Simulating QCD Jet Production in e^+e^- annihilation

– concepts of Monte Carlo event generators for collider experiments –

Steffen Schumann

Institut für Theoretische Physik

Uni Göttingen SS 24

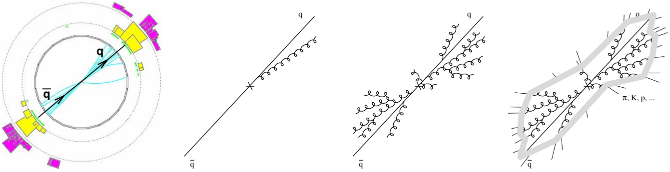
Intro Lecture – part II

12/06/24

Outline of the project

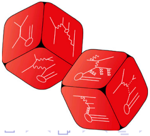
physics issues related to the modelling of high-energy collisions

- **Part I:** hard process generation, here $e^+e^- \to q\bar{q}$ at lowest order
  - integrate cross section for $e^+e^- \to q\bar{q}$
  - generate corresponding scattering events fully differentially
- **Part II:** final-state parton showering, *i.e.* soft- & collinear gluon emissions
  - simulate $q\bar{q}$-initiated final-state parton cascade
  - analyse QCD jet observables, compare to reference data

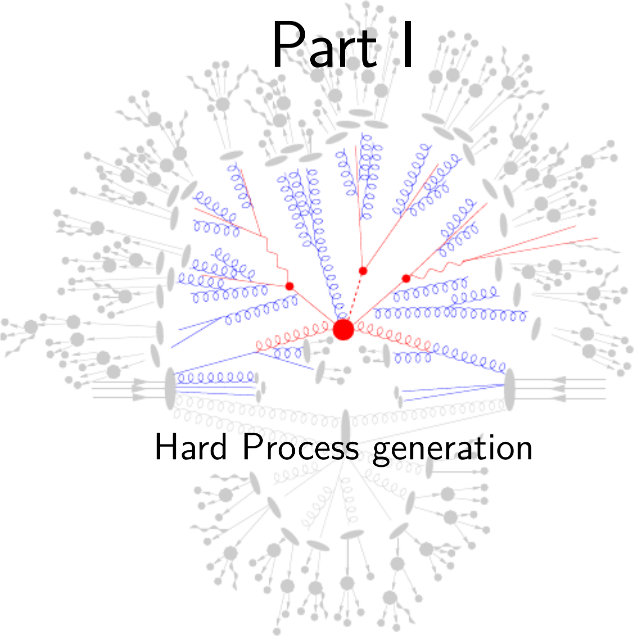


aspects of numerical methods/solutions used to tackle these

- Monte Carlo Importance Sampling
- Markov Chain parton-branching simulation
- sequential jet reconstruction algorithms



Part I



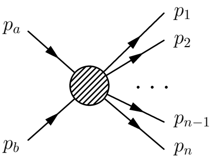
Hard Process generation

the partonic cross section

$$\frac{d\sigma_{2\to n}}{dX} = \frac{1}{\mathrm{flux}} \int d\Phi_n(p_1,\ldots,p_n)\,|\mathcal{M}_{2\to n}(\{p_n\})|^2\ \rho_n(p_1,\ldots,p_n)$$

- Φ_n is the n-particle phase space $dim[\Phi_n] = 3n-4$

$$\mathrm{d}\Phi_n = \delta^4\left(p_a + p_b - \sum_{i=1}^{n} p_i\right) \prod_{i=1}^{n} \frac{\mathrm{d}^3\vec{p}_i}{(2\pi)^3 2E_i}$$



- $\mathcal{M}_{2\to n}$ invariant QFT matrix element (perturbation theory in α_s, α)
- measurement function ρ_n projects out observable

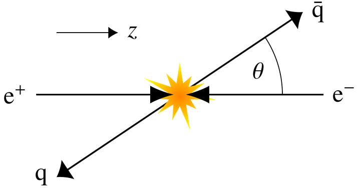
$$\rho_n = \begin{cases} 1 & : \text{ total cross section} \\ \delta(X - \chi_n(p_1,\ldots,p_n)) & : \text{ differential cross section} \end{cases}$$

$\rightsquigarrow$ implements acceptance cuts, observable definition

Part I: Hard Process generation

- consider leading-order process $e^+e^- \to q\bar{q}$

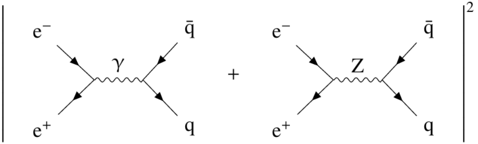
  $\rightsquigarrow$ simple most parton production channel in e^+e^-, *e.g.* at LEP



