Tutorials on Monte Carlo Integration with BASES

Junichi Kanzaki (Kavli IPMU)

Iwate Collider School 2024

@ ANA Crowne Plaza Resort Appi Kogen

February 27, 2024

Purpose of this tutorial

- We already have a great tool,
  MadGraph5_aMC@NLO, which provides
  - the computations of cross-sections,
  - the event generations, and
  - the use of tools for data manipulation and analysis.

  You only need to specify physical process information and a few parameters.

-> This tutorial aims to help you understand what MG5 is doing internally by experiencing cross-section calculations by yourselves.

"bsexamples.tar.gz" on Slack#tutorial

- Required PC environment:
  - Essential: Development with gfortran, make, ..., for installing "BASES."
  - Optional (recommended): Python3 with matplotlib for data visualization.
- "bsexamples" includes five examples.
  - Ex.1-3: 1-dimensional and multi-dimensional integrations of simple functions.
  - Ex.4-5: calculation of cross-section of physics process (e^-e^+ > Z H).
  - Both include examples of plotting with matplotlib.

Numerical Integration

- Computation of physics quantities is often reduced to calculating multivariable functions.
  - Cross-sections:

$$\frac{d\sigma}{dx_j} = f(\sqrt{s}, p_i, h_i)$$

  - If you want to observe some quantities, x_j,j=1~n (p_T of lepton or jet), you must integrate other unobserved variables.
- Monte Carlo integration using random numbers is particularly useful for higher-dimensional integrals.

Monte Carlo Integration (ref.)

- We had a good lecture by Olivier this morning.

- VEGAS: G.P. Lepage, J. Comput. Phys. 27 (1978) 192–203.
- BASES: S. Kawabata, Comput. Phys. Commun. 88, 309 (1995) - Integration/Event generation
- MG5: J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer, T. Stelzer, JHEP06 (2011) 128 - Integration/Event generation

BASES/SPRING

- BASES/SPRING is the Monte Carlo integration and event generation package developed at KEK based on the VEGAS algorithm.
- BASES, the MC integration program, includes histograming utilities, which are helpful for instantly checking distributions.
- SPRING is the unweighted event generation package using BASES integration results.

  -> I will focus on the usage of BASES today.

Tutorials with BASES

- All samples are written in Fortran 90
- The BASES program is composed of three parts:
  - initializations - call "bsinit" and "userin"
    - parameters for integration and integrand functions
  - integration part
  - termination - "call usrout" and other subroutines
    - integration results and histogram outputs

BASES parameters

- Parameters of integrand function:
  - ndim - the number of dimensions of the integrand function
  - xl(i), xu(i) - the ranges of integration variables for i = 1 to ndim

```
ndim = 8
nwild = ndim

xl(1:ndim) = 0.d0
xu(1:ndim) = 1.d0
ig(1:ndim) = 1

call bssetd(ndim,nwild,xl,xu,ig)
```

BASES parameters

- BASES has two integration phases.
  - Optimizations of grids of variables
  - Data accumulations are based on optimized grids.
  - Both phases proceed iteratively, step by step.

BASES parameters

- Parameters of integration:
  - ncall - the number of function calls per iteration step
  - it1, it2 - the maximum iterations for each program phase (it1-optimization, it2-accumulation).
  - acc1, acc2 - the accuracy to stop iterations (in %)

```
ncall = 10000
itmx1 = 10
itmx2 = 10
acc1 = 0.01
acc2 = 0.01

call bssetp(ncall,itmx1,itmx2,acc1,acc2)
```

BASES parameters

- Sample output of integration

Optimization phase

Iteration numbers

```
                 Convergency Behavior for the Grid Optimization Step
------------------------------------------------------------------------------
<- Result of  each iteration ->  <-     Cumulative result     -> < CPU  time >
 IT Eff R_Neg   Estimate  Acc %  Estimate(+- Error )order  Acc % ( h: m: sec )
------------------------------------------------------------------------------
  1 100   0.00  3.141E+00  0.042  3.141292(+-0.001308)e 00  0.042   0: 0: 0.00
  2 100   0.00  3.142E+00  0.056  3.141694(+-0.001048)e 00  0.033   0: 0: 0.00
  3 100   0.00  3.142E+00  0.064  3.141676(+-0.000931)e 00  0.030   0: 0: 0.00
------------------------------------------------------------------------------
```

