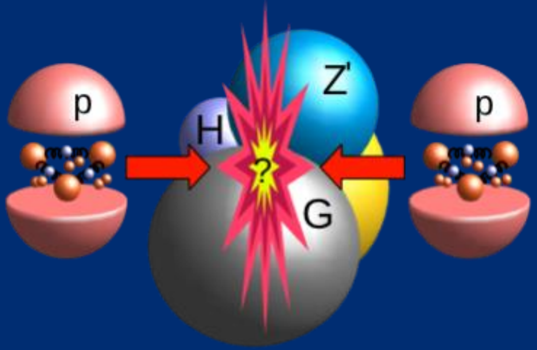


# JHUGen-MELA Tutorial

Mohit Srivastav

**2024 EFT Workshop**

**At the LPC**

**March 25th, 2024**

# Tutorial Outline

- MELA Background information
- Determining conversions between Warsaw and Higgs basis with Lexicon
- Technical Details (installation, where to find documentation, etc)
- Introduction to the C++ and Python code for the tutorial
- Running MELA
  - Gluon fusion to offshell Higgs to ZZ to 4l
  - EW to offshell Higgs to ZZ to 4l

# What *really* is MELA?

## Czech [ edit ]

### Pronunciation [ edit ]

- IPA(key): [ˈmɛla]
- Hyphenation: me‧la

### Etymology 1 [ edit ]

#### Noun [ edit ]

**mela** *f*

1. melee [synonyms ▲]
   Synonyms: tlačenice, zmatek, chaos
2. brawl [synonym ▲]
   Synonym: rvačka

## Italian [ edit ]

### Etymology [ edit ]

From Late Latin *mēla*, from *mēlum*, from Latin

### Pronunciation [ edit ]

- IPA(key): /ˈme.la/
- Audio ▶ 0:02
- Rhymes: -ela
- Hyphenation: mé‧la

### Noun [ edit ]

**mela** *f* (*plural* **mele**, *diminutive* **melétta** *or* **me**

1. apple (fruit) [synonym ▲]
   Synonym: pomo
2. (*colloquial*, *slang*) buttock, butt cheek

## Finnish [ edit ]

### Etymology [ edit ]

From Proto-Finnic **mëla*, from Proto-Finno-Permic
"oar, paddle"), Moksha миле (*milʹe*, "oar, paddle"), |

### Pronunciation [ edit ]

- IPA(key): /ˈmelɑ/, [ˈme̞lɑ̝]
- Rhymes: -elɑ
- Syllabification(key): me‧la

### Noun [ edit ]

**mela**

1. paddle (oar-like implement)

## Welsh [ edit ]

### Etymology [ edit ]

From *mêl* ("honey") + *-a*. Cognate with Cornish *mela*.

### Pronunciation [ edit ]

- (*North Wales*) IPA(key): /ˈmɛla/
- (*South Wales*) IPA(key): /ˈmeːla/, /ˈmɛla/
- Rhymes: -ɛla

### Verb [ edit ]

**mela** (*first-person singular present* **melaf**)

1. to gather nectar to make honey
2. to sweeten with honey

## Maltese [ edit ]

### Pronunciation [ edit ]

- IPA(key): /ˈmɛ.la/

### Etymology 1 [ edit ]

From Arabic بَلَى (*balā*, "why, certainly!, yes, of course!"),

#### Adverb [ edit ]

**mela**

1. certainly
2. thus; so; accordingly

#### Interjection [ edit ]

**mela**

1. so; okay; all right
   ***Mela**, ħa nibdew!*
   **So**, let's start!

## Serbo-Croatian [ edit ]

### Etymology [ edit ]

From German *Mehl*.

### Noun [ edit ]

**mela** *f* (*Cyrillic spelling* **мела**)

1. (*Kajkavian*) flour

### Related terms [ edit ]

- brašno

### Participle [ edit ]

**mela** (*Cyrillic spelling* **мела**)

1. *inflection of* **mesti**:

https://en.wiktionary.org/wiki/mela

## English [ edit ]

### Etymology [ edit ]

From Urdu میلہ (*mela*)/Hindi मेला (*melā*), from Sanskrit मेलक (*melaka*).

### Noun [ edit ]

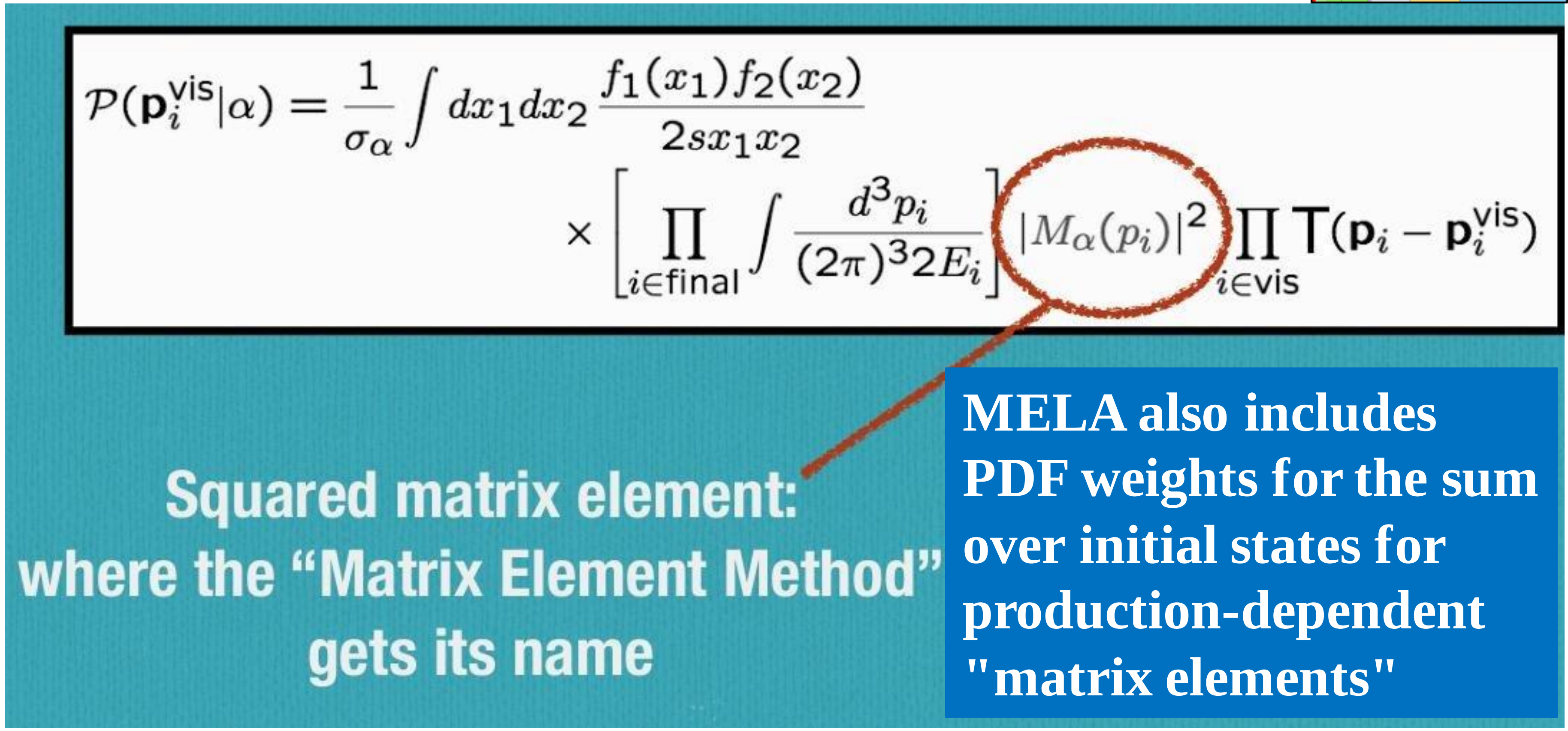
**mela** (*plural* **melas**)

1. A Hindu religious festival.
2. A South Asian fair. [from 19th c.] [quotations ▼]

# What *really* is MELA?

- Matrix
- Element

- Likelihood
- Approach



$$\mathcal{P}(\mathbf{p}_i^{\text{vis}}|\alpha) = \frac{1}{\sigma_\alpha}\int dx_1 dx_2 \frac{f_1(x_1)f_2(x_2)}{2sx_1x_2} \times \left[\prod_{i\in\text{final}}\int\frac{d^3p_i}{(2\pi)^3 2E_i}\right]|M_\alpha(p_i)|^2 \prod_{i\in\text{vis}} T(\mathbf{p}_i - \mathbf{p}_i^{\text{vis}})$$

Squared matrix element: where the "Matrix Element Method" gets its name

**MELA also includes PDF weights for the sum over initial states for production-dependent "matrix elements"**

["The Matrix Element Method for new physics discovery" - Jamie Gainer LPC "data challenge" Workshop, 2015]

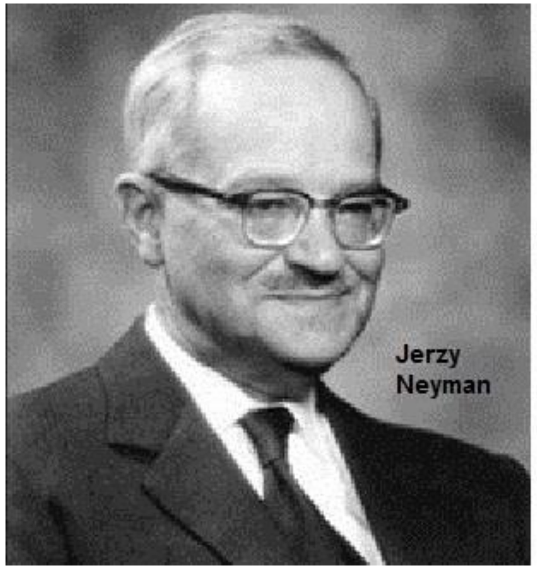
MELA: (Gao, Gritsan, Guo, Melnikov, Schulze, Tran, 2010), (Bolognesi, Gao, Gritsan, Melnikov, Schulze, Tran, Whitbeck, 2012)

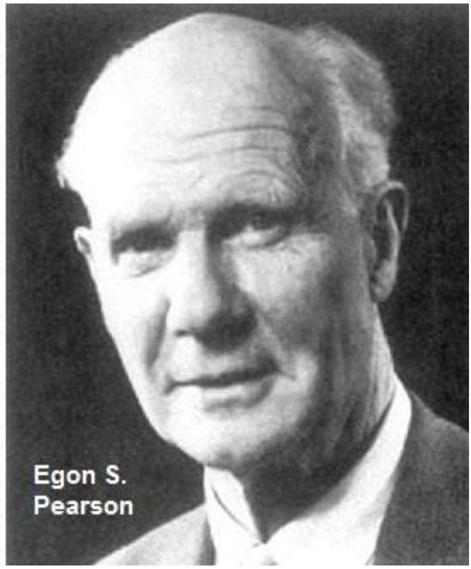
# What *really* is MELA?

- Matrix
- Element
- Likelihood
- Approach

- Utilizes the Neyman Pearson Lemma with regards to likelihood ratios
- Ratio test wiki link
- Matrix Element = likelihood based off physics knowledge!

