The EVENT2 v0.4 Program Manual

Original code by S.Catani and M.H.Seymour, edited by G.Chiurato

May 2026

Abstract

This is a collection of notes about the `EVENT2` program, which implements the Catani-Seymour algorithm for next-to-leading order corrections to two- and three-jet event observables in e^+e^- annihilation. It is not intended to describe the algorithm itself, which will one day be written up. It will hopefully evolve into proper program documentation at some stage though.

Overview

The program is written somewhat like an event generator — a main routine administers the generation of parton momenta, calls the matrix element routines to calculate their weight, and finally calls a user routine to analyse them, with the momenta stored in something vaguely resembling an event record. At present, the program is designed just to calculate the perturbative coefficients themselves, and does not even contain an α_S calculation, although of course it would be trivial to add this.

The most important point to note about the algorithm is that the singularities of the m-parton real matrix element are cancelled by subtracting several ($\sim m(m-1)(m-2)/2$) $m-1$-parton counterterms. These are weighted such that when the m-parton term diverges, so do one or more $m-1$-parton terms, so that their diverence stays finite. Therefore an 'event' consists of many separate sets of momenta with coupled weights, and it is important that the user routine takes this into account. If one is not interested in the errors on the calculated quantities this is trivial — events can be directly histogrammed. However, if error bars are needed it is important that all the sets of momenta are entered into the histogram in one go at the end of the event, so that large positive and negative weights have been cancelled already.

Phase-space generation

Although we said that the remaining cross-section is finite, it is not actually. It is guaranteed to have finite integral, but can (and does in general) contain integrable singularities, most importantly square-root singularities. These are notorious for spoiling the convergence of Monte Carlo integrals (their variance is infinite). They must therefore be removed by suitable Jacobian factors. In `EVENT2` this is done by using a fairly sophisticated multi-channel system.

The phase-space generation looks at first sight something like a parton shower — first a two-parton event is generated, then, using this as a starting point a three-parton event is generated by choosing one of the partons to split into two while the other absorbs its recoil so that all three are on mass-shell. Finally a four-parton event is generated in the same way, choosing one of the two remaining partons to absorb the recoil. However, unlike most parton showers, this can be decribed as a proper transformation of the full four-parton phase-space such that the Jacobian factor can be easily calculated. Different choices of which parton emits and which absorbs the recoil correspond to the different channels of the multi-channel integration. In principle, the convergence might be improved by assigning different weights to different channels. Algorithms even exist to optimize this

automatically, but in practice we find that the convergence is perfectly satisfactory simply assigning the same weight to all channels. As usual in multi-channel Monte Carlo, the Jacobian factor is the sum of the factors for all channels, not just the one that was used to generate the event.

The square-root singularities are removed by ensuring that for each variable x in which the full matrix element is singular as $1/x$ (i.e. two-parton invariant mass-squareds and parton energies), x is generated according to

$$\frac{dx}{x^{1-1/n}},$$

where n is an adjustable parameter. Clearly $n = 1$ gives flat phase-space, $n = 2$ removes square-root singularities, while even larger n pays even more attention to the almost-singular regions. Obviously the appropriate Jacobian factors are applied so that the final result does not depend on the value of n, only the size of the errors does. There are actually two adjustable parameters, `NPOW1` and `NPOW2`, both defaulting to 2, which control the $2 \to 3$ and $3 \to 4$ parton steps respectively. For most studies the default values are sufficient, but if for example one was studying the next-to-leading order corrections to a three-jet quantity like thrust in the extreme two-jet-like region, increasing `NPOW1` would improve the convergence. For technical reasons (certain constants are tabulated rather than being calculated from scratch) `NPOW1` and `NPOW2` are integer parameters.

The two-parton event is chosen according to the distribution

$$d\cos\theta\,\texttt{COSP}(1+\cos\theta)^2 + \texttt{COSM}(1-\cos\vartheta)^2 + \texttt{COSZ},$$

where $\cos\theta$ is the angle between the quark and the electron[1], and `COSP`, `COSM` and `COSZ` are adjustable parameters. Since the matrix elements are not very forward-backward peaked, the exact values are not too critical, but ideally one should set `COSZ` $= 1$, `COSP/M` $= 0$ for unorientated matrix elements (see below), `COSZ` $= 0$, `COSP/M` $= 1$ (the default) for photon exchange and whatever for the full Z/γ exchange.

Matrix elements

There are three sets of matrix elements included. None of them is perfect at the moment (but we're working on it!). The first is the original `ERT` set (`METYPE` $= 0$), which average over event orientation. Full colour information and most flavour information is available (see below). The second is the set provided for us by the Leiden group (`METYPE` $= 1$). These are fully oriented matrix elements, but at present only include the photon exchange term. They include the same colour and avour information as the `ERT` set. The final set is the Giele-Glover matrix elements (`METYPE` $= 2$ believe it or not), basically ripped out of their `EERAD` program but with some modifications to use the `ERT` definitions of the pole terms and a few others associated with summing over parton permutations. No colour or flavour information is available, for technical rather than fundamental reasons. In the long term this is really intended as a cross-check that our code is working rather than as a serious set.

