

Simulating dark matter with the CMS simulation chain

On the use of MadGraph5, MadAnalysis, and CMSSW to simulate and reconstruct events of the Dark Fermion model

Stephen D. Butalla

Department of Aerospace, Physics and Space Sciences
Florida Institute of Technology
stephen.butalla@cern.ch

July 21, 2023

Contents

1	Introduction	4
2	The CMS simulation chain	4
3	The basics of remote computing	7
3.1	Connecting to CERN's LXPLUS	7
3.2	Connecting to the FNAL LPC CAF (the computing cluster)	9
3.3	All things Grid-related	10
3.3.1	Getting a Grid certificate	11
3.3.2	Adding your Grid certificate to LXPLUS and the LPC CAF	12
3.3.3	Connecting to the Grid	13
3.4	Mounting your EOS disk space/CERNBox	14
3.4.1	Instructions for LXPLUS	14
3.4.2	Instructions for LPC CAF	15
4	The Hidden Valley of Particle Physics	16
4.1	The Dark Fermion Model	16
5	Building a model with FeynRules	17
5.1	The FeynRules file	17
5.1.1	File header	17
5.1.2	Model information	18
5.1.3	Gauge group definitions	18
5.1.4	Change log	18
5.1.5	Interaction order definitions	19
5.1.6	Field and gauge boson definitions	19
5.1.7	Parameter definitions	21
5.1.8	Lagrangian definition	22
5.2	Generating the UFO files	23
6	Simulating the hard-scattering events with MadGraph5	27
6.1	Installing MadGraph5	27
6.2	Setting up MG5 for simulating the f_D model	28
6.3	Using the CLI	28
6.4	Using the <code>proc_card.dat</code> and the MG5 executable	29
6.5	Modifying the parameters of the model	30
7	Hadronization, showering, and reconstruction with CMSSW	31
7.1	Installing CMSSW	31
7.1.1	Instructions for LXPLUS	31
7.2	The mechanics of running CMSSW locally	32
7.2.1	The CMSSW configuration file	32
7.2.2	Using the configuration file as input to CMSSW	33
7.3	The mechanics of running CMSSW with CRAB	33
7.3.1	The CRAB configuration file	34
7.3.2	Basic commands	36
7.3.3	CRAB configuration files	38
7.4	Step 1: GEN-SIM	38
7.4.1	Running locally	39
7.4.2	Running with CRAB	39
7.5	Step 2: GEN-SIM-RAW	41
7.5.1	Running locally	41
7.5.2	Running with CRAB	41

7.6	Step 3: DIGI I	42
7.6.1	Running locally	42
7.6.2	Running with CRAB	42
7.7	Step 4: DIGI II	42
7.7.1	Running locally	42
7.7.2	Running with CRAB	43
8	<i>n</i>-Tuple production with MuJetAnalysis	43
A	List of acronyms	44
B	Successful output of the vertices calculation using FeynRules	45
C	Example proc_card.dat file	47
D	Example of a generated CMSSW configuration file	48

DRAFT

1 Introduction

This guide is purposed to accomplish two tasks: (1) provide an overview of the full simulation chain used by the Compact Muon Solenoid (CMS) collaboration, from model building with the `FeynRules` Mathematica package to event reconstruction with the CMS software (CMSSW), and (2) provide a hands-on tutorial for generating events at the parton level and then reconstructing these events, including hadronization and showering, as well as the full simulation of the CMS detector system. These topics will be explored through concrete examples of the so-called dark fermion (or f_D) model, which is one of the many processes underneath the umbrella of the “Hidden Valley” or “Hidden Sector” theories of dark matter (DM). Note that this is just one of the many ways to accomplish the simulation and reconstruction of events—other tutorials might (and probably will) vastly differ from this one.

Before discussing the minutiae, however, I will present a brief overview of the entire simulation chain in Section 2. We’ll take a slight detour in Section 3, where we’ll discuss the basics of remote computing on the machines at LXPLUS and the FermiLab National Laboratory (FNAL) LHC Physics Center (LPC) Computing Analysis Facilities (CAF)—known as the FNAL LPC CAF—including the prerequisite steps for connecting to these machines, (Sections 3.1 and), everything you’ll need to know about grid certificates (Section 3.3.1), and finally mounting and accessing your EOS space on LXPLUS and the LPC CAF (Section 3.4).

In Section 5, I’ll review the model building step, specifically a comprehensive overview of the contents of the `FeynRules` file, and in the output of which is used as input to the event generator, which is software that simulates the hard-scattering (tree-level) process. First, however, I’ll include an introduction to the so-called “Hidden Valley” theory of dark matter and the dark fermion (f_D) model will be provided in Section 4. In Section 6, the initial configuration and the process of simulating events with `MadGraph5`, a Monte-Carlo event generator at next to leading order (`MadGraph5_aMC@NLO`, or just “MG5”), will be provided.

While this guide will be heavily geared towards event generation and reconstruction for the f_D model, the information presented will be generalizable to other models and processes. This manual is also purposed for any user that meets the following criteria: (1) the user has a regular CERN account (not a lightweight account), or (2) the user has an account on the Fermilab LPC cluster **and** has a regular CERN account. This guide will therefore be written with either of these two users in mind.

One last note on the stylistic conventions of this document: important notes or warnings will be displayed in bright **red**, while hyperlinks are displayed in **crimson**. Inline text in **monospace font** indicates either a software executable or a file name. Boxed text with a cream colored background is reserved for Bash/ZShell commands executed from the command line, e.g.,

```
$ voms-proxy-init --rfc --voms cms
```

and boxed text with a light cyan background is reserved for text in a file, usually a configuration file for `MadGraph5`, `CMSSW`, etc.

```
*****
# Number of events and rnd seed *
# Warning: Do not generate more than 1M events in a single run *
*****
10000 = nevents ! Number of unweighted events requested <-----
0 = iseed ! rnd seed (0=assigned automatically=default)
```

2 The CMS simulation chain

All physics analyses follow a similar structure, beginning with event simulation of the process under consideration, to the analysis of the real data. In general, the process will follow a similar manner as that enumerated below, although the tools used (e.g., `MadGraph5` vs. `POWHEG`, etc.) will be different.

The simulation chain, in general, contains three individual steps, some of which contain multiple sub-steps:

1. Model Building

- **FeynRules** and Mathematica is used to build the interaction Lagrangian and the supporting files which are used as input to an event generator like MadGraph5. The input files include one for the Standard Model and another for your model.
- The FeynRules (`.fr`) file contains definitions for the gauge groups, fields, bosons (both physical and unphysical), particle couplings, model parameters, and the interaction Lagrangian itself. A Mathematica notebook is used to load both the SM and model files, check Hermiticity and that the model is internally consistent, and then construct the various interaction
- The output is a directory of Python files which constitutes the universal FeynRules output (UFO) standard.

2. Event generation (hard-scattering events)

- **MadGraph5** (or a similar event generator like POWHEG; see [4]) is used to simulate hard-scattering events at the tree level or next-to-leading order. LHE files are produced from this output, which are used as input to the first reconstruction step.

3. Reconstruction

- CMSSW is used (which in turn uses Pythia8 and GEANT4 for showering/hadronization and detector response simulation, respectively) to reconstruct events.
- There are four steps to the reconstruction process:
 - (a) **GEN-SIM**: Pythia is used to simulate QCD background, including hadronization and showering of the SM process that result QCD radiation. These data are added to the underlying event data obtained from the LHE file.
 - (b) **GEN-SIM-RAW**: Simulated pile-up is overlaid on each event.
 - (c) **RECO**: The signal and background data (including pileup) are now reconstructed. This means that the various tracks throughout space are superimposed in the CMS detector, which is fully simulated, with all of the detector geometries and efficiencies.
 - (d) **MiniAOD**: Here, the analysis object data (AOD) file in its entirety is trimmed down to contain only the objects that are necessary for a particular analysis. For example, in some analyses, all information regarding hadrons is used. So, during this step the information from the hadronic calorimeter would be retained while the information from the electromagnetic calorimeter would be removed.

4. n -tuple production

- The reconstructed data in some type of AOD format (e.g., miniAOD, microAOD, nanoAOD) is processed down to an n -tuple.
- An n -tuple will contain only the physics objects (e.g., muon information such as p_T , η , and ϕ) that are necessary to the analysis.
- During the this step cuts will also be applied, such as placing constraints on the p_T of each of the final state muons.
- The final file format will be a ROOT `.root` file which contains different physics objects stored in the branches of the tree. It will essentially be a collection of arrays, ordered by event number.

These steps are neatly summarized in the flowchart in Fig. 1.

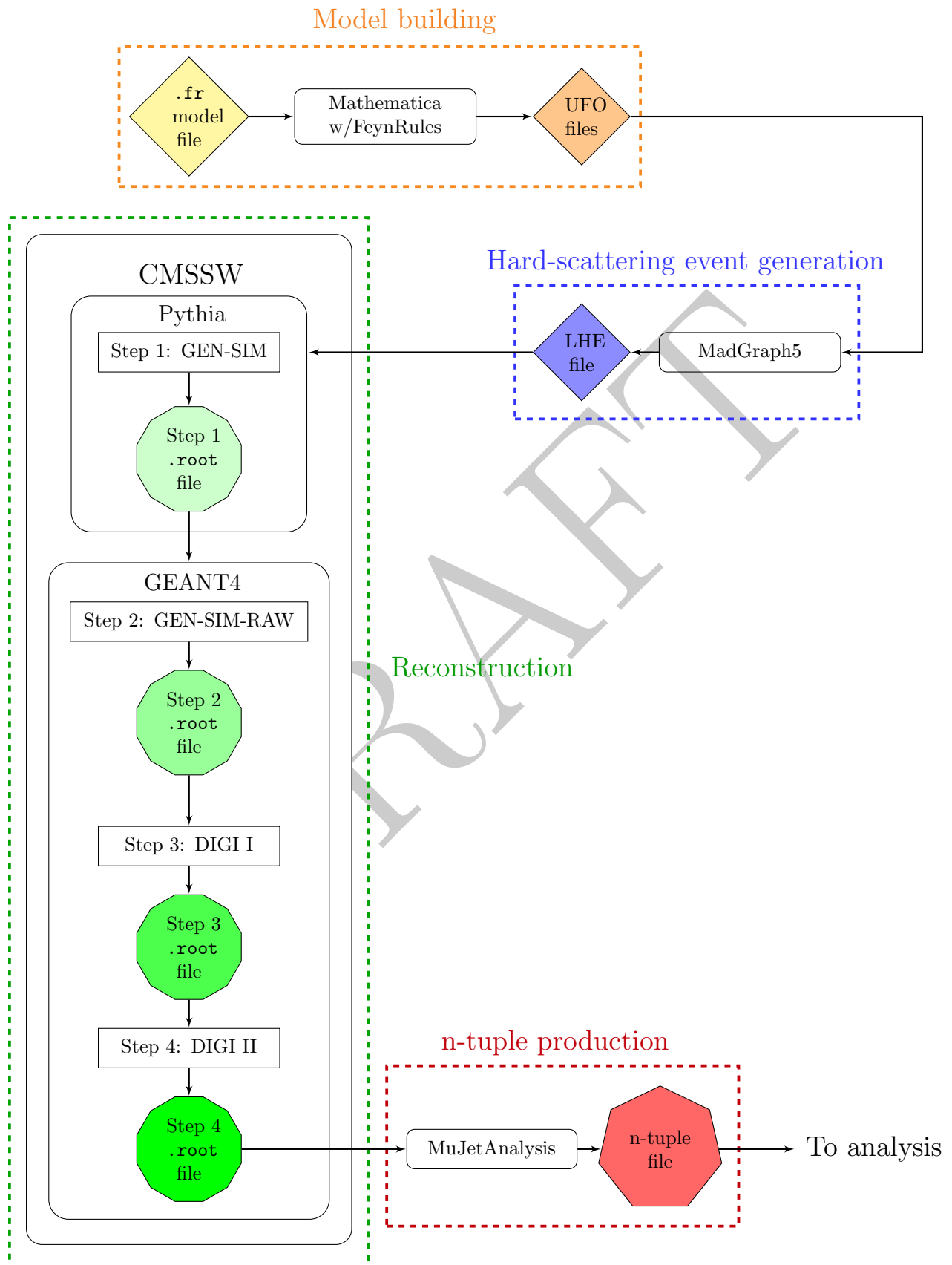


Figure 1: A flowchart of the CMS simulation chain.

