PSGen

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

Karol Kołodziej

Institute of Physics
University of Silesia

Matter To The Deepest
Recent Developments In Physics Of Fundamental Interactions
XLV International Conference of Theoretical Physics
Ustroń, 17-22 September, 2023

Outline

- Motivation
- Basics of program PSGen
- Sample results
- Preparation for running and program usage
- Summary and outlook

Based on a publication K. Kołodziej, Computer Physics Communications 292 (2023) 108870 [arXiv:2303.1204[hep-ph]].

Motivation

The Standard Model (SM) of fundamental interactions has been very successful, but hardly anyone believes it is an ultimate theory.

Projects of

- the High-Luminosity Large Hadron Collider (HL-LHC)

and electron-positron colliders as

- the Future Circular Collider (FCC–ee) and Compact Linear Collider (CLIC) at CERN
- the International Linear Collider (ILC) in Japan
- the Circular Electron–Positron Collider (CEPC) in China

may bring discoveries of phenomena beyond the scope of SM.

Motivation

The Standard Model (SM) of fundamental interactions has been very successful, but hardly anyone believes it is an ultimate theory.
Projects of

- the High-Luminosity Large Hadron Collider (HL-LHC)

and electron-positron colliders as

- the Future Circular Collider (FCC–ee) and Compact Linear Collider (CLIC) at CERN
- the International Linear Collider (ILC) in Japan
- the Circular Electron–Positron Collider (CEPC) in China

may bring discoveries of phenomena beyond the scope of SM.
They would also offer a wealth of new possibilities to test various aspects of the SM.

Motivation

The Standard Model (SM) of fundamental interactions has been very successful, but hardly anyone believes it is an ultimate theory.
Projects of

- the High-Luminosity Large Hadron Collider (HL-LHC)

and electron-positron colliders as

- the Future Circular Collider (FCC–ee) and Compact Linear Collider (CLIC) at CERN
- the International Linear Collider (ILC) in Japan
- the Circular Electron–Positron Collider (CEPC) in China

may bring discoveries of phenomena beyond the scope of SM.
They would also offer a wealth of new possibilities to test various aspects of the SM.

Motivation

Questions about the non-Abelian nature of the SM 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
- processes of a few heavy bosons production at a time.

To get insight into the nature of interactions of the heavy particles, which almost immediately decay, it is necessary to investigate multiparticle reactions, including distributions and spin correlations of light particles in the final state.

Motivation

Questions about the non-Abelian nature of the SM 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
- processes of a few heavy bosons production at a time.

To get insight into the nature of interactions of the heavy particles, which almost immediately decay, it is necessary to investigate multiparticle reactions, including distributions and spin correlations of light particles in the final state.

The multiparticle reactions can be handled with several publicly available multipurpose Monte Carlo (MC) generators as `MadGraph/MadEvent/HELAS`, `CompHEP/CalcHEP`, `ALPGEN`, `HELAC-PHEGAS`, `SHERPA/Comix`, `O'Mega/Whizard`, or `carlomat`.

Motivation

Questions about the non-Abelian nature of the SM 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
- processes of a few heavy bosons production at a time.

To get insight into the nature of interactions of the heavy particles, which almost immediately decay, it is necessary to investigate multiparticle reactions, including distributions and spin correlations of light particles in the final state.
The multiparticle reactions can be handled with several publicly available multipurpose Monte Carlo (MC) generators as MadGraph/MadEvent/HELAS, CompHEP/CalcHEP, ALPGEN, HELAC-PHEGAS, SHERPA/Comix, O'Mega/Whizard, or carlomat.

Motivation

Some multipurpose programs, as FeynArts/FormCalc, GRACE, MadGraph5_aMC@NLO, SHERPA 2.2 and HELAC-NLO, enable automatic calculation of the NLO EW or QCD corrections.

The considered multiparticle reactions, in addition to the *signal* Feynman diagrams, *i.e.* those which contain the Feynman propagators of the heavy particles of interest, receive contributions from typically several dozen thousand or even several hundred thousand *background* Feynman diagrams.

Motivation

Some multipurpose programs, as FeynArts/FormCalc, GRACE, MadGraph5_aMC@NLO, SHERPA 2.2 and HELAC-NLO, enable automatic calculation of the NLO EW or QCD corrections.

The considered multiparticle reactions, in addition to the *signal* Feynman diagrams, *i.e.* those which contain the Feynman propagators of the heavy particles of interest, receive contributions from typically several dozen thousand or even several hundred thousand *background* Feynman diagrams.

