MCSANC v1.0 Users Guide

August 3, 2012

1 Installation

MCSANC installation is performed in a standard way for autotools based code. The LHAPDF [1] is required to be installed before starting the installation. After downloading and unpacking the tarball with MCSANC from http://... perform the following steps.

```
cd mcsanc_vX.Y.Z
./configure [--with-LHAPDF=/custom/path/to/LHAPDF/installation]
make
```

If succeeded the executable will be created in `./src` directory. The program is launched with a command from `./share`:

```
cd ./share
../src/mcsanc [custom-input.cfg]
```

Upon completion an output file`mcsanc-[run_tag]-output.txt` with final results, run parameters [and histograms -¿ remove?] will be created in the current directory.

2 Configuration

The MCSANC program reads to config files upon start: `input.cfg` and `ewparam.cfg`. The first file, `input.cfg`, contains general steering parameters for a run, vegas parameters, kinematic cuts and histogram parameters organized in FORTRAN namelists.

2.1 Process namelist

processId defines a process to calculate (for available processes see section 3)(`integer`).

run_tag is an arbitrary string value (`character*256`).

sqs0 sets the beam energy at CMS (`double precision`).

beams(2) defines the incoming beams. So far only proton-proton beams are supported, which corresponds to `beams = 1,1` (`integer`).

PDFSet allows to set the parton density functions connected via LHAPDF (`character*256`).

PDFMember selects a member of PDF set. Usually 0 is used, which corresponds to central value (`integer`).

iflew(5) flags controlling electroweak components of the NLO EW computations. See below. (`integer`).

iqed =
0: disables QED corrections
1: with full QED corrections
2: only initial state QED radiation (ISR)
3: initial-final QED radiation interference term (IFI)
4: only final state QED radiation (FSR)
5: sum of initial and final state radiation contributions [IFI+FSR] (only for pid=1-3,±101-103)
6: sum of initial state and initial-final QED interference terms [ISR+IFI] (only for pid=1-3)

iew = 0/1 corresponds to disabled/enabled weak corrections.

iborn 0 or 1 selects respectively LO or NLO level of calculations.

ifgg = -1/0 – off/Born level for $F_{\gamma\gamma}$, 1/2 – [1+$F_{\gamma\gamma}$(NLO)]/[1/(1 - $F_{\gamma\gamma}$(NLO))]

irun = 0/1 corresponds to fixed/running gauge boson width

iflqcd controlls if QCD NLO is calculated (`integer`).

imsb allows to select a subtraction scheme: 0/1/2 = none/MSbar/DIS (`integer`).

irecomb electron recombination option: 0/1/2 = off/on/both histograms produced with on and off(`integer`).

2.2 VegasPar namelist

`VegasPar` regulates vegas parameters:

relAcc relative accuracy limit on the calculated cross section (`double precision`)

absAcc absolute accuracy limit on the cross section (`double precision`)

nstart number of integrand evaluations performed at first iteration (`integer`)

nincrease increment of integrand evaluations per iteration (`integer`)

nExplore number of evaluations for grid exploration (`integer`)

maxEval maximum number of evaluations (`integer`)

seed random generator seed (`integer`)

flags Cuba specific flags. See Cuba manual for more information [2] (`integer`)

Different contributions to NLO EW cross sections (real, virtual) can differ from each other by orders of magnitude. The combination of relative and absolute accuracy limits gives a suitable way to control accuracy of total cross section without need to limit each contribution. For example, the goal is 0.1% relative accuracy on total cross section. The user can set this value to the `relAcc` option with an allowance for summing of the contributions. The absolute error limit can then be calculated multiplying this value by the largest estimated contribution (usually born). This will cut off unnecessary long calculations for smaller contributions.

