Manual for the JHU generator

For simulation of a single-produced resonance at hadron colliders
(version 2.2.x, release date October 30, 2012)

The generator from [1, 2] is a model-independent generator for studying spin and parity properties of new resonances. Please cite [1, 2] if using the generator.

The code can be downloaded from [3]. The generator outputs LHE files which can be passed to parton shower programs for hadronization. The generator purposely does not output sensible cross-sections but rather deals with numbers of events which can be compared for different signal hypotheses.

Additionally, the package now includes code for computing the matrix elements standalone which can be used in a numerical matrix element analysis.

Contents

I. Installation 2

II. Configuration 2
- A. Command line configuration 2
- B. Configuration in parameter file 3
  - 1. General parameters 3
  - 2. Spin-0 parameters 4
  - 3. Spin-1 parameters 4
  - 4. Spin-2 parameters 4

III. Examples 5
- A. $J^P = 0^+_m$ resonance, $X \to ZZ$ or WW 5
- B. $J^P = 0^-_m$ resonance, $X \to ZZ$ or WW 5
- C. $J^P = 0^+_m$ resonance, $X \to \gamma\gamma$ 5
- D. $J^P = 0^-_m$ resonance, $X \to \gamma\gamma$ 6
- E. $J^P = 2^+_m$ resonance, $X \to ZZ$ or WW or $\gamma\gamma$ 6

IV. JHU Generator Matrix Elements (JHUGenME) 6

V. Release notes 7

References 7

I. INSTALLATION

Register and download the package from `www.pha.jhu.edu/spin` and untar the file. Go to the directory `JHUGenerator` where the code exists for generating events with the JHU Generator. In the `makefile`, you have two options for compiler, `Comp = ifort` or `Comp = gfort`. Then simply compile with:

```
$ make
```

II. CONFIGURATION

There are two ways to configure the program, either from the command line or in the file `mod_Parameters.F90`. The command line configurables are defined in the file `main.F90`. When one change the fortran code directly, one should also recompile the code for changes to take effect. In general, command-line configuration handles general event properties while the configuration file handles all of the couplings and physics handles.

A. Command line configuration

The list of command line configurables and the default values are (also defined in the `README`):

```
 - Collider=x    (Collider: 1=LHC, 2=Tevatron), default value:1
 - VegasNc1=x    (number of evaluations), default value:5000000
 - PChannel=x    (partonic channel: 0=glu+glu, 1=quark+antiquark, 2=both), default value:2
 - PDFSet=x      (parton distr. functions, 1=CTEQ6L1 (2001), 2=MSTW(2008),
  2xx=MSTW with eigenvector set xx=01..40), default=1
 - DecayMode1=x  (decay mode for vector boson 1), default x=0
 - DecayMode2=x  (decay mode for vector boson 2), default x=0
               x=0:  X-> Z1 Z2, Z1->2l,
               x=1:  X-> Z1 Z2, Z1->2q,
               x=2:  X-> Z1 Z2, Z1->2tau,
               x=3:  X-> Z1 Z2, Z1->2nu,
               x=4:  X-> W1 W2, W1->lnu,
               x=5:  X-> W1 W2, W1->2q,
               x=6:  X-> W1 W2, W1->taunu
               x=7:  X-> gamma gamma
 - DataFile=x    (name of the output file), default value: "output"
 - Process=x     (gives the resonance spin 0,1 or 2)
 - OffXVV=xyz    (only applies to Higgs production, x,y,z can be 1 or 0 indicating real off-shellness
  for Higgs, Z/W boson 1/2, resp.), default 000 if a parameter is not specified
 - Unweighted=x  (generates weighted or unweighted events), default value: ".true."
```

A few more details on some particular parameters:

- `VegasNc1`: This is the number of evaluations for the program to make, NOT the number of events generated. The efficiency depends on the set of parameters one uses.
- `OffXVV`: The program does not work for ZZ or WW if you set them to be on-shell (`OffXVV="\*00"`) and the mass of the X resonance to be below the m_{VV} threshold. In general, the more off-shell the process, or the more "1" you have, the less efficient the `VegasNc1` evaluations are. Specifically, if you are interested then, in producing a resonance with mass below threshold m_{VV} with a very narrow resonance, it is most efficient to generate with `OffXVV="011"`
- `PChannel`: This parameter is only meaningful in the spin-2 case. For spin-0, production is possible only via the gg process and for spin-1, production is only possible via the $q\bar{q}$ process.
- `DecayMode2=7` note: Valid for spin-0 and spin-2, only OffXVV=000 or 100 are possible.