Results and errors at each iteration

Accumulation phase

```
1
                                                    date: 24/ 2/27  08:12
                 Convergency Behavior for the Integration Step
------------------------------------------------------------------------------
<- Result of  each iteration ->  <-     Cumulative result     -> < CPU  time >
 IT Eff R_Neg   Estimate  Acc %  Estimate(+- Error )order  Acc % ( h: m: sec )
------------------------------------------------------------------------------
  1 100   0.00  3.144E+00  0.068  3.143867(+-0.002139)e 00  0.068   0: 0: 0.00
  2 100   0.00  3.140E+00  0.069  3.142203(+-0.001524)e 00  0.049   0: 0: 0.00
  3 100   0.00  3.141E+00  0.066  3.141840(+-0.001230)e 00  0.039   0: 0: 0.00
------------------------------------------------------------------------------
```

Integrand function

- The integration part and the integrand function are defined as follows:

```
call bases(func, estim,error,ctime,it1,it2)
```

  - func - the integrand function
  - Integration results:
    - estim - integrated result
    - error - its error
    - ctime - calculation time
    - it1, it2 - number of iterations in the integration

Integrand function

- Function definition

```
real(real64) function func(z)
  real(real64), intent(in) :: z(*)

  func = 2.d0*sqrt((1.d0-z(1))*(1.d0+z(1)))
```

- z(i), i=1,ndim: random numbers given to the integrand function as integration variables.

Histograming in BASES

- **Definition at initialization step:**

```
call xhinit(1,-1.d0,1.d0,50,'costh')
```

  - xhinit(id,xlow,xhigh,nbins,'title')

- **Filling in the integrand function**

```
call xhfill(1,costh,func)
```

  - xhfill(id,val,func)

- **Output at the termination step**

```
call bhplot(6)
```

  - bhplot(lun)

Tutorial-0

- We assume the development environment with gfortran.
- Install python3 with "matplotlib" to experience plotting.
- After downloading the example package, bsexamples.tar.gz, please make and install the BASES library, located under the bases50 directory.
- Apology: the bases library, libbases50.a, compiled under my environment, is erroneously included in the example file. -> Please recompile and reinstall it under your environment (see next slide).

Installation of examples

- After downloading the example package, bsexamples.tar.gz, please go to the bases50 directory and make and install the library libbases50.a.

```
kanzaki$ cd bsexamples
kanzaki$ ls
bases50/        example1/      example3/      example5/
dmp2array.py* example2/      example4/      lib/
kanzaki$ cd bases50
kanzaki$ make clean; make install
....
Loading libbases50.a ...
done
Installing libbases50.a in ../lib
kanzaki$ cd ../
kanzaki$ ls lib/
libbases50.a*
```

Installation of examples (cont'd)

- For each example, example1 - example5, go to the directory and execute "make."
- The program, "examplex.f90", includes all necessary program components.
- For example, in the case of "example1":

```
kanzaki$ cd example1/
kanzaki$ ls
Makefile       __pycache__/  dmp2array.py@ example1.f90  plot.py*
kanzaki$ make
gfortran -O -I./ -c example1.f90
gfortran example1.o -L../lib   -lbases50 -o example1
kanzaki$
kanzaki$ ls
Makefile       dmp2array.py@ example1.f90  plot.py*
__pycache__/  example1*     example1.o
kanzaki$ ./example1
```

- You can try plotting with "plot.py" if you have matplotlib.

Tutorial-1

The integration of 1-dim. function:

$$f(x) = 2\sqrt{(1 - x^2)}$$

between $-1 < x < 1$.

1. Make "example1" and execute it. Check the answer
2. Increase "ncall" and check the accuracy. You can also change it1, it2, and/or acc1, acc2 and observe results.
3. You can obtain the same answer with $0 < x < 1$ and f(x) = 4 x ...

Tutorial-1 (cont'd)

1. Learn how to define histograms in "userin."
   call xhinit(id,xmin,xmax,nbins,title)
2. Learn how to fill histograms in "bfunc":
   call xhfill(id,val,func)
3. The program generates a histogram dump file, "example.dump." If you have "matplotlib," execute the script "plot.py," which reads the file and generates a plot.
4. You can modify the script to get a better-looking plot if you have experience.

