对撞机蒙特卡洛模拟：介绍

2021对撞机唯像学暑期学校

李强　北京大学物理学院技术物理系

qliphy0@pku.edu.cn

背景及介绍

在高能对撞机物理的理论和实验领域，蒙特卡洛方法得到了很大的应用和发展。课程注重理论与实践相结合，力图使学生获得扎实的物理和编程训练。开阔眼界，应对前沿需求。

希望各位同学安装、学习如下程序：

CERN Root5/6
Python, C++, Fortran
LHAPDF5/6
MadGraph (ExRootAnalysis, lhereader)
Pythia8, HEPMC2
FastJet
Delphes
MCFM or VBFNLO or MATRIX

课程大纲

1. 开场白：

- H->bb
- CERN Root及数据分析（Fortran、Python）
- 随机数和MC积分

2. MadGraph简介：

- 对撞机物理简述
- 矩阵元及自动计算　e+e- > mu+ mu-
- LHE分析　pp->ww->lvqq
- pp->Z　LO、NLO、MLM 、CMS测量及Rivet

3. Parton Shower

- 理论介绍
- Event Display, 动画
- 数据分析：Status、copy、H decay、ZG vs Zjets

4. Others

- CEPC ISR，UFO/BSM，Delphes，Reweighting，EFT， Interference，Loop-Induced

5. MC in CMS

部分参考材料

Generator

home.thep.lu.se/~torbjorn/talks/van1ho.pdf
Introduction to Event Generators (dur.ac.uk)

Parton Shower

http://home.thep.lu.se/~torbjorn/Pythia.html
https://indico.cern.ch/event/438776/attachments/1159226/1668171/herwig.pdf
users.ictp.it/~smr2463/lect/Ellis-3.pdf
https://indico.cern.ch/event/369827/contributions/875691/attachments/1165138/1679417/gieseke.pdf

Jets

https://www.physics.sjtu.edu.cn/madgraphschool/index.php?q=node/29
https://indico.cern.ch/event/346738/contributions/813675/attachments/684118/939687/14_JetPhysics.pdf

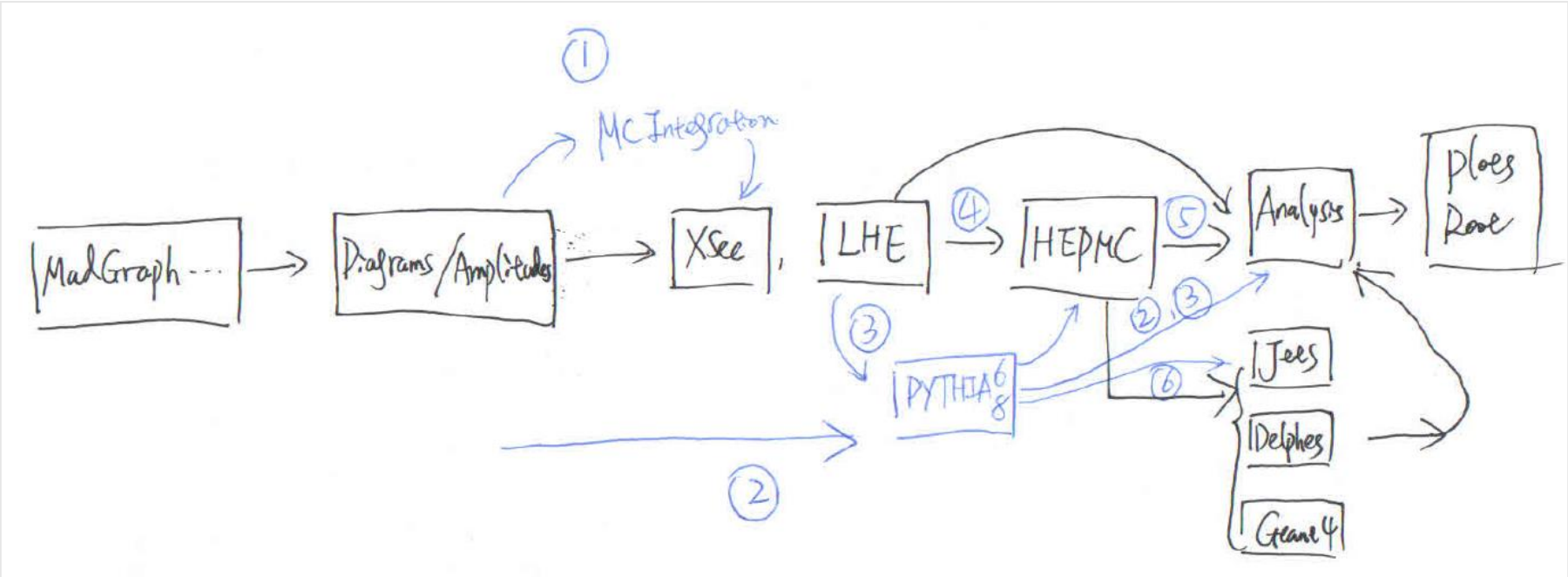
Underlying Events, LHC Xsec

www.phys.ufl.edu/~rfield/cdf/UVa_RickField_3-2-16.pdf
ICHEP 2016 Chicago (3-2016年8月10日): Total, elastic and inelastic pp cross sections at the LHC (15' + 5') · Indico (cern.ch)

Matching and Merging

https://www.physics.sjtu.edu.cn/madgraphschool/index.php?q=node/29
https://indico.cern.ch/event/669309/timetable/?view=standard

MadGraph → Diagrams/Amplitudes→MC integration → Xsec, LHE (weighted/unweighted) → Pythia6/8 → HEPMC → Jet, reconstruction, Simulation



C++ version of MG, Analysis package…

Tutorials

https://github.com/qliphy/2021Summer
https://www.notion.so/MCnet-Beijing-School-2021-436e0edf33064fcf8f267c0cf92e9ec2
https://www.notion.so/High-Energy-Physics-ML-Tutorial-f9f8a3c624cb489cbeda53d38628d5ed

qliphy / 2021Summer

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights

main | 1 branch | 0 tags | Code

qliphy first commit 2de6ec9 13 days ago 1 commit

1
2
3
README.md

High Energy Physics ML Tutorial

Preparation
Usage for these software
Events generation
- Generate LHE file
- Do parton shower and detector simulation through pythia and delphes
- Make root ntuple
- Make npz file

Neural Network
- Packages needed
- Input/Output Path
- Run the Script
- Code Overview

Decision Tree
- Input/Output Path
- Run the Script

MCnet Beijing School 2021

Indico event, GitLab , Docker Hub hepstore , IHEPBox

Setup
Tutorials
- Previous MCnet tutorials

Troubleshooting

Setup

Docker documentation, 《Docker 从入门到实践》

Recommended environment
Getting started with Docker

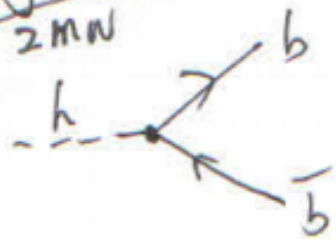
Tutorials

You are encouraged to self-educated using the tutorials in the GitLab repo. If you run into problems, the notes below might be helpful.

Herwig tutorial
Madgraph5 tutorial
Pythia tutorial
Sherpa tutorial
Rivet tutorial

Hbb

$$\frac{i g m_b}{2 m_W}$$



$$M = \frac{g^2 m_b^2}{4 m_W^2} \bar{u}_{p_1} v_{p_2}$$

$$|M|^2 = \frac{g^2 m_b^2}{4 m_W^2} \mathrm{Tr}[\not{p}_1 \not{p}_2] = \frac{g^2 m_b^2 m_h^2}{2 m_W^2}$$

$$\frac{1}{2 m_H} \frac{1}{8\pi} \times \frac{g^2 m_b^2 m_h^2}{2 m_W^2}$$

$$= \boxed{\frac{\frac{e^2}{s_W^2} m_b^2 m_h}{32\pi m_W^2}}$$

$$= \frac{\alpha}{8 s_W^2} \frac{m_b^2}{m_W^2} m_h$$

$$\frac{1}{137} \times \frac{1}{8} \times \frac{1}{0.2223} \frac{4.7^2}{80^2} \times 125$$

How to deal with decay with MadGraph?

Does the result looks ok?

CrossSections < LHCPhysics < TWiki (cern.ch)

```
qliphy@qliphy:~/Desktop/MG5_aMC/MG5_aMC_v3.1.1/MG5_aMC_v3_1_1$ python2 ./bin/mg5_aMC
WARNING:root:python2 support will be removed in last quarter 2021. If you use python2 du
 to issue with Python3, please report them on https://bugs.launchpad.net/mg5amcnlo
************************************************************
*                                                          *
*                    W E L C O M E to                      *
*             M A D G R A P H 5 _ a M C @ N L O            *
*                                                          *
*                                                          *
*                 *                       *                *
*                   *        * *        *                 *
*                     * * * * 5 * * * *                    *
*                   *        * *        *                 *
*                 *                       *                *
*                                                          *
*         VERSION 3.1.1                 2021-05-28         *
*                                                          *
*    The MadGraph5_aMC@NLO Development Team - Find us at   *
*    https://server06.fynu.ucl.ac.be/projects/madgraph     *
*                            and                           *
*            http://amcatnlo.web.cern.ch/amcatnlo/         *
*                                                          *
*               Type 'help' for in-line help.              *
```

```
MG5_aMC>install pythia8
   You are installing 'pythia8', please cite ref(s): arXiv:1410.3012.
Downloading the HEPToolInstaller at:
   http://madgraph.phys.ucl.ac.be/Downloads/HEPToolsInstaller/HEPToolsInstaller_V145.tar.
gz
```

```
MG5_aMC>install ExRootAnalysis
Downloading http://madgraph.phys.ucl.ac.be/Downloads/ExRootAnalysis/ExRootAnalysis_V1.1.5
.tar.gz
--2021-07-03 15:12:26--  http://madgraph.phys.ucl.ac.be/Downloads/ExRootAnalysis/ExRootAn
alysis_V1.1.5.tar.gz
```

```
MG5_aMC>install Delphes
   You are installing 'Delphes', please cite ref(s): arXiv:1307.6346.
Downloading http://cp3.irmp.ucl.ac.be/downloads/Delphes-3-current.tar.gz
```

```
b t~ b~ z w+ h w- ta- ta+
MG5_aMC>generate h > b b~
INFO: Trying process: h > b b~ @1
INFO: Process has 1 diagrams
1 processes with 1 diagrams generated in 0.004 s
Total: 1 processes with 1 diagrams
MG5_aMC>output hbb
INFO: initialize a new directory: hbb
INFO: remove old information in hbb
INFO: Organizing processes into subprocess groups
INFO: Generating Helas calls for process: h > b b~ @1
INFO: Processing color information for process: h > b b~ @1
INFO: Creating files in directory P1_h_bbx
INFO: Generating Feynman diagrams for Process: h > b b~ @1
INFO: Finding symmetric diagrams for subprocess group h_bbx
Generated helas calls for 1 subprocesses (1 diagrams) in 0.006 s
Wrote files for 4 helas calls in 0.039 s
ALOHA: aloha starts to compute helicity amplitudes
ALOHA: aloha creates 1 routines in  0.321 s
save configuration file to /home/qliphy/Desktop/MG5_aMC/MG5_aMC_v3.1.1/MG5_aMC_v3_1_1/hbb
/Cards/me5_configuration.txt
```

```
=== Results Summary for run: run_01 tag: tag_1 ===

    Width :   0.005391 +- 8.447e-12 GeV
    Nb of events :  10000
```

CrossSections < LHCPhysics < TWiki (cern.ch)

Higgs → 2 gauge bosons

M_H	H → gg			H → γγ			H → Zγ			H → WW			H → ZZ			Total Γ_H		
(GeV)	BR	error [%]		BR	error [%]		BR	error [%]		BR	error [%]		BR	error [%]		(GeV)	error [%]	
80.0	4.94E-02	+12.02	-11.58	9.18E-04	+6.34	-6.15	0.00E+00	+0.00	-0.00	6.27E-04	+5.89	-5.73	1.64E-04	+5.79	-5.63	1.99E-03	+5.29	-5.24
81.0	5.05E-02	+11.99	-11.55	9.46E-04	+6.32	-6.15	0.00E+00	+0.00	-0.00	6.96E-04	+5.86	-5.72	1.81E-04	+5.75	-5.62	2.01E-03	+5.28	-5.21
82.0	5.17E-02	+11.97	-11.56	9.74E-04	+6.31	-6.16	0.00E+00	+0.00	-0.00	7.74E-04	+5.89	-5.73	1.99E-04	+5.78	-5.63	2.04E-03	+5.28	-5.24
124.9	8.58E-02	+10.23	-9.97	2.28E-03	+4.99	-4.89	1.53E-03	+8.93	-8.90	2.13E-01	+4.27	-4.18	2.62E-02	+4.27	-4.18	4.06E-03	+3.95	-3.94
125.0	8.57E-02	+10.22	-9.98	2.28E-03	+4.98	-4.89	1.54E-03	+9.01	-8.83	2.15E-01	+4.26	-4.20	2.64E-02	+4.28	-4.21	4.07E-03	+3.97	-3.93
125.1	8.56E-02	+10.22	-9.95	2.28E-03	+4.91	-4.88	1.55E-03	+8.94	-8.86	2.16E-01	+4.24	-4.17	2.67E-02	+4.24	-4.18	4.08E-03	+3.94	-3.91
125.2	8.55E-02	+10.21	-9.96	2.28E-03	+4.91	-4.89	1.56E-03	+8.98	-8.81	2.18E-01	+4.23	-4.18	2.69E-02	+4.21	-4.17	4.10E-03	+3.95	-3.90
125.3	8.54E-02	+10.20	-9.95	2.28E-03	+4.91	-4.89	1.56E-03	+8.96	-8.85	2.19E-01	+4.22	-4.16	2.72E-02	+4.22	-4.16	4.11E-03	+3.94	-3.89