MELA
arXiv 1001.3396
arXiv 1208.4018





# What can MELA do?

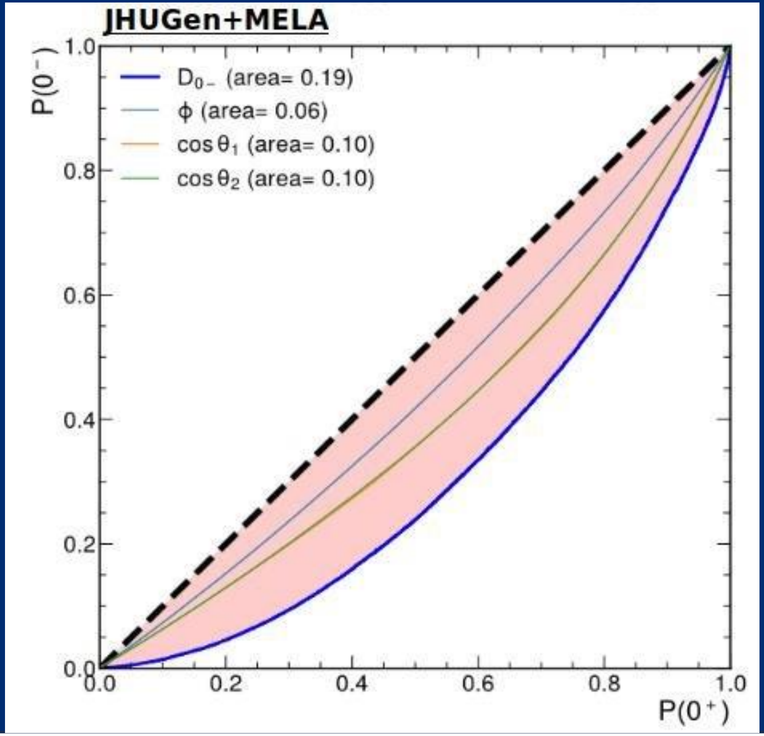
- MELA can be used to generate optimal discriminants or reweight samples from one hypothesis to another

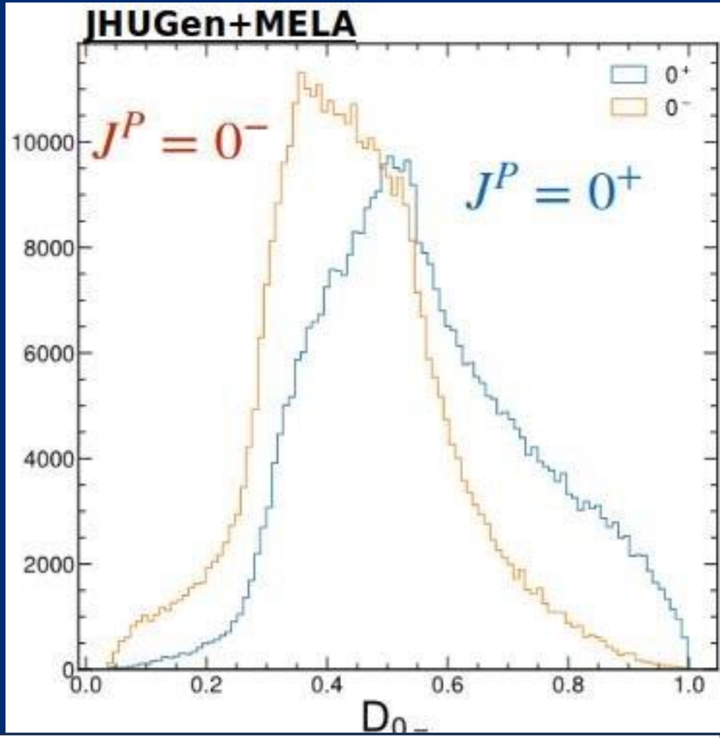
- Even though the same calculation does both, they're technically separate!
  - Reweighting is possible since the matrix element is the only difference between two processes with the same initial and final state

  - Optimal separation is possible through the Neyman-Pearson lemma!

# What can MELA do?

- In terms of separating power, it can encode all of the angles and physical quantities into a single value – the discriminant – which is better than any single one of those quantities



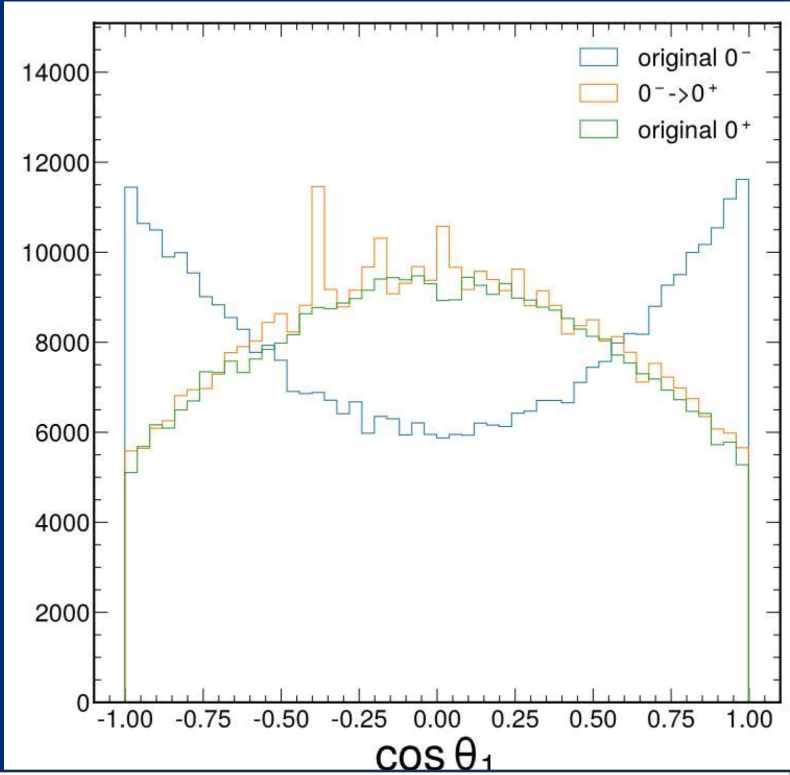


# What can MELA do?

- In terms of reweighting – if two processes are the same in all but one location (i.e., the decay coupling of the Higgs), then the matrix element can be used for reweighting samples!

- **Useful to see the effects of EFT terms to each other!**

- MELA can also reweight shape schemes!
  - Important for high-mass reweighting from POWHEG



## What is High-Mass Reweighting?

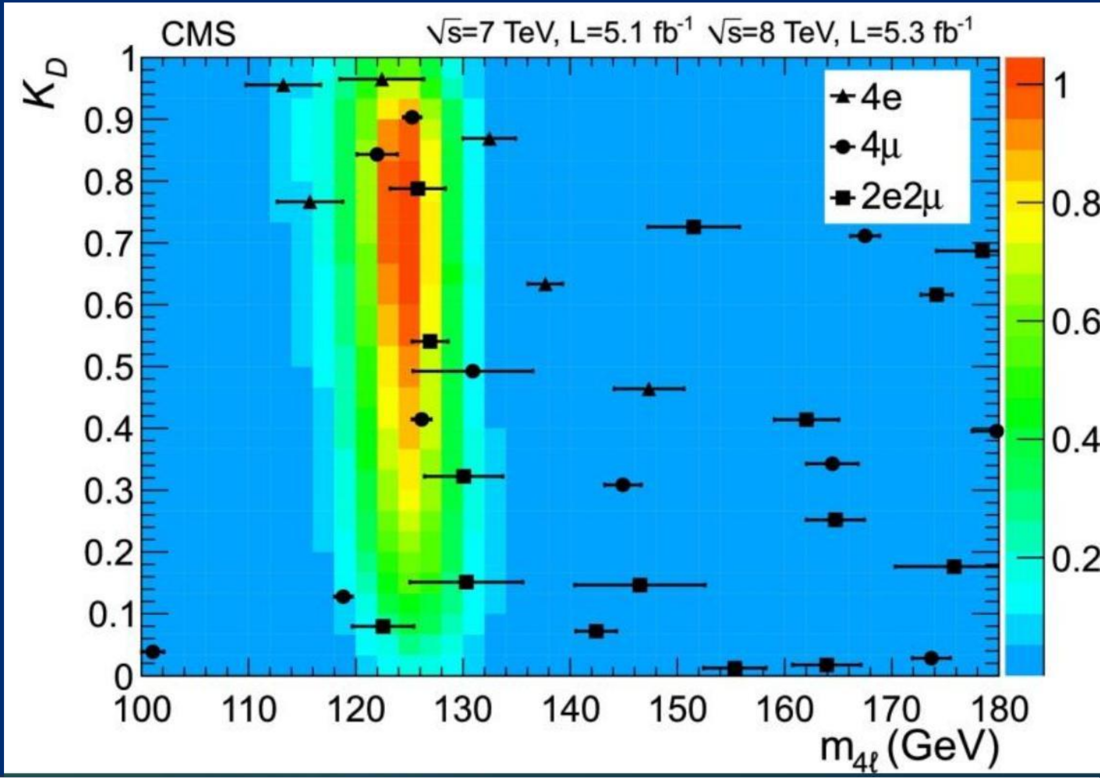
- "High Mass Reweighting" is done because POWHEG+JHUGen+PYTHIA is better tuned to simulate jets relative to MCFM+JHUGen+PYTHIA
  - Take a bunch of samples that are made in POWHEG, and reweight the shape of the mass
  - "CPS (Complex Pole Scheme) To BW (Breit-Wigner)"
  - Decay process is modelled by JHUGen

- At a high mass because it's intended for simulating offshell :)

- Not really a "Matrix Element" calculation, but MELA has the capability to do it through the propagator.
  - Makes an LO topology out of an NLO approximation

# MELA history

- Most notably, MELA was used for the Higgs discovery!
- MELA is in use for many analyses in CMS ranging from B-Physics to the offshell Higgs Boson
- It was recently introduced to an FCC framework as well



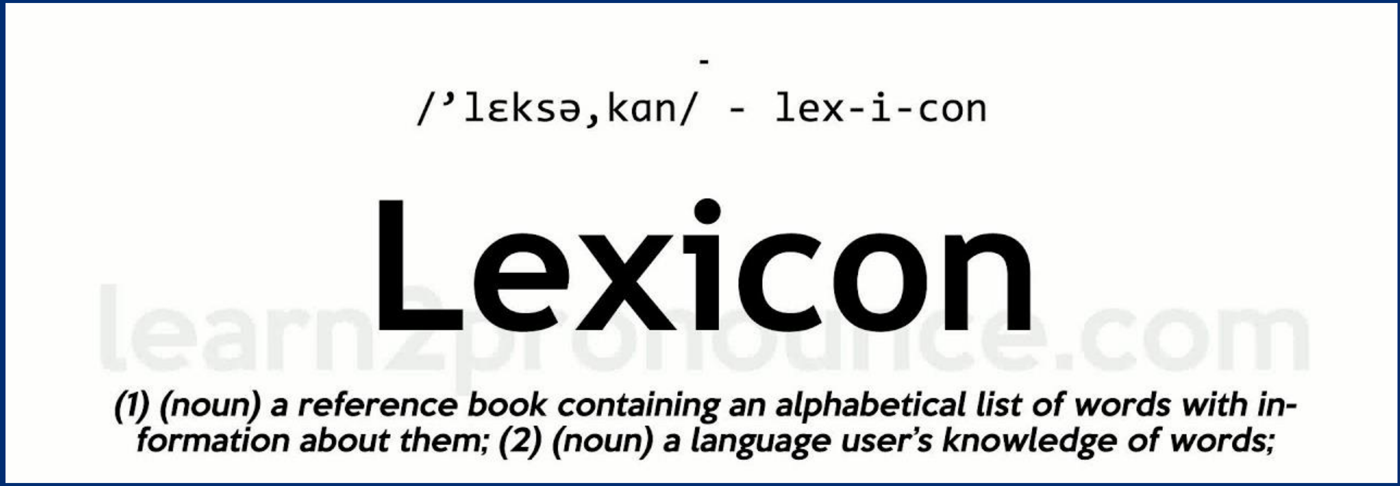
## MELA history

- The following people are JHUGen-MELA authors/developers

I. Anderson, S. Bolognesi, F. Caola, J. Davis, Y. Gao, A. V. Gritsan,
L. S. Mandacaru Guerra, Z. Guo, L. Kang, S. Kyriacou, C. B. Martin, T. Martini,
K. Melnikov, R. Pan, M. Panagiotou, R. Rontsch, J. Roskes, U. Sarica,
M. Schulze, M. V. Srivastav, N. V. Tran, A. Whitbeck, M. Xiao, Y. Zhou

## JHUGen Lexicon

- JHUGen Lexicon is a tool for converting operators from the JHUGen Higgs Basis to the Warsaw Basis
- Talk from last EFT workshop by [Jeffrey Davis]
- Similarly, see Section VII of the [JHUGen Manual]