Then, as an example of running the generator, you could do:

```
./JHUGen Collider=1 Process=0 VegasNc1=100000 PChannel=0 OffXVV=011 DecayMode1=0 DecayMode2=0 \\
Unweighted=.true. DataFile=test1
```

N.B. There is a beta-version of the generator which has improved efficiency for the generation. However, it is currently only available for gluon-gluon initiated processes. It is by default turned off, but it can be accessed in the file `main.F90`.

```
logical,parameter :: useBetaVersion=.false.
```

B. Configuration in parameter file

The `mod_Parameters.F90`, one does all the configuration of the couplings of the resonance.

1. *General parameters*

Each generation run is different when this is `.true.`

```
seed_random = .true.
```

In the case when `PChannel=2` for a spin-2 resonance, the user can define the ratio of the production of gg and $q\bar{q}$ production.

```
fix_channels_ratio = .true.
channels_ratio_fix = 0.25d0    ! desired ratio of
                               ! N_qq/(N_qq+N_gg)
```

Only for the 4l final state, one can include interference effects between the leptons (*N.B. This feature is yet to be fully validated*). The interference is controlled by the parameter:

```
includeInterference=.false.
```

The remaining parameters are more-or-less self-explanatory.

```
! we are using units of 100GeV, i.e. Lambda=10 is 1TeV
M_Z     = 91.1876d0 *GeV      ! Z boson mass (PDG-2011)
Ga_Z    = 2.4952d0  *GeV      ! Z boson width(PDG-2011)
M_W     = 80.399d0  *GeV      ! W boson mass (PDG-2011)
Ga_W    = 2.085d0   *GeV      ! W boson width(PDG-2011)
M_Reso  = 125d0     *GeV      ! X resonance mass (spin 0, spin 1, spin 2)
Ga_Reso = 5d0       *GeV      ! X resonance width
Lambda  = 1000d0    *GeV      ! Lambda coupling enters in two places
  ! overal scale for x-section and in power suppressed
! operators/formfactors (former r).
alpha_QED = 1d0/128.0d0       ! el.magn. coupling
sitW = dsqrt(0.23119d0)       ! sin(Theta_Weinberg) (PDG-2008)
Mu_Fact = M_Reso              ! pdf factorization scale
LHC_Energy=7000d0  *GeV       ! LHC hadronic center of mass energy
TEV_Energy=1960d0  *GeV       ! Tevatron hadronic center of mass energy

Br_Z_up = 0.1657d0  ! branching fraction Ga(up)/Ga(hadronic)
Br_Z_ch = 0.1657d0  ! branching fraction Ga(charm)/Ga(hadronic)
Br_Z_dn = 0.2229d0  ! branching fraction Ga(down)/Ga(hadronic)
Br_Z_st = 0.2229d0  ! branching fraction Ga(strange)/Ga(hadronic)
Br_Z_bo = 1d0-Br_Z_up-Br_Z_ch-Br_Z_dn-Br_Z_st  ! branching fraction Ga(bottom)/Ga(hadronic)
```