3 The basics of remote computing

While this is somewhat outside of the scope of this manual, I will include a brief description and a set of instructions for the basics of remote computing before we get any farther. I will discuss two separate scenarios: using the computing resources of (1) CERN’s LXPLUS and (2) the FNAL LPC CAF.

3.1 Connecting to CERN’s LXPLUS

This first option is a lot easier than connecting to the LPC cluster in terms of jumping through digital configuration hoops, but with any wealth of advantages comes a host of disadvantages (I’ll discuss this later). Before we begin, be sure to fully set up your regular CERN account (i.e., not a lightweight account). You can verify that you have a regular account if you are in the CERN phonebook. Afterwards, follow the steps enumerated below to set up your Andrew File System (AFS) space on CERN’s cluster

1. **Navigate to the CERN Account Management page.** Click on the “My Accounts” tab near the top of the page.
2. Click on the “Services” option (see Fig. 2).

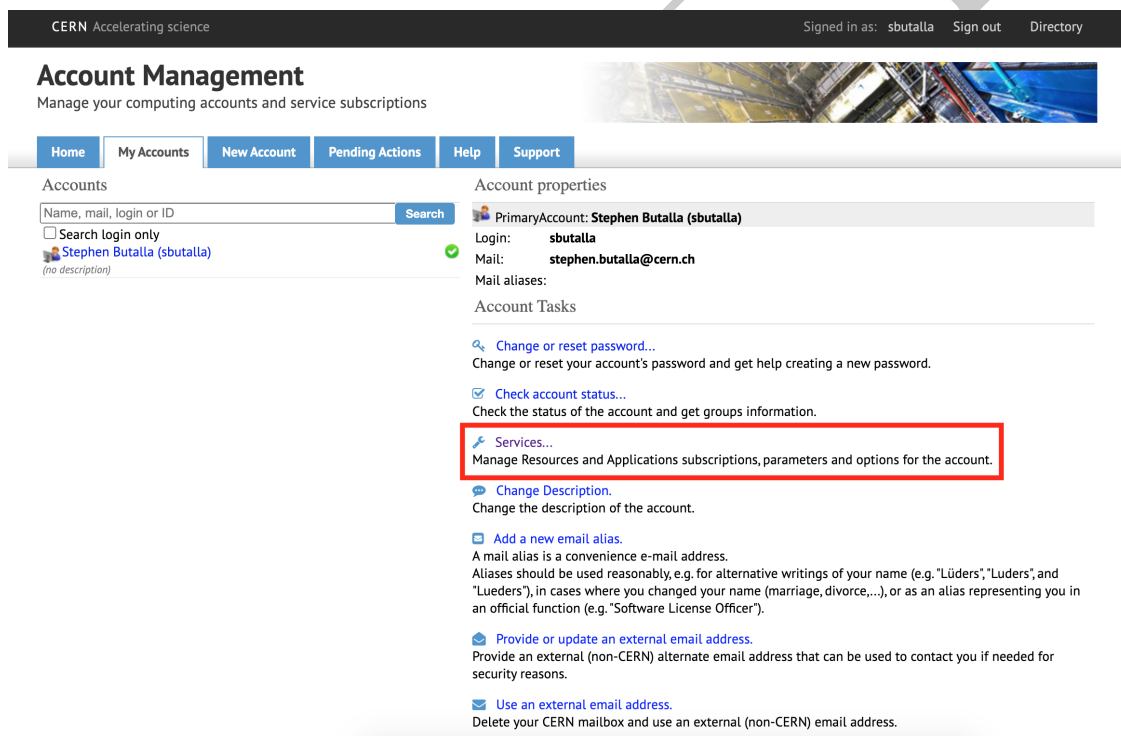


Figure 2: A screenshot of the CERN Account Management page.

3. Subscribe to the “LXPLUS and Linux services” (highlighted by a red box in Fig. 3), “AFS Workspaces” (highlighted by an orange box in Fig. 3), and “EOS/CERNBox” (highlighted by a magenta box in Fig. 3).
4. Getting access to the AFS workspace is as simple as just clicking on the “Subscribe” tab (the red box in Fig. 4), and then clicking on “Subscribe to AFS” on the next page.

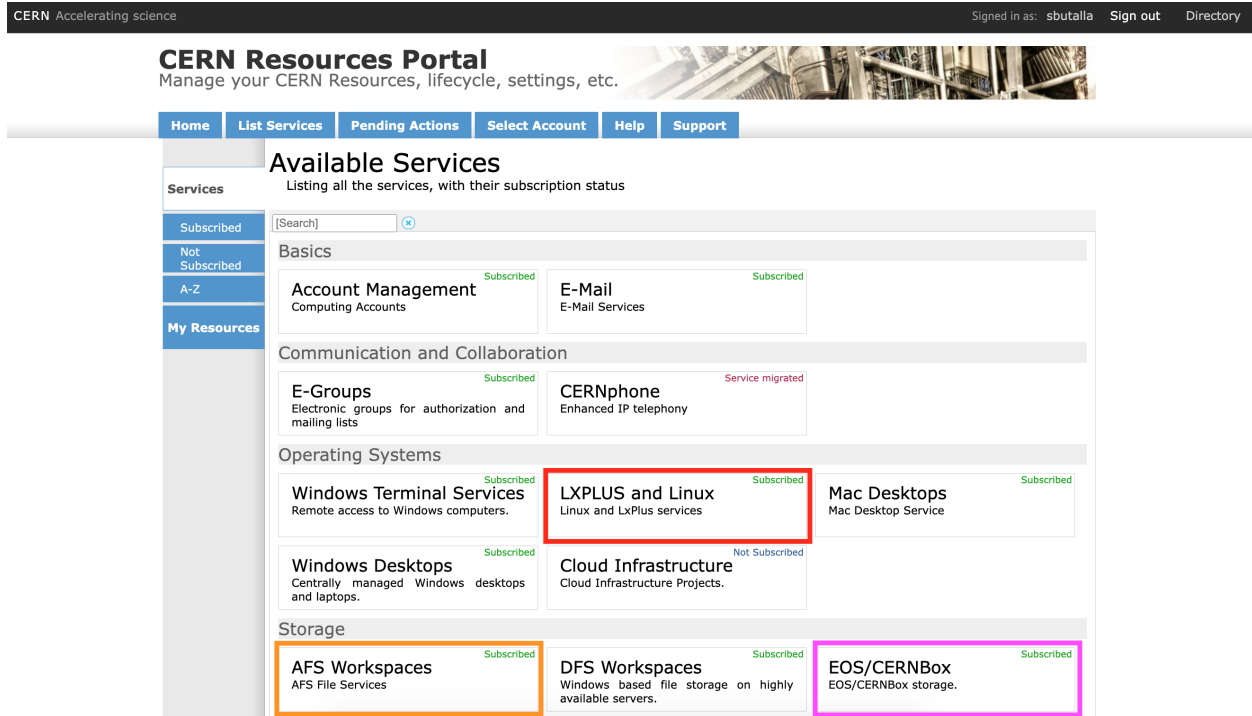


Figure 3: A screenshot of the CERN Account Management page. Subscribe to the services highlighted in the red, orange, and magenta boxes by clicking on the links and following any subsequent directions.

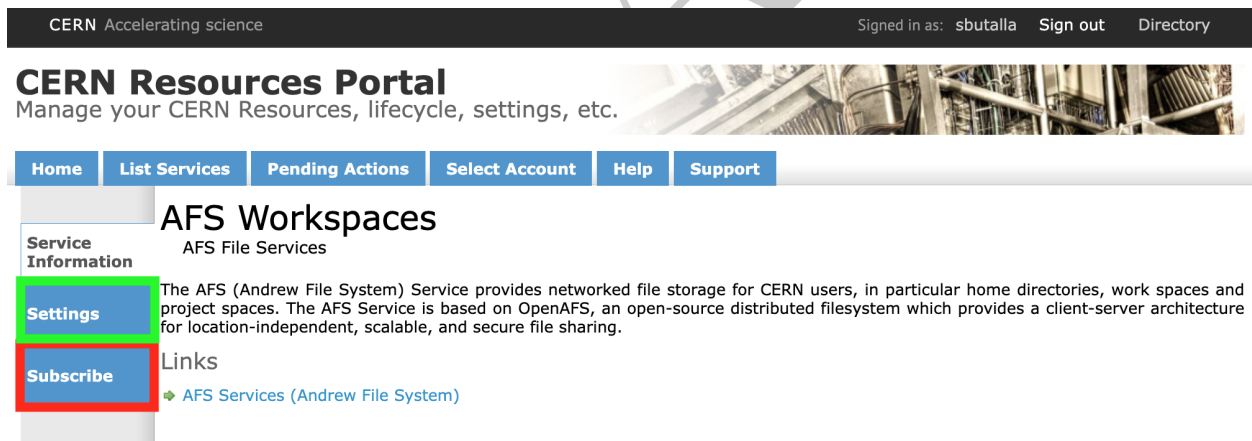


Figure 4: A screenshot of the AFS Workspaces services page.

5. While you're here, I also recommend increasing your disk space quota, as you only start with 1 GB. First, click on the "Settings" tab (highlighted by the green box in Fig. 4) and then click on the option to increase home folder quota. This will be in the area as denoted by the green box in Fig. 5. Note that you will only be able to increase your quota to 10 GB, and this has to be done in increments.

Figure 5: A screenshot of the AFS Workspaces page. Subscribe to the services highlighted in the red, orange, and magenta boxes by clicking on the links and following any subsequent directions.

If you try to create an AFS workspace and you don't have the proper permissions¹ you'll receive the notice in bright red lettering in Fig. 5. For a typical analysis you'll most likely be working with a large amount of data—hundreds of GBs, perhaps even several TBs—which is a clear problem with the stringent limit on disk quota. However, you can connect your EOS disk space to your AFS space, which can be used to run some applications and store data—with several caveats that I'll cover in Section 3.4.

6. Test out your account by connecting to LXPLUS via `ssh`:

```
$ ssh username@lxplus.cern.ch
```

It is important to note that the AFS is currently being phased out, and will most likely be replaced by EOS Open Storage (EOS) (yes, an acronym containing an acronym); see the [CERN EOS webpage](#) for more information.

3.2 Connecting to the FNAL LPC CAF (the computing cluster)

To connect to the FNAL LPC computing analysis facility (CAF) you first need to apply for computing access [here](#). Next, call the Fermilab Service Desk at 630-840-2345 to get a temporary single sign-on (SSO) password. **Also request a Kerberos² password, as you'll need this to connect to the cluster.** Once this is complete, follow the steps below:

¹You need to be a member of one of the CMS computing groups; EP-UCM membership does not qualify.

²Kerberos is a protocol for network authentication and is primarily used in the client-server scenario. See the [Kerberos webpage](#) for more information.