/ˈlɛksə,kɑn/ - lex-i-con

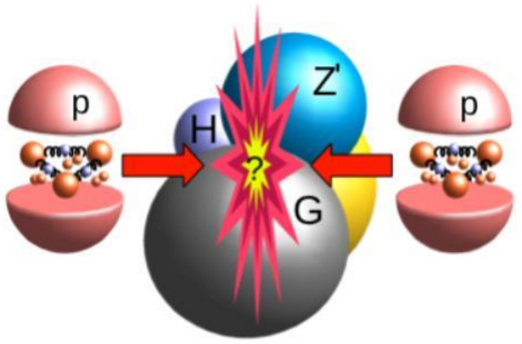
Lexicon

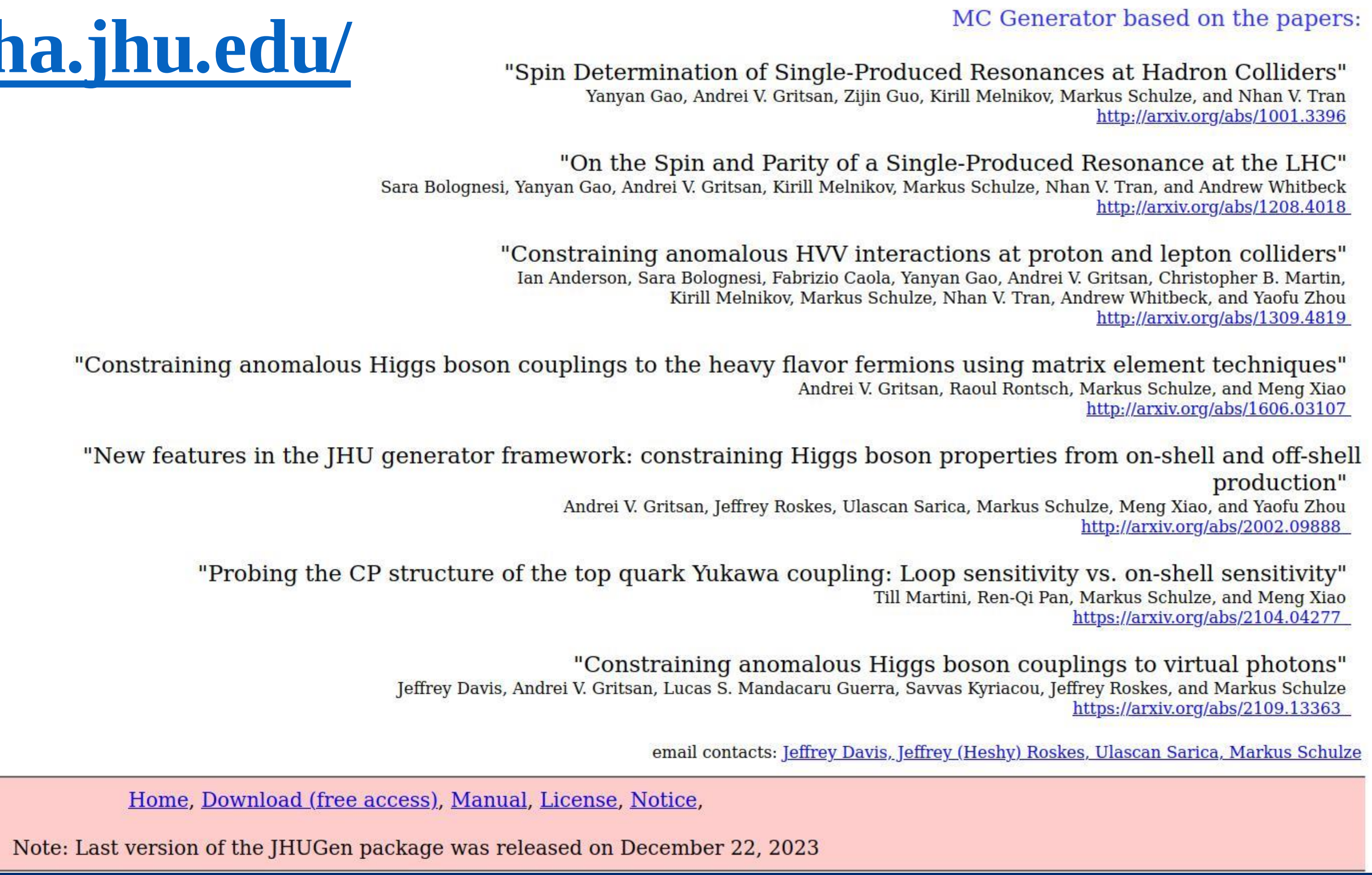
learn2pronounce.com

(1) (noun) a reference book containing an alphabetical list of words with information about them; (2) (noun) a language user's knowledge of words;

# Technical Details - BACKGROUND

https://spin.pha.jhu.edu/





MC Generator based on the papers:

"Spin Determination of Single-Produced Resonances at Hadron Colliders"
Yanyan Gao, Andrei V. Gritsan, Zijin Guo, Kirill Melnikov, Markus Schulze, and Nhan V. Tran
http://arxiv.org/abs/1001.3396

"On the Spin and Parity of a Single-Produced Resonance at the LHC"
Sara Bolognesi, Yanyan Gao, Andrei V. Gritsan, Kirill Melnikov, Markus Schulze, Nhan V. Tran, and Andrew Whitbeck
http://arxiv.org/abs/1208.4018

"Constraining anomalous HVV interactions at proton and lepton colliders"
Ian Anderson, Sara Bolognesi, Fabrizio Caola, Yanyan Gao, Andrei V. Gritsan, Christopher B. Martin, Kirill Melnikov, Markus Schulze, Nhan V. Tran, Andrew Whitbeck, and Yaofu Zhou
http://arxiv.org/abs/1309.4819

"Constraining anomalous Higgs boson couplings to the heavy flavor fermions using matrix element techniques"
Andrei V. Gritsan, Raoul Rontsch, Markus Schulze, and Meng Xiao
http://arxiv.org/abs/1606.03107

"New features in the JHU generator framework: constraining Higgs boson properties from on-shell and off-shell production"
Andrei V. Gritsan, Jeffrey Roskes, Ulascan Sarica, Markus Schulze, Meng Xiao, and Yaofu Zhou
http://arxiv.org/abs/2002.09888

"Probing the CP structure of the top quark Yukawa coupling: Loop sensitivity vs. on-shell sensitivity"
Till Martini, Ren-Qi Pan, Markus Schulze, and Meng Xiao
https://arxiv.org/abs/2104.04277

"Constraining anomalous Higgs boson couplings to virtual photons"
Jeffrey Davis, Andrei V. Gritsan, Lucas S. Mandacaru Guerra, Savvas Kyriacou, Jeffrey Roskes, and Markus Schulze
https://arxiv.org/abs/2109.13363

email contacts: Jeffrey Davis, Jeffrey (Heshy) Roskes, Ulascan Sarica, Markus Schulze

Home, Download (free access), Manual, License, Notice,

Note: Last version of the JHUGen package was released on December 22, 2023

# Technical Details - BACKGROUND

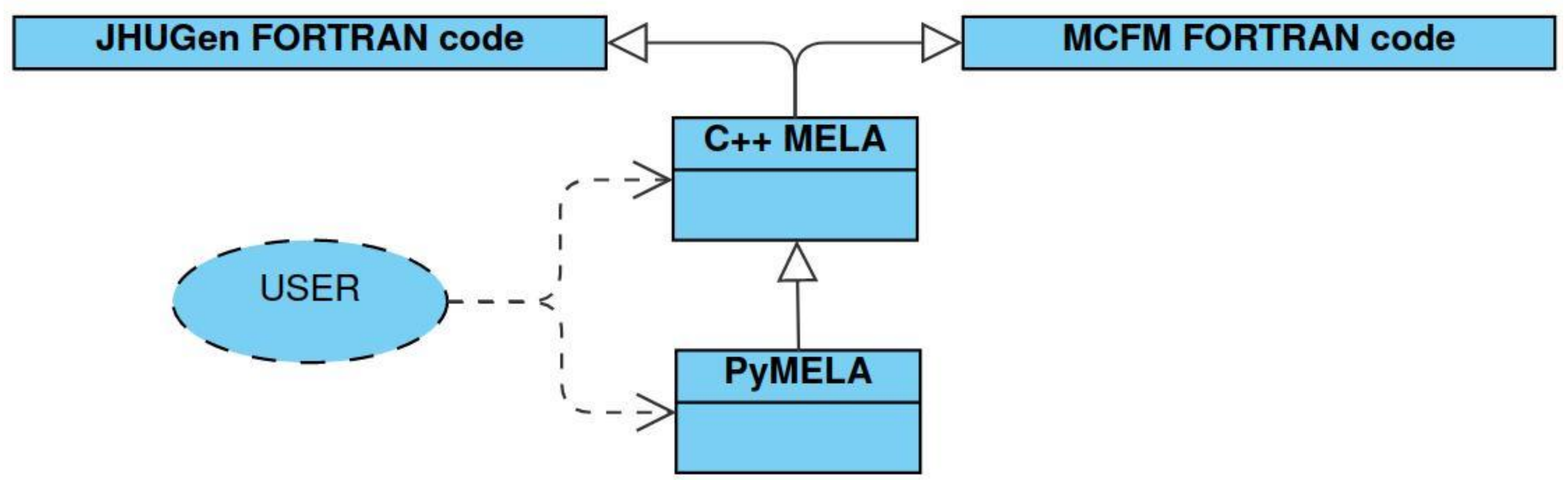
- There are *technically* 3 versions of JHUGen-MELA you can use
  - FORTRAN (yucky!) - subdivided into JHUGen and MCFM base calculations
  - C++
  - Python

- All 3 of these are congruent with one another. However, the only **supported** methods to use MELA are through either the C++ or the Python package interfaces.

- These are all interfaces to the base FORTRAN code

- We will cover the python and the C++ concurrently!

# Technical Details - BACKGROUND



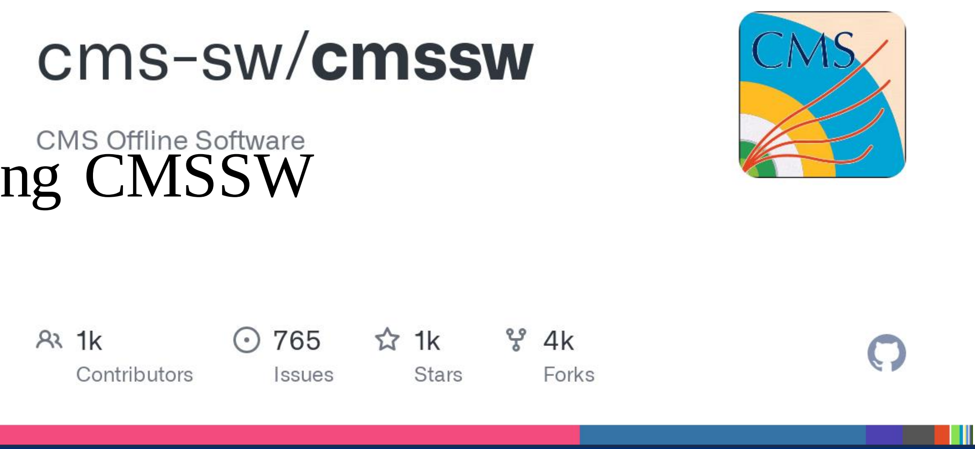
**You can either interface with the C++ or the Python, but it always comes down to the C++ using the FORTRAN**

# Technical Details - DEPENDENCIES

- There are a few dependencies for MELA

- For the FORTRAN, you need FORTRAN 90 (JHUGen) and 77 (MCFM)
- For the C++, you need CERN ROOT along with ROOFIT
- For the python, you need Python 3 and PyBind11 (this is what binds the Python to the C++ interface)

- All these packages (and many more!) are available using CMSSW
- [https://github.com/cms-sw/cmssw](https://github.com/cms-sw/cmssw)



cms-sw/**cmssw**

CMS Offline Software

1k Contributors · 765 Issues · 1k Stars · 4k Forks

# Technical Details - INSTALLING

- For the purposes of this tutorial, git clone the tutorial area inside CMSSW_14_0_0/src/
    - Can be installed by calling "cmsrel CMSSW_14_0_0"

- Now cd to CMSSW_14_0_0/src/
- **Do "cmsenv"**

# Technical Details - INSTALLING

- Can download *official* versions of JHUGen-MELA at https://spin.pha.jhu.edu/

- For the purposes of this tutorial, we have simplified your life and untar'd most things for your enjoyment here (https://github.com/MohitS704/EFT-Workshop-JHUGenMELA-tutorial)

- There are a few "official" user-input interfaces to MELA floating around
  - MELAAnalytics, MELAcalc, someone mentioned JuliaMELA once
  - These are all great! Use different config files (csv, json, etc.)
- **However, we will focus on interfacing with MELA directly without any intermediaries**