2. *Spin-0 parameters*

The `*hg*` parameters control the coupling of a spin-0 resonance to gluons in the production mechanism. The `*hz*` parameters control the decay. In practice, the production parameters are not having a large effect since angular corrections from the production mechanism are lost for spinless particles. One has the options to set the spin-0 couplings either from Eq.(9) or Eq.(11) from Ref. [2]. To switch between the two, use the parameter `generate_as`.

```
   logical, public, parameter :: generate_as = .false.

!-- parameters that define on-shell spin 0 coupling to SM fields, see note Eq.(1)
   complex(8), public, parameter :: ahg1 = (1.0d0,0d0)
   complex(8), public, parameter :: ahg2 = (0.0d0,0d0)
   complex(8), public, parameter :: ahg3 = (0.0d0,0d0)  ! pseudoscalar
   complex(8), public, parameter :: ahz1 = (1.0d0,0d0)
   complex(8), public, parameter :: ahz2 = (0.0d0,0d0)
   complex(8), public, parameter :: ahz3 = (0.0d0,0d0)  ! pseudoscalar

!-- parameters that define off-shell spin 0 coupling to SM fields, see note Eq.(2)
   complex(8), public, parameter :: ghg2 = (1.0d0,0d0)
   complex(8), public, parameter :: ghg3 = (0.0d0,0d0)
   complex(8), public, parameter :: ghg4 = (0.0d0,0d0)   ! pseudoscalar
   complex(8), public, parameter :: ghz1 = (1.0d0,0d0)
   complex(8), public, parameter :: ghz2 = (0.0d0,0d0)
   complex(8), public, parameter :: ghz3 = (0.0d0,0d0)
   complex(8), public, parameter :: ghz4 = (0.0d0,0d0)   ! pseudoscalar
```

3. *Spin-1 parameters*

The parameters below represent the couplings given in Eq. (16) from Ref. [2]. The `*left*` and `*right*` parameters control the production of the spin-1 resonance while the `*_v` and `*_a` parameters control the decay.

```
!---parameters that define spin 1 coupling to SM fields, see note
   complex(8), public, parameter :: zprime_qq_left  = (1.0d0,0d0)   !  see note Eq. (4)
   complex(8), public, parameter :: zprime_qq_right = (0.0d0,0d0)
   complex(8), public, parameter :: zprime_zz_v =  (1.0d0,0d0)!  =1 for JP=1-
   complex(8), public, parameter :: zprime_zz_a =  (0.0d0,0d0)!  =1 for JP=1+
```

4. *Spin-2 parameters*

The `a*` parameters control the coupling of a spin-2 resonance to gluons in the production mechanism. The `b*` and `c*` parameters control the decay. One has the options to set the spin-2 couplings either from Eq.(18) or Eq.(19) from Ref. [2]. To switch between the two, use the parameter `generate_bis`.

```
  logical, public, parameter :: generate_bis = .true.
  logical, public, parameter :: use_dynamic_MG = .true. ! .true. (=default),
    ! the spin-2 resonance mass with MG^2=(p1+p2)^2, otherwise fixed at M_Reso^2.

  complex(8), public, parameter :: a1 = (1.0d0,0d0)    ! g1  -- c.f. note
  complex(8), public, parameter :: a2 = (0.0d0,0d0)    ! g2
  complex(8), public, parameter :: a3 = (0.0d0,0d0)    ! g3
  complex(8), public, parameter :: a4 = (0.0d0,0d0)    ! g4
  complex(8), public, parameter :: a5 = (0.0d0,0d0)    ! pseudoscalar, g8

  complex(8), public, parameter :: graviton_qq_left  = (1.0d0,0d0)! graviton coupling to quarks
  complex(8), public, parameter :: graviton_qq_right = (1.0d0,0d0)
```

```
complex(8), public, parameter :: b1 = (1.0d0,0d0)
complex(8), public, parameter :: b2 = (0.0d0,0d0)
complex(8), public, parameter :: b3 = (0.0d0,0d0)
complex(8), public, parameter :: b4 = (0.0d0,0d0)
complex(8), public, parameter :: b5 = (0.0d0,0d0)
complex(8), public, parameter :: b6 = (0.0d0,0d0)
complex(8), public, parameter :: b7 = (0.0d0,0d0)
complex(8), public, parameter :: b8 = (0.0d0,0d0)
complex(8), public, parameter :: b9 = (0.0d0,0d0)
complex(8), public, parameter :: b10 =(0.0d0,0d0)


complex(8), public, parameter  :: c1 = (1.0d0,0d0)
complex(8), public, parameter  :: c2 = (0.0d0,0d0)
complex(8), public, parameter  :: c3 = (0.0d0,0d0)
complex(8), public, parameter  :: c41= (0.0d0,0d0)
complex(8), public, parameter  :: c42= (0.0d0,0d0)
complex(8), public, parameter  :: c5 = (0.0d0,0d0)
complex(8), public, parameter  :: c6 = (0.0d0,0d0)
complex(8), public, parameter  :: c7 = (0.0d0,0d0)
```

III. EXAMPLES

The below examples are not meant to be a complete set, but rather some interesting and relevant cases. In many cases, the example is not the only way to produce such a scenario.

