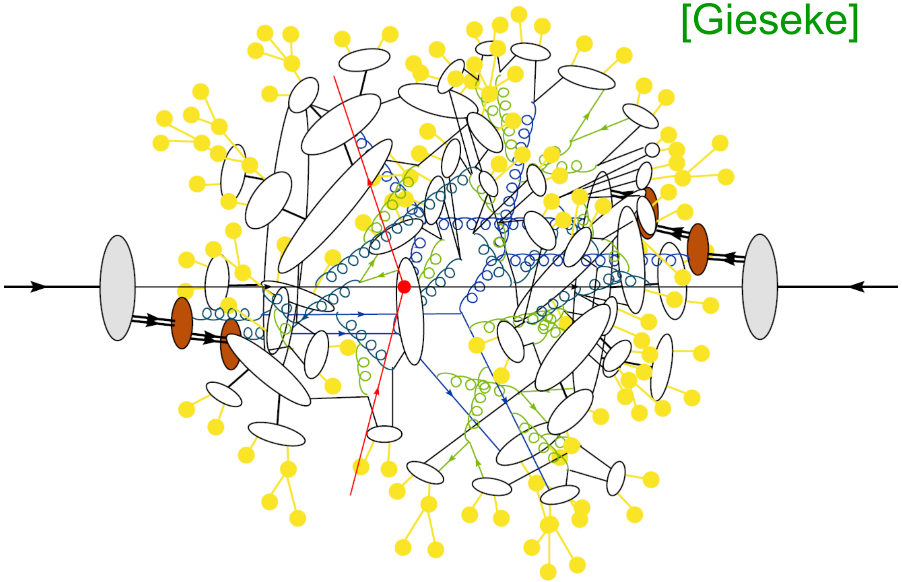
Very important tool in today's experimental analysis