The corresponding amplitudes must be added and the squared modulus of the sum must be averaged over spins and integrated over a multidimensional phase space.

Motivation

Some multipurpose programs, as FeynArts/FormCalc, GRACE, MadGraph5_aMC@NLO, SHERPA 2.2 and HELAC-NLO, enable automatic calculation of the NLO EW or QCD corrections.
The considered multiparticle reactions, in addition to the *signal* Feynman diagrams, *i.e.* those which contain the Feynman propagators of the heavy particles of interest, receive contributions from typically several dozen thousand or even several hundred thousand *background* Feynman diagrams.
The corresponding amplitudes must be added and the squared modulus of the sum must be averaged over spins and integrated over a multidimensional phase space.
Most of those programs offer a possibility of integrating the generated matrix elements over the phase space.

Motivation

Some multipurpose programs, as FeynArts/FormCalc, GRACE, MadGraph5_aMC@NLO, SHERPA 2.2 and HELAC-NLO, enable automatic calculation of the NLO EW or QCD corrections.
The considered multiparticle reactions, in addition to the *signal* Feynman diagrams, *i.e.* those which contain the Feynman propagators of the heavy particles of interest, receive contributions from typically several dozen thousand or even several hundred thousand *background* Feynman diagrams.
The corresponding amplitudes must be added and the squared modulus of the sum must be averaged over spins and integrated over a multidimensional phase space.
Most of those programs offer a possibility of integrating the generated matrix elements over the phase space.

Motivation

Reactions with multiparticle final states must also be taken into account if one wants to determine precisely hadronic contributions to the vacuum polarization 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.

The hadronic contributions to the vacuum polarization can be determined through dispersion relations from the energy dependence of the ratio

$$R_\gamma(s) \equiv \sigma^{(0)}(e^+e^- \to \gamma^* \to \mathrm{hadrons}) / \frac{4\pi\alpha^2}{3s}$$

Motivation

Reactions with multiparticle final states must also be taken into account if one wants to determine precisely hadronic contributions to the vacuum polarization 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.
The hadronic contributions to the vacuum polarization can be determined through dispersion relations from the energy dependence of the ratio

$$R_\gamma(s) \equiv \sigma^{(0)}(e^+e^- \to \gamma^* \to \mathrm{hadrons})/\frac{4\pi\alpha^2}{3s}$$

Below the J/ψ threshold, $\sigma_{e^+e^-\to\mathrm{hadrons}}(s)$ must be measured, either by the initial beam energy scan or with the use of a radiative return method, and compared with predictions of a Monte Carlo program, as e.g. PHOKARA.

Motivation

Reactions with multiparticle final states must also be taken into account if one wants to determine precisely hadronic contributions to the vacuum polarization 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.
The hadronic contributions to the vacuum polarization can be determined through dispersion relations from the energy dependence of the ratio

$$R_\gamma(s) \equiv \sigma^{(0)}(e^+e^- \to \gamma^* \to \mathrm{hadrons})/\frac{4\pi\alpha^2}{3s}$$

Below the J/ψ threshold, $\sigma_{e^+e^-\to\mathrm{hadrons}}(s)$ must be measured, either by the initial beam energy scan or with the use of a radiative return method, and compared with predictions of a Monte Carlo program, as e.g. PHOKARA.

Motivation

To obtain predictions for $e^+e^- \to$ hadrons, researchers usually spend a lot of time to program necessary matrix elements by hand within some effective model and then they must invest yet more time to prepare a routine for reliable phase space integration.

Preparation of the routine for reliable phase space integration can be facilitated with a new toll, 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 paremeterization, together with properly normalized differential volume element of the multidimensional phase space.

Motivation

To obtain predictions for $e^+e^- \to$ hadrons, researchers usually spend a lot of time to program necessary matrix elements by hand within some effective model and then they must invest yet more time to prepare a routine for reliable phase space integration.

Preparation of the routine for reliable phase space integration can be facilitated with a new toll, 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 paremeterization, together with properly normalized differential volume element of the multidimensional phase space.

Basics of program PSGen

The amplitudes of multiparticle reactions involve very many peaks, mostly due to denominators of the Feynman propagators, which cannot be mapped out with a single change of integration variables in the phase space integration element

$$\mathrm{d}^{3n_f-4}Lips = (2\pi)^4\delta^{(4)}\Big(p_1 + p_2 - \sum_{i=3}^{n} p_i\Big)\prod_{i=3}^{n}\frac{\mathrm{d}p_i^3}{(2\pi)^3 2E_i},$$

with $n_f = n - 2$.