A. $J^P = 0^+_m$ resonance, $X \to ZZ$ or WW

```
logical, public, parameter :: generate_as = .true.

complex(8), public, parameter :: ahg1 = (1.0d0,0d0)
complex(8), public, parameter :: ahg2 = (0.0d0,0d0)
complex(8), public, parameter :: ahg3 = (0.0d0,0d0)  ! pseudoscalar
complex(8), public, parameter :: ahz1 = (1.0d0,0d0)
complex(8), public, parameter :: ahz2 = (0.0d0,0d0)
complex(8), public, parameter :: ahz3 = (0.0d0,0d0)  ! pseudoscalar
```

B. $J^P = 0^-_m$ resonance, $X \to ZZ$ or WW

```
logical, public, parameter :: generate_as = .true.

complex(8), public, parameter :: ahg1 = (1.0d0,0d0)
complex(8), public, parameter :: ahg2 = (0.0d0,0d0)
complex(8), public, parameter :: ahg3 = (0.0d0,0d0)  ! pseudoscalar
complex(8), public, parameter :: ahz1 = (0.0d0,0d0)
complex(8), public, parameter :: ahz2 = (0.0d0,0d0)
complex(8), public, parameter :: ahz3 = (1.0d0,0d0)  ! pseudoscalar
```

C. $J^P = 0^+_m$ resonance, $X \to \gamma\gamma$

In practice, the example $X \to \gamma\gamma$ from this section, Sec. III C and the next Sec. III D are kinematically the same but are presented only to illustrate how one takes care of this final state.

```
 logical, public, parameter :: generate_as = .false.

 complex(8), public, parameter :: ghg2 = (1.0d0,0d0)
 complex(8), public, parameter :: ghg3 = (0.0d0,0d0)
 complex(8), public, parameter :: ghg4 = (0.0d0,0d0)   ! pseudoscalar
 complex(8), public, parameter :: ghz1 = (0.0d0,0d0)
 complex(8), public, parameter :: ghz2 = (1.0d0,0d0)
 complex(8), public, parameter :: ghz3 = (0.0d0,0d0)
 complex(8), public, parameter :: ghz4 = (0.0d0,0d0)   ! pseudoscalar
```

D. $J^P = 0^-_m$ resonance, $X \to \gamma\gamma$

```
 logical, public, parameter :: generate_as = .false.

 complex(8), public, parameter :: ghg2 = (1.0d0,0d0)
 complex(8), public, parameter :: ghg3 = (0.0d0,0d0)
 complex(8), public, parameter :: ghg4 = (0.0d0,0d0)   ! pseudoscalar
 complex(8), public, parameter :: ghz1 = (0.0d0,0d0)
 complex(8), public, parameter :: ghz2 = (0.0d0,0d0)
 complex(8), public, parameter :: ghz3 = (0.0d0,0d0)
 complex(8), public, parameter :: ghz4 = (1.0d0,0d0)   ! pseudoscalar
```

E. $J^P = 2^+_m$ resonance, $X \to ZZ$ or WW or $\gamma\gamma$

```
complex(8), public, parameter :: a1 = (1.0d0,0d0)    ! g1  -- c.f. draft
complex(8), public, parameter :: a2 = (0.0d0,0d0)    ! g2
complex(8), public, parameter :: a3 = (0.0d0,0d0)    ! g3
complex(8), public, parameter :: a4 = (0.0d0,0d0)    ! g4
complex(8), public, parameter :: a5 = (0.0d0,0d0)    ! pseudoscalar, g8
complex(8), public, parameter :: graviton_qq_left  = (1.0d0,0d0)! graviton coupling to quarks
complex(8), public, parameter :: graviton_qq_right = (1.0d0,0d0)

logical, public, parameter :: generate_bis = .true.
logical, public, parameter :: use_dynamic_MG = .true.

complex(8), public, parameter :: b1 = (1.0d0,0d0)
complex(8), public, parameter :: b2 = (0.0d0,0d0)
complex(8), public, parameter :: b3 = (0.0d0,0d0)
complex(8), public, parameter :: b4 = (0.0d0,0d0)
complex(8), public, parameter :: b5 = (1.0d0,0d0)
complex(8), public, parameter :: b6 = (0.0d0,0d0)
complex(8), public, parameter :: b7 = (0.0d0,0d0)
complex(8), public, parameter :: b8 = (0.0d0,0d0)
complex(8), public, parameter :: b9 = (0.0d0,0d0)
complex(8), public, parameter :: b10 =(0.0d0,0d0)
```