Hbb NLO QCD

```
MG5_aMC>generate h > b b~ [QCD]
The current model sm does not allow to generate loop corrections of type ['QCD'].
MG5_aMC now loads 'loop_sm'.
 import model loop_sm
INFO: Restrict model loop_sm with file models/loop_sm/restrict_default.dat .
INFO: Run "set stdout_level DEBUG" before import for more information.
INFO: Change particles name to pass to MG5 convention
Kept definitions of multiparticles l- / j / vl / l+ / p / vl~ unchanged
Defined multiparticle all = g gh gh~ d u s c d~ u~ s~ c~ a ve vm vt e- mu- ve~ vm~ vt~ e+
 mu+ b t b~ t~ z w+ h w- ta- ta+
INFO: Generating FKS-subtracted matrix elements for born process: h > b b~ [ all = QCD ]
(1 / 1)
INFO: Generating virtual matrix elements using MadLoop:
INFO: Generating virtual matrix element with MadLoop for process: h > b b~ QED<=1 [ all =
 QCD ] (1 / 1)
INFO: Generated 1 subprocesses with 2 real emission diagrams, 1 born diagrams and 1 virtu
al diagrams
MG5_aMC>output hbbnlo
```

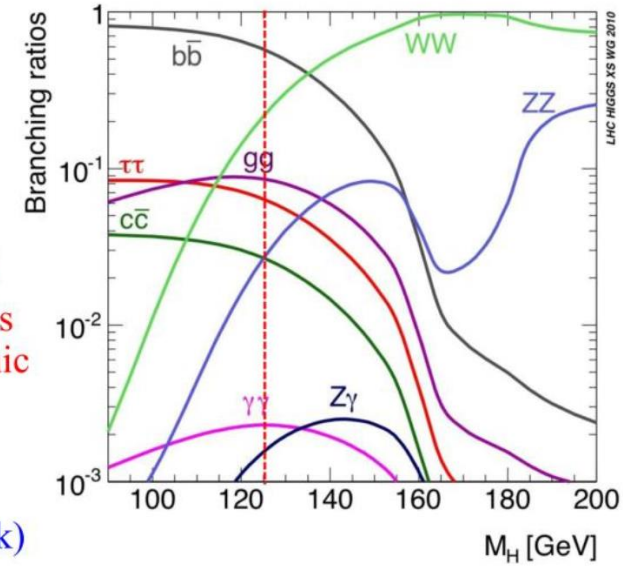
```
-----------------------------------------------------------------------
   Final results and run summary:
   Process h > b b~ [QCD]
   (Partial) decay width: 3.473e-03 +- 3.6e-05 GeV
-----------------------------------------------------------------------
   Scale variation (computed from histogram information):
       Dynamical_scale_choice -1 (envelope of 9 values):
           3.473e-03 pb  +5.0% -6.2%
-----------------------------------------------------------------------
```

Hbb NLO QCD + Yukawa corrections?

```
  13 0.000000 # mu- : 0.0
  14 0.000000 # vm : 0.0
  16 0.000000 # vt : 0.0
  21 0.000000 # g : 0.0
  22 0.000000 # a : 0.0
  24 80.419002 # w+ : cmath.sqrt(MZ__exp__2/2. + cma
*MZ__exp__2)/(Gf*sqrt__2)))

##################################
## INFORMATION FOR SMINPUTS
##################################
Block sminputs
    1 1.325070e+02 # aEWM1
    2 1.166390e-05 # Gf
    3 1.180000e-01 # aS

##################################
## INFORMATION FOR YUKAWA
##################################
Block yukawa
    5 4.2500000e+00 # ymb
    6 1.730000e+02 # ymt
   15 1.777000e+00 # ymtau

##################################
## INFORMATION FOR DECAY
##################################
-- INSERT --                                    43,1          62%
```

Higgs decay branching ratio at m_H=125 GeV

- bb: 57.7% (huge QCD background)
- WW: 21.5% (easy identification in di-lepton mode, complex background)
- ττ: 6.3% (complex final states with τ leptonic and/or hadronic decays)
- ZZ*: 2.6% ("gold-plated", clean signature of 4-lepton, high S/B, excellent mass peak)
- γγ: 0.23% (excellent mass resolution, high sensitivity)



Higgs boson production rate: 1 out of 10^{12} collision events

```
   ----------------------------------------------------------------
      Final results and run summary:
      Process h > b b~ [QCD]
      (Partial) decay width: 2.818e-03 +- 2.9e-05 GeV
```


思考： H->2photon, 2 gluon 的衰变宽度怎么用MadGraph得到？

Linux之前

1983 图灵奖：
Ken Thompson， Dennis M. Ritchie
for their development of generic operating systems theory and specifically for the implementation of the UNIX operating system.

- In 80's, Microsoft's DOS (disk operating system) was the dominated OS for PC
- Apple MAC was better, but expensive
- UNIX was much better, but much, much more expensive. Only for minicomputer for commercial applications
- **People was looking for a UNIX based system, which is cheaper and can run on PC**
- Both DOS, MAC and UNIX were proprietary, i.e., the source code of their kernel is protected
- No modification is possible without paying high license fees

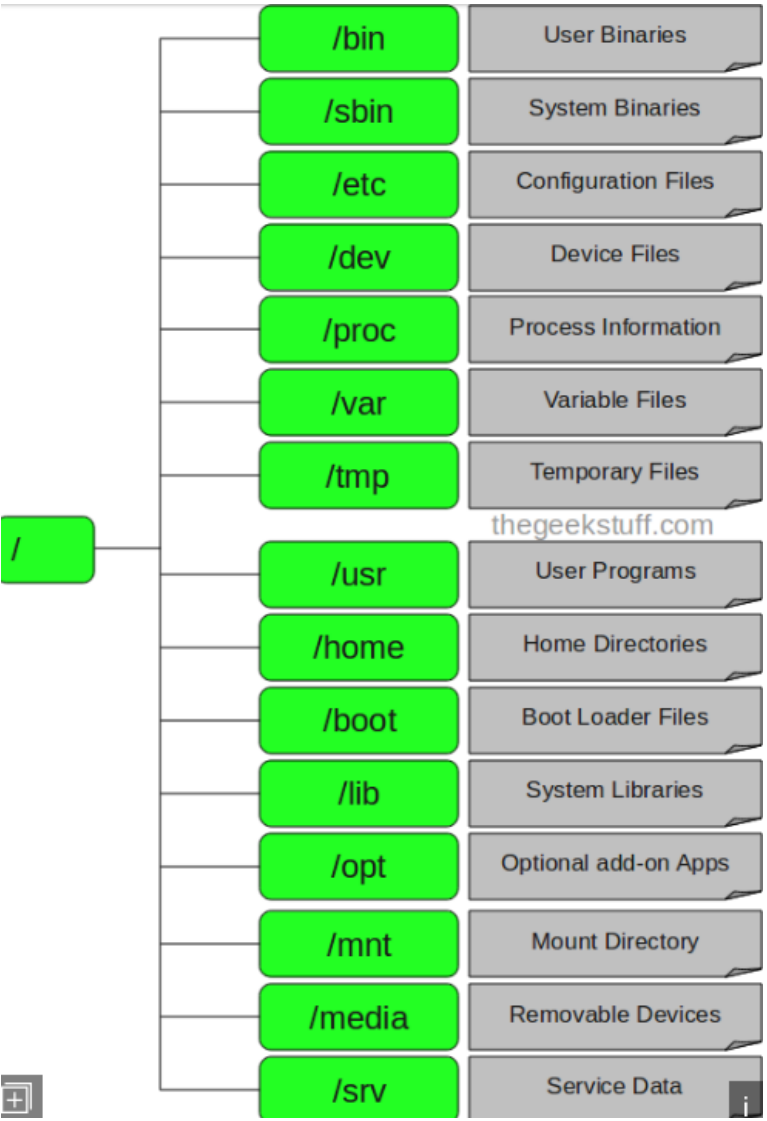
GNU Linux

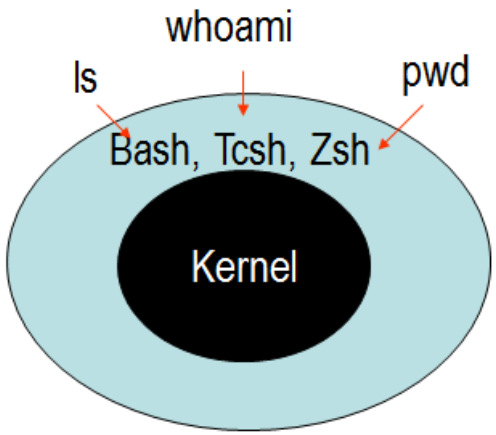
GNU是“GNU is Not Unix”的递归缩写。

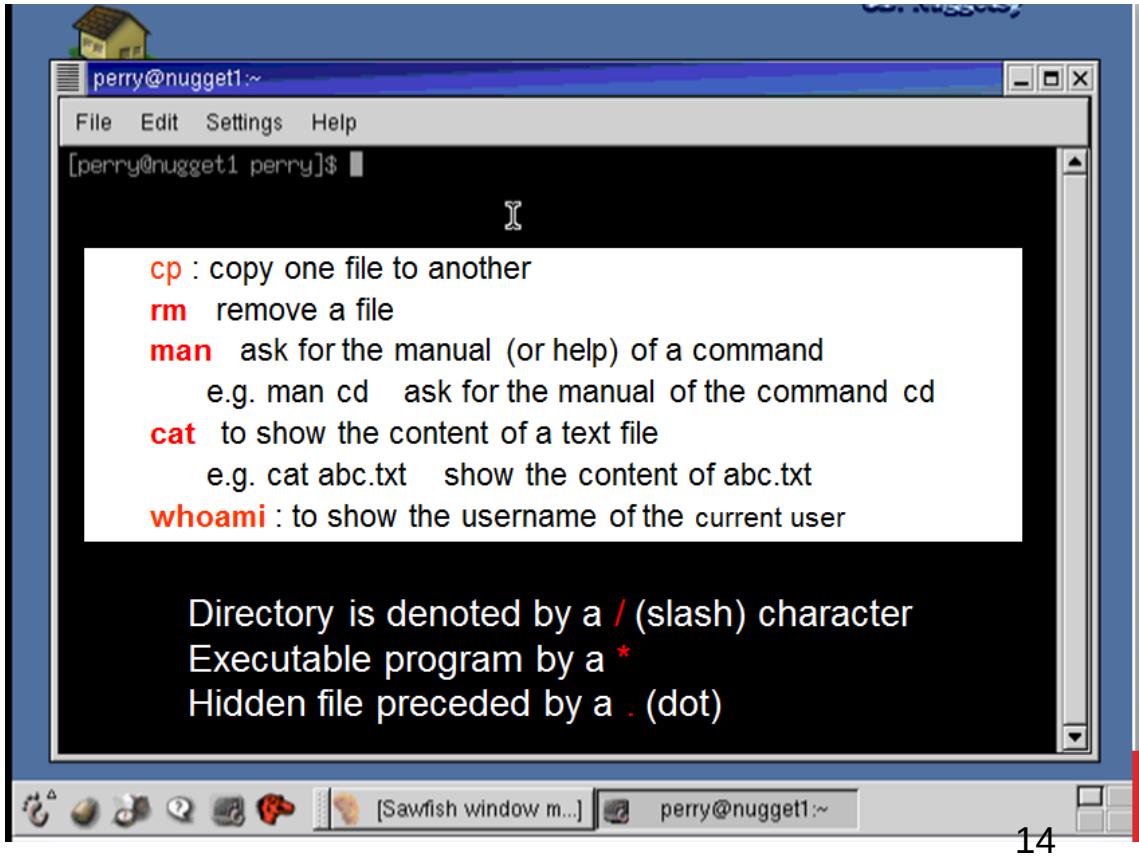
- Established in 1984 by Richard Stallman, who believes that software should be free from restrictions against copying or modification in order to make better and efficient computer programs
- GNU通用公共许可证（GNU General Public License，GPL）。即“反版权”（或称Copyleft）概念。

1991年Linus Torvalds编写出了与UNIX兼容的Linux操作系统内核并在GPL条款下发布。Linux之后在网上广泛流传，许多程序员参与了开发与修改。1992年Linux与其他GNU软件结合，完全自由的操作系统正式诞生。该操作系统往往被称为“GNU/Linux”或简称Linux。

Linux Shell以及文件系统







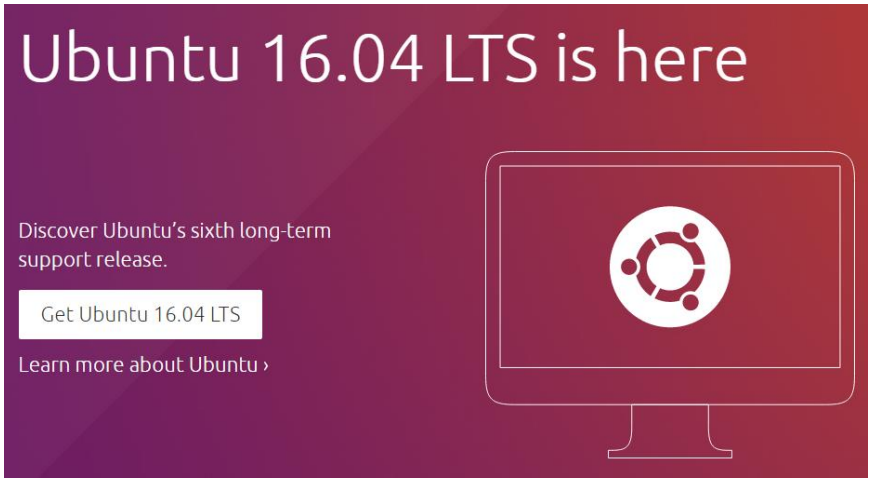
Ubuntu

Ubuntu是一个以桌面应用为主的开源GNU/Linux操作系统，Ubuntu 是基于Debian GNU/Linux，支持x86、amd64（即x64）和ppc架构，由全球化的专业开发团队（Canonical Ltd）打造的