- fully differential scattering cross section given by

$$\frac{d\sigma_{q\bar{q}}}{dsd(\cos\theta)d\phi} = f(s)\frac{1}{8\pi}\frac{1}{4\pi}\frac{1}{2s}|\mathcal{M}_{q\bar{q}}(s,\cos\theta,\phi)|^2$$

$|\mathcal{M}_{q\bar{q}}(s,\cos\theta,\phi)|^2 =$



Part I: Hard Process generation

scattering process kinematics

- collisions in centre-of-mass frame (here equal to lab frame)

  $p_{e^+} = \sqrt{s}/2(1, 0, 0, 1),\ p_{e^-} = \sqrt{s}/2(1, 0, 0, -1)$
- four-momentum conservation & on-shell conditions:

  $p_{e^+} + p_{e^-} = p_q + p_{\bar{q}} \quad \& \quad p_{e^+}^2 = p_{e^-}^2 = p_q^2 = p_{\bar{q}}^2 = 0$
- final-state momenta in spherical coordinates

$$\begin{aligned} p_q &= \sqrt{s}/2(1, -\cos\phi\sin\theta, -\sin\phi\sin\theta, -\cos\theta) \\ p_{\bar{q}} &= \sqrt{s}/2(1, +\cos\phi\sin\theta, +\sin\phi\sin\theta, +\cos\theta) \end{aligned}$$

$\leadsto$ can construct arbitrary observable X & differential cross section $d\sigma/dX$

Part I: Hard Process generation

scattering process kinematics

- collisions in centre-of-mass frame (here equal to lab frame)
  $p_{e^+} = \sqrt{s}/2(1,0,0,1),\; p_{e^-} = \sqrt{s}/2(1,0,0,-1)$
- four-momentum conservation & on-shell conditions:
  $p_{e^+} + p_{e^-} = p_q + p_{\bar{q}} \quad \& \quad p_{e^+}^2 = p_{e^-}^2 = p_q^2 = p_{\bar{q}}^2 = 0$
- final-state momenta in spherical coordinates

$$\begin{aligned} p_q &= \sqrt{s}/2(1, -\cos\phi\sin\theta, -\sin\phi\sin\theta, -\cos\theta) \\ p_{\bar{q}} &= \sqrt{s}/2(1, +\cos\phi\sin\theta, +\sin\phi\sin\theta, +\cos\theta) \end{aligned}$$

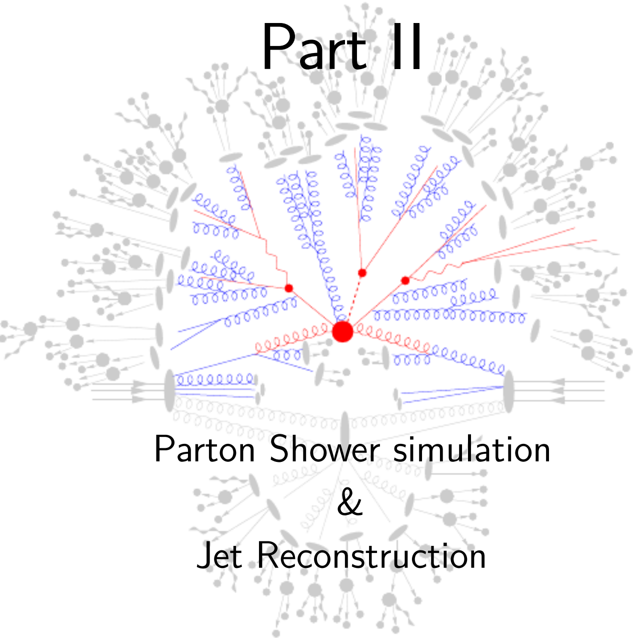
$\rightsquigarrow$ can construct arbitrary observable X & differential cross section $d\sigma/dX$

pre-defined project tasks

1. consider collisions at fixed centre-of-mass energy, *i.e.* $\sqrt{s} = M_Z$
   - Monte-Carlo integration of total cross section, uniform sampling of θ and ϕ
   - study MC error estimate, compare to VEGAS integration package

2. consider variable collision energy, *i.e.* $\sqrt{s} \in [M_Z - 3\Gamma_Z, M_Z + 3\Gamma_Z]$
   - use Breit–Wigner importance sampling to tame s integral
   - quantify integration performance increase