Therefore, the integration can be in practice performed only with the use of the multichannel MC approach, which combines different phase space parameterizations dedicated to mappings of different poles.

Basics of program PSGen

The amplitudes of multiparticle reactions involve very many peaks, mostly due to denominators of the Feynman propagators, which cannot be mapped out with a single change of integration variables in the phase space integration element

$$\mathrm{d}^{3n_f-4}Lips = (2\pi)^4\delta^{(4)}\Big(p_1 + p_2 - \sum_{i=3}^{n} p_i\Big)\prod_{i=3}^{n}\frac{\mathrm{d}p_i^3}{(2\pi)^3 2E_i},$$

with $n_f = n - 2$.

Therefore, the integration can be in practice performed only with the use of the multichannel MC approach, which combines different phase space parameterizations dedicated to mappings of different poles.

To obtain a satisfactory convergence of the MC integration, usually quite many phase space parameterizations are needed. It is vital to generate the corresponding multichannel MC integration routine in a fully automatic way.

Basics of program PSGen

The amplitudes of multiparticle reactions involve very many peaks, mostly due to denominators of the Feynman propagators, which cannot be mapped out with a single change of integration variables in the phase space integration element

$$\mathrm{d}^{3n_f-4}Lips = (2\pi)^4\delta^{(4)}\Big(p_1 + p_2 - \sum_{i=3}^{n} p_i\Big)\prod_{i=3}^{n}\frac{\mathrm{d}p_i^3}{(2\pi)^3 2E_i},$$

with $n_f = n - 2$.

Therefore, the integration can be in practice performed only with the use of the multichannel MC approach, which combines different phase space parameterizations dedicated to mappings of different poles.

To obtain a satisfactory convergence of the MC integration, usually quite many phase space parameterizations are needed. It is vital to generate the corresponding multichannel MC integration routine in a fully automatic way.

Phase space integration with PSGen

Denote i-th of N phase space parameterizations generated by

$$f_i(x) = \mathrm{d}^{3n_f-4} Lips_i\,(x)\,, \qquad i = 1, \ldots, N,$$

where $x = (x_1, ..., x_{3n_f-4})$ are random arguments, $x_i \in [0, 1]$.
It must satisfy the normalization condition

$$\int_0^1 \mathrm{d}x^{3n_f-4} f_i(x) = \mathrm{vol}(Lips).$$

All the parameterizations $f_i(x)$ are then automatically combined into a single multichannel probability distribution

$$f(x) = \sum_{i=1}^{N} a_i f_i(x),$$

with non negative weights a_i, $i = 1, ..., N$, satisfying the condition

$$\sum_{i=1}^{N} a_i = 1 \qquad \Leftrightarrow \qquad \int_0^1 \mathrm{d}x^{3n_f-4} f(x) = \mathrm{vol}(Lips).$$

Phase space integration with PSGen

Denote i-th of N phase space parameterizations generated by

$$f_i(x) = \mathrm{d}^{3n_f-4} Lips_i\,(x)\,, \qquad i = 1, \ldots, N,$$

where $x = (x_1, ..., x_{3n_f-4})$ are random arguments, $x_i \in [0, 1]$.
It must satisfy the normalization condition

$$\int_0^1 \mathrm{d}x^{3n_f-4} f_i(x) = \mathrm{vol}(Lips).$$

All the parameterizations $f_i(x)$ are then automatically combined into a single multichannel probability distribution

$$f(x) = \sum_{i=1}^{N} a_i f_i(x),$$

with non negative weights a_i, $i = 1, ..., N$, satisfying the condition

$$\sum_{i=1}^{N} a_i = 1 \qquad \Leftrightarrow \qquad \int_0^1 \mathrm{d}x^{3n_f-4} f(x) = \mathrm{vol}(Lips).$$

Phase space integration with PSGen

The actual MC integration is performed with the random numbers generated according to probability distribution $f(x)$.

Integration with PSGen can be performed iteratively, as in the template program attached.

First, the MC integral is calculated N times with a rather small number of calls to the integrand, each time with a different phase space parameterization $f_i(x)$.