Tutorial-2

- Another example of the integration of multi-dimensional function:

$$f(x) = \prod_{i=1}^{n_{\rm dim}} (2x_i)$$

- Try to change integration parameters.
- Try plotting "plot.py" for the dump file, "example2.dump."

Tutorial-3

- Another example of the integration of a 2-dimensional Gaussian function:

$$f_{\mathrm{XY}}(x,y) = \frac{1}{\sqrt{2\pi\sigma_{\mathrm{X}}^2}} \mathrm{e}^{-\frac{(x-\mu_{\mathrm{X}})^2}{2\sigma_{\mathrm{X}}^2}} \times \frac{1}{\sqrt{2\pi\sigma_{\mathrm{Y}}^2}} \mathrm{e}^{-\frac{(y-\mu_{\mathrm{Y}})^2}{2\sigma_{\mathrm{Y}}^2}}.$$

- Check results and histograms.
- Learn how to define and fill 2-dimensional histograms.
- Test "plot_1d.py" and "plot_2d.py."

2d histograms in BASES

- Definition at initialization step:

```
call dhinit(1,-1.d0,1.d0,50, &
              0.d0,2.d0,50,'costh')
```

  - dhinit(id,xlow,xhigh,nxbins, &
    ylow,yhigh,nybin,'title')

- Filling in the integrand function

```
call dhfill(1,xval,yval,func)
```

  - dhfill(id,valx,valy,func)

- Output at the termination step is the same as 1d histograms.

Tutorial-4

Calculate cross-sections of the physics process:

$$e^- e^+ \to ZH$$

See Olivier's lecture this morning for details.

$$\begin{aligned}
\sigma &= \frac{1}{2s\beta}\frac{1}{2}\frac{1}{2}\int \sum_\lambda |\sum_i M_i|^2 d\Phi \\
\text{flux factor} &= \frac{1}{2s\beta} \\
\text{spin average} &= \frac{1}{2}\times\frac{1}{2} \\
\lambda &= \text{helicities of external particles} \\
M_i &= \text{amplitude of i-th channel} \\
\Phi &= \text{phase space}
\end{aligned}$$

Tutorial-4

- Integration of cross-sections of the physics process:
  - $|M_i|^2$: Helicity amplitude by MG5
  - d\Phi: Phase space

  -> Integrand function becomes, $|M_i|^2$ x d\Phi
- I prepared the helicity amplitude with MG5, outputted with "standalone" mode. Then copy "Cards," "lib," and some files under the process directory to example4 and example5.

Tutorial-4

- Directory structure:

```
kanzaki$ cd example4
kanzaki$ ls
Cards/ lib/   src/
```

- Go to the directory, src/:

```
kanmac21:src kanzaki$ cd src/
kanmac21:src kanzaki$ ls
Makefile                  example4.f90                ngraphs.inc
__pycache__/              generate_phase_space.f90    plot_cosz.py*
coupl.inc                 matrix.f                    pmass.inc
dmp2array.py@             nexternal.inc               run_example4*
```

- I copied "matrix.f" (amplitude program) and include files (coupl.inc, nexternal.inc, ngraphs.inc, and pmass.inc) and made a few modifications(*).

Tutorial-4 (for reference)

- (*) in the previous slide:
- coupl.inc: Put "!" at the beginning of the first three lines due to the language difference of Fortran 90.

Add three "!"

```
kanzaki$ cat coupl.inc
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!c      written by the UFO converter
!cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

- Copy the definition of "ncomb" from the top part of matrix.f to "nexternal.inc" to use the variable in BASES.

matrix.f

```
      SUBROUTINE SMATRIXHEL(P,HEL,ANS)
      IMPLICIT NONE
C
C     CONSTANT
C

      INTEGER    NEXTERNAL
      PARAMETER (NEXTERNAL=4)
      INTEGER                 NCOMB
      PARAMETER (             NCOMB=12)
```

Copy two lines

nexternal.inc

```
      INTEGER    NEXTERNAL
      PARAMETER (NEXTERNAL=4)
      INTEGER    NINCOMING
      PARAMETER (NINCOMING=2)
      INTEGER                 NCOMB
      PARAMETER (             NCOMB=12)
```

Generation of phase space

We have to generate phase space:

total collision energy (input parameter) + random numbers -> four-vectors of external particles