Part II



Parton Shower simulation

&

Jet Reconstruction

QCD Bremsstrahlung

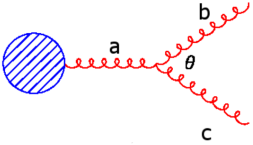
accelerated charges radiate

- QED: electrons (charged) emit photons
- QCD: quarks (coloured) emit gluons (quarks triplets, gluons octets)

  but, gluons coloured as well $\rightsquigarrow$ gluons emit gluons

$\rightsquigarrow$ **QCD radiation enhanced in the infra-red: soft / collinear emissions**

$\rightsquigarrow$ **real-emission matrix elements factorize in collinear limit [universal]**



$t = p_a^2,\ z = E_b/E_a$

$$\boxed{d\sigma_{n+1} = d\sigma_n \frac{dt}{t}\, dz \frac{\alpha_S}{2\pi} P_{ba}(z)}$$

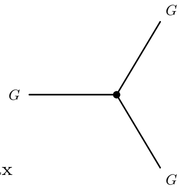
$\rightsquigarrow$ iteration / Markov process

$\rightsquigarrow$ **parton shower MC**

Parton Shower: Toy Model

one particle species G only, starting scale $t = t_{\max}$

$$d\mathcal{P}_G = P_{GG}(z)\frac{dt}{t}dz \quad \stackrel{\text{given } t}{\rightarrow} \quad I(t) = \int_{z_-(t)}^{z_+(t)} dz P_{GG}(z)$$



$\rightsquigarrow$ probability of no-branching between $t_{\max}$ and $t < t_{\max}$

$$\mathcal{P}_{G,\mathrm{no-branch}}(t, t_{\max}) = \exp\left\{ - \int_t^{t_{\max}} \frac{dt'}{t'} I(t') \right\} \quad \rightarrow \quad \text{ordering parameter t}$$

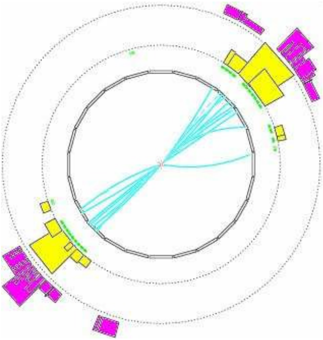
a simple shower algorithm

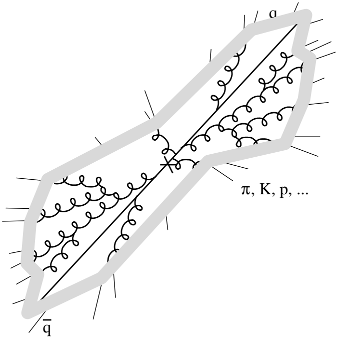
- determine scale of next emission by solving $\mathcal{P}_{G,\mathrm{no-branch}}(t, t_{\max}) = \#$ for t
  $\rightsquigarrow$ Sudakov Veto Algorithm
- select energy fraction z according to $P_{GG}(z)$
- construct kinematics of emitted particle
- reset $t_{\max} = t$ and start afresh

Parton Shower: QCD final-state cascade

The full QCD picture

- P_{qq}, P_{gq}, P_{qg}, P_{gg}, $\alpha_S(z,t)$, choice of evolution variable
- shower has to stop at some infra-red cut-off $t_o \sim \mathcal{O}(1\mathrm{GeV}^2)$
  - $\leadsto$ below t_o perturbative approach no-longer applies
  - $\leadsto$ Monte Carlo generators invoke hadronisation model





Parton Shower: The QCD running coupling

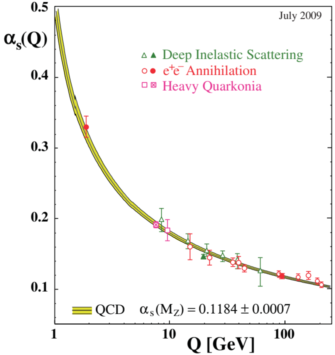
as other couplings/parameters α_s is scale dependent [momentum scale μ^2]