Ubuntu历史版本一览表

版本号	代号	发布时间
17.04	Zesty Zapus	2017/04(即将发布)
16.10	Yakkety Yak	2016/10/20
16.04LTS	Xenial Xerus	2016/04/21
15.10	Wily Werewolf	2015/10/23
15.04	Vivid Vervet	2015/4/22
14.10	Utopic Unicorn	2014/10/23
14.04 LTS	Trusty Tahr	2014/04/18
13.10	Saucy Salamander	2013/10/17
13.04	Raring Ringtail	2013/04/25
12.10	Quantal Quetzal	2012/10/18
12.04 LTS	Precise Pangolin	2012/04/26
11.10	Oneiric Ocelot	2011/10/13
11.04（Unity成为默认桌面环境）	Natty Narwhal	2011/04/28
10.10	Maverick Meerkat	2010/10/10
10.04 LTS	Lucid Lynx	2010/04/29
9.10	Karmic Koala	2009/10/29
9.04	Jaunty Jackalope	2009/04/23
8.10	Intrepid Ibex	2008/10/30
8.04 LTS	Hardy Heron	2008/04/24
7.10	Gutsy Gibbon	2007/10/18
7.04	Feisty Fawn	2007/04/19
6.10	Edgy Eft	2006/10/26
6.06 LTS	Dapper Drake	2006/06/01
5.10	Breezy Badger	2005/10/13
5.04	Hoary Hedgehog	2005/04/08
4.10（初始发布版本）	Warty Warthog	2004/10/20

Arguably the most user-friendly version of Linux.
Huge repository of (free) software available - by far the most of any Linux distro

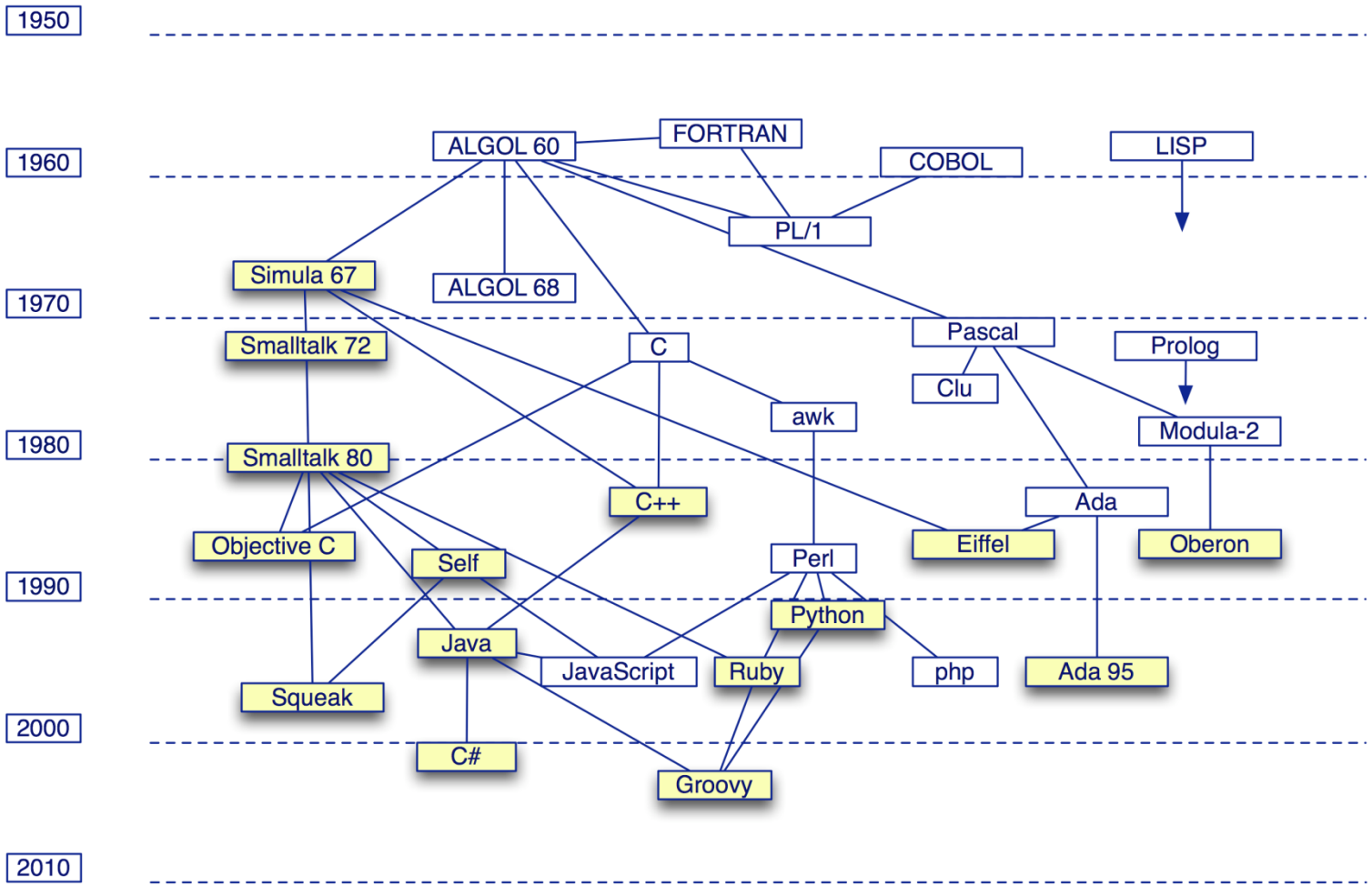


可与Windows双系统安装;可硬盘，U盘，光盘安装; 安装过程很自动

安装编译环境

sudo apt-get install build-essential
gcc, g++,....
sudo apt-get install gfortran

Object-oriented language genealogy

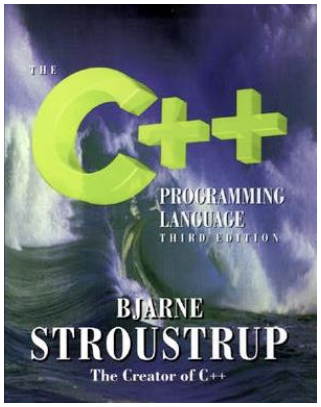


1979年， Bjarne Stroustrup到了Bell实验室，开始从事将C改良为带类的C（*C with classes*）的工作。1983年该语言被正式命名为C++。

C++简介

A *"better C"* that supports:

- Systems programming
- Object-oriented programming (*classes* & *inheritance*)
- Programming-in-the-large (*namespaces*, *exceptions*)
- Generic programming (*templates*)
- Reuse (large class & template libraries)



Most C programs are also C++ programs.

gcc hello.c –o hello

A preprocessor directive

Include standard io declarations

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

Write to standard output

char array

Indicate correct termination

"Hello World" in C++

Use the standard namespace

Include standard iostream classes

A C++ comment

```
using namespace std;
#include <iostream>
// My first C++ program!
int main(void)
{
    cout << "hello world!" << endl;
    return 0;
}
```

cout is an instance of ostream

operator overloading (two *different* argument types!)

Fortran

JOHN BACKUS

1977图灵奖

United States – 1977

CITATION

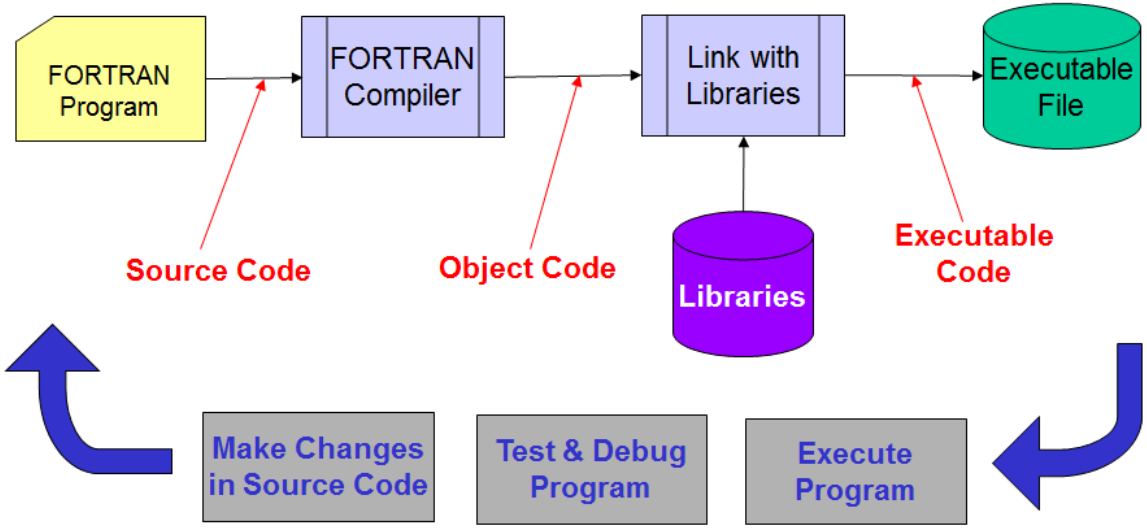
For profound, influential, and lasting contributions to the design of practical high-level programming systems, notably through his work on FORTRAN, and for seminal publication of formal procedures for the specification of programming languages.

- One of the oldest computer languages
  - created by John Backus and released in 1957
  - designed for scientific and engineering computations
- Version history
  - FORTRAN 1957
  - FORTRAN II
  - FORTRAN IV
  - FORTRAN 66 (released as ANSI standard in 1966)
  - FORTRAN 77 (ANSI standard in 1977)
  - FORTRAN 90 (ANSI standard in 1990)
  - FORTRAN 95 (ANSI standard version)
  - FORTRAN 2003 (ANSI standard version)

Fortran

- **FORTRAN was created to solve scientific and engineering problems**
- **Introduced integer and floating point variables**
- Introduced array data types for math computations
- Introduced subroutines and subfunctions
- There is a free compiler in Unix-Linux systems
  - f77, g77 - **g95, gfortran**

- FORTRAN is a complied language (like C) so the source code (what you write) must be converted into machine code before it can be executed (e.g. Make command)



Fortran 结构

- Skeleton of a program…

```
      PROGRAM MAIN
      REAL KGLO,FORC,KEL
      COMMON /STIF/KGLO(100,100)/LOAD/FORC(100)/DEF/D(100)
C     ...read in data and initialize problem...
      DO 100 IELEM=1,NELEMS
C        ...assemble global stiffness matrix...
         CALL KELEM(IELEM,KEL)
         CALL ASMBK(IELEM,KEL)
100   CONTINUE
      DO 200 ILOAD=1,NLOADS
C        ...assemble load vector...
         CALL LODVEC(ILOAD,LOAD)
200   CONTINUE
      CALL CONSTR(KDOFS)
      CALL SOLVE(NDOFS)
C     ...print out results, etc. ...
      END
```

partial list of declarations

Calculate stiffness matrix, KEL, for a single element

Add KEL to global stiffness matrix, KGLO

Construct FORC from individual loads defined in LOAD array

Must constrain problem at specified DOF's (or no solution possible)

Compute solution for displacements

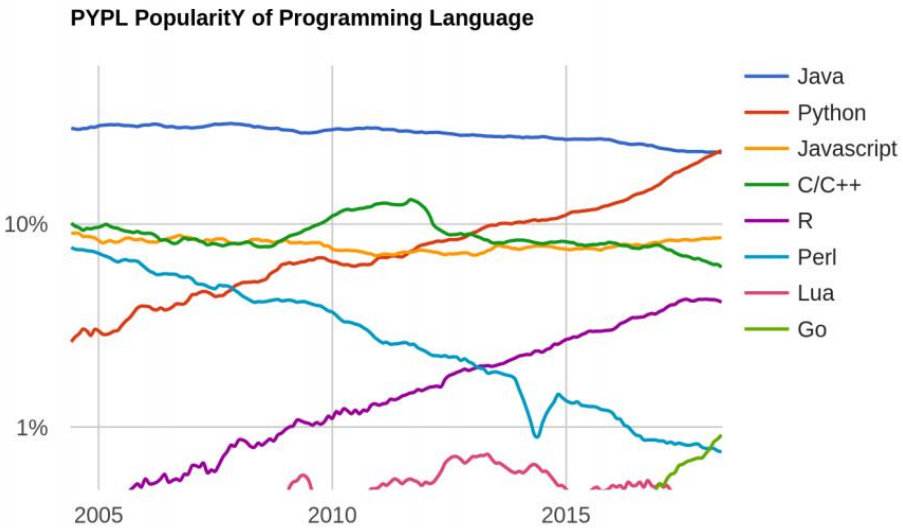
Python

- Open source general-purpose language.
- Object Oriented, Modular
- Easy to interface with C++/C/Java/Fortran
- Interactive environment
- Interpreted and therefore slower than compiled languages

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales

斐波那契数列

```
def fib(n):
    a, b = 0, 1
    while a < n:
      print(a, end=' ')
      a, b = b, a+b
    print()
fib(1000)
```



$ python3.4 1.py
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987

http://pypl.github.io/PYPL.html

end=' '不换行是python3版本的用法，python2版本无法编译

Python

>>> import antigravity
>>> import this

sudo apt install python3-pip
(python2: sudo apt install python-pip)

pip3 install matplotlib
pip3 install numpy
pip3 install pandas
pip3 install scipy
pip3 install vpython