Flavour information

For the four-parton matrix element the weight given to the user is always the sum of the contributions from $q\bar{q}gg$, $qq\bar{q}\bar{q}$ and $qq'\bar{q}\bar{q}'$. The momenta are given in the 'event record' in this order. If the cross-sections are needed for the individual processes, it is available (when using the `ERT` or Leiden matrix elements) from the variables `GGSUB`, `QQSUB` and `QPSUB` respectively, which hold the fraction of the current weight contributed by each subprocess. However, in the four-fermion case this information should be interpreted with a little care, as both sets of matrix elements neglect interference terms from which the contribution would be zero if quarks and antiquarks were averaged over, but which

[1] The truth is that I haven't checked the parity-violating parts of the code much yet, so it is possible that $\pm\cos\theta$ could be accidently interchanged.

can be non-zero if both quark-antiquark pairs are distinguished. It is hard to imagine many physical observables for which this could be a problem (baryon-baryon (as opposed to baryon-antibaryon) correlations?). Nevertheless I think we should try to include these terms in the Leiden code, just to give a complete description of the process.

Colour information

In the `ERT` and Leiden matrix elements, the variables `CFSUB`, `CASUB` and `TFSUB` give the relative contributions to the current weight from the C_F, C_A and T_f terms respectively. The colour factors themselves are removed, i.e. the normalization is

$$C_F(C_F\texttt{CFSUB} + C_A\texttt{CASUB} + N_f T_R\texttt{TFSUB}) = 1.$$

The main routine

The main routine is called `EVENT2`.

```
SUBROUTINE EVENT2(NEV,EM,NFL,USER)
INTEGER NEV,NFL
DOUBLE PRECISION EM
EXTERNAL USER
```

It takes four arguments,

`NEV` is the number of events to generate,

`EM` is the total centre-of-mass energy,

`NFL` is the number of avours (used both for the total e+ e! qq̄ normalization and in the second-order cross-section),

`USER` is the name of a user-supplied analysis routine.

The program starts by setting various parameters. Most of these are extremely standard and would never need changing, but at some stage perhaps an improved interface in which the default parameters can be modifi ed should probably be provided. The internal parameters are described below.

The user routine

The user must supply a routine to analyze events

```
SUBROUTINE USER(N,NA,ITYPE,P,WEIGHT)
INTEGER N,NA,ITYPE
DOUBLE PRECISION P(4,7),WEIGHT
```

`N` is the number of partons in the 'event record',

`NA` is the order of α_S (i.e. 0, 1 or 2),

`ITYPE` is the type of contribution (shouldn't be needed for physical observables but provided anyway): 0=tree level, 1=subtraction counterterm, 2='finite virtual' term,

P is the primitive 'event record': P$(1-4,i)$ is the momentum of the ith entry. Entries 1-N are the outgoing partons in the order given above for four-parton terms, and $q\bar{q}$ and $q\bar{q}g$ for the two- and three-parton terms. Entry 5 is their total four-momentum. Entries 6 and 7 are the incoming electron and positron.

WEIGHT is the weight of the event.

The weights are normalized such that their total value is the final cross-section. Thus they do not need to be renormalized at the end of the run. At each order in α_S a factor of $\left(\frac{\alpha_S}{2\pi}\right)^{\mathtt{NA}}$ is extracted. Thus about the simplest imagineable user routine would be something like this...

```
      SAVE A,B,C,D
      IF (NA.EQ.0) THEN
        A=A+WEIGHT
      RETURN
      ELSEIF (NA.EQ.1) THEN
        B=B+WEIGHT
      ENDIF

C---CALCULATE THE THRUST OF THE N-PARTON EVENT
     :
     :
      IF (NA.EQ.1) THEN
        C=C+(1-T)*WEIGHT
      ELSEIF (NA.EQ.2) THEN
        D=D+(1-T)*WEIGHT
      ENDIF
```

At the end of the run, A would be equal to the Born cross-section for $e^+e^- \to q\bar{q}$ at the requested energy, B would be equal to 2A, indicating that the cross-section through next-to-leading order is

$$\sigma = A\left(1+\frac{\alpha_S}{2\pi}2\right).$$

C would be equal to 2.10A and D to 45A, indicating that the average value of thrust is given by

$$\langle 1-T\rangle = \frac{2.10\frac{\alpha_S}{2\pi}+45\left(\frac{\alpha_S}{2\pi}\right)^2}{1+\frac{\alpha_S}{\pi}}.$$

There is a more complex demonstration routine built in to the program, which uses an heavily modified version of Sjostrand's GBOOK package (similar to HBOOK) to calculate the distribution of various event shapes, as well as their average values. It is recommended that you create a new user routine of your own, rather than modifying the existing one.