$\rightsquigarrow$ at lowest order one finds for $\alpha_s(\mu^2)$ $(b_0 = (33 - 2n_f)/12\pi)$

$$\frac{d\alpha_s(\mu^2)}{d\ln\mu^2} = -b_0\alpha_s^2 \quad \rightsquigarrow \quad \alpha_s(\mu^2) = \frac{\alpha_s(\mu_0^2)}{1 + b_0\alpha_s(\mu_0^2)\ln\frac{\mu^2}{\mu_0^2}} = \frac{1}{b_0\ln\frac{\mu^2}{\Lambda^2_{\mathrm{QCD}}}}$$

result expressed in terms of

- reference scale μ_0^2, e.g. M_Z^2
- non-perturbative constant $\Lambda_{\mathrm{QCD}} \simeq 0.2$ GeV
  - fundamental scale of QCD
  - sets scale for hadron masses

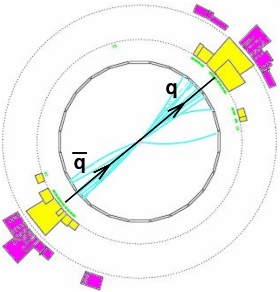


- perturbation theory valid for $\mu \gg \Lambda_{\mathrm{QCD}}$
- non-perturbative description $\mu \simeq \Lambda_{\mathrm{QCD}}$

The emergent picture: final-state jets

Jet definition (prel.): jets are collimated sprays of hadronic particles

- hard partons undergo soft and collinear showering
- hadrons closely correlated with the hard partons' directions



Counting jets

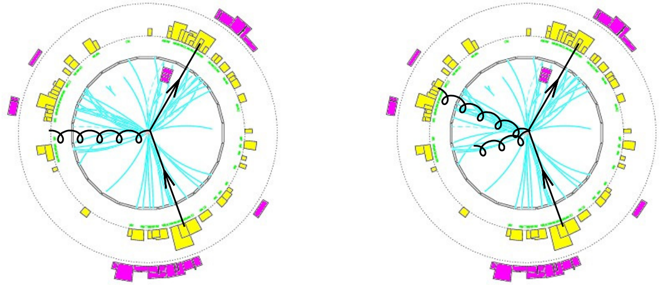
⇝ near perfect two-jet event

⇝ almost all energy contained in two cones

The emergent picture: final-state jets

Jet definition (prel.): jets are collimated sprays of particles

- hard partons undergo soft and collinear showering
- hadrons closely correlated with the hard partons' directions



Counting jets

⤳ hard emissions can induce more jets

⤳ jet counting not obvious, is this a three- or four-jet event?

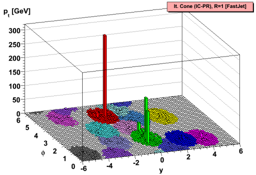
Jet algorithms

Jet definition

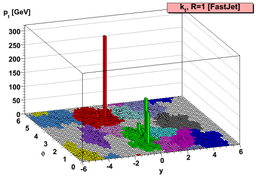
- group together particles into a common object, *i.e.* jets [jet algorithm]
- based on a distance measure that is algorithm specific
- combine momenta of jet constituents to yield jet momentum [recombination scheme]

two generic types of jet algorithms are commonly used:

- cone algorithms
  - widely used in the past at the Tevatron
  - jets have regular/circular shapes
  - some suffer from IR or collinear unsafety



- **sequential recombination algorithms**
  - widely used at LEP [Durham k_T algorithm]
  - jets can have irregular shape
  - default at the LHC experiments [anti-k_T algorithm]



Sequential recombination algorithms

A generic (final state) jet finding algorithm

1. compute a distance measure y_{ij} for each pair of final-state particles
2. determine the minimum of all y_{ij}'s
   - for smallest y_{ij}, **combine** particles ij, sum four-momenta, *i.e.* $p_{ij} = p_i + p_j$
3. go back to step one, until all particles are clustered into jets

in analyses one typically uses

- jets with inter-jet distances $y_{ij} > y_{\mathrm{cut}}$ [exclusive mode]
- jets with inter-jet distances $y_{ij} > y_{\mathrm{cut}}$ & $E > E_{\mathrm{cut}}$ [inclusive mode]

The k_T-algorithm distance measure [Catani et al. Phys. Lett. B **269** (1991), 432-438]

$$y_{ij} = \frac{2\min(E_i^2, E_j^2)(1 - \cos\theta_{ij})}{Q^2}$$

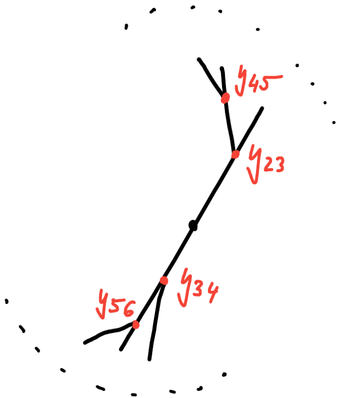
$\rightsquigarrow$ in the collinear limit: $y_{ij} \simeq \min(E_i^2, E_j^2)\theta_{ij}^2/Q^2$