Phase space integration with PSGen

The actual MC integration is performed with the random numbers generated according to probability distribution $f(x)$.
Integration with PSGen can be performed iteratively, as in the template program attached.
First, the MC integral is calculated N times with a rather small number of calls to the integrand, each time with a different phase space parameterization $f_i(x)$.
The result σ_i obtained with the i-th parameterization is used to calculate new weights according to the following formula

$$a_i = \sigma_i / \sum_{j=1}^{N} \sigma_j .$$

a_i is the probability of choosing i-th parameterization in the first iteration $\Rightarrow$ channels with small weights a_i are not chosen and will have zero weights in the following iterations.

Phase space integration with PSGen

The actual MC integration is performed with the random numbers generated according to probability distribution $f(x)$.
Integration with PSGen can be performed iteratively, as in the template program attached.
First, the MC integral is calculated N times with a rather small number of calls to the integrand, each time with a different phase space parameterization $f_i(x)$.
The result σ_i obtained with the i-th parameterization is used to calculate new weights according to the following formula

$$a_i = \sigma_i / \sum_{j=1}^{N} \sigma_j .$$

a_i is the probability of choosing i-th parameterization in the first iteration $\Rightarrow$ channels with small weights a_i are not chosen and will have zero weights in the following iterations.

Generation of kinematics routines

The core part of PSGen is subroutine `genps(nfspt)`.
It contains an algorithm for generating calls to handwritten subroutines containing different phase space parameterizations for a given number of the final state particles `nfspt`, referred to as kinematics routines.

Generation of kinematics routines

The core part of PSGen is subroutine `genps(nfspt)`.
It contains an algorithm for generating calls to handwritten subroutines containing different phase space parameterizations for a given number of the final state particles `nfspt`, referred to as kinematics routines.
`nfspt` is determined automatically from the character variable `process`, which is defined by the user in `PSGen.f` and transferred to subroutine `read_process(process)`.

Generation of kinematics routines

The core part of PSGen is subroutine `genps(nfspt)`.
It contains an algorithm for generating calls to handwritten subroutines containing different phase space parameterizations for a given number of the final state particles `nfspt`, referred to as kinematics routines.
`nfspt` 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`.

Generation of kinematics routines

The core part of PSGen is subroutine `genps(nfspt)`.
It contains an algorithm for generating calls to handwritten subroutines containing different phase space parameterizations for a given number of the final state particles `nfspt`, referred to as kinematics routines.
`nfspt` 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.

Generation of kinematics routines

The core part of PSGen is subroutine `genps(nfspt)`.
It contains an algorithm for generating calls to handwritten subroutines containing different phase space parameterizations for a given number of the final state particles `nfspt`, referred to as kinematics routines.
`nfspt` 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.

Generation of kinematics routines

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 this:

```
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`.

Generation of kinematics routines

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 this:

```
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`.

Generation of kinematics routines

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 also two integers and the type of the particle in the form of `character(1)` variable at the end of each data line are given.

- 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.

Generation of kinematics routines

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 also two integers and the type of the particle in the form of `character(1)` variable at the end of each data line are given.

- 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.

Generation of kinematics routines

After the process has been defined, `PSGen` makes a call to the 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.

Generation of kinematics routines

After the process has been defined, PSGen makes a call to the 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.

The patterns listed in file genps.dat are used to generate the subroutine kinschnl, which comprises calls to kinematics routines containing mappings smoothing peaks due to the Feynman propagators of the intermediate s-channel particles.

Generation of kinematics routines

After the process has been defined, PSGen makes a call to the 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.
The patterns listed in file genps.dat are used to generate the subroutine kinschnl, which comprises calls to kinematics routines containing mappings smoothing peaks due to the Feynman propagators of the intermediate s-channel particles.
read_process(process) also checks if there are t-channel poles in the process, or if the final state contains a photon or a gluon.

Generation of kinematics routines

After the process has been defined, PSGen makes a call to the 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.
The patterns listed in file genps.dat are used to generate the subroutine kinschnl, which comprises calls to kinematics routines containing mappings smoothing peaks due to the Feynman propagators of the intermediate s-channel particles.
read_process(process) 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) will also generate

- the file tchcalls.f, which comprises calls to kinematics routines containing mappings of the t-channel poles, or
- the subroutine kingchnl, which contain calls to subroutines with mappings of poles due to radiation of the external photon or gluon.

Generation of kinematics routines

After the process has been defined, PSGen makes a call to the 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.
The patterns listed in file genps.dat are used to generate the subroutine kinschnl, which comprises calls to kinematics routines containing mappings smoothing peaks due to the Feynman propagators of the intermediate s-channel particles.
read_process(process) 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) will also generate

- the file tchcalls.f, which comprises calls to kinematics routines containing mappings of the t-channel poles, or
- the subroutine kingchnl, which contain calls to subroutines with mappings of poles due to radiation of the external photon or gluon.