IV. JHU GENERATOR MATRIX ELEMENTS (JHUGENME)

After extracting the code, you can go to the directory `JHUGenME` to find code for computing matrix elements directly. To compile the code, simple do:

```
$ make
```

Please take note: The setup is configured for `gfort + gcc version 4.1.2 20080704 (Red Hat 4.1.2-50)` and it is highly dependent on the compiler version. Please configure for your own setup accordingly. (Using `'nm'` command will help decipher the module names you will need)

The usage of the package is straight-forward and an example is given in `testprogram.c`. There are 4 main modules:

- "modhiggs__evalamp_gg_h_vv": spin-0 matrix elements for gg initiated processes
- "modzprime__evalamp_qqb_zprime_vv": spin-1 matrix elements for $q\bar{q}$ initiated processes
- "modgraviton__evalamp_gg_g_vv: spin-2 matrix elements for gg initiated processes
- "modgraviton__evalamp_qqb_g_vv: spin-2 matrix elements for $q\bar{q}$ initiated processes

The inputs are the 4-vectors of the incoming patrons and outgoing particles in the CM frame of the object X. In addition the mass and width of the resonance are required as well as the ID of the outgoing particles. Finally the last set of inputs are the couplings themselves. They are arrays for parameters for a given spin hypothesis which mirror the parameters configurable in `mod_Parameters.F90`. As an example, the arrays are initialized in `testprogram.c` as:

```
double Hggcoupl[3];
double Hvvcoupl[4];
double Zqqcoupl[2];
double Zvvcoupl[2];
double Gqqcoupl[2];
double Gggcoupl[5];
double Gvvcoupl[10];
```

V. RELEASE NOTES

In going from `v2.2.x` to `v2.1.3`, the updates are as follows:

- Fix interference and randomization in the *beta* version
- Add the `JHUGenME` modules
- Small change for compilation on Mac OSX platforms
- Fix for tau masses in W decays

In going from `v2.0.2` to `v2.1.x`, the updates are as follows:

- Histograms are written in file (default: ./data/output.dat) and no longer on the screen. How to understand the histogram data and how to plot is briefly described in the output.dat file.
- Added tau masses
- Added lepton interference in the ZZ4l final state
- Added switch `generate_as` to choose couplings in spin-0 case (works for on- and off-shell resonance). The default is ".false.".
- Added the possibility to change graviton-quark couplings. The new parameters are `graviton_qq_left`, `graviton_qq_right` and correspond to $0.5*(1-\gamma^5)$ and $0.5*(1+\gamma^5)$ helicity projectors, respectively. Up to now the coupling was always vector-like. This is also the new default, `graviton_qq_left = graviton_qq_right =1`.
- The random seed is now fixed with gfortran.
- The call "./JHUGen help" prints out all available command line options
- Added new command line option "Unweighted=0 or 1" (default is 1)

[1] Y.Y. Gao, A. V. Gritsan, Z.J. Guo, K. Melnikov, M. Schulze and N. V. Tran, "Spin-Determination of Single-Produced Resonances at Hadron Colliders". Phys. Rev. D **81**, 075022 (2010). arXiv:1001.3396 [hep-ph].

[2] S. Bolognesi, Y.Y. Gao, A. V. Gritsan, K. Melnikov, M. Schulze, N. V. Tran and A. Whitbeck, "On the Spin and Parity of Single-Produced Resonance at the LHC". arXiv:1208.4018 [hep-ph].

[3] See webpage: www.pha.jhu.edu/spin