

PSGen, a generator of phase space parameterizations for the multichannel Monte Carlo integration

Karol Kołodziej¹

*Institute of Physics, University of Silesia
ul. 75 Pułku Piechoty 1, PL-41500 Chorzów, Poland*

Abstract

PSGen is a new general purpose Fortran program which has been written to facilitate the Monte Carlo phase space integration of the S matrix element of any $2 \rightarrow n$ scattering process, with $n = 2, \dots, 9$, provided by the user. The program is written in Fortran 90/95. It utilizes a new very fast algorithm that automatically generates calls to ready made Fortran subroutines containing different phase space parameterizations of the considered class of processes. The parameterizations take into account mappings of poles due to the Feynman propagators of unstable heavy particles decaying into 2 or 3 on shell final state particles according to predefined patterns, possible single or double t -channel poles and peaks due to one on shell photon or gluon radiation. The individual subroutines are organized in a single multichannel kinematical subroutine which can be easily called while computing the phase space integral of the S matrix element as a function of generated particle four momenta, in either the leading or higher orders of the perturbation series. The particle four momenta can be used in a quadruple precision version, if necessary.

¹E-mail: karol.kolodziej@us.edu.pl

1 Introduction

Projects of the High-Luminosity Large Hadron Collider (HL-LHC) [1] and electron–positron colliders: the Future Circular Collider (FCC–ee) [2] and Compact Linear Collider (CLIC) [3] at CERN, the International Linear Collider (ILC) [4] in Japan, or the Circular Electron–Positron Collider (CEPC) [5] in China, offer a wealth of new possibilities to test various aspects of the theory of fundamental interactions. Questions about the nonabelian nature of gauge symmetry group and the mechanism of the symmetry breaking, can be directly addressed in processes of a few heavy particles production at a time, such as the top-quark pair production, possibly associated with the Higgs or a heavy electroweak gauge boson, or processes of a few heavy bosons production at a time. In order to explore the nature of the heavy particles interaction, the corresponding multiparticle decay products of them must be studied in detail, including their distributions and spin correlations. Reactions with multiparticle final states must also be taken into account if one wants to determine precisely hadronic contributions to the vacuum polarization through dispersion relations from measurements of the ratio $R = \sigma(e^+e^- \rightarrow \text{hadrons})/\sigma(e^+e^- \rightarrow \mu^+\mu^-)$ at the centre of mass energies below the J/ψ production threshold. The hadronic contributions to the vacuum polarization are the major factor which influences precision of theoretical predictions for the muon $g - 2$ anomaly and plays an important role in the evolution of the fine structure constant $\alpha(Q^2)$ from the Thomson limit to high energy scales.

In order to fully exploit physical information contained in reactions with the multiparticle final states it is necessary to perform numerical integration over a multidimensional phase space of the corresponding squared modulus of matrix elements, often involving several dozen thousand or even several hundred thousand amplitudes of the Feynman diagrams. The amplitudes include peaks, mostly due to denominators of the Feynman propagators, which must be mapped out in order to obtain reliable results of the integration. This goal can be in practice obtained only within the multichannel Monte Carlo (MC) approach, with the corresponding integration routine being generated in a fully automatic way.

Most of processes of interest in the high energy accelerators can be handled with existing general purpose programs, such as: MadGraph/MadEvent/HELAS [6], CompHEP/CalcHEP [7], ALPGEN [8], HELAC-PHEGAS [9], SHERPA/Comix [10], O’Mega/Whizard [11], or carlomat [12], [13], [14], [15]. Some programs, as FeynArts/FormCalc [16], GRACE [17], MadGraph5_aMC@NLO [18], SHERPA 2.2 [19] and HELAC-NLO [20], enable automatic calculation of the NLO EW or QCD corrections. Most of those programs also offer a possibility of integrating the generated matrix elements over the phase space, even if it is multidimensional.