Generation of kinematics routines

File `tchcalls.f` is included in a handwritten subroutine `kintchnl` which is located in the target directory.

Calls to subroutines `kinschnl`, `kintchnl` and `kingchnl` are all included in the automatically generated subroutine `kincls`, unless the user decides otherwise by choosing appropriate values of flags `itchnl` or `igchnl` in program `PSGen`.

Generation of kinematics routines

File `tchcalls.f` is included in a handwritten subroutine `kintchnl` which is located in the target directory.
Calls to subroutines `kinschnl`, `kintchnl` and `kingchnl` are all included in the 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.

Generation of kinematics routines

File `tchcalls.f` is included in a handwritten subroutine `kintchnl` which is located in the target directory.
Calls to subroutines `kinschnl`, `kintchnl` and `kingchnl` are all included in the 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 not made, if there are no t-channel poles or the external photon or gluon are not present in the process.

Generation of kinematics routines

File `tchcalls.f` is included in a handwritten subroutine `kintchnl` which is located in the target directory.
Calls to subroutines `kinschnl`, `kintchnl` and `kingchnl` are all included in the 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 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.

Generation of kinematics routines

File `tchcalls.f` is included in a handwritten subroutine `kintchnl` which is located in the target directory.
Calls to subroutines `kinschnl`, `kintchnl` and `kingchnl` are all included in the 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 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.

The kinematics routine

All the generated routines and auxiliary files, which are also written in Fortran 90/95, are shifted to the directory `../mc_computation`, where they are used by the kinematics routine

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

- `psgen` utilizes the multichannel MC approach, *i.e.*, it combines calls to different phase space parameterizations `ikin = 1,2,...,nkin,` into a single phase space parameterization,

The kinematics routine

All the generated routines and auxiliary files, which are also written in Fortran 90/95, are shifted to the directory `../mc_computation`, where they are used by the kinematics routine

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

- `psgen` utilizes the multichannel MC approach, *i.e.*, it combines calls to different phase space parameterizations `ikin = 1,2,...,nkin,` into a single phase space parameterization,
- `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`,

The kinematics routine

All the generated routines and auxiliary files, which are also written in Fortran 90/95, are shifted to the directory `../mc_computation`, where they are used by the kinematics routine

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

- `psgen` utilizes the multichannel MC approach, *i.e.*, it combines calls to different phase space parameterizations `ikin = 1,2,...,nkin,` into a single phase space parameterization,
- `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`,
- `psgen` 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.

The kinematics routine

All the generated routines and auxiliary files, which are also written in Fortran 90/95, are shifted to the directory `../mc_computation`, where they are used by the kinematics routine

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

- `psgen` utilizes the multichannel MC approach, *i.e.*, it combines calls to different phase space parameterizations `ikin = 1,2,...,nkin,` into a single phase space parameterization,
- `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`,
- `psgen` 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.

The kinematics routine

All the generated routines and auxiliary files, which are also written in Fortran 90/95, are shifted to the directory `../mc_computation`, where they are used by the kinematics routine

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

- `psgen` utilizes the multichannel MC approach, *i.e.*, it combines calls to different phase space parameterizations `ikin = 1,2,...,nkin,` into a single phase space parameterization,
- `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`,
- `psgen` 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.

The kinematics routine

The particle four momenta computed in `psgen` for the kinematics `ikin` are returned in the `module fourmom` which is also created at the stage of code generation. The module is used in `psgen` and must also be used wherever the user wants to refer to the particle four momenta.

The template MC integration program

In the current distribution, `subroutine psgen` 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 `psgen` is called and the main MC integration program must include the command
  `use kinparams`
  `module kinparams` is automatically created at the stage of code generation.

The template MC integration program

In the current distribution, `subroutine psgen` 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 `psgen` is called and the main MC integration program must include the command
  `use kinparams`
  `module kinparams` is automatically created at the stage of code generation.
- The main MC 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 template MC integration program

In the current distribution, `subroutine psgen` 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 `psgen` is called and the main MC integration program must include the command
  `use kinparams`
  `module kinparams` is automatically created at the stage of code generation.
- The main MC 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.

Handwritten kinematics subroutines

A number of kinematics 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 and the set of the final state particle four momenta corresponding to the random arguments `x(ndim)` they are called with.