2.3 KinCuts namelist

This config section contains typical kinematic cuts for fiducial cross section calculation. Besides original cuts, the user is free to implement his own in the `src/kin_cuts.f`. The cut values are set under corresponding name in the table in `input.cfg`

```
&KinCuts
  cutName       = 'm34',  'mtr',  'pt3',  'pt4',  'eta3', 'eta4', 'dR'
  cutFlag       = 1,      0,      1,      1,      1,      1,      1
  cutLow        = 66d0,   66d0,   20d0,   20d0,   -2.5d0, -2.5d0, 0d0
  cutUp         = 116d0,  116d0,  7.d3,   7.d3,   2.5d0,  2.5d0,  1d-1
/
```

The `cutName` fields are fixed, `cutFlag` - 0/1 on/off cuts. By default seven kinematic cuts are foreseen. They are: invariant `m34` and transversal `mtr` masses, transversal momenta `pt3, pt4` rapidities `eta3, eta4` of the final particles and `dR` parameter to calculate recombined lepton momentum.

2.4 Histograms with equidistant bins

Namelist `FixedBinHist` allows to control histograms with fixed step. The histogram parameters are defined in a table: under the histogram name, the user can set printing flag, min and max histogram values and a step. The binary printing flag controlls if histogram is printed in the output (first bit) and if it's normalized to the bin width (second bit). For example to print unnormalized histograms set the flag to 1, to print normalized to 3.

```
&FixedBinHist
  fbh_name      = 'm34', 'mtr', 'pt34', 'pt3' 'pt4', 'et3',   'et4'
  fbh_flag      = 3,     3,     0,      3,     0,     3,       0
  fbh_low       = 66d0,  66d0,  20d0,   20d0,  20d0, -2.5d0,  -2.5d0
  fbh_up        = 116d0, 116d0, 58d0,   58d0,  58d0,  2.5d0,   2.5d0
  fbh_step      = 2d0,   2d0,   1d0,    1d0,   1d0,   0.125d0, 0.125d0
/
```

By default seven fixed bin histograms are foreseen. They are: invariant `m34` and transversal `mtr` masses, transversal momenta `pt3, pt4` rapidities `eta3, eta4` of the final particles and transversal momenta `pt34` of intermediate boson.

Any user defined histograms can be implemented in the `src/kin_cuts.f` file. There the histogram names have no technical meaning, the requirement is that the order of histograms in the input file must correspond to the histogram array `hist_val` and the `nhist` variable set to the number of requested histograms.

2.5 Histograms with variable bins

Variable bin histograms are defined in namelist `VarBinHist`. The format is different from `FixedBinHist`. The user have to set total number of variable step histograms with `nvbh` parameter. Then follows definition of corresponding to i'th histogram name (`vbh_name(i)`), printing flag (`vbh_flag(i)`), number of bins (`vbh_nbins(i)`) and array for the bin edges (`vbh_bins(i,1:[nbins+1]`) as a space separated string of double precision numbers. It is important that the variable bin histograms are available only with `gcc-v4.5.0` and higher.

```
&VarBinHist
  nvbh         = 7,

  vbh_name(1) = 'm34',
  vbh_flag(1) = 3,
  vbh_nbins(1) = 6,
  vbh_bins(1,1:7) = 50d0 55d0 60d0 70d0 100d0 200d0,

  vbh_name(2) = 'mtr',
  vbh_flag(2) = 0,
  vbh_nbins(2) = 6,
  vbh_bins(2,1:7) = 50d0 55d0 60d0 70d0 100d0 200d0,

  vbh_name(3) = 'pt34',
  vbh_flag(3) = 3,
  vbh_nbins(3) = 9,
  vbh_bins(3,1:10) = 25d0 30d0 50d0 70d0 90d0 110d0 130d0 150d0 200d0,

  ...

/
```

By default seven variable bin histograms are foreseen as for `FixedBinHist`.