1. First, we need to set up our `ssh` configuration directory and configuration file. Most likely you will already have the directory and file, but in case you don't:

```
$ cd ~/ && mkdir .ssh && touch .ssh/config
```

Using your favorite text editor add the following configurations to your `config` file (applicable for all *Nix distributions):

```
Host cmslpc*.fnal.gov
  GSSAPIAuthentication yes
  GSSAPIDelegateCredentials yes
  StrictHostKeyChecking no
  UserKnownHostsFile /dev/null
```

2. Now, use the client URL (cURL) command to retrieve the Kerberos configuration file which will be output to the `/etc/ssh` directory. This file *is* OS dependent, so I'll list a few below:

macOS

```
$ sudo curl https://authentication.fnal.gov/krb5conf/OSX/krb5.conf \
-o /etc/ssh/krb5.conf
```

Scientific Linux 7 (SL7)/CentOS 7

```
$ sudo curl https://authentication.fnal.gov/krb5conf/SL7/krb5.conf \
-o /etc/ssh/krb5.conf
```

If you have a different OS than the ones listed here, check out the [Fermilab Kerberos Configuration Files webpage](#) for more options.

3. Test out your connection by first generating a Kerberos ticket:

```
$ kinit username@FNAL.GOV
```

You can verify this is working by checking the hostname and the associated ticket time stamp in the output listed by the command

```
$ klist
```

After generating the ticket, connect to the LPC cluster using `ssh`:

```
$ ssh username@cmslpc-s17.fnal.gov
```

Note that you have to generate a Kerberos ticket every time you connect to the cluster (if you do not already have an active Kerberos ticket; they expire after 24 hours). If you experience any problems, check out the USCMS webpage [Connect to the LPC CAF \(Central Analysis Facility\)](#) for some excellent resources.

3.3 All things Grid-related

To access CERN's vast array of interconnected computers around the world, you'll need to get a grid certificate. This is particularly important if you are going to do any serious simulation for, say, CMSSW via the CMS Remote Analysis Builder (CRAB) or Condor.

Connection to the Grid, achieved by way of authentication by the Virtual Organization Management Service (VOMS) is essential to running CMSSW with all of the available accoutrements. For example, to simulate conditions with pileup, you need to be connected to the grid so CMSSW can access the database where these

configuration files are hosted. In Section 3.3.1, I list the steps for getting a Grid certificate, which are issued based on membership in a CERN virtual organization (VO). Since the steps of adding and configuring your grid certificate is identical on both LXPLUS and the LPC CAF, I'll discuss this process in Section 3.3.2.

3.3.1 Getting a Grid certificate

Before starting though, you need to make sure that you're in the CERN phonebook—this will confirm if you have a membership in one of the many organizations at CERN. For CMS users in our research group, most likely you will already be a CMS member (or “User CMS”, which is shortened to UCM) and affiliated with the Experimental Physics (EP) department. So you'll see the affiliation of EP-UCM in your phonebook entry.

1. Navigate to <https://ca.cern.ch/ca/> and click on **New Grid User certificate**. After logging in with your CERN SSO credentials, click on “Get Grid User certificate,” (see Fig. 6 below).

The screenshot shows the CERN Certification Authority website. At the top, it says "CERN Accelerating science" and "Signed in as: sbutalla Sign out Directory". The main heading is "CERN Certification Authority". Below this is a navigation menu with "Home", "My User Certificates", "My Host Certificates", "New Grid User Certificate" (selected), "New Grid Host Certificate", "Help", and "Support". The main content area is titled "Request a new Grid User certificate". There is a blue information box with an 'i' icon that says: "It is recommended that you provide a password to protect your certificate. If you want to install your certificate in your browser or your system, please refer to the help pages." Below this is a "Certificate Protection" section with two radio buttons: "Protect your certificate with a password (recommended)" (selected) and "Do not use a certificate password". Under the selected option, there are two input fields: "Certificate password" and "Confirm password". A "Get Grid User certificate" button is located below the form. At the bottom left, there is a "Related sites" section with links to "CERN Certification Authority Files", "Account Management", "CERN Resources Portal", and "Service Portal (Get Help)". At the bottom right is the CERN logo.

Figure 6: A screenshot of the Grid User certificate request form.

2. Enter the passphrase for the certificate, generate the certificate, and finally download it.
3. Next add the certificate to your Firefox browser (you can probably get it to work with Chrome, although it is more difficult):
 - (i) Open a browser window and go to preferences.
 - (ii) Scroll all the way down to the Certificates section and click on the View Certificates button.
 - (iii) Click on “Import” and find your recently downloaded certificate in p12 format, usually named `mycert.p12`.
4. Navigate to <https://voms2.cern.ch:8443/voms/cms/register/start.action> or <https://lcg-voms2.cern.ch:8443/voms/cms/> (both are equivalent).
5. Enter your email address if not already populated and proceed to the next step.

6. Agree to the terms and conditions of GRID use and then proceed to the next step. You will receive an email with further instructions. **Note that you have exactly one week from the date of the request to confirm your membership.**
7. Click on the link in the email to confirm your VO membership request.

3.3.2 Adding your Grid certificate to LXPLUS and the LPC CAF

1. In your home directory (/afs/cern.ch/user/firstInitial/cernUsername), make a hidden directory and navigate to it:

```
$ mkdir .globus && cd .globus
```

2. Transfer your certificate to this directory using secure copy, scp:

```
$ scp mycert.p12 cernUsername@lxplus.cern.ch:/afs/cern.ch/user/firstInitial/
  cernUsername/.globus
```

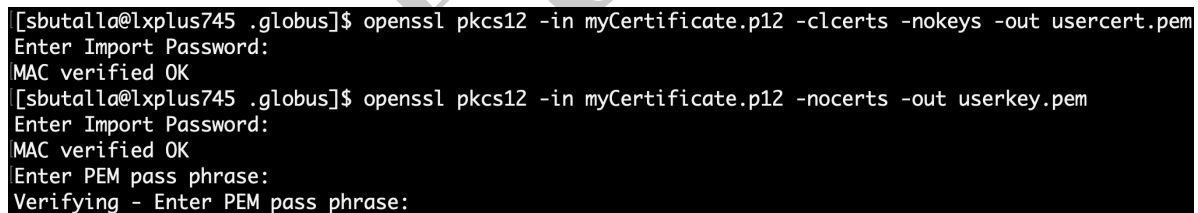
3. Remove the current user certificate and key if it exists by running:

```
$ rm -f usercert.pem && rm -f userkey.pem
```

4. Convert your certificate mycert.p12 from the PKCS#12 file format to the Privacy Enhanced Mail (PEM) format. Note: to simplify things, use the same password you used for generating the original certificate.

```
$ openssl pkcs12 -in mycert.p12 -clcerts -nokeys -out usercert.pem
$ openssl pkcs12 -in myCertificate.p12 -nocerts -out userkey.pem
```

For the FNAL cluster, you may have to rename the PKCS#12 file; if you receive an error, change the name of the certificate to usercred.p12. See Fig. 7 for a screenshot of successful output of converting your certificate.



```
[sbutalla@lxplus745 .globus]$ openssl pkcs12 -in myCertificate.p12 -clcerts -nokeys -out usercert.pem
Enter Import Password:
MAC verified OK
[sbutalla@lxplus745 .globus]$ openssl pkcs12 -in myCertificate.p12 -nocerts -out userkey.pem
Enter Import Password:
MAC verified OK
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Figure 7: Successful output for converting your certificate from the PKCS#12 to PEM format.

5. Change permissions on the certificate and key by using the running

```
$ chmod 400 usercert.pem && chmod 400 userkey.pem
```

Here, we use 400 as the parameter of the **chmod** (**change mode**) command, which changes the permissions of the file to read-only (thereby protecting against accidental overwriting).

6. Finally, test if your certificate is working

```
$ voms-proxy-init --rfc --voms cms -valid 192:00
```

If successful, the output will look like Fig. 8.

```
[sbutalla@lxplus737 src]$ voms-proxy-init --rfc --voms cms -valid 192:00
Enter GRID pass phrase for this identity:
Contacting voms2.cern.ch:15002 [/DC=ch/DC=cern/OU=computers/CN=voms2.cern.ch] "cms"...
Remote VOMS server contacted succesfully.

Created proxy in /tmp/x509up_u114726.

Your proxy is valid until Tue Mar 28 21:13:19 CEST 2023
```

Figure 8: Output of a successful grid connection test.

For more information consult the following Twiki pages:

- [For help with getting connected to the Grid.](#)
- [For help with registering in the CMS VO.](#)

3.3.3 Connecting to the Grid

To connect to the grid, you'll use a similar command to the one you used to test your certificate in the previous section. Here, you'll specify which VOMS you are a member of:

```
$ voms-proxy-init --rfc --voms cms
```

To simplify and economize connecting to the grid, define the following alias in your `.bash_profile` or `.bashrc` file (or equivalent profile for the shell of your choice):

```
alias proxy='voms-proxy-init --rfc --voms cms'
```

Be sure to source this file after editing so that the command is aliased in the current shell. This shortcut will prevent you from typing this verbose command every time you need to connect to the grid. Now, simply run:

```
$ proxy
```

and you'll create a grid session. A successful connection will resemble the output in Fig. 9 below.

```
[sbutalla@lxplus744 src]$ proxy
Enter GRID pass phrase for this identity:
Contacting voms2.cern.ch:15002 [/DC=ch/DC=cern/OU=computers/CN=voms2.cern.ch] "cms"...
Remote VOMS server contacted succesfully.

Created proxy in /tmp/x509up_u114726.

Your proxy is valid until Wed Mar 22 07:13:37 CET 2023
```

Figure 9: Output of a successful connection to the grid using the aliased command proxy.

3.4 Mounting your EOS disk space/CERNBox

3.4.1 Instructions for LXPLUS

1. After connecting to LXPLUS, export the following environment variable to your `.bashrc` or `.bash_profile`:

```
$ export EOS_MGM_URL=root://eosuser.cern.ch
```

Be sure to source the file after saving. Access your EOS directory at `/eos/user/<first initial>/<username>` (e.g., `/eos/user/s/sbutalla`). Note that this is your *personal* EOS space. A common alias for this disk space is known as “CERNBox,” which can be accessed via browser [here](#).

2. Check your EOS quote by running

```
$ eos quota
```

Successful output will resemble the screenshot displayed in Fig. 10.

```
[sbutalla@lxplus763 eos_dir]$ eos quota
By user:
└─> Quota Node: /eos/user/
┌───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┐
|user|used bytes|logi bytes|used files|aval bytes|aval logib|aval files|filled[%]|vol-status|ino-status|
├───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┘
|sbutalla|339.26 GB|169.63 GB|132.81 K|2.00 TB|1.00 TB|5.00 M|16.96 %|ok|ok|
└───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┘

By group:
└─> Quota Node: /eos/user/
┌───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┐
|group|used bytes|logi bytes|used files|aval bytes|aval logib|aval files|filled[%]|vol-status|ino-status|
├───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┘
|zh|259.16 TB|129.58 TB|74.05 M|0 B|0 B|0|100.00 %|ignored|ignored|
└───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┘
```

Figure 10: Output of the `eos quota` command.

3. To make things easy when scripting or just navigating around your workspace, make a symbolic link to your eos directory:

```
$ ln -s /eos/user/<initial>/<username> path/to/symlink
```

For example, to make the symlink called `eos_dir` from your home directory:

```
$ ln -s /eos/user/<initial>/<username> ~/eos_dir
```

Then, a simple `cd eos_dir` will take you to the base of you EOS storage.

Warning

If you plan to use CMSSW with CRAB, note that **you cannot use files stored in your EOS space as input**. This includes using LHE files as **input** for Step 1 or **any** of the output ROOT files for the succeeding steps.

3.4.2 Instructions for LPC CAF

1. After connecting to the LPC CAF, quickly check to confirm if you have an EOS space using the following command:

```
$ eosls -d /store/user/<CERN username>
```

A successful command results in the output of your username. If you receive a “No such file or directory” error then most likely the issue is that you do not have read/write permissions to the LPC cluster. This can be remedied by going to the [homepage of the LHC Physics Center](#), clicking on the “CMS Storage Space Request” option (highlighted by the pink box in Fig. 11), and filling out the form. Once you have access (you will get an email in about one to two weeks), you can write to your EOS space.

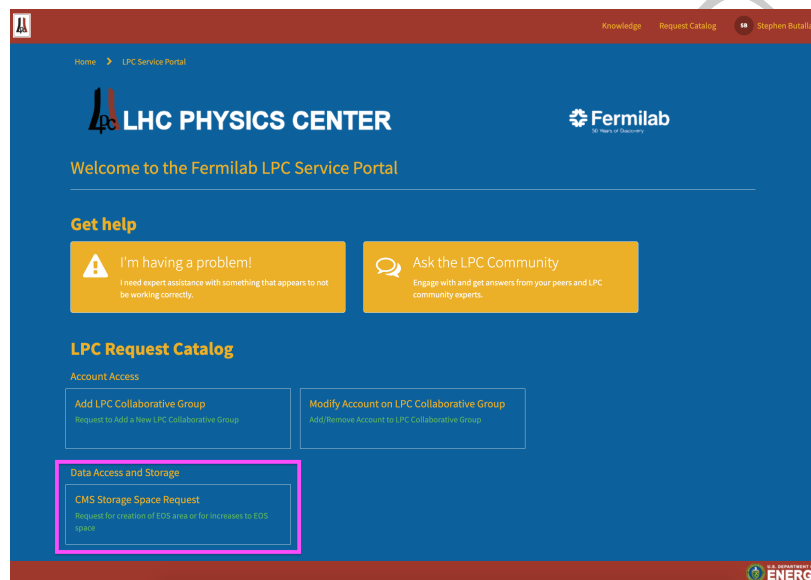


Figure 11: The LHC Physics Center Homepage with the CMS Storage Space Request highlighted by the pink box.

2. Once you confirm that you can access your EOS space, try listing the contents of your EOS:

```
$ eosls /store/user/<CERN username>
```

3. To check the disk quota, use the `eosquota` command (note that this is different than the LXPLUS syntax):

```
$ eosquota
```

Successful output is displayed in Fig. 12.

```
[sbutalla@cmslpc110 ~]$ eosquota
| Quota Node ==> /eos/uscms/store/user/ |
User      Used Space  Used Files  Avail Space  Avail Files  % Used  Space/Files
sbutalla  680.85 GB   612        2.00 TB     500000      34.04%  ok/ok
```

Figure 12: A successful execution of the `eosquota` command.

See the webpage “[Using EOS at the LPC](#)” for more information.

4 The Hidden Valley of Particle Physics

The dark fermion model is an interpretation of the “Hidden Valley” or “Hidden Sector” theory of dark matter (DM). In terms of group theory, this takes on the form of the addition of a dark unitary gauge group to the Standard Model (SM) gauge group:

$$SU(3) \otimes SU(2) \otimes U(1) \otimes U(1)_D,$$

where $U(1)_D$ is the dark, unitary gauge group. This extension modifies the SM Lagrangian, introducing:

$$\mathcal{L} \subset -\frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}Z_{D\mu\nu}Z_D^{\mu\nu} + \frac{\epsilon}{2\cos\theta}Z_{D\mu\nu}B^{\mu\nu} + \frac{1}{2}m_{Z_D}^2 Z_D^\mu Z_{D\mu}. \quad (1)$$

Here, $B_{\mu\nu}$ is the SM field strength tensor corresponding to the $U(1)$ field, with the usual definition of $B_{\mu\nu} = \partial_\mu B_\nu - \partial_\nu B_\mu$, and $Z_{D\mu\nu}$ representing the field strength tensor of the new $U(1)_D$ symmetry. The mixing term, which is the tensor product between the SM $U(1)$ and the dark $U(1)_D$ field is scaled by the ratio of the kinetic mixing coefficient $\epsilon = \sqrt{\alpha_D/\alpha}$, where α_D is the dark analog of the electroweak fine structure constant ($\alpha = 137^{-1}$), and where θ is the Weinberg mixing angle.

A spectrum of particles—analogueous to their SM counterparts—are postulated to exist within the Hidden Valley. For example, the model discussed below postulates the existence of *dark fermions*, which, as the name suggests, have similar properties to SM fermions but are charged under the dark hypercharge from the $U(1)_D$ group. Consequent to this theory, spontaneous symmetry breaking (which is analogueous to the Higgs mechanism), produces a dark vector boson, named the dark Z or Z_D . Due to kinetic mixing, the dark Z acts as a bridge to the dark sector, interacting with both SM and DM particles. This is where the name “vector portal” comes from.

4.1 The Dark Fermion Model

From the spectrum of particles contained within the hidden valley, we can consider the reaction

$$pp \rightarrow Z_D \rightarrow f_{D1} \bar{f}_{D1} \rightarrow f_{D2} \bar{f}_{D2} \mu^+ \mu^- \mu^+ \mu^-,$$

as illustrated in the Feynman diagram in Fig. 13.

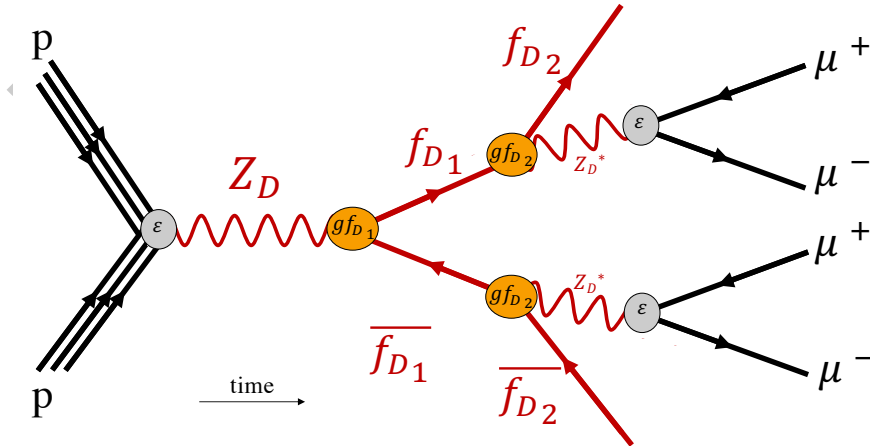


Figure 13: The Feynman diagram for the process $pp \rightarrow Z_D \rightarrow f_{D1} \bar{f}_{D1} \rightarrow f_{D2} \bar{f}_{D2} \mu^+ \mu^- \mu^+ \mu^-$.

The kinetic mixing previously discussed produces a dark Z from a proton-proton collision, which subsequently decays into two heavy, dark fermions f_{D1} . These first dark fermions couple to Z_D with strength $g_{f_{D1}}$. These dark fermions then decay into the lightest dark fermion, f_{D2} , and an *off-shell* Z_D . The dark Z bosons then kinetically mix back to SM Z bosons, which subsequently decay to dimuons.

5 Building a model with FeynRules

The first step of the simulation process is to build the model you wish to simulate. This can be done in a variety of ways; a popular one because this step has already been performed for the f_D model (and for many other models; see the [UFO model repository](#) on the MadGraph website), I will provide an abbreviated version that will familiarize the reader with the mechanics of constructing a model and generating the UFO files. Ultimately, the UFO files output at the end of this process will be used as input to the event generator (MadGraph5 in our case).

Before getting started, clone the companion repository [analysis_tutorial](#):

```
$ git clone https://github.com/sbutalla/analysis_tutorial.git
```

5.1 The FeynRules file

As mentioned in Section 2, a model is typically built using the Mathematica package FeynRules. Mirroring the subdivisions of the FeynRules file itself, this section will be divided into several subsections:

5.1.1 File header

5.1.2 Model information

5.1.3 Gauge groups

5.1.4 Change log

5.1.5 Interaction order

5.1.6 Fields and gauge bosons

5.1.7 Parameters

5.1.8 Lagrangian

In the proceeding subsections I'll reproduce the particular section of the FeynRules file ([FD_KM_new.fr](#)), accompanied by a discussion of its most salient and less-than-self-explanatory parts. The other, accompanying file in the [feynrules/Models/DMsimp](#), named [SM_new.fr](#), contains all of the Standard Model definitions.

5.1.1 File header

This one is pretty self-explanatory—it's just a non-functional header that contains the model and author names:

```
(* ***** *)
(* ***** *)
(* ***** FeynRules model file: Simplified DM models ***** *)
(* ***** s-channel spin-1 mediator ***** *)
(* ***** Author: A. Martini, K. Mawatari mehdi ***** *)
(* ***** J. Wang, C.Zhang (EW) ***** *)
(* ***** B. Zaldivar (lepton) ***** *)
(* ***** B. Fuks (monotop) ***** *)
(* ***** *)
(* ***** *)
```

5.1.2 Model information

As the name implies, this section contains all of the information regarding the model, including the model name, stored in the variable `M$ModelName` variable (which is essentially a dictionary), as well an “electronic signature” [1] of the model, stored in `M$Information`.

```
(* ***** *)
(* ***** Information ***** *)
(* ***** *)
M$ModelName = "DMsimp_s_spin1";

M$Information = {
  Authors      -> {"A. Martini, K. Mawatari, J. Wang, C. Zhang, B. Zaldivar, B. Fuks"},
  Institutions -> {"Universite catholique de Louvain, LPSC Grenoble,
                  Johnnas Gutenberg University of Mainz, Brookhaven National Laboratory,
                  LAPTh Ancecy, LPTHE / UPMC"},
  Emails       -> {"kentarou.mawatari@lpsc.in2p3.fr", "cenzhang@bnl.gov"},
  URLs         -> "http://feynrules.irmp.ucl.ac.be/wiki/DMsimp/",
  References    -> {"O. Mattelaer et al., arXiv:1508.00564",
                  "M. Backovic et al., arXiv:1508.05327",
                  "M. Neubert et al., arXiv:1509.05785",
                  "J. Andrea et al., arXiv:1106.6199"},
  Version      -> "2.1",
  Date         -> "2016.10.27"
};
```

5.1.3 Gauge group definitions

This section defines any gauge groups *additional to the SM gauge groups* that are featured in the model.

```
(* ***** *)
(* ***** Gauge groups ***** *)
(* ***** *)
M$GaugeGroups = {
  (* Add U(1)D for new gauge boson *)
  U1X == { Abelian -> True, GaugeBoson -> X, Charge -> QX, CouplingConstant -> ee}
};
```

Here we define `M$GaugeGroups` “list” (which is really a dictionary), in which we add the definition of the $U(1)_D$ group, which includes if it is or isn’t Abelian, the gauge boson produced from symmetry (note that it is defined as X and not ZD), the charge of the boson, and the coupling constant.

5.1.4 Change log

This one is also self-explanatory; it just lists the release dates and versions of any changes that have been made to this particular file.

```
(* ***** *)
(* ***** Change log ***** *)
(* ***** *)

(* 2015.08.24 v1.0 - release version. *)
(* 2016.06.02 v1.1 - changed the PDG numbers for parton-shower programs. *)
(* 2016.09.23 v2.0 - included the gamma-lepton interactions (B.Zaldivar) *)
(* 2016.10.27 v2.1 - included the monotonop interactions (B. Fuks) *)
```

5.1.5 Interaction order definitions

This “section” defines the interaction order limit and also the hierarchy of interactions (i.e., the strength/importance of the various interactions) during simulation. Here we define the dark matter vectoral interaction, DMV, to be of the same precedence as quantum electrodynamic (QED) couplings (see `M$InteractionOrderHierarchy`). We also specifically allow the DMV interaction up to the limit of $n = 4$ during perturbative expansion.