```
kanzaki$ cat generate_phase_space.f90
subroutine generate_phase_space(z, sqrts, pmass, &
      p, nexternal, phfact, ierr)
```

In this example:

The final state particles are Z and H.

Calculate four-vectors of external particles, p_{e^-}, p_{e^+}, p_Z, p_H, from input random numbers.

* MG5 does it for you.

Two-body phase space

Two-body phase space:

$$d\Phi_2 = \frac{1}{8\pi}\overline{\beta}(\frac{m_1^2}{s}, \frac{m_2^2}{s}) \int_{-1}^{+1} \frac{d\cos\theta}{2} \int_0^{2\pi} \frac{d\phi}{2\pi}$$

Total energy: W(=\sqrt{s}) -> $P_1(m_1)$ + $P_2(m_2)$

1. Determine energies of P_1 and P_2 (E_1 and E_2) from total energy, W
2. Determine directions (\theta and \phi) of final particles with two random numbers and calculate four momenta for P1 and P2.

   \theta: -1 < cos(\theta) < +1

   \phi: 0 < \phi < 2 \pi

(Three body phase space)

Then go to three-body phase space:

$$d\Phi_3 = \int_{(m_1+m_2)^2}^{(\sqrt{s}-m_3)^2} \frac{dq^2}{2\pi} \int d\Phi_2(P = q + p3) \int d\Phi2(q = p_1 + p_2)$$

$$= \frac{1}{2} \frac{1}{32\pi^2} \frac{1}{32\pi^2} \int_{(m_1+m_2)^2}^{(\sqrt{s}-m_3)^2} dq^2 \overline{\beta}\left(\frac{q^2}{s}, \frac{m_3^2}{s}\right) \overline{\beta}\left(\frac{m_1^2}{q^2}, \frac{m_2^2}{q^2}\right)$$

$$\times \int_{-1}^{+1} d\cos\theta_3 \int_0^{2\pi} d\phi_3 \int_{-1}^{+1} d\cos\hat{\theta}_1 \int_0^{2\pi} d\hat{\phi}_1$$

W(=\sqrt{s}) -> $p_1(m_1)$ + $p_2(m_3)$ + $p_3(m_3)$

-> Combination of two two-body phase space:

PS(p_3,q=p_1+p_2) + PS(p_1,p_2)

(Three body phase space) (cont'd)

1. Determine the invariant mass of the system of p_1+p_2, q^2, with a random number:

$$m_1+m_2 < q^2 < W-m_3$$

2. Generate four momenta of p_3 and p_1+p_2 with two body phase space.
3. Then generate p_{10} and p_{20} at the rest frame of p_1+p_2 with q^2.
4. Rotate&Boost generated p_{10} and p_{20} to the lab. Frame, p_1 and p_2.

Tutorial-4 (cont'd)

Go to ./src, make executable, and use "run_example4.sh" to execute the program. This script includes values collision energy and "ncall." Try plotting with "plot_cosz.py."

1. Learn how we can calculate cross-sections of the physics process.
2. Learn about phase space generation.
3. If you wish, try other physics processes and calculate cross-sections after learning more about using "MG5." Of course, MG5 automatically does it for you.

Tutorial-5

Calculate cross-sections of the Standard Model physics processes with decays:

$$e^-e^+ \to Z(\to \mu^-\mu^+)H(\to b\bar{b})$$

1. The procedure is almost the same as "Tutorial-4."
2. Try plotting with "plot.py," which generates invariant mass distributions.
3. Try the four-body phase space with decays.

Generation of phase space

Phase space generation for the decay of a resonance:

-> See the detailed description and example in Olivier's lecture.

All necessary functions and subroutines are included in "generate_phase_space.f90" in the src/directory. It is far more complicated than the two body cases in "example4." If you are interested, please read it carefully.

Summary

- We already have many excellent software tools for physics studies. They have much automatic functionality, which helps your analysis.
- Still, I want you to understand what procedures/technologies such software uses, not just using it.
- I hope my examples help you experience the MC integration and the calculation of cross-sections and understand the calculation processes.

Summary

- If possible, try to modify these codes and test them by yourself:
  - check ncall dependence of integration accuracies,
  - apply kinematical cuts to final state particles,
  - try other physics processes in which you have some interest
  - etc.