# Technical Details - INSTALLING

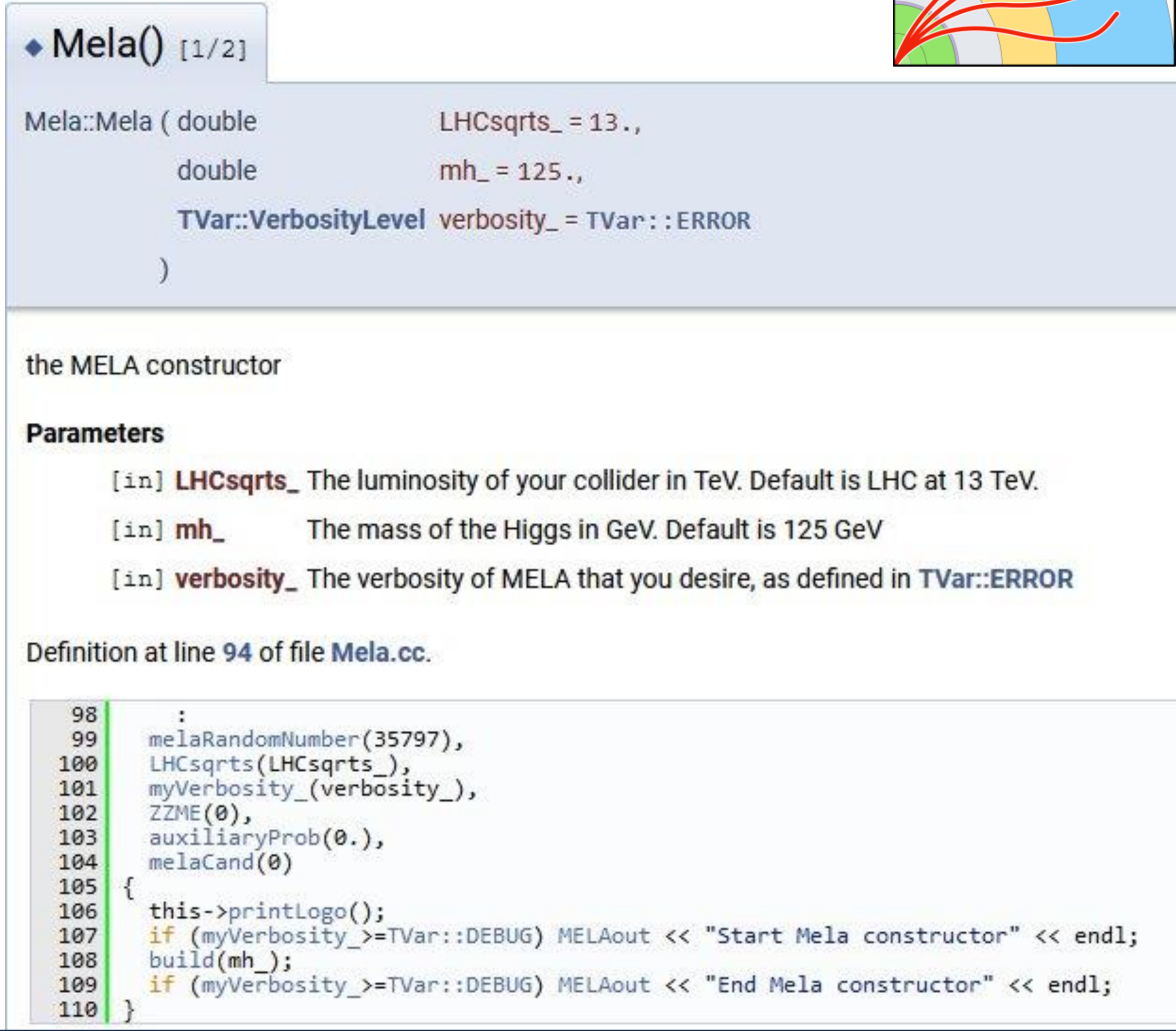
- You need to **compile MELA!**

- **Do the following:**
- **Cd JHUGenMELA (from CMSSW_14_0_0/src/EFT-Workshop-JHUGenMELA-tutorial)**
- **./setup.sh; eval $(./setup.sh env) inside JHUGenMELA/**

- Your MELA is now set up!

## Technical Details - DOCUMENTATION

- Documentation for C++ functions in MELA are currently here:
- https://spin.pha.jhu.edu/MELA/classMela.html
- https://spin.pha.jhu.edu/MELA/
- This is very much still a work in progress! However, it is helpful for various functions
- Currently written by yours truly :)



◆ Mela() [1/2]

```
Mela::Mela ( double                  LHCsqrts_ = 13.,
             double                  mh_ = 125.,
             TVar::VerbosityLevel    verbosity_ = TVar::ERROR
           )
```

the MELA constructor

**Parameters**

| | | |
|---|---|---|
| [in] | **LHCsqrts_** | The luminosity of your collider in TeV. Default is LHC at 13 TeV. |
| [in] | **mh_** | The mass of the Higgs in GeV. Default is 125 GeV |
| [in] | **verbosity_** | The verbosity of MELA that you desire, as defined in TVar::ERROR |

Definition at line 94 of file Mela.cc.

```
  :
  melaRandomNumber(35797),
  LHCsqrts(LHCsqrts_),
  myVerbosity_(verbosity_),
  ZZME(0),
  auxiliaryProb(0.),
  melaCand(0)
{
  this->printLogo();
  if (myVerbosity_>=TVar::DEBUG) MELAout << "Start Mela constructor" << endl;
  build(mh_);
  if (myVerbosity_>=TVar::DEBUG) MELAout << "End Mela constructor" << endl;
}
```

# C++ Portion - Initialization

- Go to the folder named CPP
- The file to edit is called tutorial_link.cpp, the output file will be probs_output.txt
- Notice the include statements! TVar.hh and TCouplingsBase.hh have vital content for MELA in C++
- Everything else should be accessible through Mela.h!
- Run the makefile to link the code to the MELA package

# C++ Portion – Code Introduction

- The Mela object is initialized with the following arguments:
  - Luminosity (set to 13 by default here)
  - Onshell Mass of the Higgs (set to 125 by default here)
  - Verbosity (initialized to TVar::SILENT, but can be changed)

```
Mela m = Mela(13, 125, TVar::SILENT);
```

- Every time you call computeP from the event loop it wipes the information from the MELA object, so it must be reset!
  - **ComputeP edits the variable passed *in place*!!**

- Set any couplings and input events in the for loop as a consequence

# Python Portion – Initialization

- Go to the folder named PY

- The file to edit is called tutorial_link.py, the output file will be probs_output.txt

- Mela is added to $PYTHONPATH via the setup script, so nothing should need to be done
  - If you are curious about how the python binding is created, take a look at JHUGenMELA/MELA/python/mela_binding.cpp

- The Mela package should have everything you need (all the enumerations from Tvar and TCouplingsbase are included)

# Python Portion – Code Introduction

- The Mela object is initialized with the following arguments:
  - Luminosity (set to 13 by default here)
  - Onshell Mass of the Higgs (set to 125 by default here)
  - Verbosity (initialized to TVar::SILENT, but can be changed)

```
m = Mela.Mela(13, 125, Mela.VerbosityLevel.SILENT)
```

- Every time you call computeP from the event loop it wipes the information from the MELA object, so it must be reset!
  - **ComputeP *returns a value* in Python**

- Set any couplings and input events in the for loop as a consequence

# START YOUR ENGINES

- Let's Run MELA!
- We'll start off by setting up a GGZZ offshell process to reweight
- left->right:

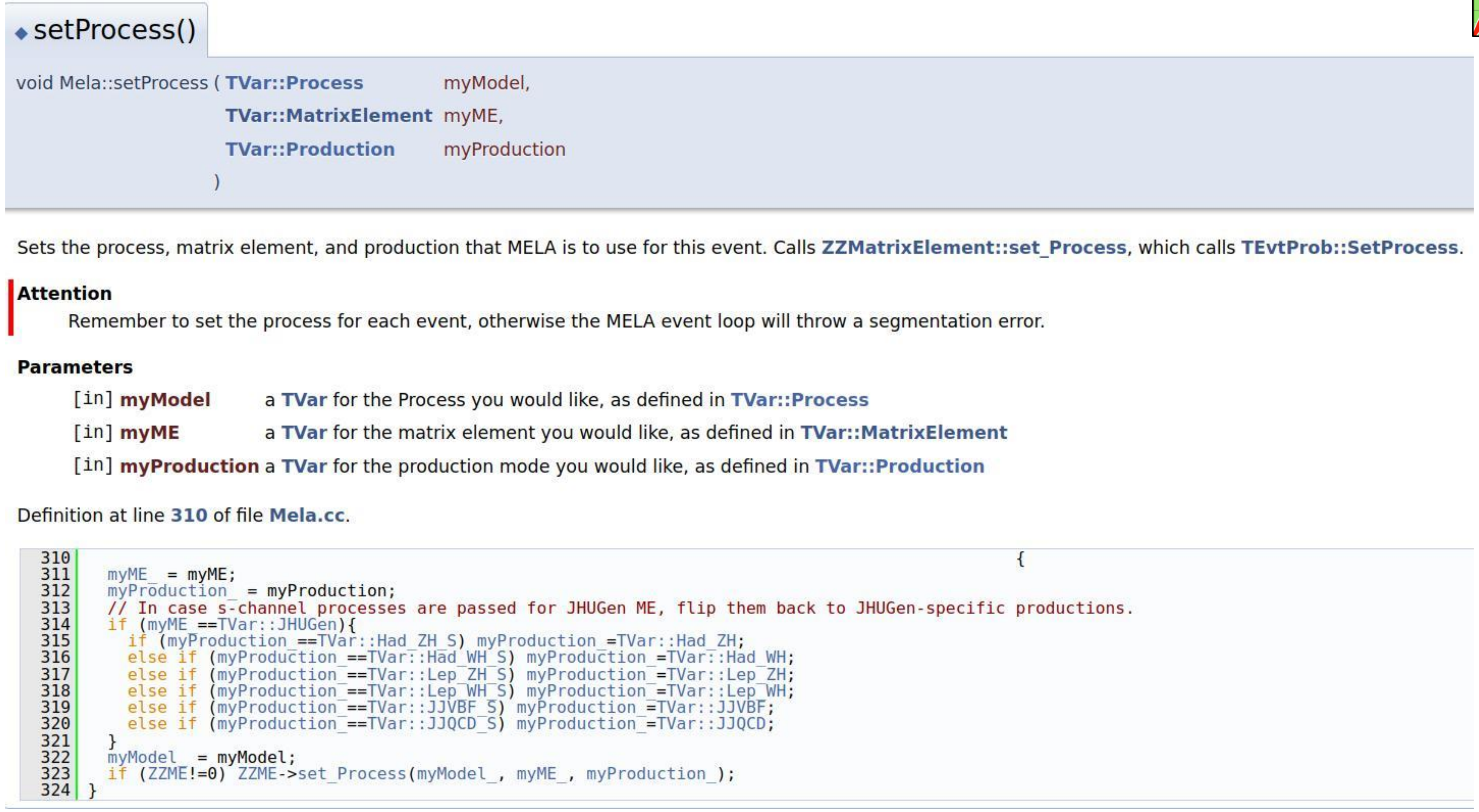
Ian Anderson , Ulascan Sarica, Andrei Gritsan

Chris Martin, Roberto Covarelli

# SETTING UP THE MELA MODE

- MELA needs to know the process you're running, the production mode for that process, as well as the "MatrixElement" you are using
- Processes in MELA
  - Things like "HSMHiggs" for MCFM, and things like "SelfDefine_Spin{NUMBER}" for JHUGen
  - TVar::Process
- Production modes
  - These are things like GGZZ, JJQCD, or JJVBF
  - TVar::Production
- Matrix elements
  - JHUGen when doing a JHUGen calculation, or MCFM when doing an MCFM calculation
  - Either TVar::JHUGen or TVar::MCFM in TVar::MatrixElement

# SETTING UP THE MELA MODE



◆ setProcess()

```
void Mela::setProcess ( TVar::Process         myModel,
                        TVar::MatrixElement   myME,
                        TVar::Production      myProduction
                      )
```

Sets the process, matrix element, and production that MELA is to use for this event. Calls **ZZMatrixElement::set_Process**, which calls **TEvtProb::SetProcess**.

**Attention**
Remember to set the process for each event, otherwise the MELA event loop will throw a segmentation error.

**Parameters**

| | | |
|---|---|---|
| [in] | **myModel** | a **TVar** for the Process you would like, as defined in **TVar::Process** |
| [in] | **myME** | a **TVar** for the matrix element you would like, as defined in **TVar::MatrixElement** |
| [in] | **myProduction** | a **TVar** for the production mode you would like, as defined in **TVar::Production** |

Definition at line **310** of file **Mela.cc**.

```
310                                                                        {
311   myME_ = myME;
312   myProduction_ = myProduction;
313   // In case s-channel processes are passed for JHUGen ME, flip them back to JHUGen-specific productions.
314   if (myME_==TVar::JHUGen){
315     if (myProduction_==TVar::Had_ZH_S) myProduction_=TVar::Had_ZH;
316     else if (myProduction_==TVar::Had_WH_S) myProduction_=TVar::Had_WH;
317     else if (myProduction_==TVar::Lep_ZH_S) myProduction_=TVar::Lep_ZH;
318     else if (myProduction_==TVar::Lep_WH_S) myProduction_=TVar::Lep_WH;
319     else if (myProduction_==TVar::JJVBF_S) myProduction_=TVar::JJVBF;
320     else if (myProduction_==TVar::JJQCD_S) myProduction_=TVar::JJQCD;
321   }
322   myModel_ = myModel;
323   if (ZZME!=0) ZZME->set_Process(myModel_, myME_, myProduction_);
324 }
```

# SETTING UP THE MELA MODE

**C++**

- Set the process, then the MatrixElement then the production in m.setProcess.
- Remember to use the TVars!

```
m.setProcess(
   TVar::HSMHiggs,  TVar::MCFM,TVar::ZZGG);
```

**Python**

- Set the process, then the MatrixElement then the production in m.setProcess.
- TVars are stored in enumerated items!
  - See mela_binding.cpp line 1122 if you are curious

```
m.setProcess(Mela.Process.HSMHiggs,
Mela.MatrixElement.MCFM,
Mela.Production.ZZGG)
```

# SETTING COUPLINGS

- Couplings are what define physical processes in MELA! No set couplings = no values!
- These correspond to interactions between particles
  - Couplings between the Z and the Higgs, or the Higgs and gluons, etc.