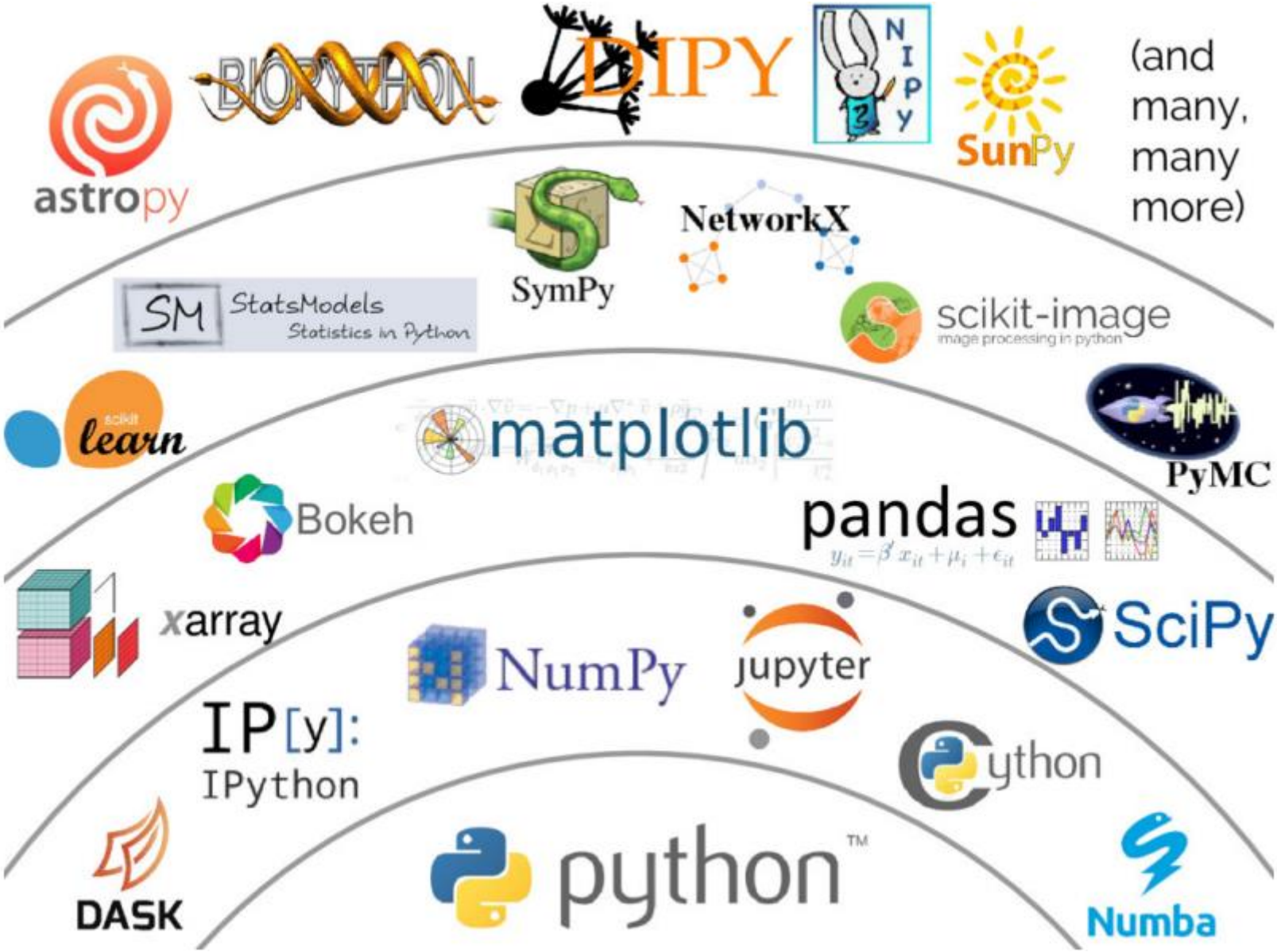
pip3 install --upgrade pip
pip3 install jupyter
jupyter notebook

- C++:
  - Fast
  - Compiled
  - Statically typed
    - int i = 0;
  - Access to pointers
  - Whitespace irrelevant

- python:
  - Slower
  - Interpreted
  - Dynamically typed:
    - i = 0
  - No pointers
  - Whitespace matters

Best case is to have your “human” handling with python and your hardcore computer code in C++ .
Then call the C++ code from python.
This is what scipy, numpy do, etc.

Python-ecosystem

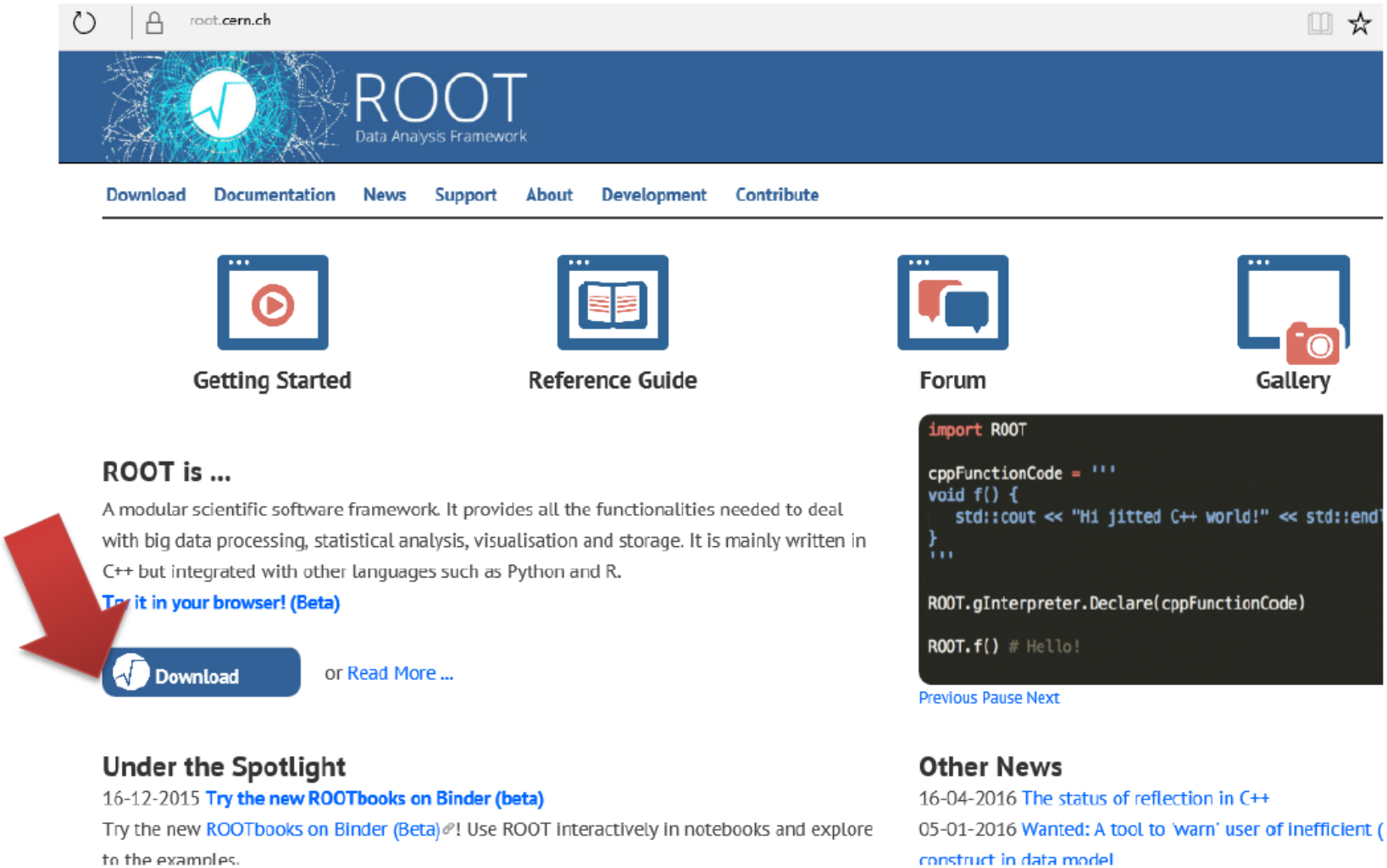


CERN Root

欧洲核子中心(CERN)开发的大型科学数据处理软件

基于Cint(C/C++ interpreter, C-int)是一个C++解释器, 和GCC、VC等编译器不同,它是解释执行C++代码的

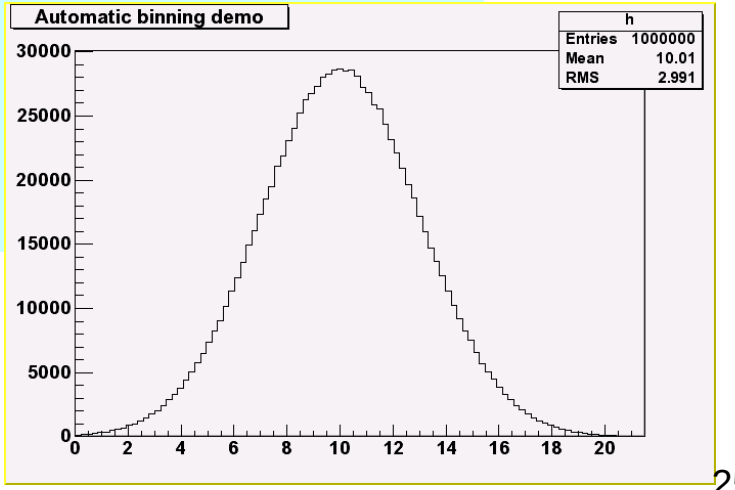
https://root.cern.ch/



CERN Root: Histogram

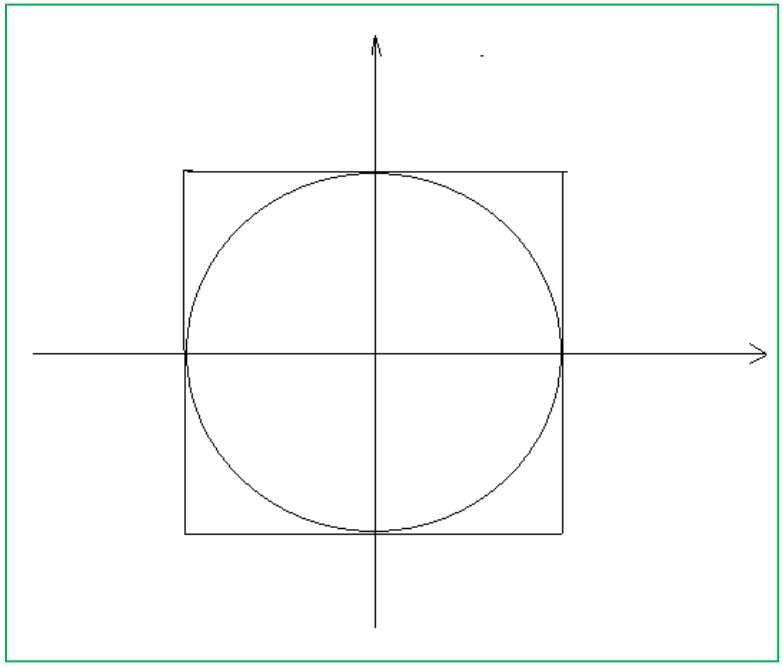
Bin，Events，Mean，RMS

```
#include "TH1.h"
#include "TF1.h"

void demoauto() {
   TF1 *f1 = new TF1("f1","gaus",0,30);
   f1->SetParameters(1,10,3);
   TH1F *h = new TH1F("h","Automatic binning demo",100,0,20);
   for (Int_t i=0;i<1000000;i++) {
      h->Fill(f1->GetRandom());
   }
   h->Draw();
}
```



CERN Root： 随机数计算Pi

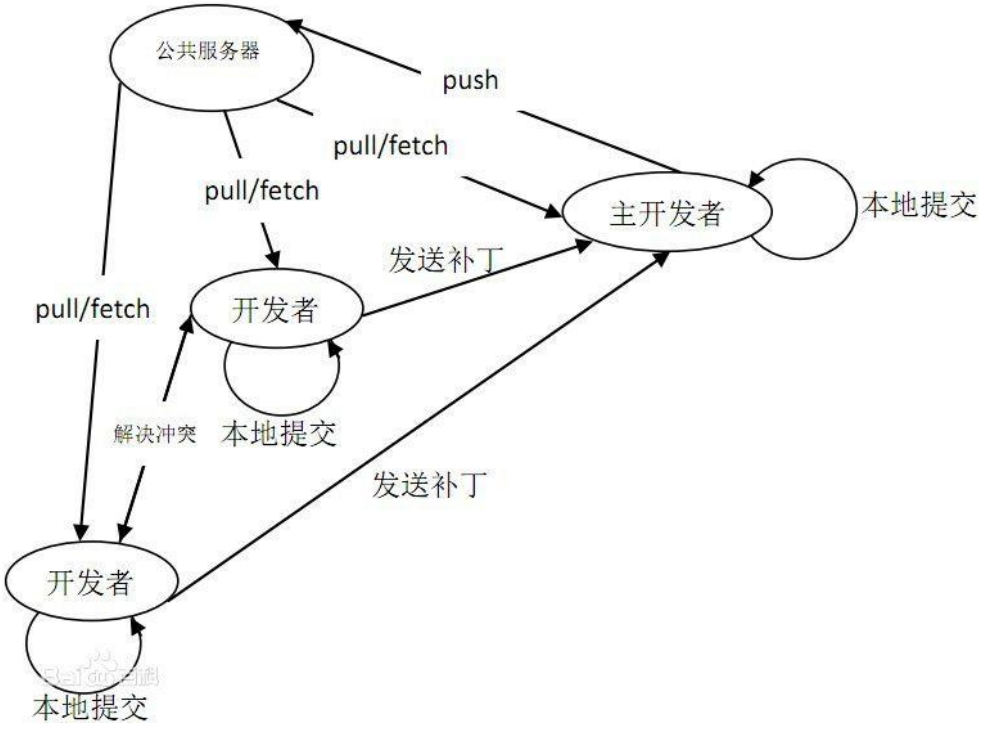
```
void random()
{
//TRandom3, is based on the "Mersenne Twister
generator", and is the recommended one, since
it has good random proprieties (period of about
10**6000 ) and it is fast
        // create random number generator
        gRandom = new TRandom3(0);
        gRandom2 = new TRandom3(1);
        int n=20000;
        int ftot=0;
        for (int i = 0; i < n; ++i) {
          double x=gRandom->Uniform(-1,1);
          double y=gRandom2->Uniform(-1,1);
          double r=sqrt(x*x+y*y);
          if(r<1.0) {ftot++;}
        }

       cout<<4*float(ftot)/float(n)<<endl;

}
```