The whole of the EVENT2 program is Lorentz invariant. The momenta can be specified in any frame in routine GENTWO, which generates the original $q\bar{q}$ pair, and the choice will carry through the whole procedure. However, the default frame is the e^+e^- centre-of-mass, so it should be safe to assume that in the user routine. For die-hard dogmatists it is possible to write any event shape in a Lorentz-invariant way though, so it should be possible to avoid any assumption about the frame used.

The common blocks

There are three common blocks. Two are intended to be internal, while the third gives extra information to the user routine. CONCOM holds all the constants used by the program,

```
INTEGER METYPE,NF
DOUBLE PRECISION CF,CA,TR,PI,PISQ,HF,CUTOFF,CQ,ONF
COMMON /CONCOM/ CF,CA,TR,PI,PISQ,HF,CUTOFF,CQ,ONF,NF,METYPE
```

All values are set at the beginning of EVENT2. At present there is no possibility to change them without recompiling. They are not copied to the EERAD common blocks, so if using the Giele-Glover matrix elements, some care is needed. The first few should be reasonably obvious, (with TR = HF = $\frac{1}{2}$).

CUTOFF is an infrared cut on the phase-space. If any pair of partons has $m_{ij}^2 <$ CUTOFFQ^2 the whole event is thrown away. Strictly speaking, the algorithm does not need an infrared cutoff — all integrals are finite, so it can be set to zero. However, for very small m_{ij} one ends up taking the difference between two very large numbers and the numerical inaccuracy of the machine becomes important. The default value CUTOFF $= 10^{-8}$ works for the Leiden matrix elements. For ERT, it could even be decreased to about 10^{-10} if necessary, but the Giele-Glover matrix elements seem to be extremely unreliable and something more like 10^{-6} is necessary. We have explicitly checked by working in quadruple precision that this is a numerical problem not a physics problem or a bug. A study of the CUTOFF-dependence is needed, but has not yet been done.

CQ holds the coupling constant, CQ $= \sum_q^{N_f} e_q^2$. It will eventually become an array once Z exchange is included. NF is just a copy of the user input NFL and ONF holds 1/NF for convenience. Setting NFL = 0 is safely dealt with[2]. METYPE was already described above.

SAMCOM holds all the constants associated with importance sampling.

```
INTEGER NPOW1,NPOW2
DOUBLE PRECISION POWINT(10),XPOW1,XPOW2,COSP,COSM,COSZ
COMMON /SAMCOM/ POWINT,XPOW1,XPOW2,COSP,COSM,COSZ,NPOW1,NPOW2
```

The last five were already described above. XPOWi is simply shorthand for 1-1/NPOWi, while POWINT pretabulates the definite integral

$$\texttt{POWINT}(n) = \int_0^1 dx\sqrt{1-x^n} = \frac{\sqrt{(\pi)}\Gamma(1/n)}{2^{2/n}n\Gamma(1/2+17n)}.$$

Since it only extends to 10, NPOWi can only be up to 10, but this is easily big enough for most imagineable applications.

SUBCOM holds the sub-process contributions to the current weight (for the α_S^2 terms only) as described earlier,

```
DOUBLE PRECISION CFSUB,CASUB,TFSUB,GGSUB,QQSUB,QPSUB
COMMON /SUBCOM/ CFSUB,CASUB,TFSUB,GGSUB,QQSUB,QPSUB
```

EERAD common blocks

The EERAD common blocks are set in routine setup. They are reasonably well explained in the comments. A couple of points worth noting are: ymin has to be set to a non-zero value even though it is never used; as is the strong coupling constant, which has to be set to 2π to match EVENT2's normalization; iorder has to be left at 0 otherwise the subtraction terms will not match the real phase-space; there is no check that nf is the same as that used by EVENT2, but the integrals will not converge if it is not. setup prints lots of spurious messages, which can be ignored.

[2] Actually it isn't at the moment. This is currently the only known bug.

Timing

The first useful results can be obtained by running the program with around 10^8 events. Publication-quality results require a minimum of 10^9 events. Updated benchmarks have not yet been computed, but as a rough estimate, we found that on a modern pc, running the program single-core, the results could be obtained in less than 1h with 10^8 events and around 7h with 10^9 events.

Usage with runcard

Since version 0.4, the program behaviour can be configured with an external runcard, following the same principles as the latest version of `EERAD3`. Additional flags can also be passed via the command line to set global options, which are described first below.