- Couplings are stored in arrays – many of them 3D
  - 2 possible Higgs candidates
  - Coupling array sizes vary (see TCouplingsBase.hh)
  - Real and imaginary portion of the coupling

- Coupling arrays are declared in Mela.h, and are documented here:

https://spin.pha.jhu.edu/MELA/classMela.html

# SETTING COUPLINGS

**C++**

- Below is the procedure for setting ghz1
- The index corresponds to the value of gHIGGS_VV_1

```
•  m.selfDHzzcoupl[0][gHIGGS_VV_1][0] = 2;
•  m.selfDHzzcoupl[0][gHIGGS_VV_1][1] = 0;
```

**Python**

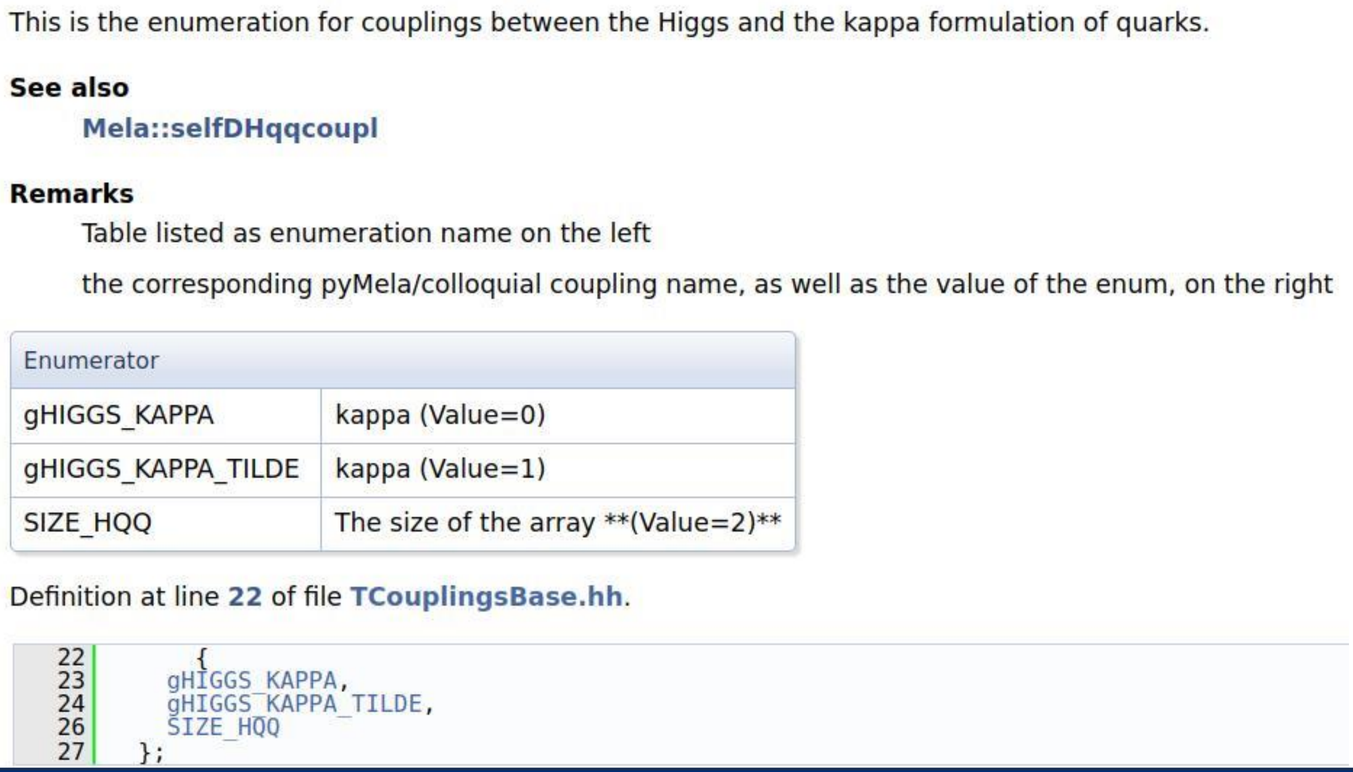
- In Python the couplings are stored in identical arrays
- However, there are *also* direct links to named couplings!
    - Arrays need to be called as functions to maintain references

```
m.selfDHzzcoupl()[
   0,Mela.CouplingIndex_HVV.gHiggs_VV_1
] = [2,0]
m.ghz1 = [2,0] #equivalent!
```

# SETTING COUPLINGS

- Q: How do I find what the coupling arrays correspond to??
- A: Read the documentation! When that fails (WIP, remember) - the source code!
- Documentation (for one of the arrays)

- Names in TCouplingsBase are a bit obfuscated – when in doubt - ask!



This is the enumeration for couplings between the Higgs and the kappa formulation of quarks.

**See also**

Mela::selfDHqqcoupl

**Remarks**

Table listed as enumeration name on the left

the corresponding pyMela/colloquial coupling name, as well as the value of the enum, on the right

| Enumerator | |
|---|---|
| gHIGGS_KAPPA | kappa (Value=0) |
| gHIGGS_KAPPA_TILDE | kappa (Value=1) |
| SIZE_HQQ | The size of the array **(Value=2)** |

Definition at line **22** of file **TCouplingsBase.hh**.

```
     {
   gHIGGS_KAPPA,
   gHIGGS_KAPPA_TILDE,
   SIZE_HQQ
 };
```

## SETTING COUPLINGS

- Q: How do I find what the coupling names in Python correspond to??
- A: Read the documentation! When that fails (WIP, remember) - the source code!
- mela_binding.cpp L 410

```
MAKE_COUPLING_REAL_IMAGINARY_SPIN_ZERO(selfDHggcoupl, ghg2, gHIGGS_GG_2, 0)
MAKE_COUPLING_REAL_IMAGINARY_SPIN_ZERO(selfDHggcoupl, gh2g2, gHIGGS_GG_2, 1)
```

- Array that it comes from, the name, the TCouplingsBase index, and the Higgs index
  - Higgs index is 0 or 1 depending on if you're using the first or second Higgs (second Higgs only applicable in the MCFM matrix element)

# SETTING COUPLINGS

- **Q: The Python is slower, why use it??**
- **A: Metaprogramming!**
- You sacrifice a bit of speed for the ability to more easily configure things
- This is the age-old fight between C++ and Python!

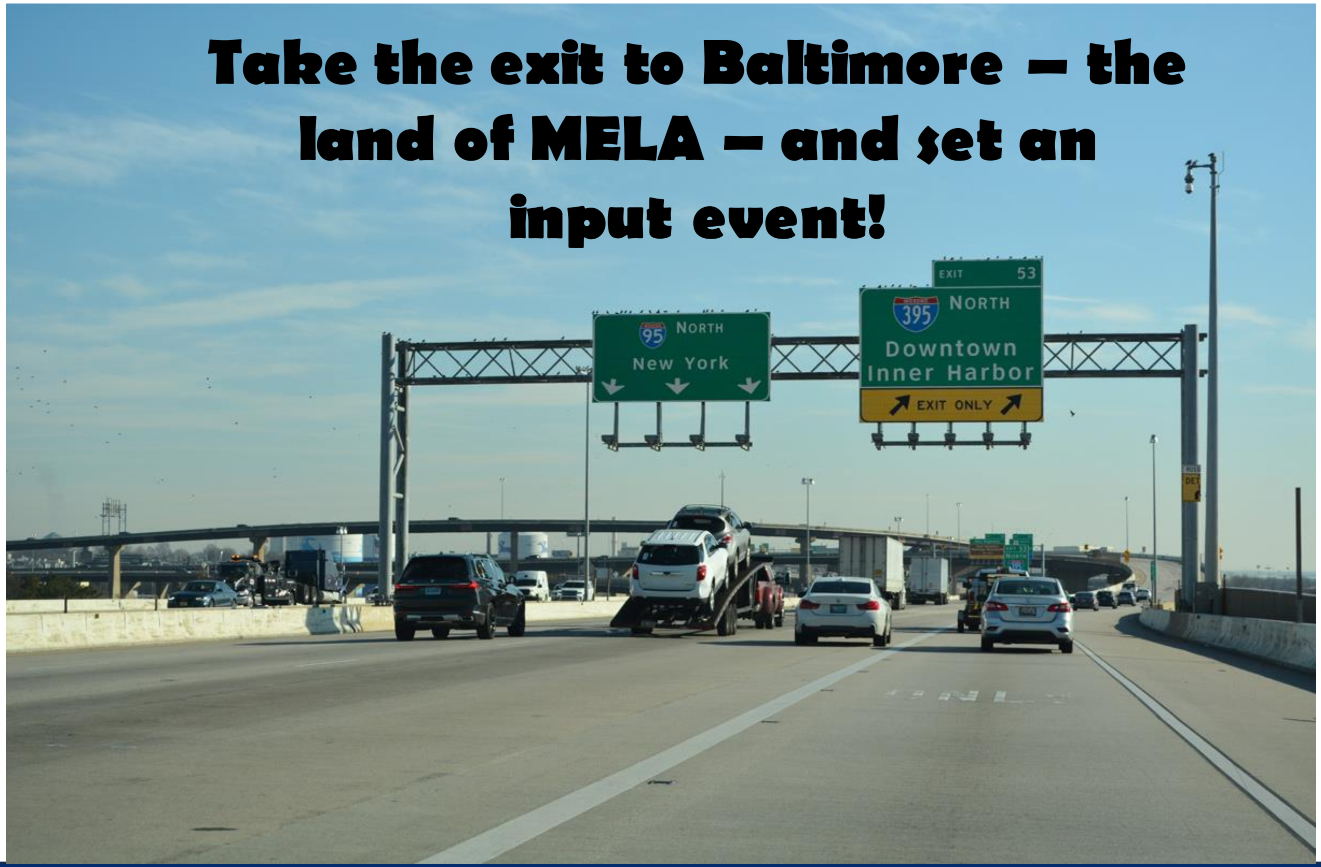
```
couplings = [
    {
        "COUPLING1":[1,0],
        "COUPLING2":[0,1]
    },
    {
        "COUPLING1":[0,1],
        "COUPLING2":[1,0]
    }
]
for hypo_dict in couplings:
    for coup, coup_val in hypo_dict.items():
        setattr(m, coup, coup_val)
```

# SETTING COUPLINGS – FOR YOU

- Within the for loop, change the couplings to what you want!
- For now, we are going to generate matrix element values for a standard model ggH offshell dataset.
- Try, in the for loop, setting the standard model HZZ and HGG coupling to 1 (*hint: this corresponds to "gHIGGS_VV_1" and "gHIGGS_GG_2"!)*
- Make the file and run the executable for C++ or run the Python file in Python. What gets printed?

# HUH!???

- Of course you'll get a bunch of zeros back!! You haven't set any events!


Take the exit to Baltimore – the land of MELA – and set an input event!
95 NORTH
New York
EXIT 53
395 NORTH
Downtown
Inner Harbor
EXIT ONLY

- **Q: Well, how do I do that?**
- **A: Go to the next slide.**

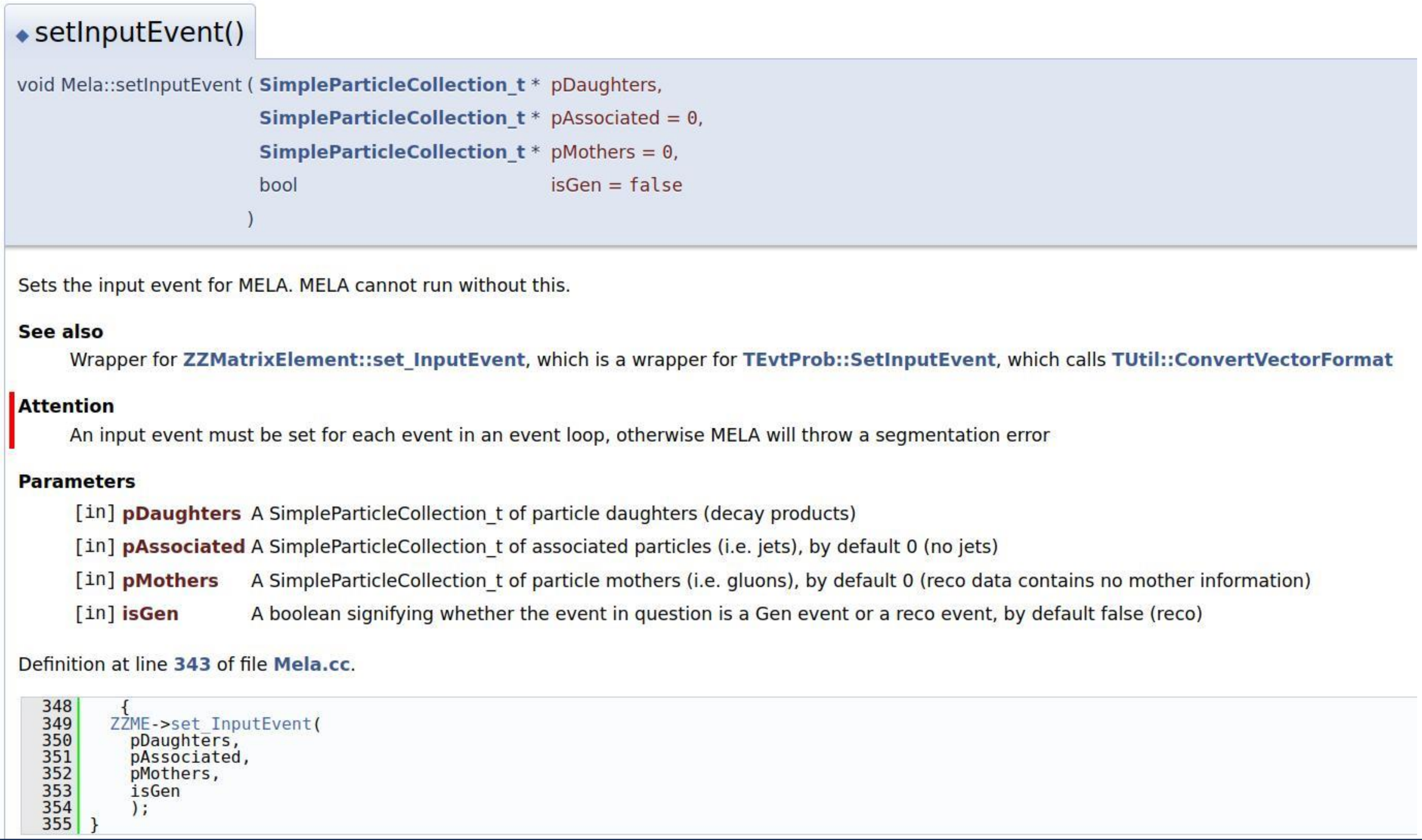
# SETTING INPUT EVENTS

- Input events are set using [Mela::setInputEvent]
- They take in a SimpleParticleCollection_t (a fancy vector made of SimpleParticle_t)
- A SimpleParticle_t is a pair consistent of a PDG ID and a Lorentz Vector
- From TVar.hh:

```
// typedefs for use in simple_event_record
typedef std::pair<int, TLorentzVector> SimpleParticle_t;
typedef std::vector<SimpleParticle_t> SimpleParticleCollection_t;
```

- An input event **MUST** be set for any calculation to be successful!