→ everybody should have a rough idea what goes in there



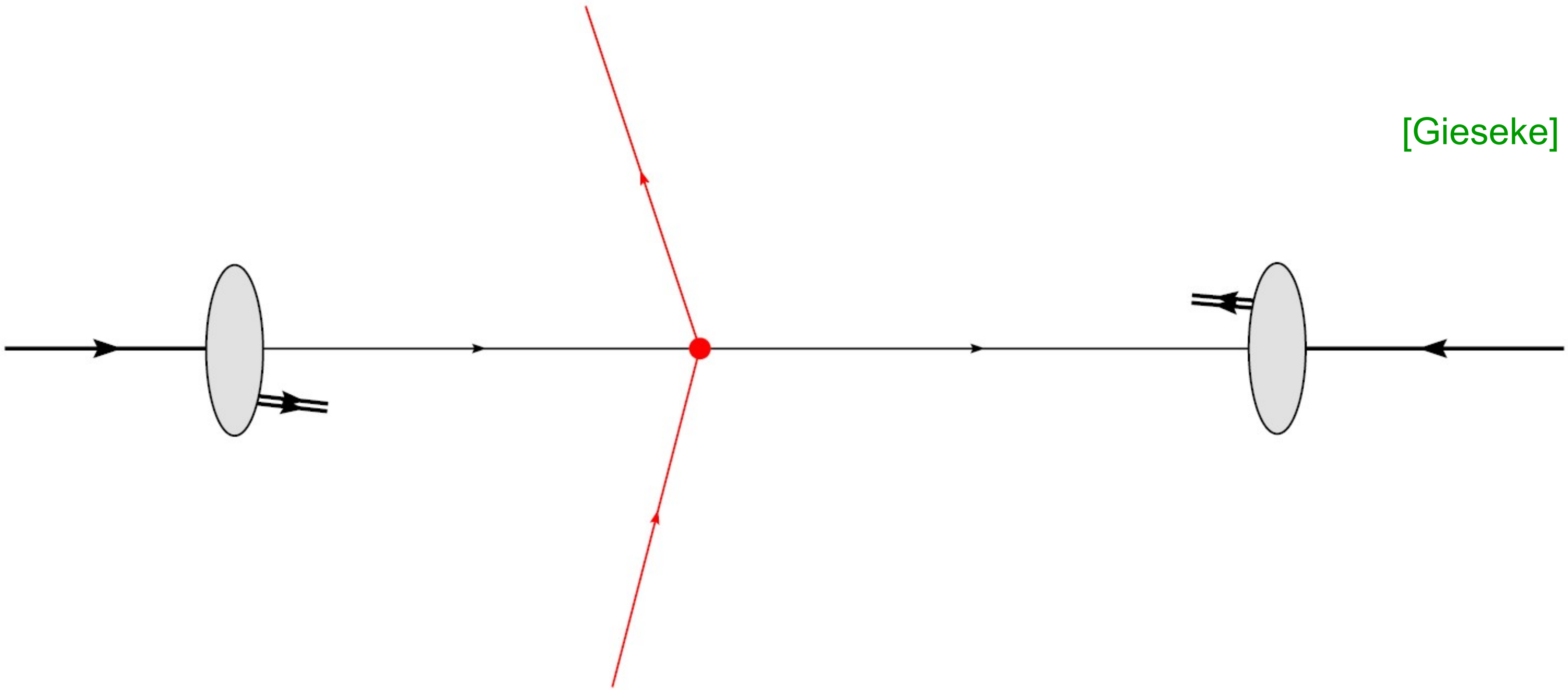
what we might see at the LHC

Higgs event



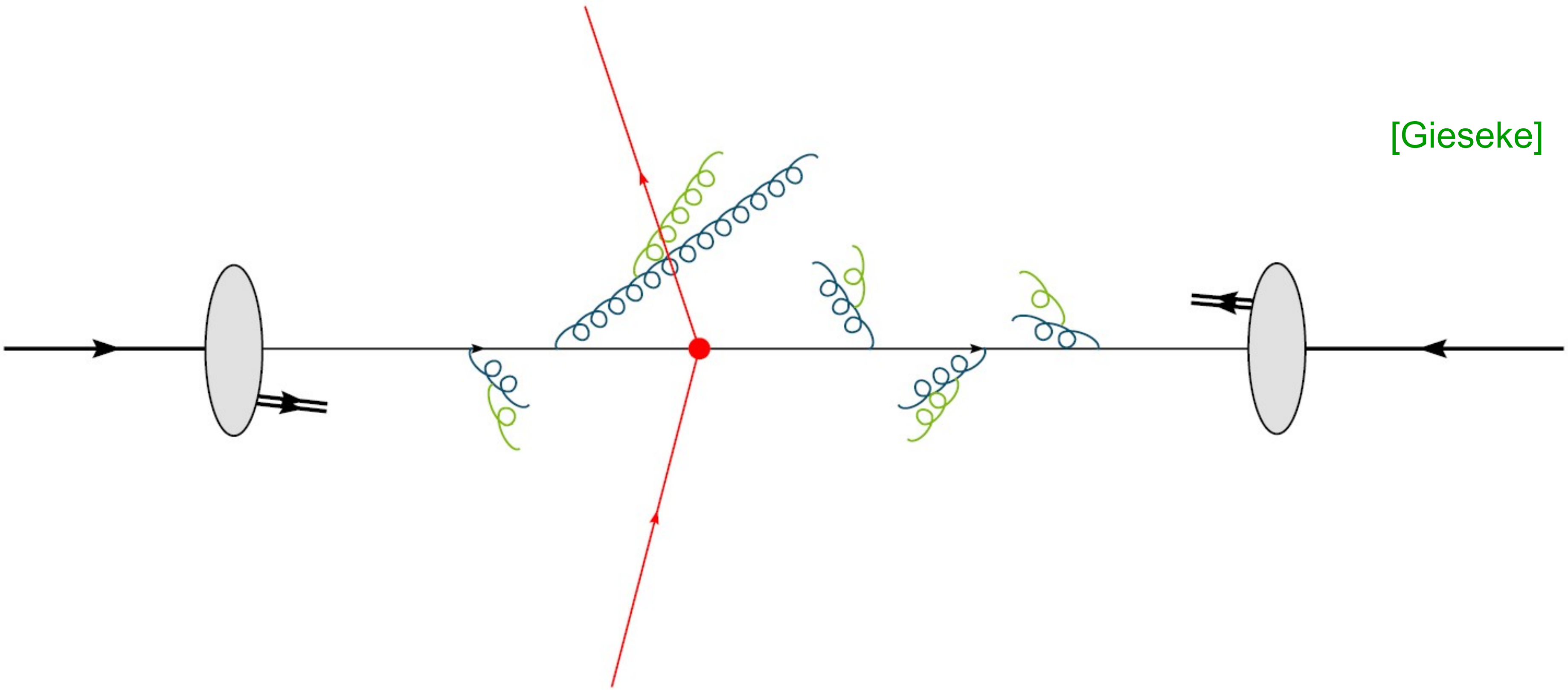
how we understand it

Event generators – a closer look



[Gieseke]