However, some interesting issues, as e. g. above mentioned determination of hadronic contributions

to the vacuum polarization, require dedicated studies of multiparticle reactions within some effective models. In such cases, researchers usually spend a lot of time to program necessary matrix elements by themselves and then they must invest yet more time to prepare a routine for reliable phase space integration. The present work tries to meet their needs in a sense that it provides a new tool, called PSGen, which automatically generates a stand-alone Fortran 90/95 subroutine which, if called with random arguments by any MC integration routine, delivers the corresponding particle four momenta calculated for one selected phase space parameterization together with properly normalized differential volume element of the multidimensional phase space.

The rest of the paper is organized in the following way. Basics of program PSGen are described in Section 2 and preparation for running and program usage are described in Section 3.

2 Basics of PSGen

PSGen is a Fortran 90/95 program which automatically generates calls to kinematical subroutines of the user defined process. The generated subroutines and auxiliary files are moved to the target directory, where they are organized in a subroutine containing the differential multichannel phase space volume parameterization that can be easily called by any MC integration routine.

2.1 Generation of kinematical routines

The core part of PSGen is subroutine `genps(nfspt)`. It contains an algorithm for generating calls to in advance prepared subroutines containing different phase space parameterizations, further referred to as kinematical routines, for a given number of the final state particles `nfspt`. The latter is determined automatically from the character variable `process`, which is defined by the user in `PSGen.f` and transferred to subroutine `read_process(process)`. The algorithm is based on user defined patterns which are collected in a data file `genps.dat`. Each pattern consists of one line that contains the following data: a number of the final state particles, their names and, after a colon, the mass and width of the intermediate particle(s) they are coupled to. For example, in the current version of file `genps.dat`, among others, there are the following lines:

```
2 u u~ : mg,zero,
2 e- e+ : mz,gamz.
```

The first line means that a pair of the final state quarks $u\bar{u}$ couples to the intermediate gluon of mass `mg` and width `zero` and the second line says that the e^-e^+ pair couples to the Z boson of mass `mz`

and width `gamz`. There are also entries in `genps.dat` which look like the following one

```
3 b~ d u~ : mw,gamw,mt,gamt.
```

It consists of 3 final state particles $\bar{b}d\bar{u}$ which couple to two intermediate particles: the $d\bar{u}$ -quark pair couples the W boson of mass `mw` and width `gamw` and the W boson and \bar{b} -quark couple to the top quark of mass `mt` and width `gamt`. If the number of particles is 0, then the whole line is treated as a comment, e.g. the line

```
0 quark-quark-gluon:
```

is a comment. The names of particles, their masses and widths in file `genps.dat` must conform with those listed in file `particles.dat`, where in addition to the name and width also some other characteristics of each particle are given, namely two integers equal 1 or 0 each, and the type of the particle in the form of character(1) variable at the end of each data line. The first integer specifies whether the particle couples (= 1) or not (= 0) to the photon, the second specifies if it couples to the gluon and the one character variable specifies the type of particle, i.e., `n` stands for a neutrino, `l` for a charged lepton, `q` for quark, etc.

After the process has been defined and a few flags, which will be explained below, have been specified, program `PSGen` makes a call to subroutine `read_process(process)`. It reads the initial and final state particles from character variable `process` and checks if all the particles are contained in data file `particles.dat` and whether they couple to the photon or gluon. It also checks if there are t -channel poles in the process, or if the final state contains a photon or a gluon. If it is so, then `genps(nfspt)`, in addition to subroutine `kinschnl`, which comprises calls to kinematical routines containing mappings smoothing peaks due to the Feynman propagators of the above mentioned intermediate particles, will also generate file `tchcalls.f`, which comprises calls to kinematical routines containing mappings of the t -channel poles, or subroutine `kingchnl`, which contain calls to subroutines with mappings of poles due to radiation of the external photon or gluon. File `tchcalls.f` is included in ready made subroutine `kintchnl` which is located in the target directory.

Calls to subroutines `kinschnl`, `kintchnl` and `kingchnl` are all included in automatically generated subroutine `kincls`, unless the user decides otherwise by choosing appropriate values of flags `itchnl` or `igchnl` in program `PSGen`. If `itchnl` (`igchnl`) is set to 0, then a call to `kintchnl` (`kingchnl`) in `kincls` is cancelled. Obviously, the calls to them are also not made, if there are no t -channel poles or the external photon or gluon are not present in the process. Yet one more flag `iquadp` is present in program `PSGen`. If `iquadp=1` then the quadruple precision for denominators of the Feynman propagators, the particle masses and four momenta is used, otherwise the double precision arithmetic is used.