2.6 Electroweak parameters

Electroweak parameters are set in ewparams.cfg config file. It contains `&EWPars` name list with the following fields.

gfscheme flag selects electroweak scheme in which the calculation is performed (`integer`). The possible values are 0:$\alpha(0)$-scheme, 1:G_μ-scheme ($\alpha(G_\mu)$ in Pure Weak NLO part), 2:G'_μ-scheme ($\alpha(0)$ in Pure Weak NLO part), 3:(G''_μ ($\alpha(G_\mu)$ in soft and hard parts), 4:$\alpha(M_Z)$-scheme (only for pid=001,002,003).

fscale is the factorization scale (`double precision`). When less than zero, the value appropriate for requested process is set (e.g. final lepton invariant mass for DY process). Otherwise the given value is used.

rscale - renormalization scale defined with the same rules as `fscale` (`double precision`).

ome is the parameter separating contributions from soft and hard photon radiation (`double precision`). The total NLO (EW,QCD) cross section must be independent on it.

alpha, gf, alphas, conhc - a list of constants and coefficients: α_{EM}, G_μ, α_S, conversion constant (`double precision`)

mw, mz, mh boson masses W, Z, higgs (`double precision`)

wz, ww, wh, wtp widths for bosons W, Z, higgs and the top quark (`double precision`)

Vqq CKM matrix (`double precision`)

men, mel, mmn, mmo, mtn, mta leptonic masses $\nu_e, e, \nu_\mu, \mu, \nu_\tau, \tau$(`double precision`)

mdn, mup, mst, mch, mbt, mtp quark masses u, d, s, c, b, t(`double precision`)

mfact quark mass factor(`double precision`). The total NLO EW cross sections should be independent on this parameter as the quark masses should cancel.

rmf1 is a 12 element array of fermion masses for technical purposes (`double precision`)

Vegas grids have to be generated from scratch if these parameters are changed.

3 Processes included in MCSANC

pid	$f(p_1) + f(p_2) \to$	ref.
01	$e^+(p_3) + e^-(p_4)$	[3, 4]
02	$\mu^+(p_3) + \mu^-(p_4)$	
03	$\tau^+(p_3) + \tau^-(p_4)$	
04	$Z^0(p_3) + H(p_4)$	[5]
101	$e^+(p_3) + \nu_e(p_4)$	[6, 4]
102	$\mu^+(p_3) + \nu_\mu(p_4)$	
103	$\tau^+(p_3) + \nu_\tau(p_4)$	
-101	$e^-(p_3) + \bar{\nu}_e(p_4)$	
-102	$\mu^-(p_3) + \bar{\nu}_\mu(p_4)$	
-103	$\tau^-(p_3) + \bar{\nu}_\tau(p_4)$	
104	$W^+(p_3) + H(p_4)$	-
-104	$W^-(p_3) + H(p_4)$	
105	$t(p_3) + \bar{b}(p_4)$ (s-channel)	[7, 8]
106	$t(p_3) + q(p_4)$ (t-channel)	
-105	$\bar{t}(p_3) + b(p_4)$ (s-channel)	
-106	$\bar{t}(p_3) + q(p_4)$ (t-channel)	

4 Persistency

Cuba library [2] allows to save the vegas grid for further restoration. The integration stops when requested accuracy is reached. The last version of the grid and histograms state are saved in `*.vgrid` and `*.hist` files. If the user desires to improve precision he can set the new limits in `input.cfg` and restart the program in the directory with saved grids. Care should be taken that all other configs (PDF, electroweak parameters) are remained the same. If the user wishes to change the histograms he can use old grids, but remove `*.hist` files and fill the histograms from scratch. The histogram errors will correspond to less precision than the total cross section.

The grid exploration stage is omitted if grid state file is found.

5 Output

The main output of the program will be written to `mcsanc-[run_tag]-output.txt` file. The file contains final table with cross sections from the components and the summed total cross section. After the table follows the CPU usage, and a list of input parameters, followed by a list of cuts applied in the process of integration.