`-i` Sets the input runcard path. Defaults to `./card.input`.

`-o` Sets the output folder. Defaults to `./results`.

`-sa` Sets the first half of the random generator. Defaults to 123450.

`-sb` Sets the second half of the random generator. Defaults to 678900.

`-c` Sets the run identifier. Defaults to 00000.

These options make it straightforward to set up an automated multi-run campaign by sweeping one of the two seed halves and varying the run identifier.

On the other hand, the runcard allows to setup the core run parameters without the need to recompile the whole code (as was in every verion prior to the current one). A generic runcard is structured as follows:

```
! Example run card

! General run parameters
NEV     =   5M
NF      =   5

! Binning parameters:

! LH: logarithmic binning
LH      =   1

! B: Bin, L: low, H: high

TB      =   100
TL      =   0.0
TH      =   0.5
```

Each line starting with a `!` is ignored by the program. The available options and their description are listed in the following table.

Parameter	Description	Type	Default
NEV	Event No.	Int	No
NF	Flavor No.	Int	5
EM	COM Energy	Double	91.187
CUTOFF	$m_{ij}^2 <$ CUTOFFQ^2	Double	10^{-8}
CUTUP	$m_{ij}^2 >$ CUTUPQ^2	Double	1.0
LH	Global log. bins	Int	0
xB	Bin No., obs. x	Int	Variable
xL	Obs. x lower limit	Double	Variable
xH	Obs. x upper limit	Double	Variable

The currently implemented observables x are:

C C-Parameter.

D D-Parameter.

T Thrust.

Y y_3.

E Energy-energy correlation.

To set the histogram property of the desired observable, simply replace x in the input card option with the required character.

Output structure

In the last version of the program the output structure has significantly changed. Until version `0.3`, all observable histograms were written into a single `gtopdraw.top` file, which could be directly fed into the `topdrawer` plotter. Now, each observable is saved into a separate file named `gtopdraw_OBS_RUNID.dat`, with a fixed header containing some information about the histogram. A typical output file looks like:

```
NEW FRAME
SET WINDOW X 4.9 9.5 Y 2 9
SET FONT DUPLEX
TITLE TOP 'THRUST_00001'
SET LIMITS X   2.00000009E-03  0.500000000
SET X LOG BINS
SET PATT 0.02 0.08
SET ORDER X Y DY
HIST ID            3
HIST IDERR         103
HIST NBINS          50
DATA
   NLO DATA...
HIST          ; PLOT
HIST ID           13
HIST IDERR         113
HIST NBINS          50
DATA
   LO DATA...
JOIN DASH     ; PLOT
```

While this change made the ouput processing more manageable, it also broke the file formati, which now requires some manual changes to be used in `topdrawer`. This is an issue that is currently under investigation[3].

As a final note, running the program two times with the same run id. and output folder will not overwrite the previous results, but create a new file with the prefix $_i$, where i is the copy number, appended before the file extension.

`EVENT2` merge utility

In the latest version of `EVENT2`, the `event2_merge` utility has been added. This program merges histograms from different runs into a single result, computed as the bin-by-bin weighted average of each observable histogram, using squared errors as weights. The utility reads histograms previously generated by `EVENT2` from the `./results/` folder and writes its output to the `./merge` folder. Changing these folders currently requires editing the source code and recompiling; external configuration is planned for a future release.

The future

The program is not yet complete. The main problem is the reliance on Giele-Glover code. We are gradually moving away from this, but still need them at present because we have not yet implemented the Leiden matrix elements for Z exchange. The eventual aim is that they will only be a cross-check of the Leiden code. The eventual aim is to also work for arbitrary currents including W exchange and polarized leptons, but that's still a way o yet. These are the main changes that can be anticipated. Smaller cosmetic changes can also be expected once work gets more seriously under way on the DIS program, to make the different programs as similar as possible. There are also some othe minor changes and improvements planned, which are listed in the last paragraph of this document.

Modification Log

0.0 18/01/96: First prerelease version put on web page.

0.1 26/02/96: Bug fix in event orientation.

0.2 15/03/96: Removed need for Lorentz boosts.

0.3 10/11/97:

- Improved numerical convergence and safety against arithmetic errors.
- Added `CUTUP` variable

0.4 05/06/26:

- Separated histogram output into individual files.
- Added runcard input.
- Added support for variable bin sizes.
- Seeds can now be externally set.
- Added log-binning capability.

[3] Look also at the last section of this document.

Issues and planned features

- Histograms:
  - Increase histogram max. bin number.
  - Allow for variable histogram ids.
  - Restore `topdrawer` compatibility.
- Core:
  - Investigate suspicious normalization difference between linear and log binning.
- Merge utility:
  - Add externally configurable I/O paths.
  - Handle variable histogram ids.