# SETTING INPUT EVENTS



◆ setInputEvent()

```
void Mela::setInputEvent ( SimpleParticleCollection_t * pDaughters,
                           SimpleParticleCollection_t * pAssociated = 0,
                           SimpleParticleCollection_t * pMothers = 0,
                           bool                         isGen = false
                         )
```

Sets the input event for MELA. MELA cannot run without this.

**See also**

Wrapper for **ZZMatrixElement::set_InputEvent**, which is a wrapper for **TEvtProb::SetInputEvent**, which calls **TUtil::ConvertVectorFormat**

**Attention**

An input event must be set for each event in an event loop, otherwise MELA will throw a segmentation error

**Parameters**

| | | |
|---|---|---|
| [in] | **pDaughters** | A SimpleParticleCollection_t of particle daughters (decay products) |
| [in] | **pAssociated** | A SimpleParticleCollection_t of associated particles (i.e. jets), by default 0 (no jets) |
| [in] | **pMothers** | A SimpleParticleCollection_t of particle mothers (i.e. gluons), by default 0 (reco data contains no mother information) |
| [in] | **isGen** | A boolean signifying whether the event in question is a Gen event or a reco event, by default false (reco) |

Definition at line **343** of file **Mela.cc**.

```
  {
  ZZME->set_InputEvent(
    pDaughters,
    pAssociated,
    pMothers,
    isGen
    );
}
```

pDaughters = decay particles (4l, 2l2nu, etc.)

pAssociated = jets, Z decay from ZH production, etc.

pMothers = gluons, quarks

# SETTING INPUT EVENTS – STEP 1

**C++**

- Create a SimpleParticle_t object
- The constructor is just that of a pair of an integer (id) and a ROOT TLorentzVector

```
TLorentzVector g1 = TLorentzVector(0, 0, 500,
500);
SimpleParticle_t melaParticle =
SimpleParticle_t(21, g1);
```

**Python**

- Create a SimpleParticle_t object
- The constructor in Python is different! It takes in the id, and either
  - (px, py, pz, E, False)
  - (pt, eta, phi, m, True)
- See the constructors in mela_binding.cpp line 311

```
melaParticle = Mela.SimpleParticle_t(
    21, 0, 0, 500, 500, False)
```

# SETTING INPUT EVENTS – STEP 2

**C++**

- Push back SimpleParticle_t objects into a SimpleParticleCollection_t

```
SimpleParticleCollection_t* mothers = new
SimpleParticleCollection_t();
mothers->push_back(melaParticle );
```

**Python**

- 2 ways to do this
  - Add particles one by one
  - Initialize a list of particles
  - Add a column of particles!
    - Same as SimpleParticle_t constructor, except now all the inputs (except the last) are lists!

```
mothers = Mela.SimpleParticleCollection_t()
mothers.add_particle(melaParticle)

mothers = Mela.SimpleParticleCollection_t (idList
pxList, pyList, pzList, EList, False)
```

## SETTING INPUT EVENTS STEPS 1 & 2

- In the C++ - interface with the ROOT file and figure out where the mothers, daughters, and associated particles are (hint – they're named accordingly!)

- In the Python – use the uproot dictionaries to find out where the mothers, daughters, and associated particles are

- Assign your particles!!

# SETTING INPUT EVENTS – STEP 3

**C++**

- Set the input event using m.setInputEvent!
- Notice the 0!
    - There are no associated particles for GGZZ

```
m.setInputEvent(daughters, 0, mothers, true);
```

**Python**

- Set the input event using m.setInputEvent!
- Notice the empty SimpleparticleCollection!
    - Instead of a 0!

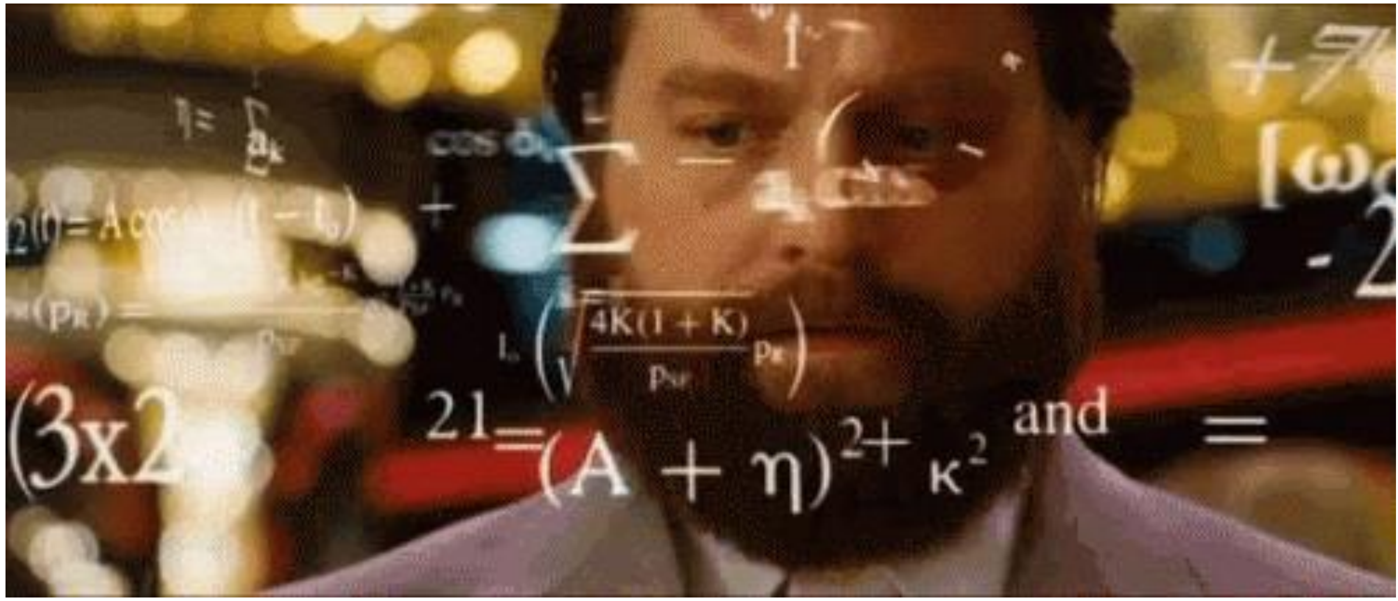
```
associated = Mela.SimpleParticleCollection_t()
m.setInputEvent(daughters, associated, mothers,
True)
```

# COMPUTING FUNCTIONS!

- There are 3 computing functions when dealing with coupling reweighting

- computeP
- computeProdDecP
- computeProdP

- Everything gets reset after you call one of these functions, **hence why is it in the for loop**



# COMPUTING FUNCTIONS computeP

- Mela::computeP is the catch-all for lots of processes decay-side
- Every JHUGen matrix element process uses this!
- For MCFM matrix elements (offshell!!) the following processes use this:
  - GG->ZZ
  - QQbar->ZZ
  - Production Independent -> ZZ
  - Z+jets

- **This is what you are using in this example!**

# COMPUTING FUNCTIONS computeProdP

- Mela::computeProdP does production side calculations

- Only able to do through the JHUGen Matrix Element

# COMPUTING FUNCTIONS computeProdDecP

- Mela::computeProdDecP combines production and decay probabilities into one calculation

- MCFM really shines here – all processes but GGZZ should go through here for the MCFM matrix element

- **This is what you want to use in the next example**

# COMPUTING VALUES

**C++**

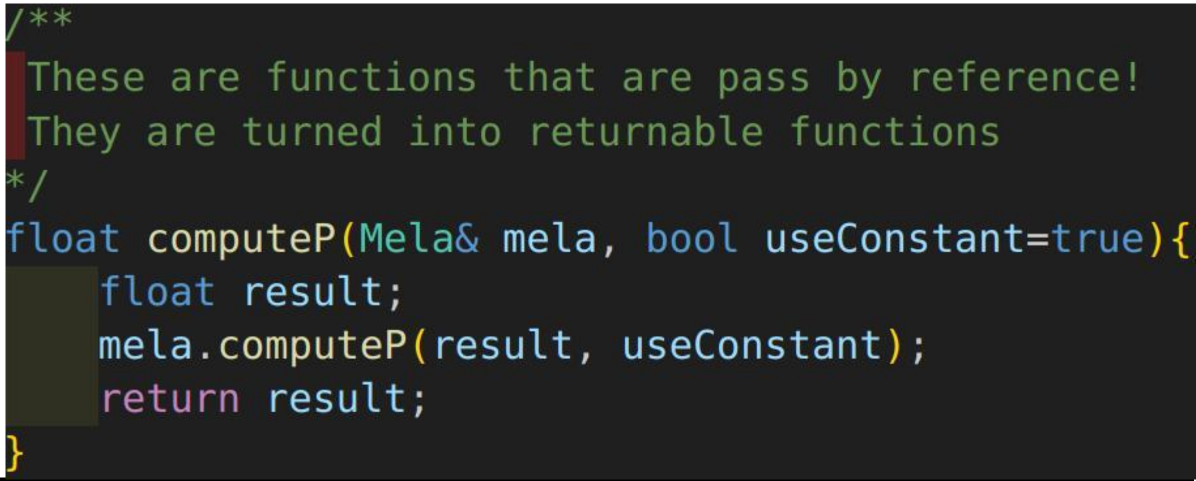
- Probability is ***passed by reference!***

```
float prob = 0;
m.computeP(prob,  false);
```

**Python**

- Probability is a ***value returned!***

Line 13 in mela_binding.cpp



```
/**
 These are functions that are pass by reference!
 They are turned into returnable functions
*/
float computeP(Mela& mela, bool useConstant=true){
    float result;
    mela.computeP(result, useConstant);
    return result;
}
```

```
prob = m.computeP(False)
```

# SUMMARY FOR GGZZ PORTION

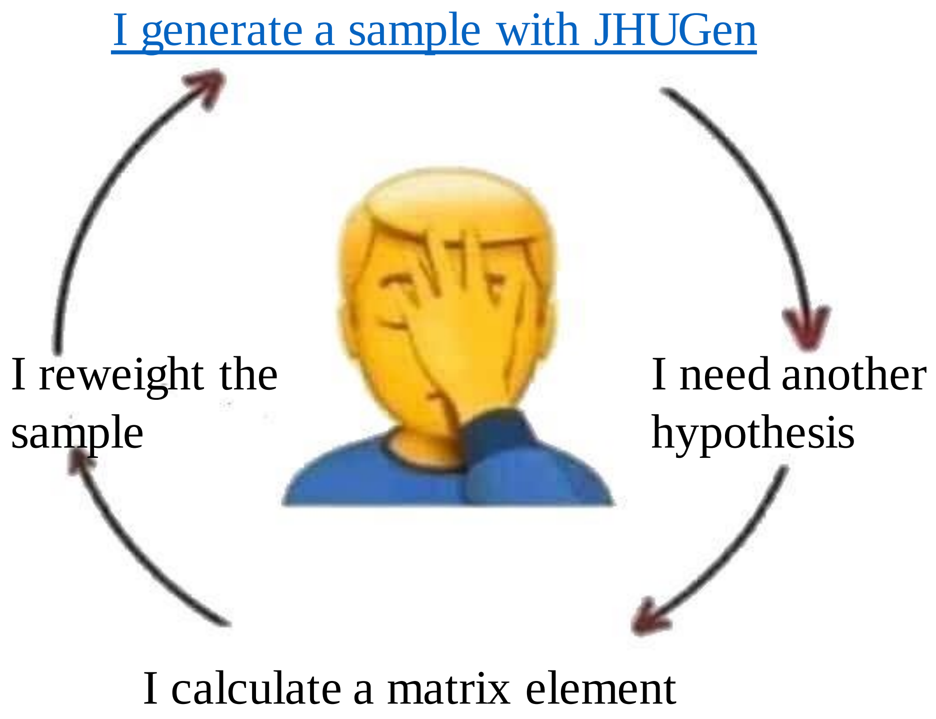
- We've set the process, matrix element, and production!
- We've set the necessary couplings!
- We've set input events!
- We've run computeP!
- **All of it was done in the event loop!**