$\rightsquigarrow$ relative transverse momentum, normalized to total energy

$\rightsquigarrow$ soft/collinear particles get clustered first

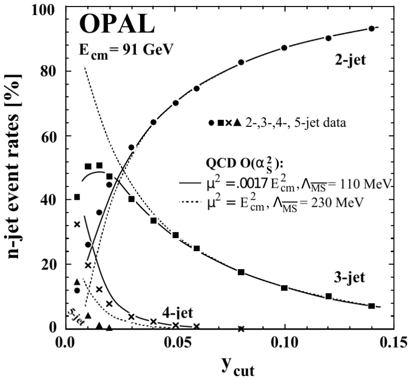
Jet algorithms at work: k_T jets at work

differential k_T scales

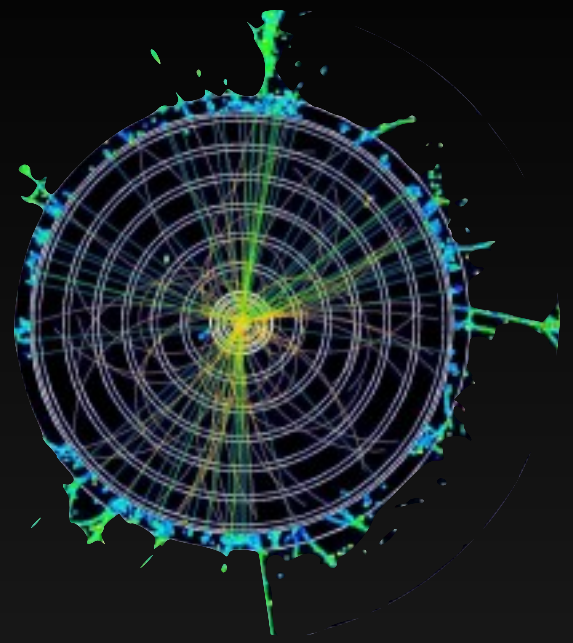


cluster sequence: … $< y_{56} < y_{45} < y_{34} < y_{23}$

k_T jet fractions @ LEP



e.g. 3-jet rate: $y_{23} \geq y_{\text{cut}}$



AdvCompPhys Lab 2024

Project: MC Simulations for Particle Physics

Enrico Bothmann – 12th June 2024

Quick Start

Quick Start Cheatsheet

1. get the project worksheet from Stud.IP
2. get utility code and reference data (see below)

```
git clone git@gitlab.gwdg.de:bothmann/advanced-computational-physics.git

mkdir my-project
cp -r advanced-computational-physics/{utils,sherpa.yoda} my-project
cd my-project
git init ; git add --all ; git commit -m "Add libs"

# inspect libs and create your own script
gedit utils/vector.py
gedit my_solution