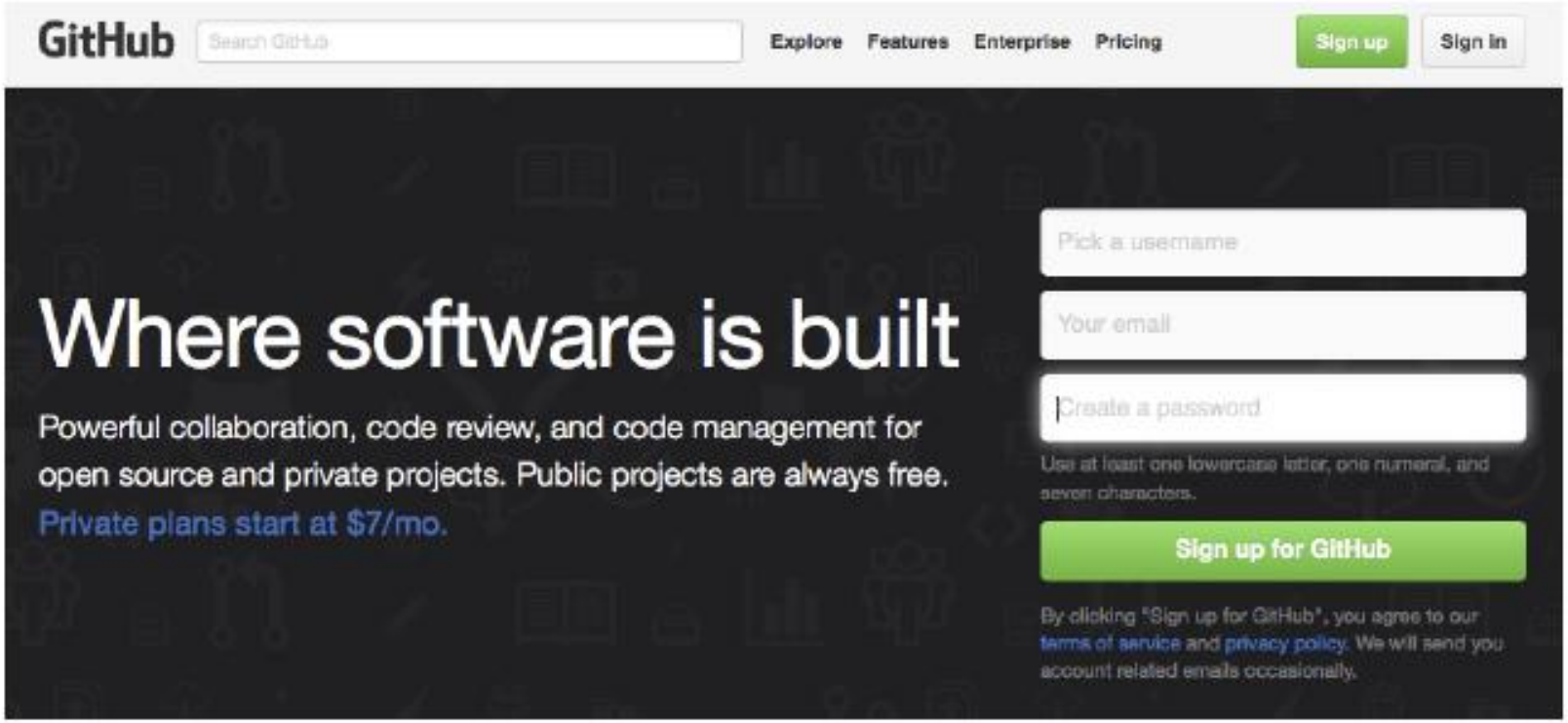
版本控制

- CVS (concurrent versions system)
- SVN (subversion)
- git

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.



Sign Up a Github Account



You will receive an activation email to confirm your email address

Set Up Git

https://help.github.com/articles/set-up-git/

1. Download and install the latest version of Git.

In terminal:

1. $ git config --global user.name "*YOUR NAME*"
2. $ git config --global user.email "*YOUR EMAIL ADDRESS*"

Authenticating with GitHub from Git

• Connecting over HTTPS (recommended)

If you clone with HTTPS, you can cache your GitHub password in Git using a credential helper.

https://help.github.com/articles/caching-your-github-password-ingit/

• Connecting over SSH

If you clone with SSH, you must generate SSH keys on each computer you use to push or pull from GitHub.

CERN Root随机数产生器

1. From the original implementation in FORTRAN by Fred James as part of CLHEP. The initialisation is carried out using a **Multiplicative Congruential generator** using formula constants of L'Ecuyer as described in "F.James, Comp. Phys. Comm. 60 (1990) 329-344".
https://root.cern.ch/doc/master/TRandom1_8cxx_source.html

2. Random number generator class based on the **maximally quidistributed combined Tausworthe generator** by L'Ecuyer. The period of the generator is 2**88 (about 10**26) and it uses only 3 words for the state.
https://root.cern.ch/doc/master/classTRandom2.html

3. TRandom3, is based on the "**Mersenne Twister generator**", and is the recommended one, since it has good random proprieties (period of about 10**6000) and it is fast
http://root.cern.ch/root/html/TRandom3.html

gRandom = new TRandom1/2/3(0);

随机性的统计检验

- 设有在区间$[0,1]$上的伪随机数序列为$\{\xi_1, \xi_2 ... \xi_n\}$
- 如果该伪随机数是均匀分布的，则将$[0,1]$区间分成k 个相等的子区间后，落在每个子区间的伪随机数个数应当近似为$\mathrm{m_k} = \frac{\mathrm{N}}{\mathrm{k}}$，此数为**理论频数**。
- 统计随机数落在第k个子区间的**实际频数**$\mathrm{n_k}$，它应当趋近于理论频数$\mathrm{m_k}$。
- 注意此处的$\mathrm{n_k}$是整数而$\mathrm{m_k}$可以是小数。
- 考察统计量χ^2，它定义为：

$$\chi^2 = \sum_{k=1}^{k} \frac{(n_k - m_k)^2}{m_k}$$

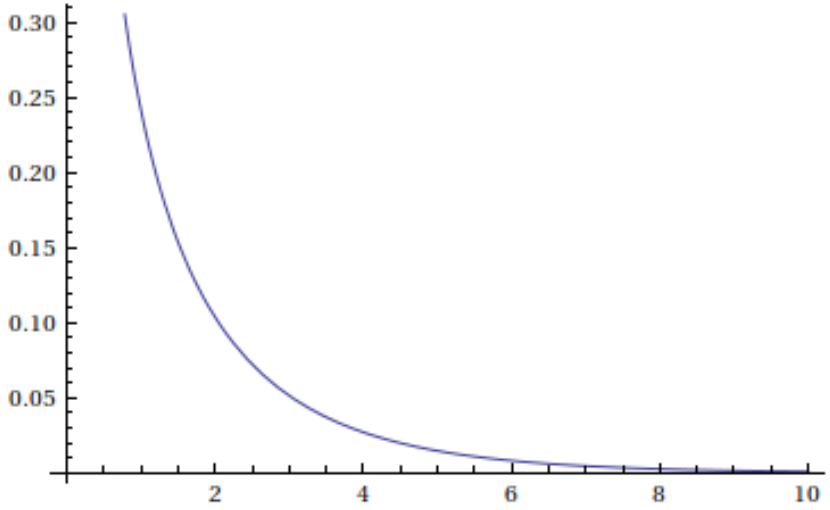
- 如果χ^2值很大，表示远远偏离理想值，因此要求χ^2值尽可能小。通常求和中的每一项的大小约为1，因此χ^2的值约为k
- 概率论中的Pearson定理说明，(15)式的极限概率分布是$\chi^2(k-1)$分布

随机性的统计检验

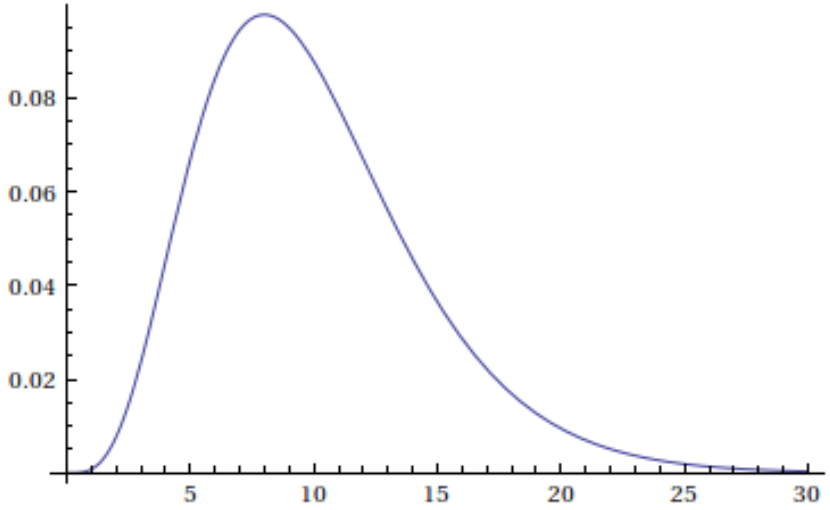
$$P(\chi^2 \le x \mid \upsilon) = \frac{1}{2^{\upsilon/2}\Gamma(\upsilon/2)} \int_0^x t^{(\upsilon-2)/2} e^{-t/2} dt$$

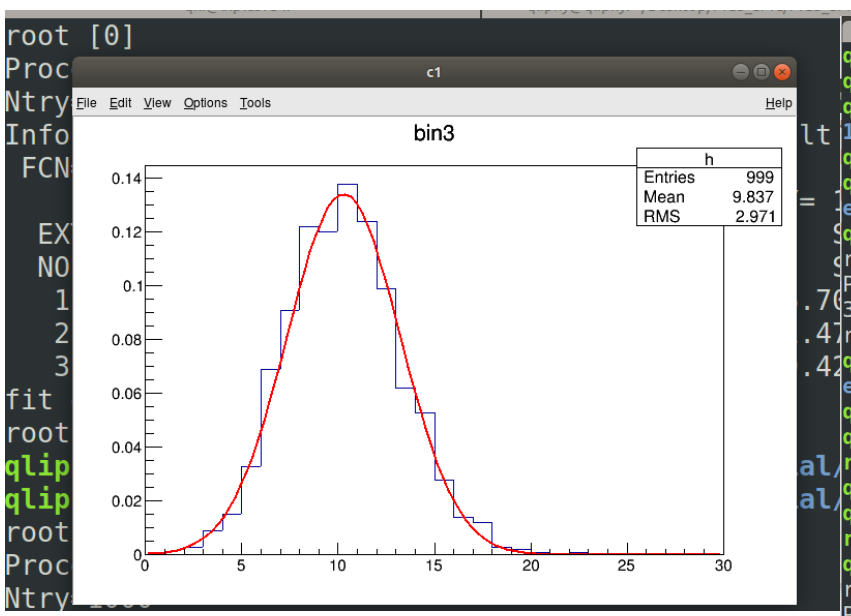
- 它给出了(15)式中的其中$\chi^2 \le x$的概率。整数υ 是系统的自由度，表示独立测量的次数。
- 但是有一个约束条件存在，$\sum_{k=1}^{k} m_k = N$,故自由度$\upsilon = k-1$。
- 据此可以假定一个显著性水平值来进行检验。当给定显著水平α后(或置信度$1-\alpha$)，由方程$P(\chi_\alpha^2 \mid \upsilon) = 1-\alpha$解出$\chi_\alpha$ 值，或者直接从χ^2表查得$\mathrm{k}-1$个自由度的显著水平为α时的χ_α值。
- 如果由(16)式计算出来的χ小于χ_α，则认为在此置信度下，原伪随机数在$[0,1]$区间是均匀分布的假定是正确的。
- 如果由(16)式计算出来的χ大于χ_α，则认为在α 的显著水平下，伪随机数不满足均匀性的要求。