The remaining of the output file is a section containing Histograms listing. Each histogram header contains histogram name and its integral with an error. The histograms are printed bin-by-bin in the order: bin edges, lower and upper, value and statistical error.

Each histogram is output separately in a text file for plotting convenience. The file name format is `[run_tag]_[hist_name]-XXX`, where XXX stands for `born, tota` or `delt` for LO, NLO and $\delta\%$ histograms.

6 Multiprocess calculations

The Cuba library [2], used as a Monte-Carlo integrating tool in the MCSANC program, supports multiprocessing. Having a multicore CPU the user can request the program to run on several cores by specifying `CUBACORES` environment variable.

Due to the features of interprocess communications, the efficiency of multiprocessing is not always optimal. Figure 1 shows NLO EW cross section calculation time depending on the number of CPU cores active. The test was run on a dual-processor Intel® Xeon® machine with 12 real (24 virtual) cores with Linux operating system.

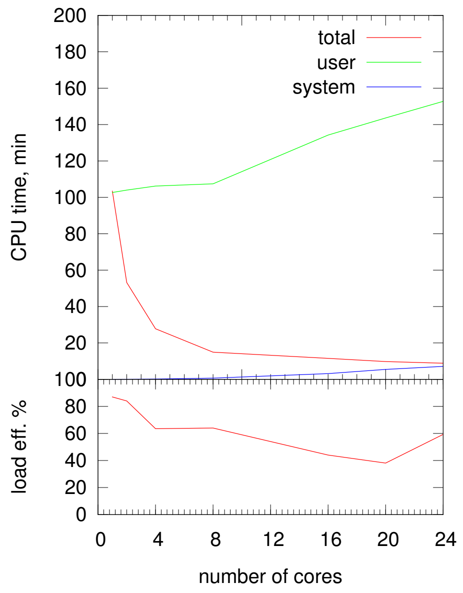

Figure 1: CPU usage and load efficiency for MCSANC program depending on the number of processor cores.

The upper plot summarizes multicore CPU productivity: "total" is the wallclock time passed during the run; "user" is the CPU time consumed by the program (roughly equals to wallclock time multiplied by the number of cores in case of 100% efficiency); "system" is the time spent by the operating system on the multiprocessing service. One can see that the multiprocessing is efficient with number of cores up to 8, after which the total run time does not significantly decrease and the CPU time ("user") grows. It is also clear from the lower plot that the average CPU load efficiency drops below 50% with more than 8 cores active.

References

[1] M. R. Whalley, D. Bourilkov, and R. C. Group, `hep-ph/0508110`.

[2] T. Hahn, *Comput.Phys.Commun.* **168** (2005) 78–95, `hep-ph/0404043`.

[3] A. Arbuzov, D. Bardin, S. Bondarenko, P. Christova, L. Kalinovskaya, *et al.*, *Eur.Phys.J.* **C54** (2008) 451–460, `arXiv:0711.0625`.

[4] A. Andonov, A. Arbuzov, S. Bondarenko, P. Christova, V. Kolesnikov, *et al.*, *Phys.Atom.Nucl.* **73** (2010) 1761–1769, `0901.2785`.

[5] D. Bardin, S. Bondarenko, L. Kalinovskaya, G. Nanava, L. Rumyantsev, *et al.*, *Comput.Phys.Commun.* **177** (2007) 738–756, `hep-ph/0506120`.

[6] A. Arbuzov, D. Bardin, S. Bondarenko, P. Christova, L. Kalinovskaya, *et al.*, *Eur.Phys.J.* **C46** (2006) 407–412, `hep-ph/0506110`.

[7] D. Bardin, S. Bondarenko, P. Christova, L. Kalinovskaya, V. Kolesnikov, *et al.*, `1110.3622`.

[8] D. Bardin, S. Bondarenko, P. Christova, L. Kalinovskaya, L. Rumyantsev, *et al.*, `1207.4400`.