``` example content of my_solution:
#!/usr/bin/env python

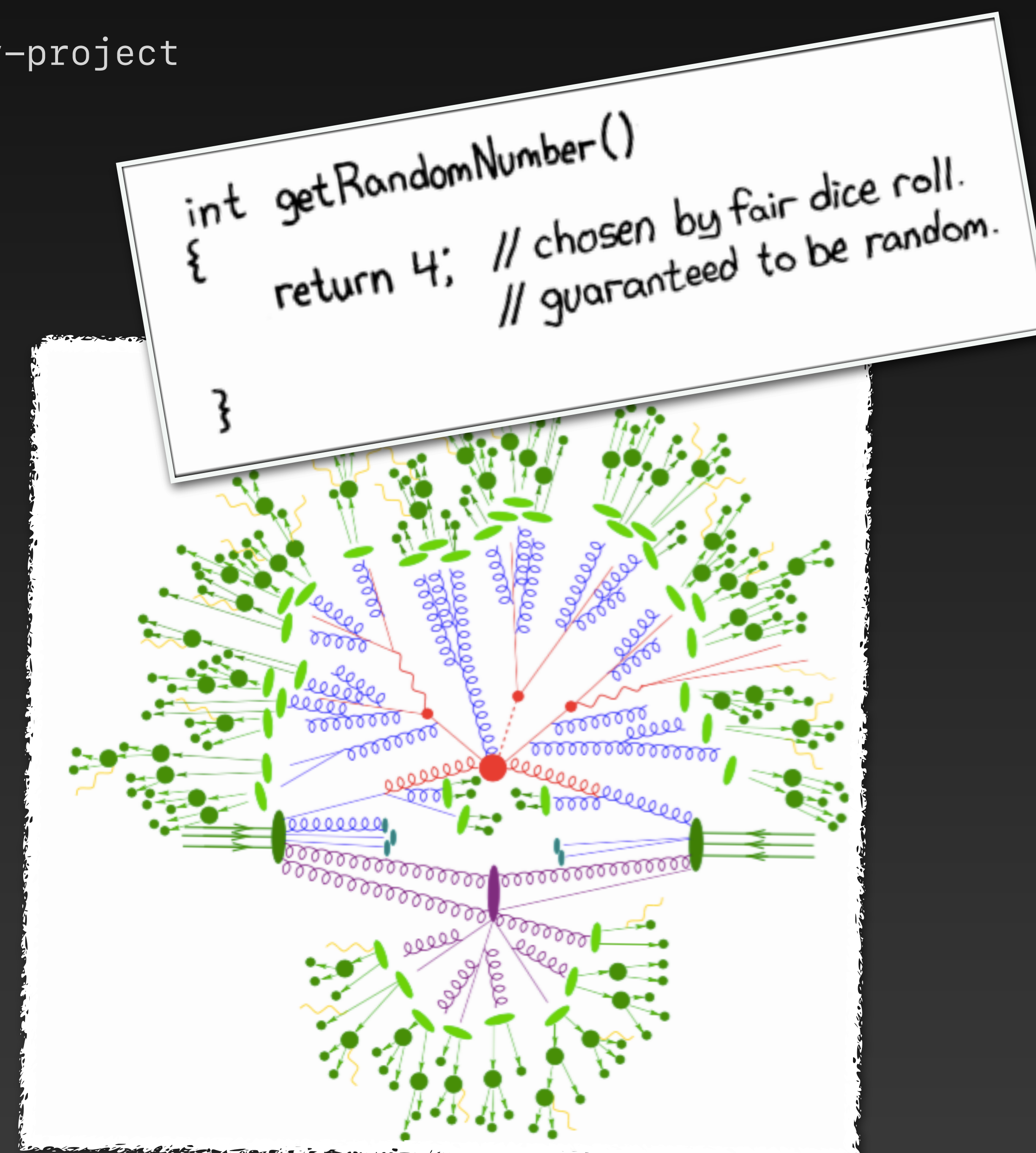
from utils.vector import Vec4

momentum = Vec4(128.9, 14.1, 3.3, 89.9)
print(momentum.invariant_mass())
```

make executable and run
chmod +x my_solution
./my_solution
→ 91.2...

get external libs, to make e.g. `import vegas` work
pip3 install vegas # or maybe pip3 install --user vegas
```

https://vegas.readthedocs.io/en/latest/tutorial.html



Best practices

- Mostly the same „Criteria for grading“ as in other projects
  40 % code, 10 % formal aspect, 50 % report, see project worksheet
- Use `python3`
  - language of provided library code
- Consider using `git`
  not just for sharing, but for organising your own work
- Readable code
  - simple code statements
  - add comments when useful
    unnecessary if code is truly self-explanatory

huh?!

```
if (ic<0 || jc<0 || kc<0)
  THROW(fatal_error,"Invalid PS tree");
double ws, mu2;
int flip(jc<ic), swap(jc<campl->NIn() && flip);
if (swap) std::swap<int>(ic,jc);
int type((ic<campl->NIn()?1:0)|(kc<campl->NIn()?2:0));
Splitting s=p_clus->KT2
   (campl->Leg(ic),campl->Leg(jc),campl->Leg(kc),
    lij->Flav(),campl->Kin(),type,1|(swap?2:0),ws,mu2);
s.p_s=lmap[lampl->IdLeg(lij->K())];
s.p_c=lmap[lij];
(*----m_ampls.end())->SetSplit(s);
if (!flip || swap) RecoCheck(*----m_ampls.end(),swap);
```

huh?!!?

~~wtf!!!~~
argh!!! 🤯

¿Questions?

- now?
- Stud.IP AdvCompPhys Lab forum
- enrico.bothmann@uni-goettingen.de
- Q & A sessions (Online meeting link will be announced on Stud.IP)

  Wed, 2:15pm-4pm (→ CIP Pool C.00.106)

  on-demand + online: Wed, 4pm–6pm, Fri 10:15am–12am

First session: Wed 19th June 2:15pm

➡ opportunity to get in contact among yourselves