- **You should get standard model output values out for your matrix element calculations!**

# Using Lexicon

- We will now repeat the same calculation but with new Warsaw couplings

-
  ```
  ./JHUGenLexicon.out
  HZ_couplings_only=true
  input_basis=warsaw
  output_basis=eft_jhu alpha=7.815553E-003
  cHbx=4,0
  tcHG=1,0
  ```



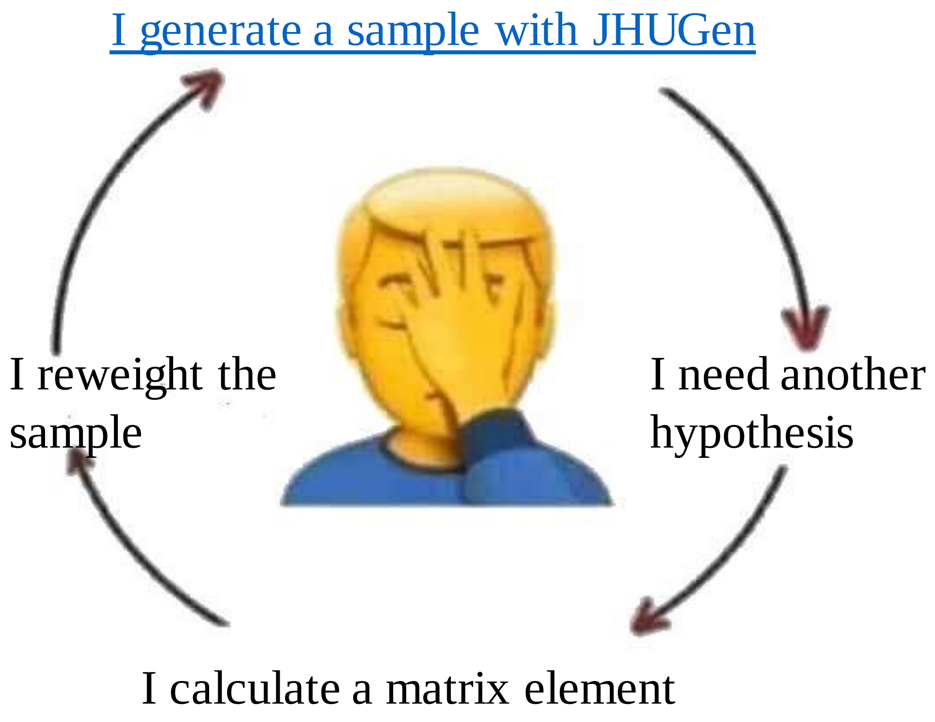
I generate a sample with JHUGen
I need another hypothesis
I calculate a matrix element
I reweight the sample

# Put Your Money Where Your MELA Is!

- ghg2=0,0
  ghgsgs4=0,0
  ghzgs4=0,0
  ghgsgs2=0,0
  ghzgs2=0,0
  ghz4=0,0
  ghg4=-0.121249,0
  ghz2=0,0
  ghz1_prime2=0,0
  ghz1=2.48499,0

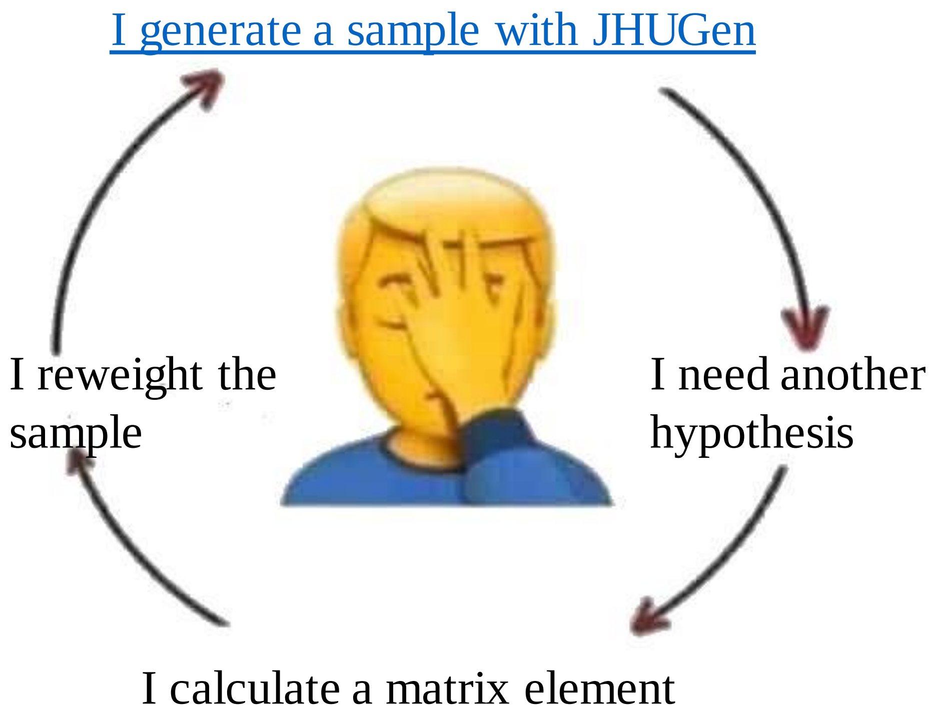
**gHIGGS_GG_4**

**gHIGGS_VV_1**


I generate a sample with JHUGen
I need another hypothesis
I calculate a matrix element
I reweight the sample

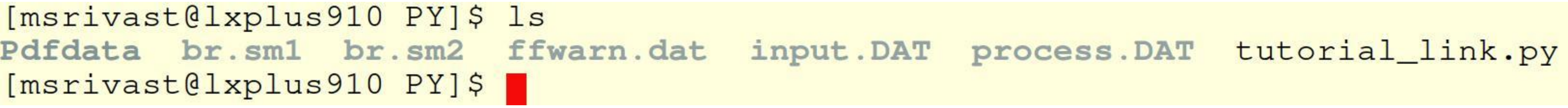
# Put Your Money Where Your MELA Is!

- Do you see output?


I generate a sample with JHUGen
I need another hypothesis
I calculate a matrix element
I reweight the sample

# WHAT ARE THESE PESKY FILES!?

- In the area you run your MELA calculation, you should see a few new files!



```
[msrivast@lxplus910 PY]$ ls
Pdfdata  br.sm1  br.sm2  ffwarn.dat  input.DAT  process.DAT  tutorial_link.py
[msrivast@lxplus910 PY]$ 
```

- The Pdfdata directory
  - NNPDF30_lo_as_0139.LHgrid, cteq61.tbl, cteq611.tbl
- The files br.sm1, br.sm2, ffwarn.dat, input.DAT, process.DAT

- These are all files that MELA needs when running calculations
  - You can delete them, but they'll just come right back!

# REWEIGHTING!!!!!!!!!!!!!!

- To get the weights for the sample, you must **replace** the matrix element in the expression from slide 4.
- You should multiply the expression by (NEW HYPO/NATIVE HYPO) to get weights for the sample successfully
- Lucky for you – the native hypothesis of the GGZZ sample was just ghz1=1!
- If you want to plot this reweighted value – feel free! I do not give you the code to do so, but it is certainly something you must do eventually

# REWEIGHTING!!!!!!!!!!!!!!

- To get the weights for the sample, you must **replace** the matrix element in the expression from slide 4.
- You should multiply the expression by (NEW HYPO/NATIVE HYPO) to get weights for the sample successfully
- Imagine new_hypo_values and old_hypo_values are lists

```
np.histogram(m4l_distro,  weights=(new_hypo_values/old_hypo_values))
```

```
int i = 0;
while(myReader.next()){ //reading with TTreeReaderValue
Hist -> Fill((*m4l_distro)[i], new_hypo_values[i]/old_hypo_values[i]);
i += 1;
}
```

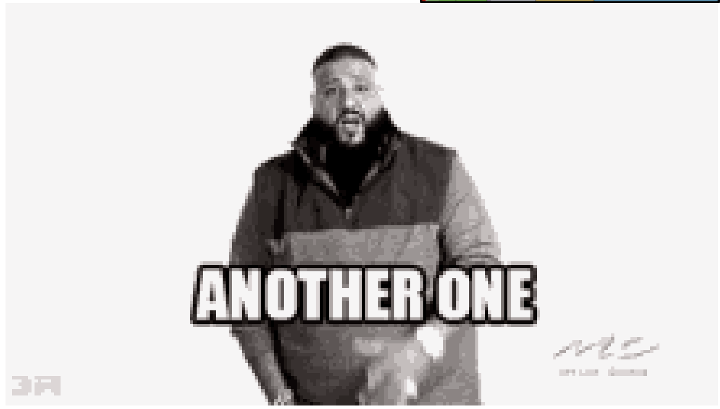
# OPTIMAL DISCRIMINANT!!!!

- Take the matrix element calculations that you have and do the following:

- HYPO1/(HYPO1 + C*HYPO2)
  - C is some constant to make the plot prettier (cross at 0.5)

- If you want to plot this value – feel free! I do not give you the code to do so, but it is certainly something you must do eventually

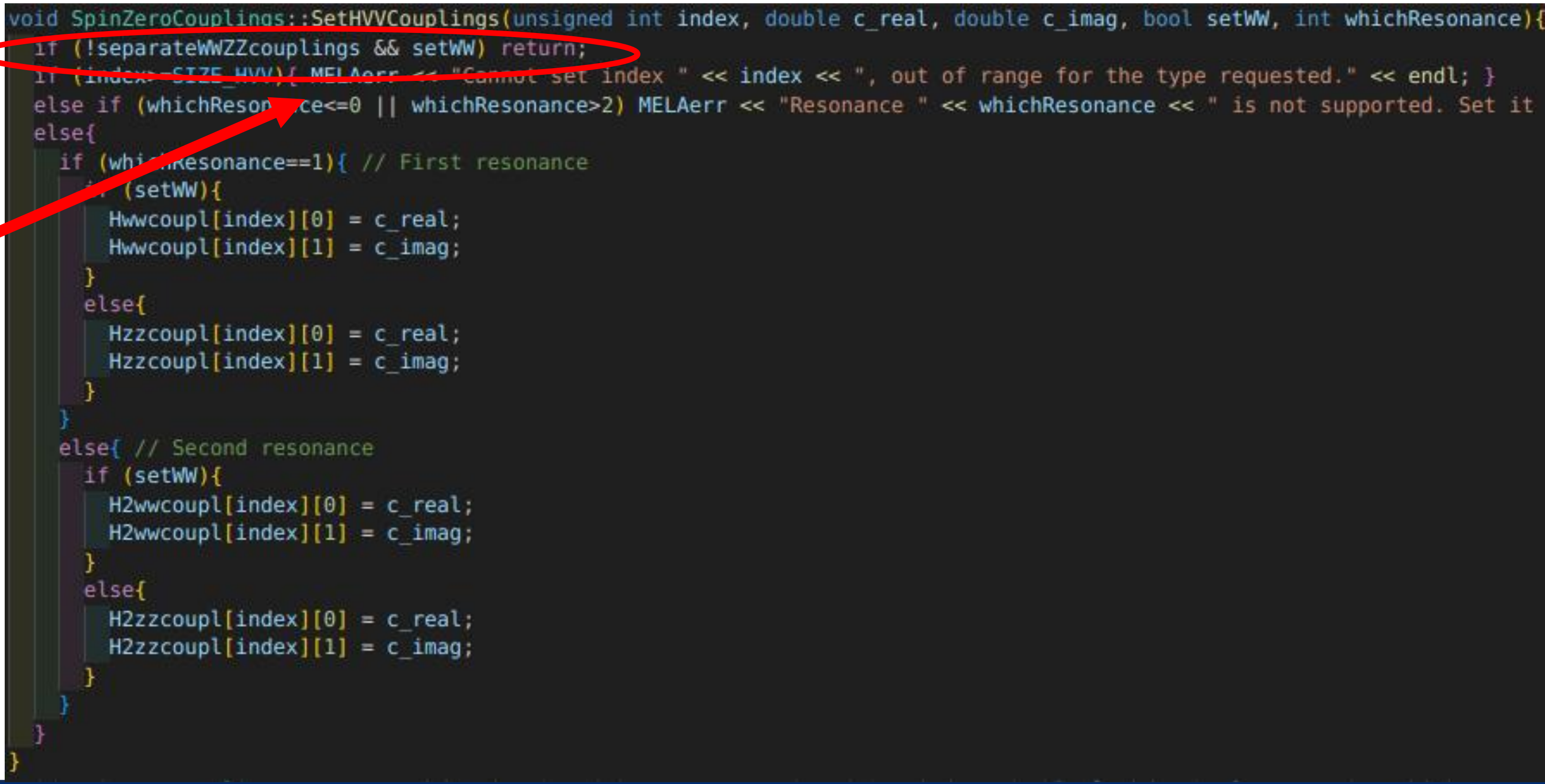
# Let's do the same thing now for EW!