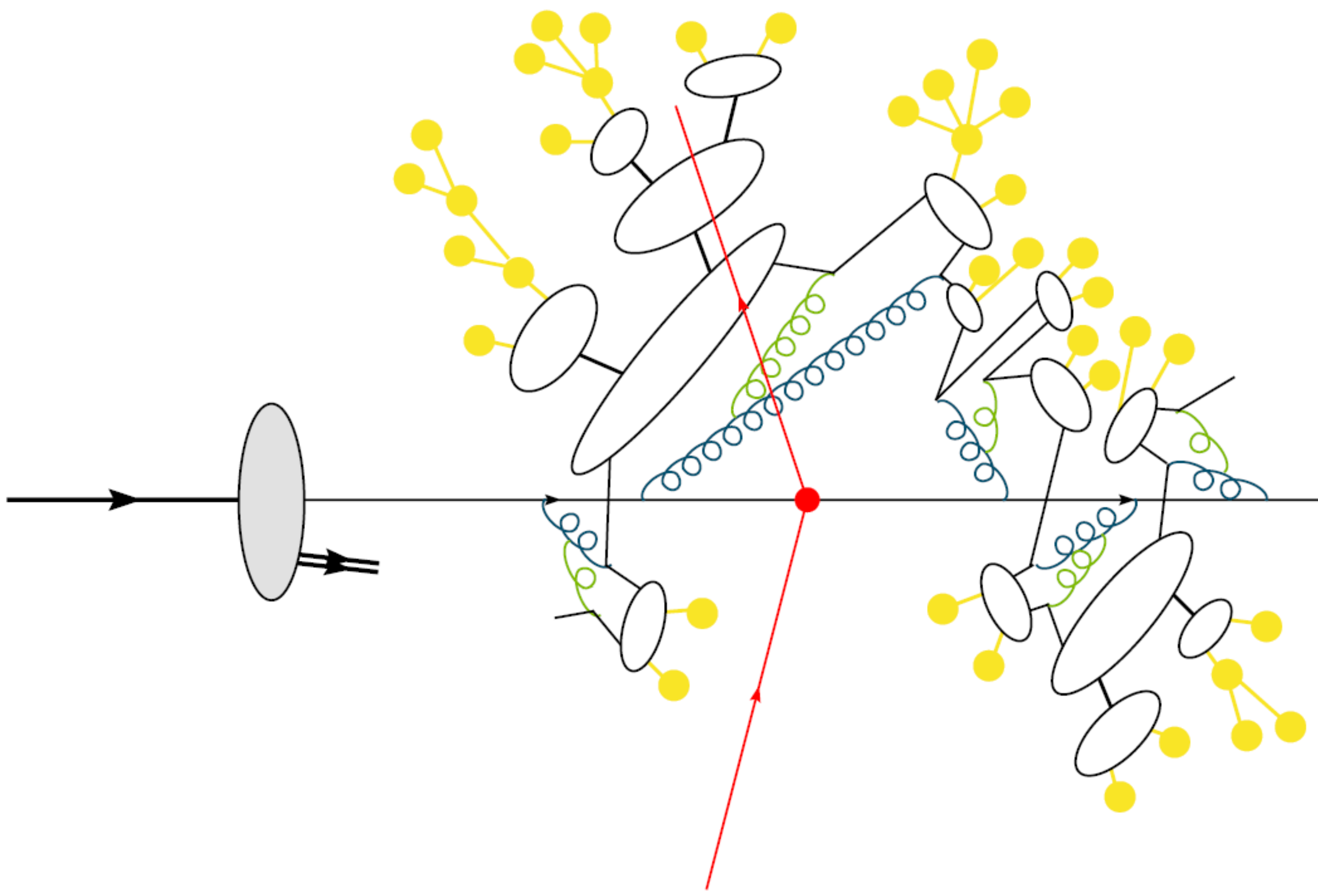
Event generators – a closer look



[Gieseke]