Handwritten kinematics subroutines

A number of kinematics 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 and the set of the final state particle four momenta corresponding to the random arguments `x(ndim)` they are called with.
As most of those handwritten subroutines contain the Fortran 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 few seconds.

Handwritten kinematics subroutines

A number of kinematics 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 and the set of the final state particle four momenta corresponding to the random arguments `x(ndim)` they are called with.
As most of those handwritten subroutines contain the Fortran 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 few seconds.
It may happen, however, that for some new user defined patterns in `genps.dat` or for some processes for which the program has not been tested yet, new kinematics subroutines will be necessary.

Handwritten kinematics subroutines

A number of kinematics 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 and the set of the final state particle four momenta corresponding to the random arguments `x(ndim)` they are called with.
As most of those handwritten subroutines contain the Fortran 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 few seconds.
It may happen, however, that for some new user defined patterns in `genps.dat` or for some processes for which the program has not been tested yet, new kinematics subroutines will be necessary.

Handwritten kinematics subroutines

The user can easily add by hand a call to an own made kinematics subroutine by modifying the automatically generated subroutine `kincls` which collects calls to all kinematics subroutines of different type.

The own made kinematics subroutine must conform to the automatically generated subroutines `kinschnl` or `kingchnl`.

Handwritten kinematics subroutines

The user can easily add by hand a call to an own made kinematics subroutine by modifying the automatically generated subroutine `kincls` which collects calls to all kinematics subroutines of different type.

The own made kinematics subroutine must conform to the automatically generated subroutines `kinschnl` or `kingchnl`.

The instruction where the call to them should be put is contained in `kincls.f`.

Handwritten kinematics subroutines

The user can easily add by hand a call to an own made kinematics subroutine by modifying the automatically generated subroutine `kincls` which collects calls to all kinematics subroutines of different type.

The own made kinematics subroutine must conform to the automatically generated subroutines `kinschnl` or `kingchnl`.

The instruction where the call to them should be put is contained in `kincls.f`.

Note, that the number of kinematics channels given by parameter `nkin` in `kinparams.f` must then be adjusted appropriately by hand according to the following formula

$$\mathtt{nkin} = \mathtt{nkinag} + \mathtt{nkinua},$$

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

Handwritten kinematics subroutines

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

$$\texttt{nkin} = \texttt{nkinag} + \texttt{nkinua},$$

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

Physical input parameters

All the physical input parameters are defined in the `module inprms_ps,` located in the 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.

Let us compare the LO cross sections of a few reactions obtained with the MC integration of the matrix elements generated by `carlomat_4.4`, integrated with the phase parameterization based on topologies of the Feynman diagrams, as automatically generated by `carlomat_4.4`, and the phase space parameterization obtained with `PSGen`.

Physical input parameters

All the physical input parameters are defined in the `module inprms_ps,` located in the 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.

Let us compare the LO cross sections of a few reactions obtained with the MC integration of the matrix elements generated by `carlomat_4.4`, integrated with the phase parameterization based on topologies of the Feynman diagrams, as automatically generated by `carlomat_4.4`, and the phase space parameterization obtained with `PSGen`.

Sample results

The LO cross sections at $\sqrt{s} = 0.5\ \mathrm{TeV}$ and $\sqrt{s} = 1\ \mathrm{TeV}$ are calculated with the cuts

$$5^\circ < \theta(l, \mathrm{beam}),\ \theta(\gamma, \mathrm{beam}) < 175^\circ, \quad \theta(\gamma, l) > 5^\circ,$$
$$E_l > 5\ \mathrm{GeV}, \quad E_\gamma > 1\ \mathrm{GeV}.$$

Reaction	$\sigma(0.5\ \mathrm{TeV})$		$\sigma(1\ \mathrm{TeV})$	
	carlomat_4.0	PSGen	carlomat_4.0	PSGen
$e^+e^- \to b\mu^+\nu_\mu\bar{b}\mu^-\bar{\nu}_\mu$	6.565(4) fb	6.593(3) fb	2.332(8)	2.375(2) fb
$e^+e^- \to be^+\nu_e\bar{b}\mu^-\bar{\nu}_\mu$	6.624(4) fb	6.622(4) fb	2.621(11)	2.594(6) fb
$e^+e^- \to b\mu^+\nu_\mu\bar{b}\mu^-\bar{\nu}_\mu\gamma$	1.602(5) fb	1.551(11) fb	0.722(4)	0.715(5) fb

The standard deviation of the MC integration, which is given for every entry in the parentheses, is comparable for both integrations. However, PSGen generates much less kinematics channels than carlomat_4.4. This results in much shorter time of the code generation and compilation and shorter program execution time.