```
(**** Setting for interaction order (as e.g. used by MadGraph 5) *****)
M$InteractionOrderLimit = {
  {DMV, 4}
};

M$InteractionOrderHierarchy = {
  {QCD, 1}, {DMV, 2}, {QED, 2}
};
```

5.1.6 Field and gauge boson definitions

Because we are working in the parlance and framework of quantum field theory (QFT), we define need to define the fields and the gauge bosons. This is essentially two separate sections combined into one, the various entries of which are stored in the list `M$ClassesDescription`. For the fields portion, we have:

```
(* ***** *)
(* ***** Fields ***** *)
(* ***** *)
M$ClassesDescription = {

F[7] == { ClassName -> Fd11,
          SelfConjugate -> True,
          Mass -> {MFd11, 5},
          Width -> {WFd1, 1.00000000e-05},
          PDG -> 5000521,
          TeX -> Subscript[F,d11],
          FullName -> "Dirac DM" },

F[9] == { ClassName -> Fd21,
          SelfConjugate -> True,
          Mass -> {MFd21, 2},
          Width -> 0,
          PDG -> 5000523,
          TeX -> Subscript[F,d21],
          FullName -> "Dirac DM" },
```

Here we are defining the fields of the f_{D_1} and f_{D_2} particles, which are represented by the classes `Fd11` and `Fd21` in FeynRules, respectively. We also indicate whether the particles are self-conjugate (i.e., identical to their antiparticle, like a neutron), their mass, which is in the form of a list and includes the variable representation and their mass, e.g., `Mass -> MFd21, 2`, their decay width, given in the same format (note that the width for f_{D_2} is zero, meaning that it is a stable particle and will not decay). The Particle Data Group (PDG)³ identifier is also provided; the identifiers for the f_{D_1} and f_{D_2} have been chosen such that they do not conflict with any of the existing PDG identifiers; for example, the d , u , and s quarks are 1, 2, and 3, respectively, and the charmed baryon Ξ_c^{*+} is labelled with 4324. The TeX description (which is used when rendering Feynman diagrams) is also provided, and the full name of the particle type is also listed.

³The PDG is like the National Institute of Standards and Technology (NIST) of particle physics. It is essentially a repository for all of the measurements of the particle masses, decay widths, etc. See the latest PDG Particle Physics Booklet [here](#).

For the Gauge bosons, we define most of the same information, including whether it is a physical or unphysical boson, the `PropagatorType`, which is how the propagator line will be represented when printing the Feynman diagrams, as well as if an arrow should be displayed on the propagator line in the Feynman diagram (`PropagatorArrow`). Some of the bosons defined below also contain definitions for rotation in the mass eigenbasis. Here, we provide how the field should transform during a rotation. The `[mu_]` and `[mu]` are the indices (upper and lower, respectively) of the fields, `A`, `Z`, and `Zd` are the SM photon, SM Z , and the Z_D , respectively. The variables `cw` and `sw` are the cosine and sine of the Weinberg angle in the electroweak (EW) theory, respectively.

```
(***** Gauge Bosons *****)
V[22] == { ClassName -> Zd,
          SelfConjugate -> True,
          Indices -> {},
          Mass -> {MZd, 125},
          Width -> {WZd, 1e-05},
          PropagatorLabel -> "Zd",
          PropagatorType -> Sine,
          PropagatorArrow -> None,
          PDG -> 1023,
          FullName -> "Zd" },

V[210] == { ClassName -> Bd,
           SelfConjugate -> True,
           Unphysical -> True,
           Indices -> {},
           Mass -> 0,
           Width -> 0,
           Definitions -> {Bd[mu_] :> cw A[mu] -sw ca Z[mu] + sw sa Zd[mu]}},

V[220] == {ClassName -> Xp,
           SelfConjugate -> True,
           Unphysical -> True,
           Indices -> {},
           Mass -> 0,
           Width -> 0,
           Definitions -> {Xp[mu_] :> sa Z[mu] + ca Zd[mu]}},

(* Gauge bosons: Q = -1 *)

V[61] == {ClassName -> X,
          SelfConjugate -> True,
          Definitions -> {X[mu_] -> Eps Eta Xp[mu]},
          Indices -> {},
          Mass -> 0,
          Unphysical -> True}

};
```

5.1.7 Parameter definitions

Here, the various parameters of the model are defined in the `M$Parameters` list. This includes the couplings between the dark matter particles, the coupling of the f_{D_1} to the f_{D_2} , etc.

```
(* ***** *)
(* ***** Parameters ***** *)
(* ***** *)
M$Parameters = {

  gVFd11Fd21 == {
    ParameterType -> External,
    InteractionOrder -> {DMV, 1},
    BlockName -> DMINPUTS,
    TeX -> Subscript[g,Fd11Fd21],
    Description -> "Fd11 Fd21 coupling",
    Value -> 1.42E-01 },
  gAFd11Fd21 == {
    ParameterType -> External,
    InteractionOrder -> {DMV, 1},
    BlockName -> DMINPUTS,
    TeX -> Subscript[g,Fd11Fd21],
    Description -> "Fd11 Fd21 coupling",
    Value -> -1.42E-01 },

  gVFd1 == {
    ParameterType -> External,
    InteractionOrder -> {DMV, 1},
    BlockName -> DMINPUTS,
    TeX -> Subscript[g,VFd1],
    Description -> "Fd1-ZD vector coupling",
    Value -> -1.42E-01 },

  gAFd1 == {
    ParameterType -> External,
    InteractionOrder -> {DMV, 1},
    BlockName -> DMINPUTS,
    TeX -> Subscript[g,AFd1],
    Description -> "Fd1-ZD axial-vector coupling",
    Value -> 1.42E-01 },

  MZO == {
    ParameterType -> Internal,
    Value -> MZ,
    Description -> "Z mass before mixing"},

  MX =={
    ParameterType -> Internal,
    Value -> MZd,
    Description -> "X mass before mixing"},

  CapitalDeltaZ =={
    ParameterType -> Internal,
    Value -> MX^2/MZO^2,
    ParameterName -> DZ,
    Description -> "Ratio of scales"},

  Thetaa == {
    TeX -> Subscript[\[Theta], \[Alpha]],
    ParameterType -> Internal,
    Value -> ArcTan[-2 sw Eta/(1-sw^2 Eta^2 -CapitalDeltaZ)]/2,
    ParameterName -> alp,
    Description -> "Mixing in the weak sector"},
```

```

sa == {
  TeX -> Subscript[s, \[Alpha]],
  ParameterType -> Internal,
  Value -> Sin[Thetaa],
  Description -> "Sine of alp"},

ca == {
  TeX -> Subscript[c, \[Alpha]],
  ParameterType -> Internal,
  Value -> Cos[Thetaa],
  Description -> "Cosine of alp"},

Eta == {
  ParameterType -> External,
  BlockName -> HIDDEN,
  ParameterName -> eta,
  Value -> 0.01,
  Description -> "U(1)X - U(1)Y mixing parameter"},

Eps == {
  ParameterType -> Internal,
  Value -> (Sqrt[1+4 Eta^2] - 1)/2/Eta,
  ParameterName -> eps,
  Description -> "kinetic mixing parameter"}
};

```

Much the same information is provided here as in the field and gauge boson definitions section, with a few extra parameters. The most noticeable difference is that of the `ParameterType` variable, which can be set to either `External` or `Internal`. An external parameter is one that we physically set, either because we have that information from empirical study, or because it is a free parameter in our model. An internal parameter is one that is dependent upon others (either internal or external). For example, in the context of the f_D model, we define the kinetic mixing coefficient, ε , as an *internal* parameter, defining is such that it depends on the mixing strength between the $U(1)_Y$ and $U(1)_X$ fields [SM $U(1)$ and dark $U(1)$], which we denote as η :

$$\varepsilon = \frac{-1 + \sqrt{1 + 4\eta^2}}{2\eta}$$

5.1.8 Lagrangian definition

Finally, we have the Lagrangian definition. Here, we define each interaction Lagrangian separately for clarity, and then form a composite Lagrangian when all of those definitions are complete.

```

(* ***** *)
(* ***** Lagrangian ***** *)
(* ***** *)

LU1 := 1/4 (del[B[nu], mu] - del[B[mu], nu])^2 - 1/4 (del[X[nu], mu] - del[X[mu], nu])^2
      + Eps/2 (del[X[nu], mu] - del[X[mu], nu]) (del[B[nu], mu] - del[B[mu], nu]);

L1X := Fd11bar.Ga[mu].(gVFD1 + gAFd1 Ga[5]).Fd11 Zd[mu]
      + HC[Fd11bar.Ga[mu].(gVFD1 + gAFd1 Ga[5]).Fd11 Zd[mu]];

LqG := gs Ga[mu, s, r] T[a, i, j] uqbar[s, f, i].uq[r, f, j] G[mu, a]
      + gs Ga[mu, s, r] T[a, i, j] dqbar[s, f, i].dq[r, f, j] G[mu, a];

L1Fd11Fd21Zd := Fd11bar.Ga[mu].(gVFD11Fd21 + gAFd11Fd21 Ga[5]).Fd21 Zd[mu]
               + HC[Fd11bar.Ga[mu].(gVFD11Fd21 + gAFd11Fd21 Ga[5]).Fd21 Zd[mu]];

```

```

LBright := -2ee/cw B[mu]/2 lbar.Ga[mu].ProjP.l      (*Y_1R=-2*)
          + 4ee/3/cw B[mu]/2 uqbar.Ga[mu].ProjP.uq  (*Y_uR=4/3*)
          - 2ee/3/cw B[mu]/2 dqbar.Ga[mu].ProjP.dq;  (*Y_dR=-2/3*)

LBleft := -ee/cw B[mu]/2 vlbar.Ga[mu].ProjM.vl     (*Y_LL=-1*)
          - ee/cw B[mu]/2 lbar.Ga[mu].ProjM.l      (*Y_LL=-1*)
          + ee/3/cw B[mu]/2 uqbar.Ga[mu].ProjM.uq  (*Y_QL=1/3*)
          + ee/3/cw B[mu]/2 dqbar.Ga[mu].ProjM.dq ; (*Y_QL=1/3*)

L1DM := L1X + L1Fd11Fd21Zd + LqG + LU1 + LBright + LBleft;

```

The first Lagrangian, LU1, defines the information provided in Equation 1, i.e., the kinetic mixing between the $U(1)$ and $U(1)_D$ fields. The next component, L1X, defines the vectoral and axial interactions between Z_D and f_{D_1} . A term is added to correct issues with quark-gluon couplings, which is represented by the Lagrangian LqG. Governing the interactions between Z_D , f_{D_1} , and f_{D_2} is the Lagrangian L1Fd11Fd21Zd. Lastly, for the weak interactions, we have the right- and left-hand components of the Lagrangian given by LBright and LBleft. The final product is then the sum of each of these parts, stored in L1DM. For more information on the various classes, syntax, and operators, see [1].

5.2 Generating the UFO files

Now that the FeynRules file is defined, we need to use it to generate the UFO files. To do this, we create a Mathematica notebook to compute the matrix elements of the model. First, though, you'll need to get a Mathematica license. Consult with Dr. Hohlmann about transferring the current license or purchasing a new one. Below, I'll review each cell of the Mathematica notebook, discussing the important lines of code and providing screenshots of successful output.

1. In the companion repository, `analysis.tutorial` and change into the `feynrules` directory:

```
$ cd feynrules
```

2. Now open the Mathematica notebook `DMsimp_easy.nb`.
3. In the first cell we set the `$FeynRulesPath` variable, which is just the path to the FeynRules workspace. We also set the current working directory and then load both the Standard Model and f_D model FeynRules file. Successful output is displayed in Fig. 14

```

$FeynRulesPath = SetDirectory["~/analysis/feynrules-current"];
SetDirectory[$FeynRulesPath <> "/Models/DMsimp"];
LoadModel["SM_new.fr", "FD_KM_new.fr"];

```

4. Next, we check the Hermiticity of the interaction Lagrangian using `CheckHermiticity`; next. If this step is successful, we then calculate the Feynman rules (i.e., forming each possible vertex). A truncated version of the successful output is provided in Fig. 15; for the rest of the output, see Appendix B.