2.2 The target kinematical routine

In the current distribution of the program, all the generated routines and auxiliary files, which are also written in Fortran 90/95, are shifted to directory `../mc_computation`, where they are used by the target kinematical routine

```
subroutine psgen(ikin,ecm,x1,x2,x,ndim,flux,dlips). (1)
```

Subroutine (1) utilizes the multichannel MC approach, i.e., it combines calls to different phase space parameterizations containing the mappings discussed in Section 2.1 in a single phase space parameterization. It is self-consistent in a sense that it can be easily called by any program which integrates the S matrix element in either the leading or higher orders of the perturbation series. Its automatically generated ingredients can be used in a quadruple precision version, if necessary.

In the current distribution subroutine (1) is called from a template function `cs_sect(x,ndim)`, that is integrated by a template program `PSGen_test_mpi` with the use of MC integration routine `carlos`. Both the routine from which (1) is called and the main MC integration program must include the command

```
use kinparams,
```

where module `kinparams` is automatically created at the stage of code generation. Moreover, the main integration program must include the command

```
call param_trans(unit),
```

which should be located below the command that opens the output file associated with the same `unit` number and before the first call to the actual MC integration routine used.

The dummy arguments of (1) are the following

- `ikin = 1, 2, ..., nkin`, where `nkin` is a number of kinematical channels calls to which have been generated; parameter `nkin` is defined in automatically generated module `kinparams`,
- `ecm` is the user defined centre of mass energy,
- `x1, x2` are the beam energy fractions carried by the initial state particles, i.e., if `x1=x2=1` then the initial state particles scatter at fixed energy `ecm`,
- `x` is an array of random numbers of dimension `ndim`, with `x(ndim)` being delivered by the MC integration routine actually used,
- `flux` and `dlips` are, respectively, the initial flux factor and the differential phase space element, both calculated by (1).

The particle four momenta computed by (1) for a given value of `ikin` are returned in module `fourmom` which is also created at the stage of code generation. It is used in (1) and must also be used wherever the user wants to refer to the particle four momenta.

2.3 Ready made kinematical subroutines

A number of kinematical subroutines corresponding to 2, 3, ..., 9 final state particles have been written and tested. Each of them calculates a volume of the Lorentz invariant phase space volume element as generally defined in Eq. (2) of [15] and the set of the final state particle four momenta corresponding to the random arguments `x(ndim)` they are called with. If a double t -channel pole is present in the process then it is mapped out in a way described in Section 3.1 of [15]. Several new subroutines have been written which map the single t -channel poles in a similar way and subroutines which map out peaks due to the photon or gluon radiation with transformations described in Section 3.1 of [15]. Lists of dummy arguments of all those subroutines take into account patterns defined in file `genps.dat`, as discussed in Section 2.1. It may happen, however, that for some new user defined patterns in file `genps.dat` or for some processes for which the program has not been tested yet, new kinematical subroutines will be necessary. Note that, as most of those ready made subroutines contain `kind` type parameters which are set at the stage of code generation, they must be recompiled each time the MC code is generated anew. However, this is not a problem at all, as the compilation usually takes a few seconds time.

The user can easily add by hand a call to an own made kinematical subroutine by modifying the automatically generated subroutine `kincls` which collects calls to all kinematical subroutines of different type. The own made kinematical subroutine should be written in the similar way as the automatically generated subroutines `kinschnl` or `kingchnl`. The instruction where the call should be put is contained in `kincls`. Note, that the number of kinematical channels given by parameter `nkin` in `kinparams.f` must be then adjusted appropriately by hand according to the following formula

$$nkin = nkinag + nkinua,$$

where `nkinag` is the number of the kinematical channels automatically generated and `nkinua` is the number of channels added by the user.

All physical input parameters are defined in module `inprms_ps`, located in directory `mc_computation`, where in particular numerical values of all the particle masses and widths introduced in files `genps.dat` and `particles.dat` must be specified.