Sample results

The LO cross sections at $\sqrt{s} = 0.5\ \mathrm{TeV}$ and $\sqrt{s} = 1\ \mathrm{TeV}$ are calculated with the cuts

$$5^\circ < \theta(l, \mathrm{beam}),\ \theta(\gamma, \mathrm{beam}) < 175^\circ, \quad \theta(\gamma, l) > 5^\circ,$$
$$E_l > 5\ \mathrm{GeV}, \quad E_\gamma > 1\ \mathrm{GeV}.$$

Reaction	$\sigma(0.5\ \mathrm{TeV})$		$\sigma(1\ \mathrm{TeV})$	
	carlomat_4.0	PSGen	carlomat_4.0	PSGen
$e^+e^- \to b\mu^+\nu_\mu\bar{b}\mu^-\bar{\nu}_\mu$	6.565(4) fb	6.593(3) fb	2.332(8)	2.375(2) fb
$e^+e^- \to be^+\nu_e\bar{b}\mu^-\bar{\nu}_\mu$	6.624(4) fb	6.622(4) fb	2.621(11)	2.594(6) fb
$e^+e^- \to b\mu^+\nu_\mu\bar{b}\mu^-\bar{\nu}_\mu\gamma$	1.602(5) fb	1.551(11) fb	0.722(4)	0.715(5) fb

The standard deviation of the MC integration, which is given for every entry in the parentheses, is comparable for both integrations. However, PSGen generates much less kinematics channels than carlomat_4.4. This results in much shorter time of the code generation and compilation and shorter program execution time.

Preparation for running and program usage

PSGen is distributed as a single tar.gz archive PSGen.tgz which can be downloaded from the CPC Program Library or from:
http://kk.us.edu.pl/PSGen.html.
When extracted 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 psgen and all other handwritten files listed in a makefile of mc_computation.

Preparation for running and program usage

PSGen is distributed as a single tar.gz archive PSGen.tgz which can be downloaded from the CPC Program Library or from:
http://kk.us.edu.pl/PSGen.html.
When extracted 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 psgen and all other handwritten files listed in a makefile of mc_computation.

- Set the process and desired options in PSGen.f and execute
  make code
  from the command line in code_generation.

Preparation for running and program usage

PSGen is distributed as a single `tar.gz` archive `PSGen.tgz` which can be downloaded from the CPC Program Library or from:
`http://kk.us.edu.pl/PSGen.html.`
When extracted 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 `psgen` and all other handwritten files listed in a makefile of `mc_computation`.

- Set the process and desired options in `PSGen.f` and execute
  `make code`
  from the command line in `code_generation`.

The template MC integration program

A template program `PSGen_test_mpi` allows to perform the MC integration, utilizing the Message Passing Interface (MPI), of a template function `cs_sect(x,ndim)`, both located in directory `mc_computation`.

Function `cs_sect(x,ndim)` calls the kinematics routine `psgen`.

The template MC integration program

A template program `PSGen_test_mpi` allows to perform the MC integration, utilizing the Message Passing Interface (MPI), of a template function `cs_sect(x,ndim)`, both located in directory `mc_computation`.

Function `cs_sect(x,ndim)` calls the kinematics routine `psgen`.

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 of the considered process, restricted by kinematics cuts, at the centre of mass energies defined in `PSGen_test_mpi` by array `aecm(ne)`.

The template MC integration program

A template program PSGen_test_mpi allows to perform the MC integration, utilizing the Message Passing Interface (MPI), of a template function cs_sect(x,ndim), both located in directory mc_computation.

Function cs_sect(x,ndim) calls the kinematics routine psgen.

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 of the considered process, restricted by kinematics cuts, at the centre of mass energies defined in PSGen_test_mpi by array aecm(ne).

The cuts can be defined in the subroutine define_cuts and imposed by setting icuts=1 in PSGen_test_mpi.

The template MC integration program

A template program PSGen_test_mpi allows to perform the MC integration, utilizing the Message Passing Interface (MPI), of a template function cs_sect(x,ndim), both located in directory mc_computation.

Function cs_sect(x,ndim) calls the kinematics routine psgen.

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 of the considered process, restricted by kinematics cuts, at the centre of mass energies defined in PSGen_test_mpi by array aecm(ne).