Parton shower

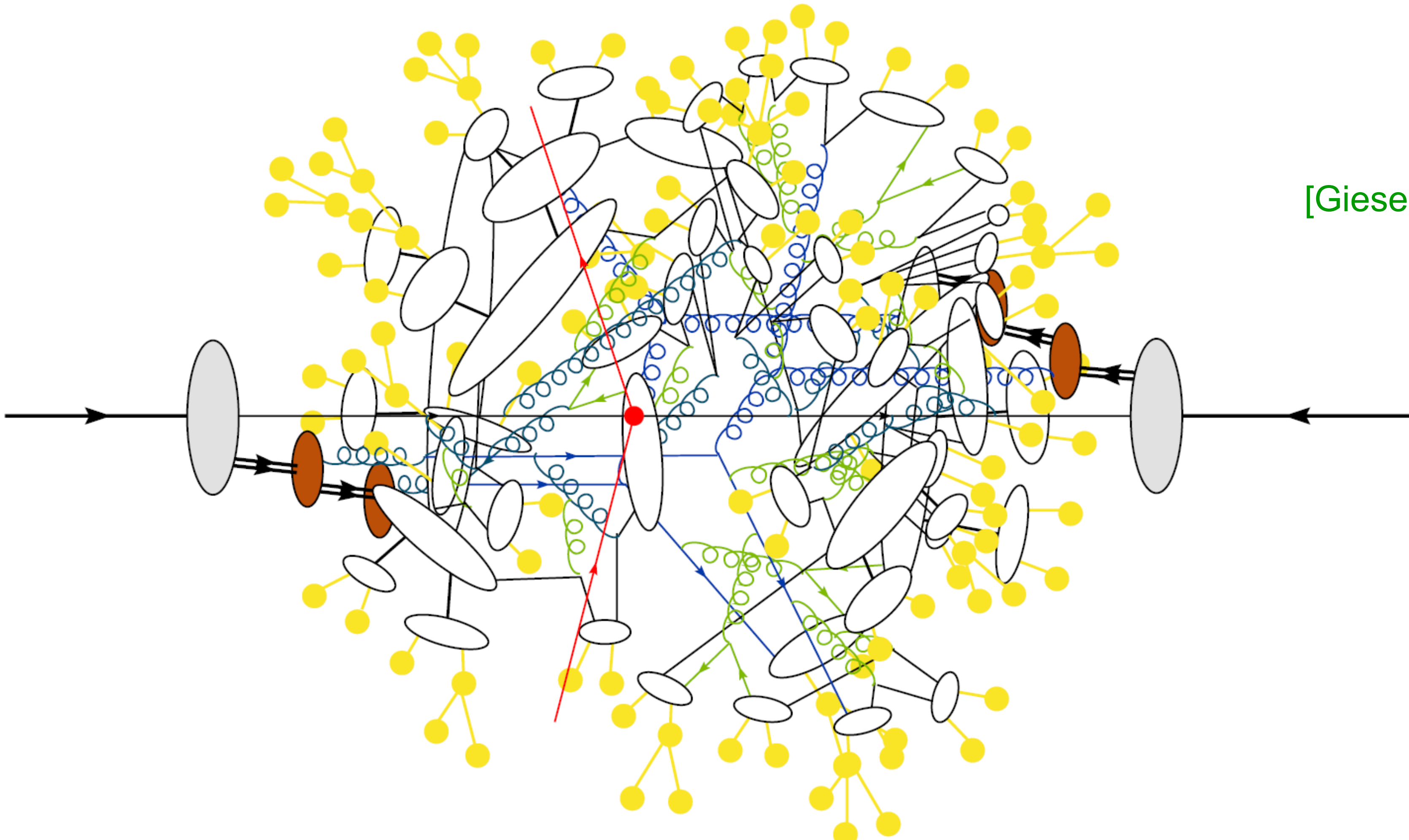
Event generators – a closer look



[Gieseke]

Hadronisation

Event generators – a closer look



[Gieseke]

underlying event

Hadronic cross sections

[Gieseke]

$$d\sigma = d\sigma_{\text{hard}} dP(\text{partons} \to \text{hadrons})$$

partonic
cross section

$$\int dP(\text{partons} \to \text{hadrons}) = 1 \,,$$

$$\begin{aligned} dP(\text{partons} \to \text{hadrons}) = \; & dP(\text{resonance decays}) && [\Gamma > Q_0] \\ & \times dP(\text{parton shower}) && [\text{TeV} \to Q_0] \\ & \times dP(\text{hadronisation}) && [\sim Q_0] \\ & \times dP(\text{hadronic decays}) && [O(\text{MeV})] \end{aligned}$$

Complex simulation → Herwig, Pythia

Literature

[1] S. Ermakow. *Die Monte-Carlo-Methode und verwandte Fragen*. VEB Deutscher Verlag der Wissenschaften, 1975.

[2] T. Fliessbach. *Statistische Physik*. Wissenschaftsverlag, 1993.

[3] W. Gibbs. *Computation in Modern Physics*. World Scientific, 1994.

[4] J. Hammersley and D. Handscomb. *Monte Carlo Methods*. Chapman & Hall, 1992.

[5] F. James. MONTE CARLO THEORY AND PRACTICE. *Rept. Prog. Phys.*, 43:1145, 1980.

[6] M. Kalos and P. Whitlock. *Monte Carlo Methods, Volume I: Basics*. John Wiley & Sons, 1986.

[7] W. Kinzel and G. Reents. *Physics by Computer*. Springer, 1998.

[8] D. Knuth. *The art of computer programming, Volume 2 Seminumerical Algorithms*. Addison Wesley, 2008.

[9] M. Kolonko. *Stochastische Simulation*. Vieweg+Teubner, 2008.

[10] J. Monathan. *Numerical Methods of Statistics*. Cambridge University Press, 2001.

[11] S. Weinzierl. Introduction to Monte Carlo methods. *hep-ph/0006269v1*.

[12] U. Wolff. *Computational Physics II, Vorlesungsskipt*.

The End