3 Preparation for running and program usage

Program PSGen is distributed as a single tar.gz archive PSGen.tgz which can be downloaded from: <http://kk.us.edu.pl/PSGen.html>. When untared with the command

```
tar -xzvf PSGen.tgz
```

it will create directory PSGen_1.0 with sub directories: code_generation, mc_computation and test_output.

The preparation for running requires the following steps

- Choose a Fortran 90/95 compiler in a makefile of code_generation and possibly change the target directory to which the generated files should be moved from `./mc_computation/` to a directory of your choice. Note that the target directory must include subroutine (1) and all other ready made files listed in a makefile of mc_computation.
- Specify the process and required options in PSGen.f and execute the command
`make code`
from the command line in code_generation.

As already mentioned in Section 2.2, the current distribution of PSGen contains a template program PSGen_test_mpi which allows to perform the MC integration, utilizing the Message Passing Interface (MPI), of a template function `cs_sect(x, ndim)`, both are located in directory mc_computation. Function `cs_sect(x, ndim)` calls the target kinematical routine (1). As the conversion constant and the matrix element are both set to 1 in `cs_sect(x, ndim)`, the integral is the phase space volume, restricted by kinematical cuts, of the considered process at the centre of mass energies defined in PSGen_test_mpi by array `aecm(ne)`. The cuts can be defined in subroutine `define_cuts` and imposed by setting `icuts=1` in PSGen_test_mpi. The template program can be executed with the single command

```
make mc
```

executed in directory mc_computation. The MC integration is performed using the multichannel approach with integration weights adjusted anew after every iteration, as described in Section 2 of [15]. Obviously, the templates can be used to calculate the cross section of the user defined process, if a call to user's own subroutine calculating the corresponding matrix element at the four momenta calculated by (1) is made and the conversion constant is appropriately taken into account function in `cs_sect(x, ndim)`. The basic output of the MC run is written in file `tot_0_name`, where `name` is created automatically if the assignment for character variable
`nicknm='auto'`

in `PSGen.f` is not changed to arbitrary user's defined name. The output of other MPI processes are written to files `tot_i_name`, with `i` being the MPI process identification number.

Whenever the Fortran compiler is changed, or a compiled program is transferred to another computer with a different processor, all the object and module files should be deleted by executing the commands:

```
rm *.o
rm *.mod
```

and the command `make mc` should be executed anew.

References

- [1] G. Apollinari, et al., CERN Yellow Report CERN-2015-005, pp.1-19 [arXiv:1705.08830 [physics.acc-ph]].
- [2] FCC Collaboration (A. Abada, et al.), “FCC Physics Opportunities: Future Circular Collider Conceptual Design Report Volume 1”, Eur. Phys. J. C **79** (2019) 474;
FCC Collaboration (A. Abada, et al.), “FCC-ee: The Lepton Collider: Future Circular Collider Conceptual Design Report Volume 2”, Eur. Phys. J. ST **228** (2019) 261;
FCC Collaboration (A. Abada, et al.), “FCC-hh: The Hadron Collider: Future Circular Collider Conceptual Design Report Volume 3”, Eur. Phys. J. ST **228** (2019) 755.
- [3] M. Aicheler et al. (eds.), “A Multi-TeV Linear Collider based on CLIC Technology: CLIC Conceptual Design Report” (CERN) (2012). DOI:10.5170/CERN-2012-007;
CLIC and CLICdp Collaborations (P. Roloff, et al.), “The Compact Linear e^+e^- Collider (CLIC): Physics Potential”, arXiv:1812.07986 [hep-ex];
CLICdp and CLIC Collaborations (P.N. Burrows, et al.), “The Compact Linear Collider (CLIC) – 2018 Summary Report”, CERN Yellow Rep. Monogr. 1802 (2018) 1-98, arXiv:1812.06018 [physics.acc-ph], CERN-2018-005-M;
L. Linssen et al., “Physics and Detectors at CLIC: CLIC Conceptual Design Report (2012)”, DOI:10.5170/CERN-2012-003, arXiv:1202.5940 [physics.ins-det];
T. K. Charles et al., CLICdp, CLIC, “The Compact Linear Collider (CLIC) - 2018 Summary Report”, CERN Yellow Rep. Monogr.1802(2018), ed. by P. N. Burrows et al. 1, DOI:10.23731/CYRM-2018-002, arXiv:1812.06018 [physics.acc-ph];
P. Roloff et al., CLIC, CLICdp, “The Compact Linear e^+e^- Collider (CLIC): Physics Potential (2018)”, arXiv:1812.07986 [hep-ex].