The cuts can be defined in the subroutine define_cuts and imposed by setting icuts=1 in PSGen_test_mpi.

Details on how to prepare PSGen_test_mpi for running are given on http://kk.us.edu.pl/PSGen.html.

The template MC integration program

A template program PSGen_test_mpi allows to perform the MC integration, utilizing the Message Passing Interface (MPI), of a template function cs_sect(x,ndim), both located in directory mc_computation.
Function cs_sect(x,ndim) calls the kinematics routine psgen.
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 of the considered process, restricted by kinematics cuts, at the centre of mass energies defined in PSGen_test_mpi by array aecm(ne).
The cuts can be defined in the subroutine define_cuts and imposed by setting icuts=1 in PSGen_test_mpi.
Details on how to prepare PSGen_test_mpi for running are given on http://kk.us.edu.pl/PSGen.html.
It can be run with a command

```
make mc
```

from the command line in mc_computation.

The template MC integration program

A template program PSGen_test_mpi allows to perform the MC integration, utilizing the Message Passing Interface (MPI), of a template function cs_sect(x,ndim), both located in directory mc_computation.
Function cs_sect(x,ndim) calls the kinematics routine psgen.
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 of the considered process, restricted by kinematics cuts, at the centre of mass energies defined in PSGen_test_mpi by array aecm(ne).
The cuts can be defined in the subroutine define_cuts and imposed by setting icuts=1 in PSGen_test_mpi.
Details on how to prepare PSGen_test_mpi for running are given on http://kk.us.edu.pl/PSGen.html.
It can be run with a command

```
make mc
```

from the command line in mc_computation.

Summary and outlook

Efficient integration of the multiparticle reaction cross sections over a multidimensional phase space is a challenge.

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 $2 \to n$ scattering process, with $n = 2, ..., 9$, provided by the user.

Summary and outlook

Efficient integration of the multiparticle reaction cross sections over a multidimensional phase space is a challenge.

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 $2 \to n$ scattering process, with $n = 2, ..., 9$, provided by the user.

The program is written in Fortran 90/95. It uses a new very fast algorithm that automatically generates calls to Fortran subroutines containing different phase space parameterizations of the considered class of processes.

Summary and outlook

Efficient integration of the multiparticle reaction cross sections over a multidimensional phase space is a challenge.

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 $2 \to n$ scattering process, with $n = 2, ..., 9$, provided by the user.

The program is written in Fortran 90/95. It uses a new very fast algorithm that automatically generates calls to 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.

Summary and outlook

Efficient integration of the multiparticle reaction cross sections over a multidimensional phase space is a challenge.
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 $2 \to n$ scattering process, with $n = 2, ..., 9$, provided by the user.
The program is written in Fortran 90/95. It uses a new very fast algorithm that automatically generates calls to 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.

Summary and outlook

- The individual subroutines are organized in a single multichannel kinematics subroutine which can be easily called, as a function of generated particle four momenta, while computing the phase space integral of the S matrix element 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.
- If necessary, user defined parameterizations can be included in the program by hand.

Summary and outlook

- The individual subroutines are organized in a single multichannel kinematics subroutine which can be easily called, as a function of generated particle four momenta, while computing the phase space integral of the S matrix element 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.
- If necessary, user defined parameterizations can be included in the program by hand.
- Inclusion of new phase space parameterizations is planed.

Summary and outlook

- The individual subroutines are organized in a single multichannel kinematics subroutine which can be easily called, as a function of generated particle four momenta, while computing the phase space integral of the S matrix element 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.
- If necessary, user defined parameterizations can be included in the program by hand.
- Inclusion of new phase space parameterizations is planed.
- If there is a need, C++ version of PSGen will be written.

Summary and outlook

- The individual subroutines are organized in a single multichannel kinematics subroutine which can be easily called, as a function of generated particle four momenta, while computing the phase space integral of the S matrix element 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.
- If necessary, user defined parameterizations can be included in the program by hand.
- Inclusion of new phase space parameterizations is planed.
- If there is a need, C++ version of PSGen will be written.

Thank you for your attention.

Summary and outlook

- The individual subroutines are organized in a single multichannel kinematics subroutine which can be easily called, as a function of generated particle four momenta, while computing the phase space integral of the S matrix element 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.
- If necessary, user defined parameterizations can be included in the program by hand.
- Inclusion of new phase space parameterizations is planed.
- If there is a need, C++ version of PSGen will be written.

Thank you for your attention.