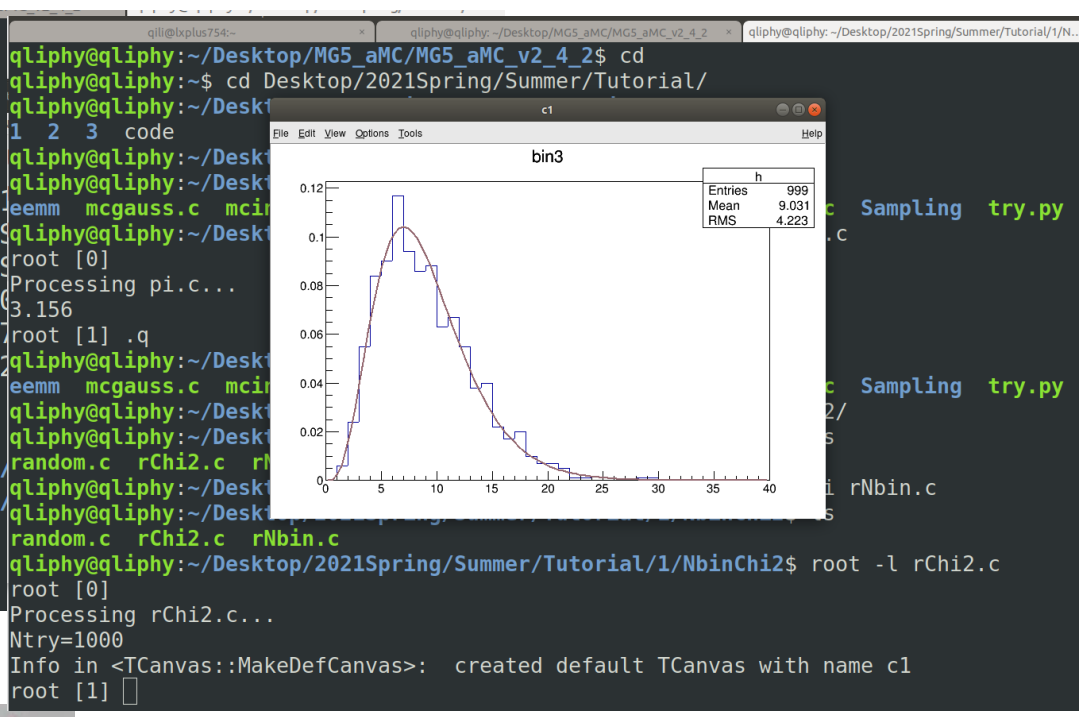
```
Plot[(x^{n/2-1}*Exp[-x/2]/2^{n/2}/Gamma[n/2]) /. {n → 1}, {x, 0, 10}]
```




```
Plot[(x^{n/2-1}*Exp[-x/2]/2^{n/2}/Gamma[n/2]) /. {n → 10}, {x, 0, 30}]
```







$$\text{新}\ N^{(k)} \to \propto e^{-x^2/2}$$

$$e^{-r^2/2}\,dx\,dy\,dz\ldots \Rightarrow e^{-r^2/2}\,r^{n-1}\,dr$$

$$x = r^2 \Rightarrow e^{-x/2}\,dx^{n/2}$$

$$= \boxed{e^{-x/2}\,x^{\frac{n}{2}-1}\,du}$$

随机性的统计检验

```
root [0] TMath::Prob(10.82,1)
(Double_t)1.00409489093039703e-03
root [1] TMath::Prob(10.83,1)
(Double_t)9.98686379180259171e-04
root [2] TMath::Prob(3.84,1)
(Double_t)5.00435212487051889e-02
```

- 通常置信度水平选取为0.99或者0.95。为了反映均匀性分布的特性，k的取值不宜太小，但也不能太大。一般选取的k值，要能使每个子区间有若干个伪随机数时就比较合适。

Degrees of freedom (df)	χ^2 value[19]										
1	0.004	0.02	0.06	0.15	0.46	1.07	1.64	2.71	3.84	6.64	10.83
2	0.10	0.21	0.45	0.71	1.39	2.41	3.22	4.60	5.99	9.21	13.82
3	0.35	0.58	1.01	1.42	2.37	3.66	4.64	6.25	7.82	11.34	16.27
4	0.71	1.06	1.65	2.20	3.36	4.88	5.99	7.78	9.49	13.28	18.47
5	1.14	1.61	2.34	3.00	4.35	6.06	7.29	9.24	11.07	15.09	20.52
6	1.63	2.20	3.07	3.83	5.35	7.23	8.56	10.64	12.59	16.81	22.46
7	2.17	2.83	3.82	4.67	6.35	8.38	9.80	12.02	14.07	18.48	24.32
8	2.73	3.49	4.59	5.53	7.34	9.52	11.03	13.36	15.51	20.09	26.12
9	3.32	4.17	5.38	6.39	8.34	10.66	12.24	14.68	16.92	21.67	27.88
10	3.94	4.87	6.18	7.27	9.34	11.78	13.44	15.99	18.31	23.21	29.59
P value (Probability)	0.95	0.90	0.80	0.70	0.50	0.30	0.20	0.10	0.05	0.01	0.001

the probability of observing a test statistic *at least* as extreme in a chi-squared distribution

The central limit theorem

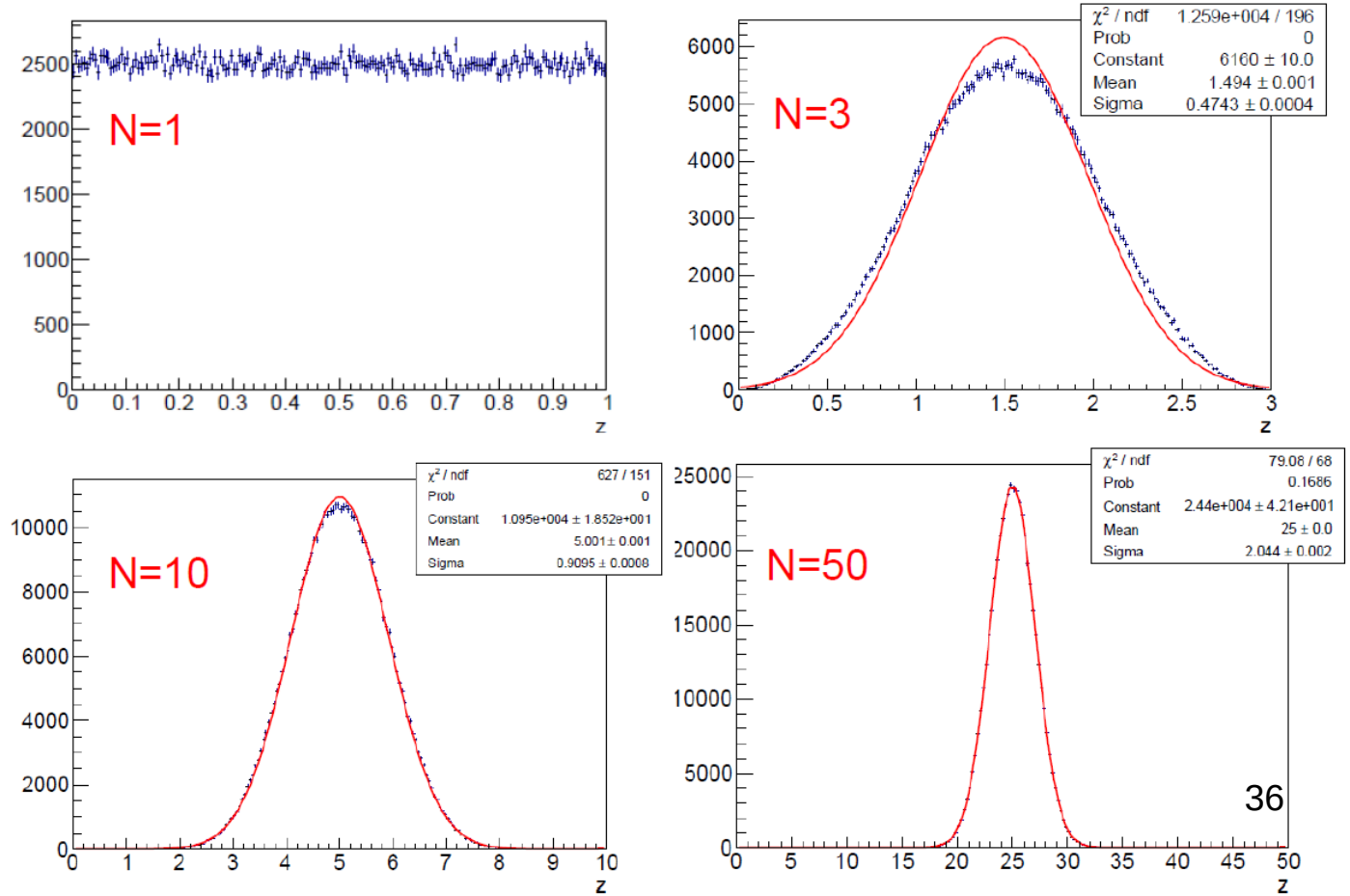
Test central limit theorem

The sum of many random variables will tend to a Gaussian distribution!

This is at the basis of MUCH of the statistics practice we do

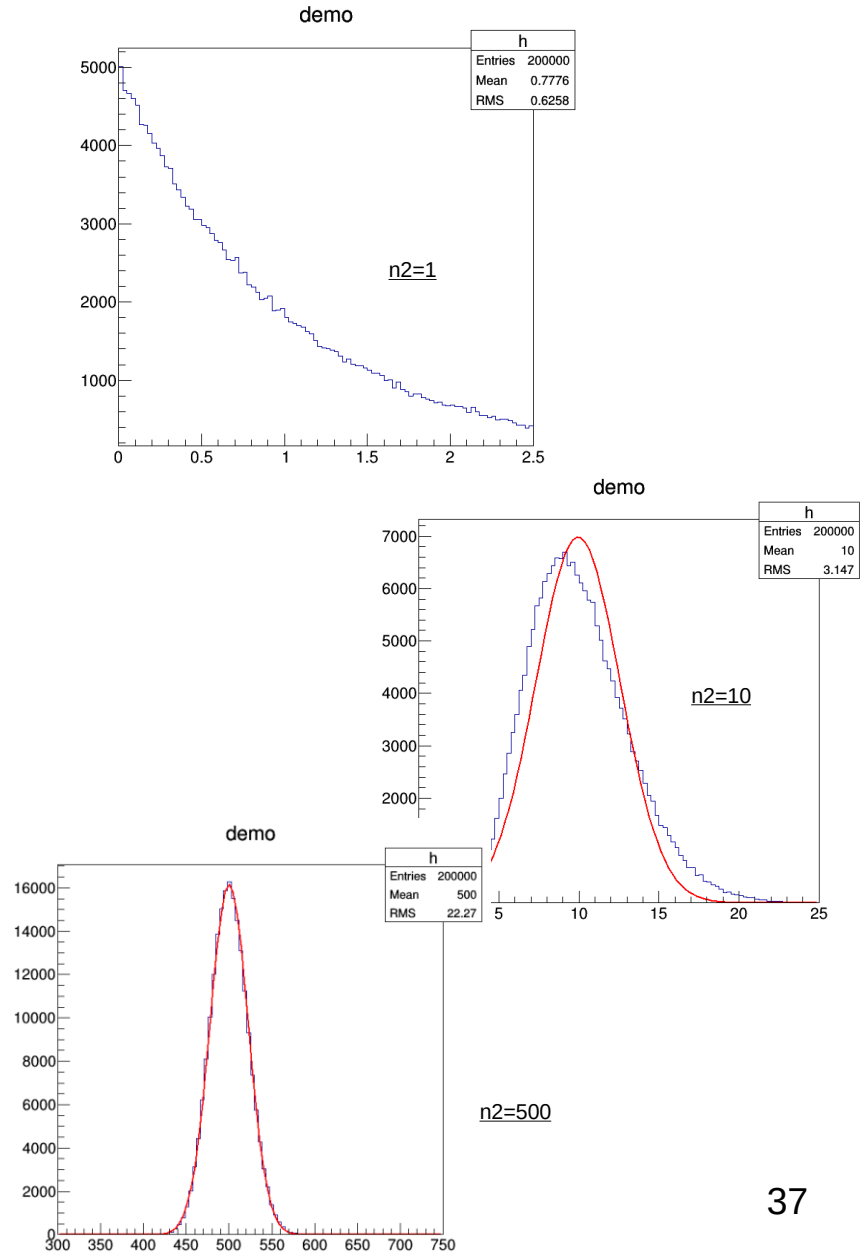
$$z = \sum_{i=1}^{N} x_i$$

Here we take x_i from a Uniform distribution and plot z when N becomes large



The central limit theorem

```
void random0()
{   TCanvas *c1= new TCanvas("c1", "demo",
10,10,700,700);
      gRandom = new TRandom3(0);
      int n=200000;
      int n2=500;
      double ftot;
      TF1 *f1 = new
TF1("f1","gaus",0.6*n2,1.5*n2);
      TH1F *h = new
      TH1F("h","demo",100,0.6*n2,1.5*n2);
      for (int i = 0; i < n; ++i) {
        ftot=0.0;
          for (int j = 0; j < n2; ++j) {
          double x=gRandom->Exp(1);
          ftot+=x; }
        h->Fill(ftot);
      }
      h->Fit("gaus");
      TF1 *fit = h->GetFunction("gaus");
      Double_t chi2 = fit->GetChisquare();
      cout<<chi2<<endl;
      h->Draw();
      c1->SaveAs("Exp_500.png");
}
```



MC积分：平均值法

- 平均值法是建立在大数定理的基础之上；
- 如果函数f(x)在$[a,b]$区间，以均匀的概率分布密度随机地取N个数x_i，对每个x_i 计算出函数值$f(x_i)$。大数法则告诉我们这些函数值之和除以N所得的值将收敛于函数f的期望值，即：

$$\langle f\rangle = \frac{1}{N}\sum_{i=1}^{N} f(x_i) = \frac{1}{(b-a)}\int_a^b f(x)dx$$

$$Var(<f>) = \frac{V^2}{N^2}\sum_{1}^{N} Var(f) = V^2\frac{Var(f)}{N}$$

- 故有：

$$\int_a^b f(x)dx = \frac{(b-a)}{N}\sum_{i=1}^{N} f(x_i)$$

- 在简单抽样中，我们由均匀分布中选取随机数，并不考虑被积函数的具体情况。因此，被积函数的极大处和极小处有相同的抽样权重。
- 但是对积分贡献较大的更多在函数值较大处，故直观上就可以看出，非权重的简单抽样效率较低。即为了获得较高计算结果的精度，需要大量的抽样。