- [4] T. Behnke, J. E. Brau, B. Foster, J. Fuster, M. Harrison, et al., “The International Linear Collider Technical Design Report - Volume 1: Executive Summary”, arXiv:1306.6327;
H. Baer, T. Barklow, K. Fujii, Y. Gao, A. Hoang, et al., “The International Linear Collider Technical Design Report - Volume 2: Physics”, arXiv:1306.6352.
- [5] CEPC Study Group, CEPC Conceptual Design Report: Volume 1 – Accelerator, arXiv:1809.00285[physics.acc-ph];
CEPC Study Group, CEPC Conceptual Design Report: Volume 2 — Physics & Detector, arXiv:1811.10545[physics.acc-ph].
- [6] T. Stelzer, W.F. Long, *Comput. Phys. Commun.* **81** (1994) 357;
F. Maltoni, T. Stelzer, *JHEP* 02 (2003) 027;
J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer, T. Stelzer, *JHEP* 06 (2011) 128;
H. Murayama, I. Watanabe, K. Hagiwara, KEK-91-11.
- [7] A. Pukhov, et al., arXiv:hep-ph/9908288;
E. Boos, et al., *Nucl. Instrum. Meth. A* 534 (2004) 250;
A. Belyaev, N.D. Christensen, A. Pukhov, *Comput. Phys. Commun.* **184** (2013) 1729.
- [8] M.L. Mangano, M. Moretti, F. Piccinini, R. Pittau, A. Polosa, *JHEP* 0307 (2003) 001.
- [9] A. Kanaki, C.G. Papadopoulos, *Comput. Phys. Commun.* **132** (2000) 306;
C.G. Papadopoulos, *Comput. Phys. Commun.* **137** (2001) 247;
A. Cafarella, C.G. Papadopoulos, M. Worek, *Comput. Phys. Commun.* **180** (2009) 1941.
- [10] T. Gleisberg, et al., *JHEP* 0402 (2004) 056;
T. Gleisberg, et al., *JHEP* 0902 (2009) 007;
T. Gleisberg, S. Höche, *JHEP* 0812 (2008) 039.
- [11] M. Moretti, T. Ohl, J. Reuter, arXiv:hep-ph/0102195;
W. Kilian, T. Ohl, J. Reuter, *Eur. Phys. J.* **C71** (2011) 1742.
- [12] K. Kołodziej, *Comput. Phys. Commun.* **180** (2009) 1671.
- [13] K. Kołodziej, *Comput. Phys. Commun.* **185** (2014) 323.
- [14] K. Kołodziej, *Comput. Phys. Commun.* **196** (2015) 563;
K. Kołodziej, carlomat_3.1, <http://kk.us.edu.pl/carlomat.html>.
- [15] K. Kołodziej, *Comput. Phys. Commun.* **276** (2022) 108330.

- [16] J. Küblbeck, M. Böhm, A. Denner, *Comput. Phys. Commun.* **60** (1990) 165;
T. Hahn, *Nucl. Phys. Proc. Suppl.* **89** (2000) 231;
T. Hahn, *Comput. Phys. Commun.* **140** (2001) 418.
- [17] H. Tanaka, T. Kaneko, Y. Shimizu, *Comput. Phys. Commun.* **64** (1991) 149;
F. Yuasa, et al., *Prog. Theor. Phys. Suppl.* 138 (2000) 18;
G. Belanger, et al., *Phys. Rept.* 430 (2006) 117.
- [18] J. Alwall, et al. *JHEP07* (2014) 079 [arXiv:1405.0301 [hep-ph]].
- [19] E. Bothmann et al., *SciPost Phys.* 7 (2019) 034 [arXiv:1905.09127 [hep-ph]].
- [20] G. Bevilacqua, et al., *Comput. Phys. Commun.* **184** (2013) 986.