- Exact same calculation, but with a few changes
  - **Let's try standard model for this one (ghz1=2)**
- **Change the file to the EW one**
- Production is now JJEW ("JJ Electroweak")
- Compute function is now computeProdDecP
- We have different particles to input!
  - We have associated particles (resultant jets from VBF production/ decaying Z leptons)
  - We have mothers (quarks)
  - We have daughters (4 leptons)

## Let's do the same thing now for EW!

- By default, the MELA parameter "bool differentiate_HWW_HZZ" is set to false



```
void SpinZeroCouplings::SetHVVCouplings(unsigned int index, double c_real, double c_imag, bool setWW, int whichResonance){
  if (!separateWWZZcouplings && setWW) return;
  if (index>=SIZE_HVV){ MELAerr << "Cannot set index " << index << ", out of range for the type requested." << endl; }
  else if (whichResonance<=0 || whichResonance>2) MELAerr << "Resonance " << whichResonance << " is not supported. Set it
  else{
    if (whichResonance==1){ // First resonance
      if (setWW){
        Hwwcoupl[index][0] = c_real;
        Hwwcoupl[index][1] = c_imag;
      }
      else{
        Hzzcoupl[index][0] = c_real;
        Hzzcoupl[index][1] = c_imag;
      }
    }
    else{ // Second resonance
      if (setWW){
        H2wwcoupl[index][0] = c_real;
        H2wwcoupl[index][1] = c_imag;
      }
      else{
        H2zzcoupl[index][0] = c_real;
        H2zzcoupl[index][1] = c_imag;
      }
    }
  }
}
```

- What does this mean?
- Setting HZZ sets HWW
- NOT vice versa!

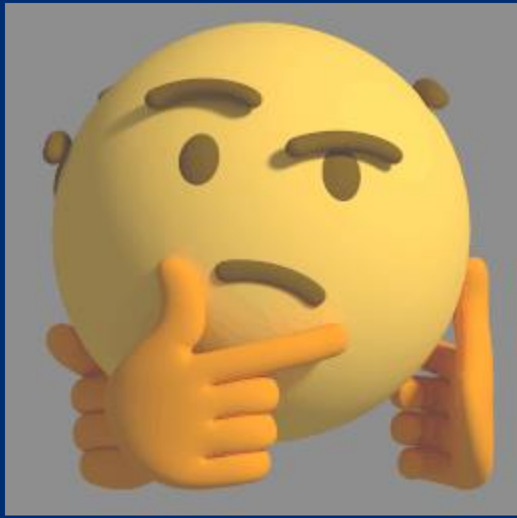
- What does this mean?

# LET'S THINK!

MELA is, by default, H->ZZ decay!

1) If there are no couplings set between the Higgs and the Z, can HZZ decay occur?

2) How would you go about isolating WW VBF production in MELA?

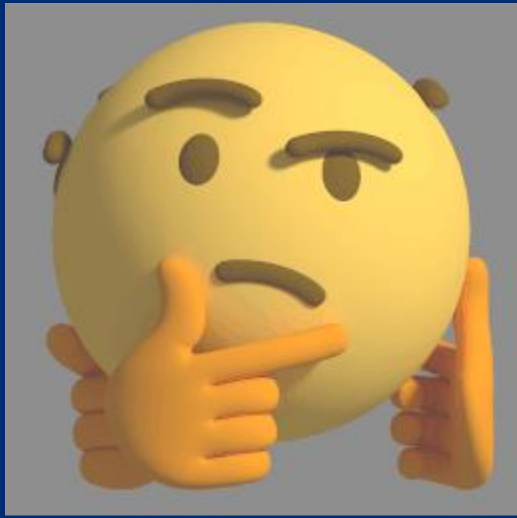


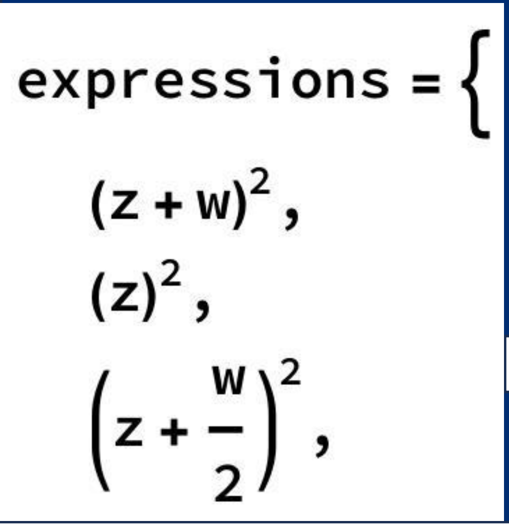
# LET'S THINK!

**1) The returned probability would be 0!**

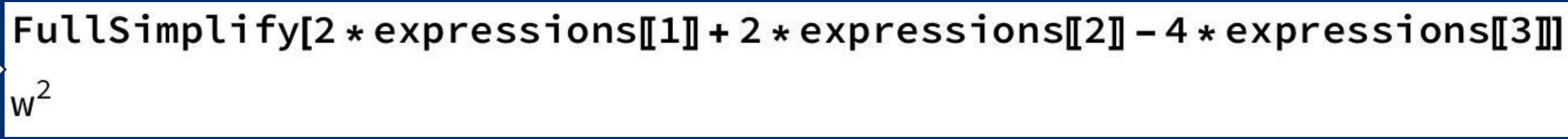
2) Linear combinations!

- Set combinations of Z and W couplings then add/subtract them!





$$\text{expressions} = \left\{ (z+w)^2,\ (z)^2,\ \left(z+\frac{w}{2}\right)^2, \right.$$



```
FullSimplify[2*expressions[[1]]+2*expressions[[2]]-4*expressions[[3]]]
```

$w^2$

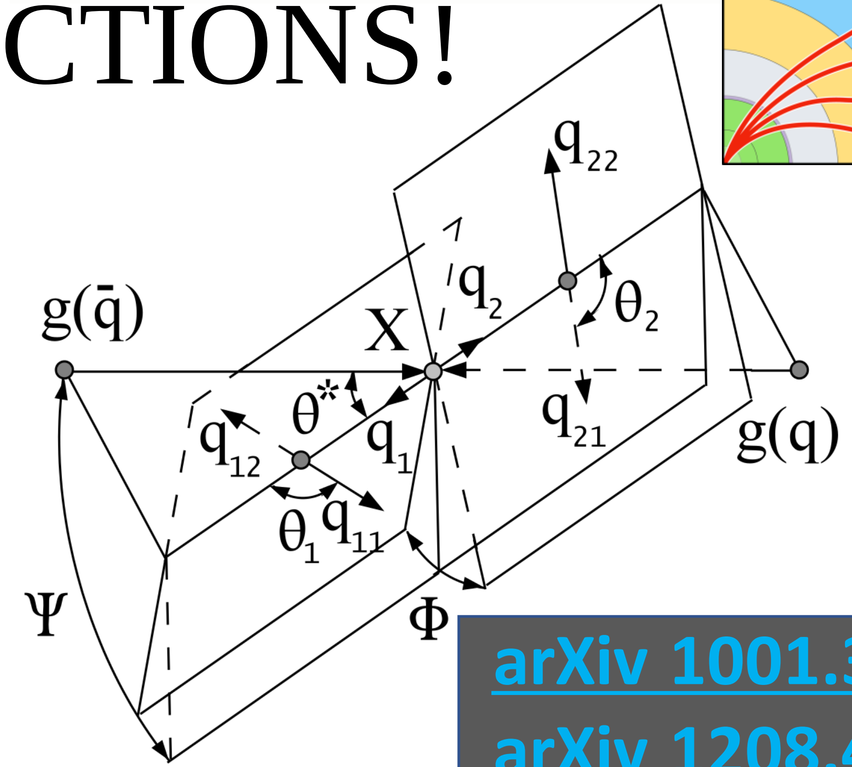
# OTHER MELA OPTIONS!

- The best option is to parse the documentation for MELA to look for functions that you think would be useful – but here are some useful functions for your pleasure!

- **SetCandidateDecayMode** sets the decay mode you want from TVar::CandidateDecayMode
- **ResetMass** and **ResetWidth** set the mass and widths of various particles
- **SetMelaHiggs{Mass, Width, MassWidth}** sets their respective values for the Higgs (useful for the propagator in offshell)

# OTHER MELA FUNCTIONS!

- The laundry list of angle calculations
  - computeDecayAngles
  - computeVBFAngles
  - computeVBFAngles_ComplexBoost
  - computeVHAngles



arXiv 1001.3396

arXiv 1208.4018

- Other methods exist like Mela::getHiggsWidthAtPoleMass(mass)
- Best to look at the documentation or at Mela.h to find every function you can use

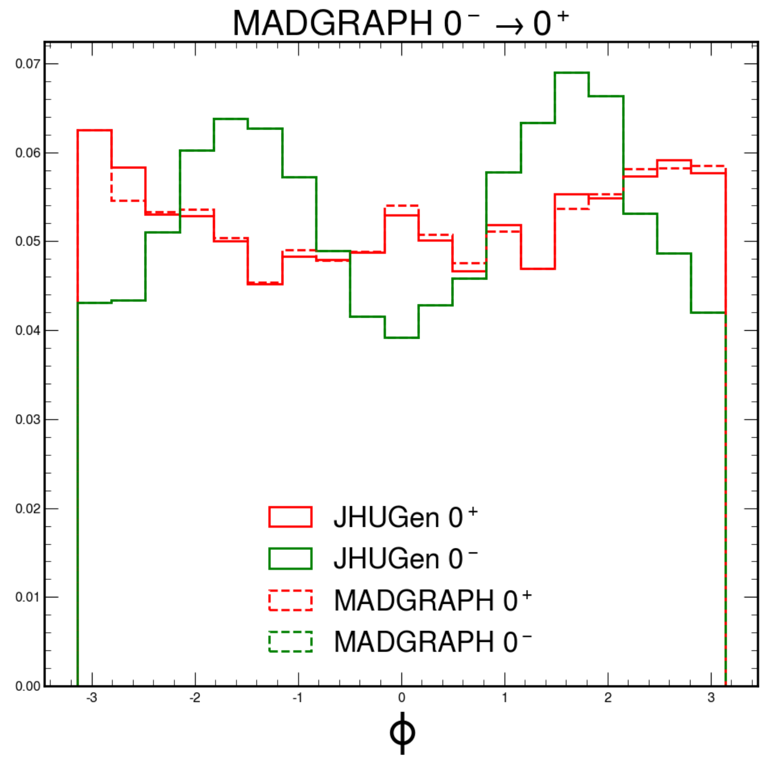
# OTHER MELA FUNCTIONALITY!

- You can get input/output records for MELA parton-by-parton!
- You can also get the actual matrix-element array!
  - Look at the MELAIO class (also implemented in the python under the MELA package)

- If there is ever some functionality you want in MELA - **contact the developers!**
  - **Send an email to the email list on spin.pha.jhu.edu**
  - The Python has functions bound from the C++ **one-by-one!**
  - So, tell us when it's missing something you want!

# MADMela????

- Currently there is a **beta** version of MELA that includes MADGRAPH matrix elements for gg --> Higgs --> Z/Gamma --> 4l using SMEFTSIM 3



- Plot is for ghz4=1 **AKA:**
  mdl_chwtil = -6.34078
  mdl_chbtil = -1.90674
  mdl_chg = -8.24752
  mdl_chwbtil = -6.9542
  mdl_chbox = -16.495
- To standard model **AKA** all terms 0 (in SMEFTSIM)

# MADMela????

- We want this tool to be widely used for EFT :)
- If there are any other processes you want us to add specifically – **tell us!** (same email link)
- It will take us less time to do it if you give us the MADGRAPH reweighting area to begin with!
- Currently planning to do VBS and offshell ggH

# I NEED HELP!

- The **official** documentation for MELA exists, although still very much a work-in-progress, here
  - https://spin.pha.jhu.edu/MELA/
- An **experimental** version of the documentation exists here
  - https://msrivast.web.cern.ch/MELADoc/Docs/html/index.html
  - More up to date – but filled with a few assumptions that are yet to be double checked
- Please keep in mind that both versions are still very much in "beta"
  - If you have any suggestions, please reach out to us!

- **If you encounter ANY bugs – PLEASE reach out to us!**
  - **Send an email to the email list on spin.pha.jhu.edu**

# FINAL QUESTIONS?

- Hopefully this was helpful!
- Any questions are welcome!