中心极限定理

- 与大数法则一样，概率论中的中心极限定理同样是MonteCarlo方法应用于统计计算的基础。它是对积分的蒙特卡洛估计值收敛于该积分的正确结果的收敛程度的研究
- 中心极限定理告诉我们：在有足够大，但又有限的抽样数N的情况下，蒙特卡洛估计值的误差分布。即：

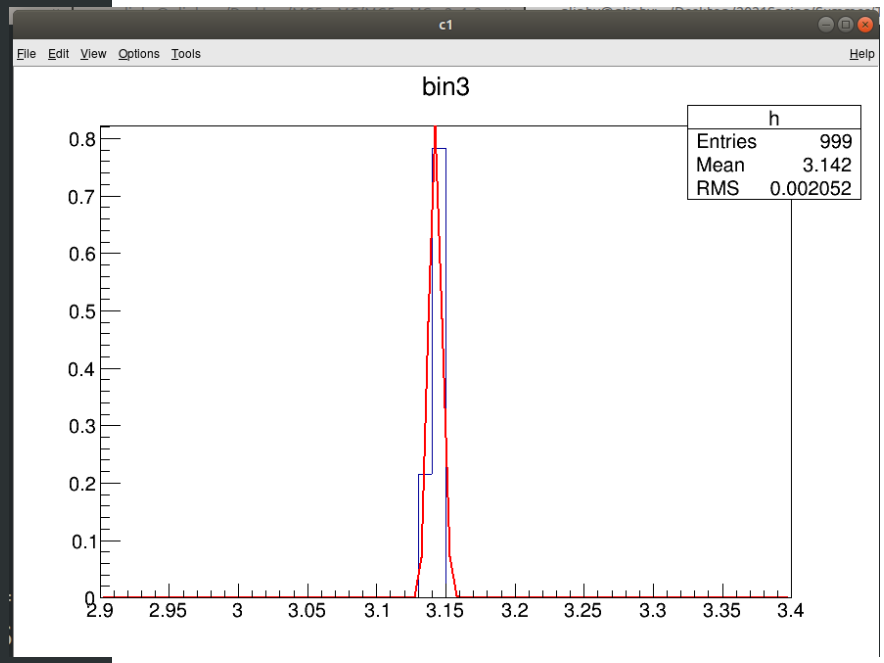
$$P\left\{\left|\frac{\langle f\rangle-\mu}{\sigma_f/\sqrt{N}}\right|<\beta\right\}\to\Phi(\beta)$$

- 其中$\mu=\frac{1}{b-a}\int_a^b f(x)dx$，$\sigma_f$是函数f(x)的标准偏差，$\Phi(\beta)$是Gauss正态分布，那么对于积分值得标准偏差$\sigma_s$，就有：

$$\sigma_s=|\langle f\rangle-\mu|\propto\frac{\sigma_f}{\sqrt{N}}$$

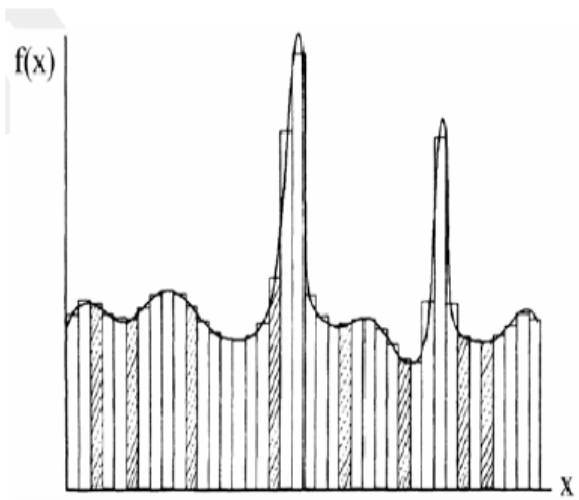
- 从上式可以看到蒙特卡洛积分的误差和σ_f与N 有关。在方差固定时，增加模拟次数可以有效地减小误差。

```
  int n;
  n=100000;
  double crude_mc, x, sum_sigma, fx, variance;
  gRandom = new TRandom2(0);

  TH1F *h = new TH1F("h","bin3",50,2.9,3.4);
//-------Ntry-----------------
  for(int j=1; j<Ntry; j++) {
     crude_mc = sum_sigma=0. ;
     variance =0.;
     for (int i = 1; i <= n; i++){
        x=gRandom->Uniform(0,1);
        fx=4/(1.+x*x);
        crude_mc += fx;
        sum_sigma += fx*fx;
     }
     crude_mc = crude_mc/((double) n );
     sum_sigma = sum_sigma/((double) n );
     variance=sum_sigma-crude_mc*crude_mc;
     variance=variance/double(n);
     cout<<sqrt(variance)<<endl;
  h->Fill(crude_mc,1.0/double(Ntry));
 }
//-------Ntry-----------------
     h->Fit("gaus");
     TF1 *fit = h->GetFunction("gaus");
     Double_t k2 = fit->GetChisquare();
```

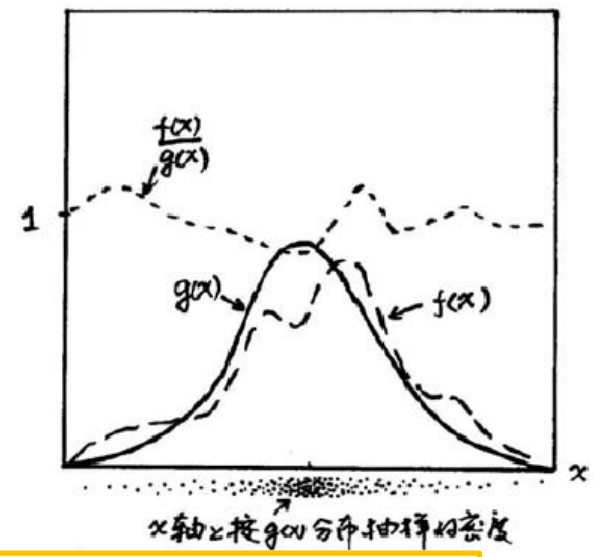


重要抽样法

简单抽样法积分近似度较差



重要抽样法示意图



- 重要抽样法的原理起源于数学上的变量代换方法的思想，即：

$$\int_0^1 f(x)dx = \int_0^1 \frac{f(x)}{g(x)} g(x)dx = \int \frac{f(x)}{g(x)} dG(x)$$

- 此时随机点的选择不再是简单抽样法中的均匀选择，而是以分布函数G(x)分布的
- 这里g(x)称为偏倚分布密度函数。

- 例如，计算积分：

$$I = \int_0^1 \exp(-x/2)dx$$

- 该式当然可以解析算出，这里主要举例说明重要抽样法的求解过程。
- 式中的指数项随x增加而迅速减小，g(x)必须是另一有相同行为的函数，我们用指数函数在$x = 0$附近的渐近展开来近似：

$$g(x) = 1 - \frac{x}{2} + \frac{x^2}{8} - \frac{x^3}{48} \ldots \approx 1 - \frac{x}{2}$$

- 由反函数法抽样：

$$G(x) = \int_0^x (1 - x/2)dx = x - x^2/4$$

$$\eta = 2(1 \pm \sqrt{1-\xi})$$

- 由于$G(0) = 0$，故上式中括号内应取负号，即：

$$\eta = 2(1 - \sqrt{1-\xi})$$

- 原积分即可表示成：

$$I = \frac{1}{N}\sum_{i=1}^{N} \exp \frac{-(1-\sqrt{1-\xi_i})}{\sqrt{1-\xi_i}}$$

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cmath>
using namespace std;
double func(double x);
int main()
{ srand(unsigned(time(0)));  // seed the randomizer
   int i, n;
   double crude_mc, x, sum_sigma, fx, variance;
   cin >> n;
   crude_mc = sum_sigma=0. ;
   // a crude Monte-Carlo method
   for ( i = 1; i <= n; i++){
      x=rand()/(double)RAND_MAX;
      fx=func(x);
      crude_mc += fx;
      sum_sigma += fx*fx;   }
   crude_mc = crude_mc/((double) n );
   sum_sigma = sum_sigma/((double) n );
   variance=sum_sigma-crude_mc*crude_mc;
   variance=sqrt(variance)/double(n); }

double func(double x
{
   double value;
   value = exp(-x/2.0);
   return value;
} // end of function to evaluate
```

```
Read in the number of Monte-Carlo samples
1000
 variance= 0.0129141 Integral = 0.782019
 Exact= 0.786939
```

```
//  int_0^1 exp(-x/2) dx =2(1-exp(-0.5))
//  importance sampling: g(x)=1-x/2.0
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cmath>
using namespace std;
double func(double x);
int main()
{ srand(unsigned(time(0)));  // seed the randomizer
   int i, n;
   double crude_mc, tx, x, sum_sigma, fx, variance;
   cin >> n;
   crude_mc = sum_sigma=0. ;
   for ( i = 1; i <= n; i++){
   tx=3.0/4.0*rand()/(double)RAND_MAX;
   //0=<tx<=3/4;
    x=2.*(1-sqrt(1.0-tx));
    fx=3.0/4.0*func(x);
    crude_mc += fx;
    sum_sigma += fx*fx;  }
   crude_mc = crude_mc/((double) n );
   sum_sigma = sum_sigma/((double) n );
   variance=sum_sigma-crude_mc*crude_mc;
   variance=sqrt(variance)/double(n);}
double func(double x)
{ double value;
   value = exp(-x/2.0)/(1.-x/2.0);
   return value;
} // end of function to evaluate
```

```
Read in the number of Monte-Carlo samples
1000
 variance= 0.00178519 Integral = 0.788191
 Exact= 0.786939
```

VEGAS

ELSEVIER

Journal of Computational Physics

Volume 27, Issue 2, May 1978, Pages 192-203

A new algorithm for adaptive multidimensional integration ☆

G Peter Lepage

Abstract

A new general purpose algorithm for multidimensional integration is described. It is an iterative and adaptive Monte Carlo scheme. The new algorithm is compared with several others currently in use, and shown to be considerably more efficient than all of these for a number of sample integrals of high dimension (n ≳ 4).

The basic technique for importance sampling in VEGAS is to construct, adaptively, a multidimensional weight function g that is *separable*,

$$p \propto g(x, y, z, \ldots) = g_x(x) g_y(y) g_z(z) \ldots$$

Such a function avoids the K^d explosion in two ways: (i) It can be stored in the computer as d separate one-dimensional functions, each defined by K tabulated values, say — so that $K \times d$ replaces K^d. (ii) It can be sampled as a probability density by consecutively sampling the d one-dimensional functions to obtain coordinate vector components $(x, y, z, \ldots)$.

The optimal separable weight function can be shown to be

$$g_x(x) \propto \left[\int dy \int dz \ldots \frac{f^2(x, y, z, \ldots)}{g_y(y) g_z(z) \ldots} \right]^{1/2}$$

GSL - GNU Scientific Library

GNU Operating System

*Sponsored by the **Free Software Foundation***

Introduction

The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers. It is free software under the GNU General Public License.

The library provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1000 functions in total with an extensive test suite.

VEGAS

https://pypi.org/project/vegas/

vegas 3.3.4

```
pip install vegas
```

Project description

This package provides tools evaluating multidimensional

integrals numerically using an enhanced version of

the adaptive Monte Carlo vegas algorithm (G. P. Lepage, J. Comput. Phys. 27(1978) 192).

$$C \int_{-1}^{1} dx_0 \int_{0}^{1} dx_1 \int_{0}^{1} dx_2 \int_{0}^{1} dx_3 \; \mathrm{e}^{-100 \sum_d (x_d - 0.5)^2},$$

where constant C is chosen so that the exact integral is 1. The following code shows how this can be done:

```
import vegas
import math

def f(x):
    dx2 = 0
    for d in range(4):
        dx2 += (x[d] - 0.5) ** 2
    return math.exp(-dx2 * 100.) * 1013.2118364296088

integ = vegas.Integrator([[-1, 1], [0, 1], [0, 1], [0, 1]])

result = integ(f, nitn=10, neval=1000)
print(result.summary())
print('result = %s    Q = %.2f' % (result, result.Q))
```

using `nitn=10` iterations of the `vegas` algorithm, each of which uses no more than `neval=1000` evaluations of the integrand. Each iteration produces an independent estimate of the integral. The final estimate is the weighted average of the results from all 10 iterations, and is returned by `integ(f ...)`. The call `result.summary()` returns a summary of results from each iteration.

This code produces the following output:

```
itn   integral        wgt average     chi2/dof        Q
-------------------------------------------------------
  1   2.4(1.9)        2.4(1.9)            0.00     1.00
  2   1.19(32)        1.23(32)            0.42     0.52
  3   0.910(90)       0.934(87)           0.68     0.51
  4   1.041(70)       0.999(55)           0.76     0.52
  5   1.090(43)       1.055(34)           1.00     0.41
  6   0.984(34)       1.020(24)           1.24     0.29
  7   1.036(27)       1.027(18)           1.07     0.38
  8   0.987(22)       1.011(14)           1.20     0.30
  9   0.995(18)       1.005(11)           1.11     0.35
 10   0.993(17)       1.0015(91)          1.02     0.42

result = 1.0015(91)     Q = 0.42
```

`result.mean` — weighted average of all estimates of the integral;

`result.sdev` — standard deviation of the weighted average;

`result.chi2` — χ^2 of the weighted average;

`result.dof` — number of degrees of freedom;

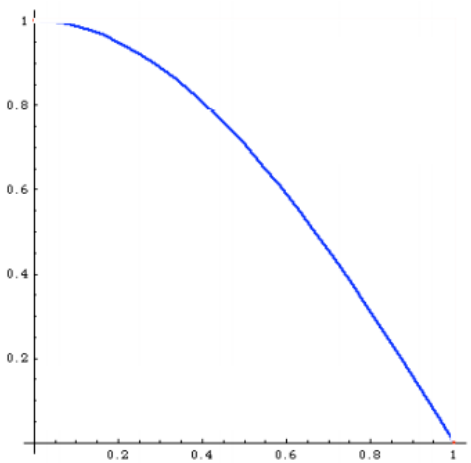
`result.Q` — Q or *p-value* of the weighted average's χ^2;

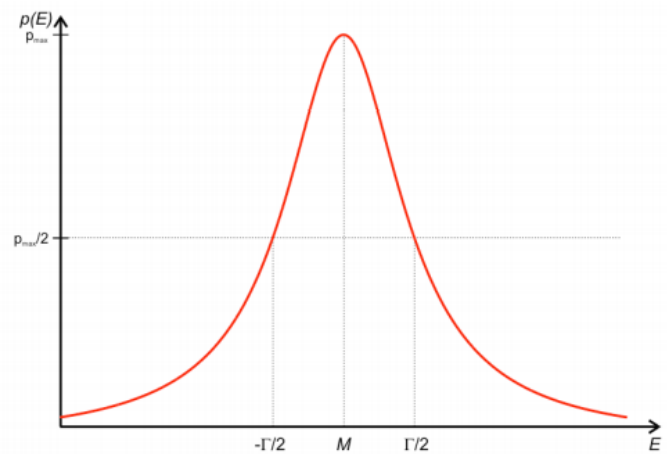
`result.itn_results` — list of the integral estimates from each iteration.

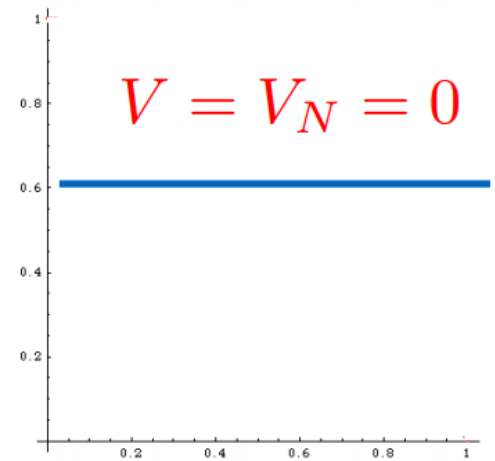
In this example the final Q is 0.42, indicating that the χ^2 for this average is not particularly unlikely and thus the error estimate is most likely reliable.

MC integration at Colliders

$$I = \int_0^1 dx \cos\frac{\pi}{2}x \qquad \int \frac{dq^2}{(q^2 - M^2 + iM\Gamma)^2} \qquad \int dx\, C$$







$$V = V_N = 0$$

$$I = \int_{x_1}^{x_2} f(x)dx \quad \Rightarrow \quad I_N = (x_2 - x_1)\frac{1}{N}\sum_{i=1}^{N} f(x)$$

$$V = (x_2 - x_1)\int_{x_1}^{x_2} [f(x)]^2 dx - I^2 \quad \Rightarrow \quad V_N = (x_2 - x_1)^2 \frac{1}{N}\sum_{i=1}^{N}[f(x)]^2 - I_N^2$$

$$I = I_N \pm \sqrt{V_N/N}$$

Can be minimized!

Phase Space at Colliders

$$d\Phi_2 = (2\pi)^4 \delta^4(Q-p_1-p_2) \frac{d^3p_1}{(2\pi)^3 2E_1} \frac{d^4p_2}{(2\pi)^3} \delta(p_2^2-m_2^2)$$

$$= \frac{1}{(2\pi)^2} \delta((Q-p_1)^2-m_2^2) \frac{d^3p_1}{2E_1} \quad \frac{|p_1|^2 d|p_1| d\cos\theta d\varphi}{2E_1}$$

Q 质心系下

$$= \frac{1}{(2\pi)^2} \delta\left[S - 2\sqrt{S}E_1 + m_1^2 - m_2^2\right] \frac{|p_1|^2 d|p_1| d\cos\theta d\varphi}{2E_1}$$

$$= \frac{1}{8\pi^2} \delta(\cdots) \frac{|p_1| \, d|p_1|^2 \, d\cos\theta d\varphi}{E_1} \qquad (E^2-m^2) \qquad 2dE_1$$

$$= \frac{1}{8\pi^2} \frac{\sqrt{E_1^2-m_1^2}}{2 \times 2\sqrt{S}} \, 2 \, d\cos\theta \, d\varphi$$

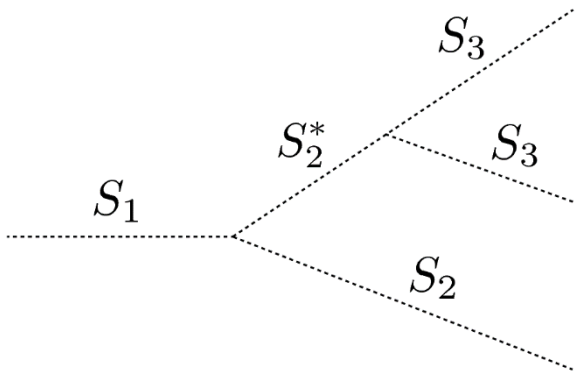
← 质心系中能量

$$E_1 = \frac{S+m_1^2-m_2^2}{2\sqrt{S}}$$

$$\lambda = \frac{\sqrt{(S-m_1^2-m_2^2)-(4m_1^2m_2^2)}}{S}$$

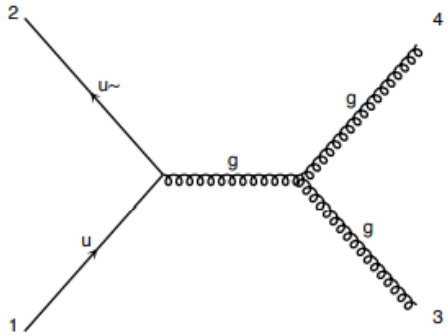
$$\Rightarrow \frac{d\cos\theta d\varphi}{32\pi^2} \lambda(S, m_1^2, m_2^2)$$

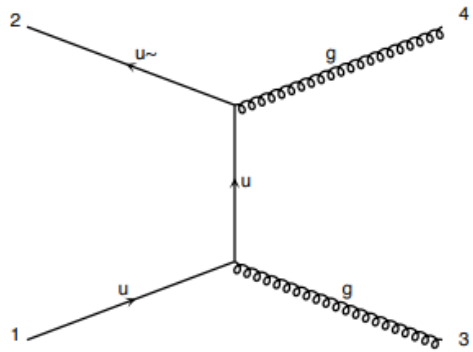
$$d^4p\,\delta(p^2-m^2) = d^4p\,\delta(E^2-p^2-m^2)$$
$$= d^3p \; dE\,\delta(E^2-p^2-m^2)$$
$$= \frac{d^3p}{2E_p}$$

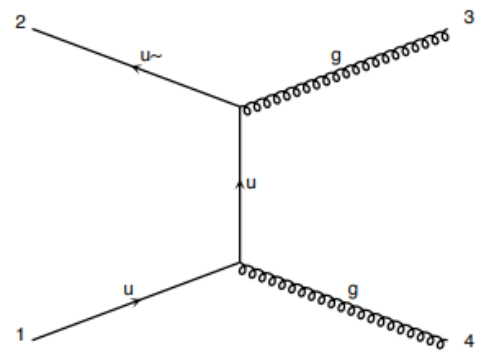


$$d\Phi_n(P, p_1, ..., p_n) = \frac{1}{2\pi} dQ^2 d\Phi_j(Q, p_1, ..., p_j) d\Phi_{n-j+1}(P, Q, p_{j+1}, ..., p_n).$$

MadEvent







$$\propto \frac{1}{\hat{s}} = \frac{1}{(p_1 + p_2)^2} \qquad \propto \frac{1}{\hat{t}} = \frac{1}{(p_1 - p_3)^2} \qquad \propto \frac{1}{\hat{u}} = \frac{1}{(p_1 - p_4)^2}$$

Three very different pole structures contributing to the same matrix element.

MadEvent

Multi-channel based on single diagrams*

Trick in MadEvent: Split the complexity

$$\int |M_{tot}|^2 = \int \frac{\sum_i |M_i|^2}{\sum_j |M_j|^2} |M_{tot}|^2 = \sum_i \int \underbrace{\frac{|M_i|^2}{\sum_j |M_j|^2}}_{\approx 1} |M_{tot}|^2$$

Key Idea

- Any single diagram is "easy" to integrate (pole structures/ suitable integration variables known from the propagators)
- Divide integration into pieces, based on diagrams
- All other peaks taken care of by denominator sum

MadEvent

$$\int |M_{tot}|^2 = \int \frac{\sum_i |M_i|^2}{\sum_j |M_j|^2} |M_{tot}|^2 = \sum_i \int \frac{|M_i|^2}{\sum_j |M_j|^2} |M_{tot}|^2$$

P1_qq_wpwm

s= 725.73 ± 2.07 (pb)

Graph	Cross-Section ↓	Error	Events (K)	Unwgt	Luminosity
G2.2	377.6	1.67	142.285	7941.0	21
G3	239	1.16	220.04	10856.0	45.5
G1	109.1	0.378	70.88	3793.0	34.8

P1_gg_wpwm

s= 20.714 ± 0.332 (pb)

Graph	Cross-Section ↓	Error	Events (K)	Unwgt	Luminosity
G1.2	20.71	0.332	7.01	373.0	18

- In practice not all diagram are included

- We do **not include** diagram with 4 point interaction

MadEvent

Pure interference

$$\int |M_{tot}|^2 = \int \frac{\sum_i |M_i|^2}{\sum_j |M_j|^2} |M_{tot}|^2 = \sum_i \int \frac{|M_i|^2}{\sum_j |M_j|^2} |M_{tot}|^2$$

$\neq 1$

Status

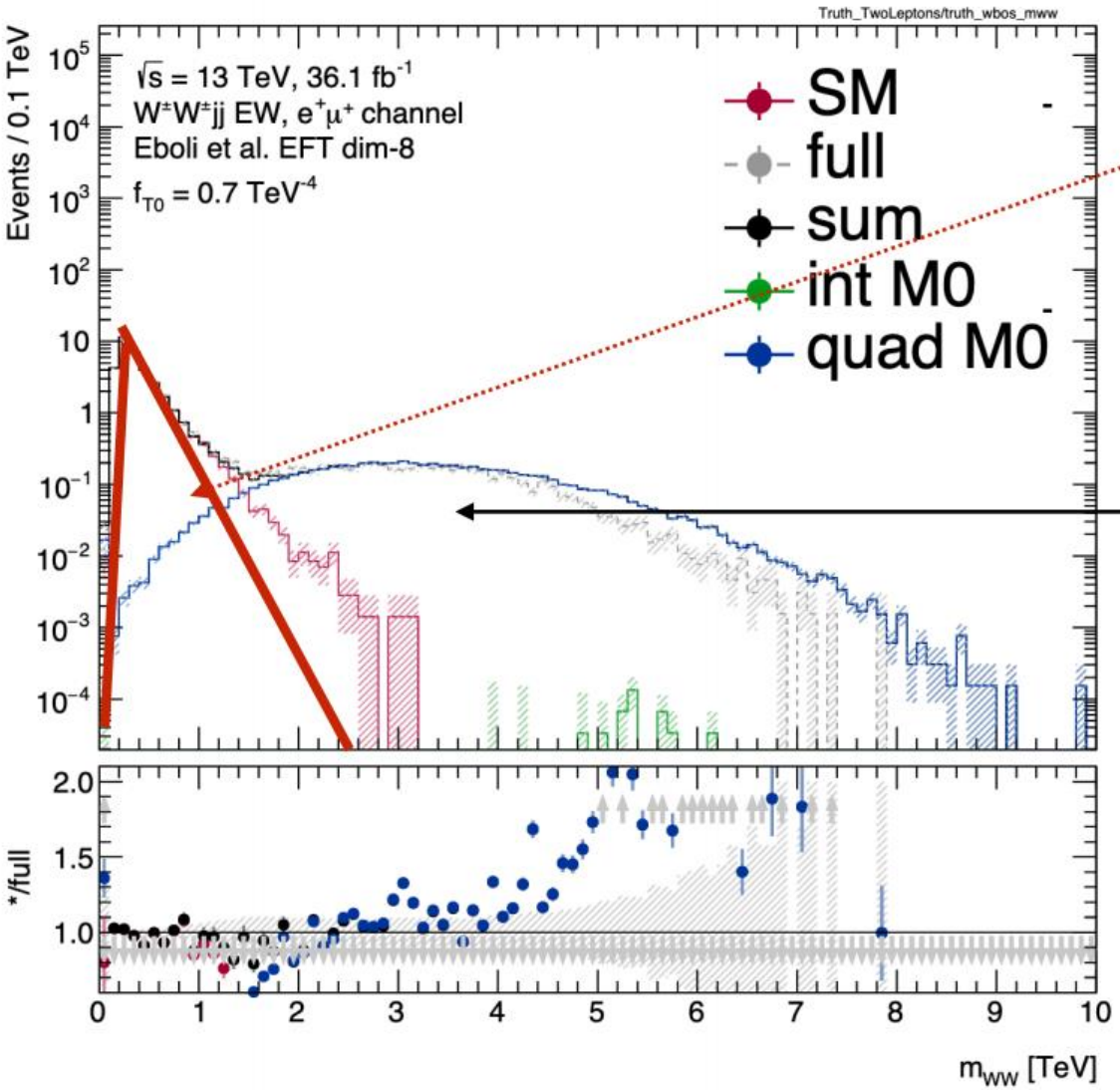
Single diagram enhancement method does not hold for pure interference

- No alternative implemented in MG5aMC
- Integration should be consider as **suspicious** by default

Advice

- do SM + EFT interference

MadEvent



MadGraph expects

Did not know about resonances features

Relies on Machine Learning algorithm to detects it and compensate
-> efficiency issue
-> bias issue