```

CheckHermiticity[L1DM];
vertsDMsimp = FeynmanRules[L1DM, ScreenOutput -> True];

```

5. Finally, we write the UFO files using the `WriteUFO` function, the successful output of which is displayed in Fig. 16.

```
WriteUFO[L1DM];
```

6. This will output to a new directory with the name `DMsimp_s_spin1_UFO` to the `~/Models/DMsimp` directory. (Here the \sim denotes the base directory of your FeynRules workspace, **not** your home directory.)

```

H111: $FeynRulesPath = SetDirectory["~/analysis/feynrules-current"];
<< FeynRules
SetDirectory[$FeynRulesPath < "/Models/DMSimp"];
LoadModel["SM_new.fr", "FD_new.fr"];
- FeynRules -
Version: 2.3.36 (28 November 2019).
Authors: A. Alloul, N. Christensen, C. Degrande, C. Duhr, B. Fuks

Please cite:
- Comput.Phys.Commun.185:2259-2300,2014 (arXiv:1310.1921);
- Comput.Phys.Commun.188:1614-1643,2009 (arXiv:0806.4194).

http://feynrules.phys.ucl.ac.be

The FeynRules palette can be opened using the command FRPalette[].
(General) Last::normal : Nonatomic expression expected at position 1 in Last[DMINPUTS].
(General) Last::normal : Nonatomic expression expected at position 1 in Last[DMINPUTS].
(General) Last::normal : Nonatomic expression expected at position 1 in Last[DMINPUTS].
(General) Last::normal : Nonatomic expression expected at position 1 in Last[DMINPUTS].
(General) Part::partd : Part specification OrderBlock[DMINPUTS][1, 1] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[DMINPUTS][1, 1] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[DMINPUTS][1, 1] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[DMINPUTS][1, 1] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[DMINPUTS][1, 2] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[DMINPUTS][1, 2] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[DMINPUTS][1, 2] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[DMINPUTS][1, 2] is longer than depth of object.
(General) Last::normal : Nonatomic expression expected at position 1 in Last[HIDDEN].
(General) Last::normal : Nonatomic expression expected at position 1 in Last[HIDDEN].
(General) Last::normal : Nonatomic expression expected at position 1 in Last[HIDDEN].
(General) Last::normal : Nonatomic expression expected at position 1 in Last[HIDDEN].
(General) Part::partd : Part specification OrderBlock[HIDDEN][1, 1] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[DMINPUTS][1, 2] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[DMINPUTS][1, 2] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[DMINPUTS][1, 2] is longer than depth of object.
(General) Last::normal : Nonatomic expression expected at position 1 in Last[HIDDEN].
(General) Last::normal : Nonatomic expression expected at position 1 in Last[HIDDEN].
(General) Last::normal : Nonatomic expression expected at position 1 in Last[HIDDEN].
(General) Last::normal : Nonatomic expression expected at position 1 in Last[HIDDEN].
(General) Part::partd : Part specification OrderBlock[HIDDEN][1, 1] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[HIDDEN][1, 1] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[HIDDEN][1, 1] is longer than depth of object.
(General) Part::partd : Part specification OrderBlock[HIDDEN][1, 1] is longer than depth of object.
(General) General::stop : Further output of Part::partd will be suppressed during this calculation.
(General) General::stop : Further output of Part::partd will be suppressed during this calculation.
(General) General::stop : Further output of Part::partd will be suppressed during this calculation.
(General) Part::partw : Part -3 of PRIVATE`elemfunc[PRIVATE`tempfunc1406[Last[DMINPUTS]]] does not exist.
(General) Part::partw : Part -3 of PRIVATE`elemfunc[PRIVATE`tempfunc1406[Last[DMINPUTS]]] does not exist.
(General) Part::partw : Part -3 of PRIVATE`elemfunc[PRIVATE`tempfunc1406[Last[DMINPUTS]]] does not exist.
(General) Part::partw : Part -3 of PRIVATE`elemfunc[PRIVATE`tempfunc1406[Last[DMINPUTS]]] does not exist.
(General) Part::partw : Part -2 of PRIVATE`elemfunc[PRIVATE`tempfunc1406[Last[DMINPUTS]]] does not exist.
(General) Part::partw : Part -2 of PRIVATE`elemfunc[PRIVATE`tempfunc1406[Last[DMINPUTS]]] does not exist.
(General) Part::partw : Part -2 of PRIVATE`elemfunc[PRIVATE`tempfunc1406[Last[DMINPUTS]]] does not exist.
(General) Part::partw : Part -2 of PRIVATE`elemfunc[PRIVATE`tempfunc1406[Last[DMINPUTS]]] does not exist.
(General) Part::partw : Part -3 of PRIVATE`elemfunc[PRIVATE`tempfunc1420[Last[HIDDEN]]] does not exist.
(General) Part::partw : Part -3 of PRIVATE`elemfunc[PRIVATE`tempfunc1420[Last[HIDDEN]]] does not exist.
(General) Part::partw : Part -3 of PRIVATE`elemfunc[PRIVATE`tempfunc1420[Last[HIDDEN]]] does not exist.
(General) Part::partw : Part -3 of PRIVATE`elemfunc[PRIVATE`tempfunc1420[Last[HIDDEN]]] does not exist.
(General) General::stop : Further output of Part::partw will be suppressed during this calculation.
(General) General::stop : Further output of Part::partw will be suppressed during this calculation.
(General) General::stop : Further output of Part::partw will be suppressed during this calculation.
(General) General::stop : Further output of Part::partw will be suppressed during this calculation.
A. Martini, K. Mawatari, J. Wang, C. Zhang, B. Zaldivar, B. Fuks
Model Version: 2.1
Please cite
O. MatteLaer et al., arXiv:1508.00564
M. Backovic et al., arXiv:1508.05327
M. Neubert et al., arXiv:1509.05785
J. Andrea et al., arXiv:1106.6199
http://feynrules.irmp.ucl.ac.be/wiki/DMSimp/
For more information, type ModelInformation[].

- Loading particle classes.
- Loading gauge group classes.
- Loading parameter classes.

Model DMSimp_s_spin1 loaded.
(General) General: Further output of Part::partw will be suppressed during this calculation.
(General) Part: Part -3 of PRIVATE`elemfunc[PRIVATE`tempfunc3182[Last[HIDDEN]]] does not exist.
(General) Part: Part -2 of PRIVATE`elemfunc[PRIVATE`tempfunc3139[Last[DMINPUTS]]] does not exist.
(General) Part: Part -3 of PRIVATE`elemfunc[PRIVATE`tempfunc3139[Last[DMINPUTS]]] does not exist.
(General) General: Further output of Part::partd will be suppressed during this calculation.
(General) Part: Part specification OrderBlock[HIDDEN][1, 1] is longer than depth of object.
(General) Last: Nonatomic expression expected at position 1 in Last[HIDDEN].
(General) Part: Part specification OrderBlock[DMINPUTS][1, 2] is longer than depth of object.
(General) Part: Part specification OrderBlock[DMINPUTS][1, 1] is longer than depth of object.
(General) Last: Nonatomic expression expected at position 1 in Last[DMINPUTS].
Merging model-files...
(General) General: Further output of Part::partw will be suppressed during this calculation.

```

Figure 14: A screenshot of successful output of setting the requisite paths and loading the SM and f_D model FeynRules files.


```

In[5]:= CheckHermiticity[L1DM];
vertsDMSimp = FeynmanRules[L1DM, ScreenOutput -> True];
Checking for hermiticity by calculating the Feynman rules contained in L-HC[L].
If the lagrangian is hermitian, then the number of vertices should be zero.
Starting Feynman rule calculation.
Expanding the Lagrangian...
No vertices found.
0 vertices obtained.
The lagrangian is hermitian.
Starting Feynman rule calculation.
Expanding the Lagrangian...
Collecting the different structures that enter the vertex.
16 possible non-zero vertices have been found -> starting the computation: 16 / 16.
16 vertices obtained.
(* * * * * *)
Vertex 1
Particle 1 : Dirac , dq̄
Particle 2 : Dirac , dq
Particle 3 : Vector , G
Vertex:
i g_s γ_{s1,s2}^{μ3} δ_{f1,f2} T_{m1,m2}^{a3}
(* * * * * *)
Vertex 2
Particle 1 : Dirac , uq̄
Particle 2 : Dirac , uq
Particle 3 : Vector , G
Vertex:
i g_s γ_{s1,s2}^{μ3} δ_{f1,f2} T_{m1,m2}^{a3}
(* * * * * *)
Vertex 3
Particle 1 : Majorana , Fd11

```

Figure 15: A screenshot of successful output of running the command to check the Hermiticity of the Lagrangian, as well as the calculation of the Feynman Rules. For the remaining screenshots, see Appendix B.

```

WriteUFO[L1DM];
... Last: Nonatomic expression expected at position 1 in Last[DMINPUTS].
... Last: Nonatomic expression expected at position 1 in Last[HIDDEN].
--- Universal FeynRules Output (UFO) v 1.1 ---
... Part: Part 3 of PRIVATE`ReorderEParamEntry[External, {DMINPUTS, Last[DMINPUTS], PRIVATE`tempfunc$3139[Last[DMINPUTS]]}] does not exist.
... Part: The expression {All, NumericalValue[All]} cannot be used as a part specification.
... Part: The expression {All, False} cannot be used as a part specification.
... Part: The expression Null cannot be used as a part specification.
... General: Further output of Part::pkspec1 will be suppressed during this calculation.
Starting Feynman rule calculation.
Expanding the Lagrangian...
Collecting the different structures that enter the vertex.
16 possible non-zero vertices have been found -> starting the computation: 16 / 16.
16 vertices obtained.
Flavor expansion of the vertices: 16 / 16
... Part: Part specification OrderBlock[DMINPUTS][[1, 1]] is longer than depth of object.
... Part: Part specification OrderBlock[HIDDEN][[1, 1]] is longer than depth of object.
... Part: Part -2 of PRIVATE`elemfunc[PRIVATE`tempfunc$3139[Last[DMINPUTS]]] does not exist.
... Part: Part -2 of PRIVATE`elemfunc[PRIVATE`tempfunc$3162[Last[HIDDEN]]] does not exist.
... General: Further output of Part::partw will be suppressed during this calculation.
- Saved vertices in InterfaceRun[ 1 ].
Computing the squared matrix elements relevant for the 1->2 decays:
29 / 29
Squared matrix element compute in 1.44195 seconds.
Decay widths computed in 0.009149 seconds.
Preparing Python output.
- Splitting vertices into building blocks.
Splitting of vertices distributed over 4 kernels.
- Optimizing: 44/44 .
- Writing files.
Done!

```

Figure 16: A screenshot of successful output of the writing of the UFO files to disk.

At the conclusion of the creation of the UFO files, you should `scp` them as a compressed archive to your MadGraph5 installation. For example, to prepare and then transfer the archive to LXPLUS:

```
$ tar -xvaf DMsimp_s_spin1_UFO.tar.xz DMsimp_s_spin1_UFO
$ scp DMsimp_s_spin1_UFO.tar.xz username@lxplus.cern.ch:/afs/cern.ch/user/<first initial>/<
  username>/path/to/madgraph/models
```

(where one would remove the information contained in the “<” and “>” and replace it with the user’s information).

6 Simulating the hard-scattering events with MadGraph5

This section outlines the steps for simulating hard-scattering processes at the parton level. The information of these processes are stored in *Les Houches Event* (LHE) files (file extension `.lhe`). This file type is the new and improved version of the file format specified by the Les Houches Accord (LHA). Etymologically, the LHA file is named from an agreement made by the particle physicists at the *Physics at TeV Colliders Workshop* in 2001 in the city of Les Houches in eastern France; see Ref. [2]. This accord specified that the output from parton-level event generators should be structured in a generic format where each event inside the file consists of a block of Fortran code. The LHE file format, introduced five years later, improves upon the LHA file by organizing these Fortran blocks into a file following XML syntax [3].

Preparing your MG5 installation for simulating hard-scattering events of the f_D model is reviewed in Section 6.2. You can simulate events using two methods: (1) by using the command line interface (CLI) directly, or (2) by passing a process card file (`proc_card.dat`) to the MG5 executable. These methods are reviewed in Sections 6.3 and 6.4, respectively.

6.1 Installing MadGraph5

The installation procedure of MadGraph5 is computing resource-independent (and mostly OS-independent), whether it be your own personal machine, on LXPLUS or the LPC CAF. The following is a brief guide for installing MG5 and MadAnalysis5 on any *Nix machine.

1. Navigate to the downloads page for MG5 and download the tarball of the version of your choice. Or, if you prefer to do things from the terminal and know exactly which version you need, modify the `wget` command below:

```
$ wget https://launchpad.net/mg5amcnlo/2.0/2.9.x/+download/MG5_aMC_v2.9.6.tar.gz
```

Decompress and un-tar using

```
$ tar -xvf MG5_aMC_v2.9.6.tar.gz
```

The good thing about this package is that it comes with the MG5 binary ready to work out of the box; no need to compile from source. To launch MadGraph5, change into the base directory of the software and launch the executable using

```
$ cd MG5_aMC_v2_9_6
$ ./bin/mg5_aMC
```

While optional, it is recommended to install some ancillary packages such as `pythia8`, `zlib`, `boost`, `lhpdf6` (or `lhpdf5`), `collier`, `hepmc`, `mg5amc.py8.interface`, `ninja`, `oneloop`, `MadAnalysis5`. Specifically, to get access to the multitudinous parton distribution functions available, install **LHAPDF6**. At the very least you should download **MadAnalysis5** for various plotting/processing tasks. From within the MG5 command line interface CLI, execute:

```
MG5_aMC> install MadAnalysis
```

6.2 Setting up MG5 for simulating the f_D model

1. Install MadGraph5 using your installation procedure of choice.
2. Add the directory with the UFO files to the `models` directory:

```
$ mv fD_model.tar path/to/madgraph/models/fD_model.tar &&\
tar -xvf path/to/madgraph/models/fD_model.tar
```

3. Move the process card file (`proc_card.dat`) file to the base directory of your MG5 installation (`~/path/to/madgraph/`). The `proc_card.dat` file contains the instructions for MG5, such as what model to import, custom particle definitions, what process to generate, etc. See Appendix C for an example.
4. For a freshly generated set of UFO files using the FeynRules Mathematica package, one would generate the parameter card (`param_card.dat`) with `write_param_card.dat`, but for now use the one supplied in the archive. The `param_card.dat` contains various parameters such as physical constants (e.g., the SM couplings like the fine structure constant α , the f_{D1} and f_{D2} coupling constants $g_{f_{D1}}$ and $g_{f_{D2}}$, etc.), particle masses, and decay widths.

6.3 Using the CLI

To simulate events using the CLI, follow the steps below:

1. Launch the CLI by calling the binary directly from the base directory.

```
$ ./bin/mg5_aMC
```

2. Import the model (the name of the directory housing the UFO files):

```
MG5_aMC > import model <model_name>
```

in our case, this is

```
MG5_aMC > import model fD_model
```

3. Then define the contents of the proton:

```
MG5_aMC > define p = u u~ d d~ s s~ g
```

where the tilde indicates the anti-particle, i.e., u and $u\sim$ are the up and anti-up quarks, respectively. The g represents the gluon. This is ultimately handled by the parton distribution function (pdf), but one must provide *which* partons can be selected from the pdf. It can be useful to simulate the production process of Z_D with heavier quarks, so we can also define a proton composed of all three generations of quarks and the eight gluons:

```
MG5_aMC > define j = g u u~ d d~ c c~ s s~ b b~ t t~
```

4. Now generate the events:

```
MG5_aMC > generate j j > Zd, (Zd > FD11 FD11, (FD11 > FD21 mu+ mu-))
```

5. Output the process to a MadEvent directory:

```
MG5_aMC > output <output_dir_name> -nojpeg
```

for this example we'll name the output directory `fD_model`:

```
MG5_aMC > output fD_model -nojpeg
```

This directory is where the event generator binary and ancillary files are located.

- To generate events, first change to the output directory:

```
$ cd fD_model
```

- If desired, change the number of events in the run card (`run_card.dat`), located in the `Cards` directory in the output directory:

```
*****
# Number of events and rnd seed                *
# Warning: Do not generate more than 1M events in a single run      *
*****
10000 = nevents ! Number of unweighted events requested <-----
0     = iseed  ! rnd seed (0=assigned automatically=default))
```

In the example code above 10^4 events will be generated.

- Now, from the output directory (`~/fD_model/`), run the event generator:

```
$ ./bin/generate_events
```

The LHE file will be output to a new directory created in the `Events` directory, which is numerically labeled (`~/fD_model/Events/run_YY`). **Note that from here you can change the particle masses in the `param_card.dat` file located in the `Cards` directory. You do not need to produce a new event generator every time a parameter is modified.** The runs will be labeled numerically, like `Events/run_01`, `Events/run_02`, etc., so keep a log of the configuration of each set of events. See Section 6.5 for more information on modifying the masses of the dark particles.

6.4 Using the `proc_card.dat` and the MG5 executable

One can more efficiently create the event generator by feeding the process card directly to the MG5 binary. These steps are listed below.

- Create the event generator:

```
./bin/mg5_aMC proc_card.dat
```

The process card contains all of the commands manually entered to the CLI in the previous section. See Appendix C for an example.

- To generate events, first change to the output directory:

```
$ cd fD_model
```

- If desired, change the number of events in the run card (`run_card.dat`), located in the `Cards` directory in the output directory (i.e., `~/fD_model/Cards`):

```
*****
# Number of events and rnd seed                *
# Warning: Do not generate more than 1M events in a single run      *
*****
10000 = nevents ! Number of unweighted events requested <-----
0     = iseed  ! rnd seed (0=assigned automatically=default))
```

In the example code above 10^4 events will be generated.

- Now, from the output directory run the event generator:

```
$ ./bin/generate_events
```

The LHE file will be output to a new directory created in the `Events` directory, which is numerically labeled (`~/fd_model/Events/run_YY`). **Note that from here you can change the particle masses in the `param_card.dat` file located in the `Cards` directory. You do not need to produce a new event generator every time a parameter is modified.** The runs will be labeled numerically, (e.g., `Events/run_01`, `Events/run_02`, etc.), so keep a log of the run configuration for each set of events. See Section 6.5 for more information on modifying the masses of the dark particles.

6.5 Modifying the parameters of the model

We are interested in the physics of the f_D model at different masses of the mediator (Z_D), and both of the dark fermions (f_{D_1} and f_{D_2}). To change these values, you'll need to edit the `param_card.dat` file (the one in the `Cards/` directory!), specifically the entries for m_{Z_D} , $m_{f_{D_1}}$, and $m_{f_{D_2}}$ (marked by `<-----`):

```
#####
## INFORMATION FOR MASS
#####
Block mass
 5.040000e-03 # MD
 2.550000e-03 # MU
 3 1.010000e-01 # MS
 4 1.270000e+00 # MC
 5 4.700000e+00 # MB
 6 1.720000e+02 # MT
11 5.110000e-04 # Me
13 1.056600e-01 # MMU
15 1.777000e+00 # MTA
23 9.120000e+01 # MZ
25 1.250000e+02 # MH
1023 1.000000e+03 # MZd <-----
5000521 9.000000e+00 # MFd11 <-----
5000523 7.000000e+00 # MFd21 <-----
## Dependent parameters, given by model restrictions.
## Those values should be edited following the
## analytical expression. MG5 ignores those values
## but they are important for interfacing the output of MG5
## to external program such as Pythia.
12 0.000000e+00 # ve : 0.0
14 0.000000e+00 # vm : 0.0
16 0.000000e+00 # vt : 0.0
21 0.000000e+00 # g : 0.0
22 0.000000e+00 # a : 0.0
24 7.983998e+01 # w+ : cmath.sqrt(MZ__exp__2/2. + cmath.sqrt(MZ__exp__4/4. - (aEW*cmath.pi
 *MZ__exp__2)/(Gf*sqrt__2)))
```

Where the usual particle ID (PID) convention is used. Because these particles are unique to our model (and completely theoretical at this point), Z_D is denoted by PID 1023, f_{D_1} by 5000521, and f_{D_2} by 5000523. Remember, the masses of the particles must respect conservation of mass. So rules (1) and (2) below must be obeyed:

$$m_{Z_D} \geq 2m_{f_{D_1}} \quad (2)$$

$$m_{f_{D_1}} \geq m_{f_{D_2}} + 2m_\mu \quad (3)$$

7 Hadronization, showering, and reconstruction with CMSSW

One has two options for running CMSSW: either locally in your computing workspace on a cluster (e.g., LXPLUS, the FNAL LPC cluster, the TAMU cluster, etc.) or by submitting *jobs* where the data are processed on the grid at the various computing centers around the world. The latter is (surprisingly) the easier of the two options, and will most likely be quicker, as you have more cores available to perform computations. I will discuss each method for each reconstruction step.

I first outline the installation procedure for CMSSW on both LXPLUS and the FNAL LPC CAF in Section 7.1. Next, in Section 7.2, we'll discuss the mechanics of locally using CMSSW for reconstructing MC-simulated hard-scattering events. Next, we'll discuss the easier method, which is that of using CRAB to handle all of the computing tasks for you. We'll then discuss, in detail, the four reconstruction steps in Sections 7.4–7.7.

7.1 Installing CMSSW

7.1.1 Instructions for LXPLUS

1. To keep things organized, make a directory for all of your CMSSW versions and change to this directory:

```
$ mkdir cmssw && cd cmssw
```

2. Now, to get and build your CMSSW version, we first want to find the latest release. This is performed using SCRAM, which is essentially a package manager. See the [SCRAM Twiki page](#) for more information. The output of listing all of the CMSSW versions is piped to grep which filters out the most recent CMSSW approved (no beta versions):

```
$ scram list | grep CMSSW
```

3. Select the most recent version and use `cmsrel` to make a new directory with the software (basically like git clone):

```
$ cmsrel CMSSW_X_Y_Z
```

where X is the major release number, Y is the minor release, and Z is an update.

4. Export the correct SCRAM architecture environment variable (you can also optionally define this in your `.bash_profile`; remember, this is only valid for the **current** shell session):

```
$ export SCRAM_ARCH=s1c6_amd64_gcc700
```

5. Change directories to the source code (`src`) directory of the latest CMSSW release:

```
$ cd CMSSW_X_Y_Z/src/
```

and source the environment configuration for CMS analysis by invoking the `cmsenv` command:

```
$ cmsenv
```

6. Clone the `MuJetAnalysis` code from GitHub:

```
$ git clone https://github.com/cms-tamu/MuJetAnalysis.git
```

7. Clean any logs, temporary files, or previously compiled files:

```
$ scram b clean
```

and then compile the software using the `build` command of SCRAM:

```
$ scram b -j 6
```

8. Make a new directory under `src` and copy the Pythia configuration file to it:

```
$ mkdir Configuration && mkdir Configuration/GenProduction\  
  && mkdir Configuration/GenProduction/python  
$ mv ~/path/to/Pythia8HadronizerFilter_13TeV_cfi.py Configuration/GenProduction/Python
```

9. Clean and build again:

```
$ scram b clean  
$ scram b -j 6
```

With CMSSW installed and your LHE files in-hand, you're now ready to more realistically simulate the entirety of the event, including background radiation, and the responses of the detector systems in the CMS experiment. This can be done in two ways: running CMSSW locally, and submitting CRAB jobs to economize the process.

7.2 The mechanics of running CMSSW locally

Running CMSSW locally consists of producing a configuration file (if one doesn't already exist), and then passing that configuration file to CMSSW for processing. We'll discuss the former case in Section 7.2.1, and the latter in Section 7.2.2.

7.2.1 The CMSSW configuration file

To generate the configuration file, one uses the `cmsDriver.py` script. As an example, see the generic command (modeled after the command for reconstruction step 1) below:

```
$ cmsDriver.py\  
<path/to/pythia_config_file.py>\  
--fileout file:<OUT_FILE.root>\  
-mc\  
--eventcontent <EVENTCONTENT>\  
--datatier <DATATIER>\  
--conditions <CONDITIONS>\  
--beamspace <BEAMSPOT_CONFIG>\  
--step <RECO_STEP>\  
--geometry <DETECTOR_GEOMETRY>\  
--era Run2_2017\  
--filetype <FILE_EXTENSION>\  
--filein file:<INPUT_FILE.lhe>\  
--python_filename <OUTPUT_CONFIG_FILE_NAME.py>\  
--no_exec\  
-n <NUM_EVENTS>
```

[Note that items between the `<` and `>` will be replaced with their appropriate values, (sans the `<` and `>`, of course).]

Executing this command produces the generically named configuration file `<OUTPUT_CONFIG_FILE_NAME.py>`. This Python file contains all of the various “configurations” necessary to run the GEN-SIM step for CMSSW. These configurations are really just importing some CMSSW Python-specific modules/classes/functions, loading different configurations, setting attributes of classes, etc. Because this is still probably abstract, I include a working example below:


```
$ cmsDriver.py\  
Configuration/GenProduction/python/Pythia8HadronizerFilter_13TeV_cfi.py\  
--fileout file:out_sim.root\  
--mc\  
--eventcontent RAWSIM\  
--datatier GEN-SIM\  
--conditions 93X_mc2017_realistic_v3\  
--beamspot Realistic25ns13TeVEarly2017Collision\  
--step GEN,SIM\  
--geometry DB:Extended\  
--era Run2_2017\  
--filetype LHE\  
--filein file:unweighted_events.lhe.gz\  
--python_filename gen_sim_cfg.py\  
--no_exec\  
-n 10000
```

See Appendix D for the configuration file produced from this command.

7.2.2 Using the configuration file as input to CMSSW

Following the generation of the configuration file, a one-line command is used to run CMSSW:

```
$ cmsRun <CONFIG_FILE.py>
```

So for the previous example given, one would run

```
$ cmsRun gen_sim_cfg.py
```

Once the command is finished running, your ROOT file will be placed in either your current directory or the directory specified in the output file name option when running the `cmsDriver.py` command.

One last note: you can, of course, modify an existing configuration file with updated file names, eras, etc. It is usually much easier to select this option. However, if you are adding any new options, it is best to run the `cmsDriver.py` command again.

7.3 The mechanics of running CMSSW with CRAB

1. While using CMSSW you can **write** the various output files to your EOS space. However, **you cannot read files stored here in any capacity**, whether that be through CRAB or if you're locally running CMSSW. One either has to use a storage location that is connected to the grid, or transfer the files from your storage space and then input them locally.
2. If you are running CMSSW locally (i.e., you're not connected to the grid), you won't be able to include pileup, or any other files included in the configuration files that start with the database identifier (`db: /`), as a connection to the grid is necessary to access the central database.
3. In previous years you could increase your AFS workspace disk quota to a TB (if requested). As of (roughly) May 2021, there is now a 10 GB limit, which is severely encumbering⁴. So, if you are running CMSSW from LXPLUS, it is highly recommended that you change your storage location to the FNAL LPC cluster or another suitable Tier 2 or 3 (T2 and T3, respectively) location.
4. If you have access to the LPC, the best solution to these quandaries is to run the first reconstruction step in your AFS space, and then run all of the other steps from your EOS workspace.

5. **Keep a log of your CRAB submissions and status reports.** The information contained therein will, in the case of the submission output, contain the task name and also a command that you can copy and paste, which will produce the status of the CRAB job. The latter, if completed, will give you the location and dataset name. This information is critical when you need to later retrieve the output files from a job, and also when you need to use them for input to the next CRAB reconstruction step.

Before proceeding any farther, **you will need Tier 2 (T2) or Tier 3 (T3) user or group storage space in order to use CRAB.** With CERN's computing environment, your EOS space will be sufficient (remember that **the EOS is not connected to the grid**). Following the directions enumerated in Section 3.4.2, you can use your T3 storage at the LPC.

In most cases CRAB is the easier (and more efficient) option. All that is required is one CMSSW configuration file (discussed in Section 7.2) and one CRAB configuration file for each step. I'll discuss the CRAB configuration file in Section 7.3.1 The CRAB configuration file is written in Python and is significantly shorter and simpler than the CMSSW configuration file; see the example below, which is a configuration file for the first reconstruction step:

7.3.1 The CRAB configuration file

```
from WMCore.Configuration import Configuration

config = Configuration()

config.section_('General')
config.General.workArea          = 'fd_2018_step1_cmssw_10_2_18'
config.General.requestName      = 'step1_cmssw_10_2_18'

config.section_('JobType')
config.JobType.psetName         = 'GEN-SIM_step1_cfg_2018_edited.py'
config.JobType.pluginName       = 'PrivateMC'
config.JobType.inputFiles       = ['unweighted_events_noDecayWidth.lhe']
config.JobType.allowUndistributedCMSSW = True

config.section_('Data')
config.Data.outputDatasetTag    = 'cmssw_10_2_18_mc_gen-sim_mzd_125_15_2018'
config.Data.publication         = True
config.Data.publishDBS         = 'phys03'
config.Data.unitsPerJob        = 1000
config.Data.splitting           = 'EventBased'
config.Data.outputPrimaryDataset = 'fd_model'
config.Data.totalUnits          = 10000
config.Data.outLFNDirBase      = '/store/user/sbutalla/'

config.section_('Site')
config.Site.storageSite        = 'T3_US_FNALLPC'
```

Below are explanations and some general rules and conventions for the various parameters assigned in the CRAB configuration file. These annotations are organized by the various “sections” of the configuration file.

General

- `config.General.workArea` (str): The name of the directory where all CRAB metadata and log files will be saved.
- `config.General.requestName` (str): The name of the directory that will be created for the particular process that you are currently running. For example, if you're running step 1 for many different masses, you can set the work area name to `step1_work_area`, and for each mass point you need to run step 1 on, you can just change the request name; say `fd_mzd_150_mfd1_5`, `fd_mzd_150_mfd1_15`, `fd_mzd_150_mfd1_20`, etc. So for a single mass point, the CRAB logs will be located in, say, `step1_work_area/fd_mzd_150_mfd1_20`. **If your CRAB submission fails, and you cannot re-submit successfully, you must change, at minimum, the request name.** Although it will not produce an error, if you name the request and work area the same name as a previously submitted and/or failed process, the submission will fail. One would have to “officially” delete the published dataset hosted in the CERN database to regain the freedom of naming the request name.

Job Type

- `config.JobType.psetName` (str): The parameter set; the name of the CMSSW configuration file.
- `config.JobType.pluginName` (str): The type of production (e.g., private, central, etc.), The input files are collated into a list (with standard Python syntax) and are assigned to
- `config.JobType.inputFiles` (list; str): A list (in standard Python syntax), of input files. I.e.,

```
config.JobType.inputFiles = ['file0.lhe', 'file1.lhe']
```

It is important to note that even if you have only a single file, you **must** enter it as a list.

- `config.JobType.allowUndistributedCMSSW` (bool): Used to determine whether or not to perform reconstruction in the event that you're using a CMSSW version that might not be available at all sites. The advantage of setting this to **False** is, of course, that it maximizes the number of sites that can be used to process your job, resulting in quicker computation times (much time is spent waiting for resources). If **True**, the number of potential computing sites narrows.

Data

- `config.Data.outputDatasetTag` (str): A database identifier for a project. For instance, while the work area and request name will change, this should stay constant. All samples for the f_D model, say, will all be under the output dataset tag '`fd_model1`'. Generally it is named such that it contains identifiable information about the underlying process [what version of CMSSW you're using, the masses of the important particles (such as m_{Z_D} and $m_{f_{D1}}$, in the context of the f_D model)]. To publish your dataset (which is good as a
- `config.Data.publication` (bool): To publish in the database, set to **True**. The advantage to publishing your dataset is the ease with which other collaborators can access it. (It's also good for book-keeping purposes.)
- `config.Data.publishDBS` (str): The name of the database in which you'd like to publish your dataset. Currently, this can only be `phys03`.
- `config.Data.unitsPerJob` (int): The number of events to process in a set. This controls the number of output ROOT files. For example, if you set this to 1000, and you have 10^4 events total, you'll end up with ten output ROOT files.
- `config.Data.splitting` (str): How to split your data during processing. Depending on your circumstances, you might want to split the jobs based on luminosity (i.e., if you have different luminosities in the same input dataset), `LumiBased`, or based on each file in your input list, `EventBased`.
- `config.Data.outLFNDirBase` (str): The logical file name (LFN) of your base directory for file storage. For most circumstances, this will be something like `/store/user/<username>/`. One can also add a subdirectory after the `<username>/`

Site

- The storage location of your files. If one has LPC CAF access and read/write permissions, this should be set to 'T3_US_FNALLPC'. For storage on your LXPLUS EOS, this would read

The subsequent configuration files contain a few more parameters that should be highlighted. See the example below, which is a step 2 configuration file:

```
from WMCore.Configuration import Configuration

config = Configuration()

config.section_("General")
config.General.requestName      = 'sD_MZD_125_MSD_40_2018_step2_cmssw_10_2_18'
config.General.workArea        = 'sD_MZD_125_MSD_40_step2_cmssw_10_2_18_2018'

config.section_("JobType")
config.JobType.pluginName      = 'ANALYSIS'
config.JobType.psetName        = 'GEN-SIM-RAW_step2_cfg_2018_original.py'
config.JobType.maxMemoryMB     = 16000
config.JobType.numCores       = 8
config.JobType.allowUndistributedCMSSW = True

config.section_("Data")
config.Data.inputDBS           = 'phys03'
config.Data.inputDataset      = '/sD_model/sbutalla-
cmssw_10_2_18_mc_gen_lhe_sD_2018_MZD_125_MSD_40-64d5b857b62366c21ce655e0d424c60b/USER'
config.Data.splitting         = 'FileBased'
config.Data.unitsPerJob      = 200
config.Data.publication      = True
config.Data.publishDBS       = 'phys03'
config.Data.outLFNDirBase    = '/store/user/sbutalla/'

config.section_("Site")
config.Site.storageSite      = 'T3_US_FNALLPC'
```

As before, I'll organize the new parameters into their respective sections and explain them below:

JobType

- `config.JobType.maxMemoryMB` (int): The maximum memory quota for jobs. Here, we set it to a factor of eight times the default value.
- `config.JobType.numCores` (int): The number of cores to use for computation. Note that we increase the maximum memory quota because we are using more than the default number of cores (one). Increasing the number of cores significantly expedites the computation.

Data

- `config.Data.inputDataset` (str): The name of the path to the input dataset. For CRAB storage locations this will (usually) be of the form `/outputPrimaryDataset/name-of-dataset/USER`. Note that this is only valid for datasets published in the database bookkeeping system (DBS).

The configuration files for Steps 3 and 4 are not much different than for Step 2. However, should you have additional questions or want to read more on parameters in the CRAB configuration file, consult the excellent [CRAB configuration file Twiki page](#).

Before continuing our journey to the first reconstruction step, we need to review some basic CRAB commands.

7.3.2 Basic commands

After you run the CRAB job, the data will be sent to your specified storage site. How do you locate and access these files? How can you see which files (and also how many) are contained within each dataset? These questions will all be answered below:

Finding your dataset

For this operation, I refer you back to Rule 5 presented in Section 7.3. It is much easier to find a dataset and the files contained therein if you already have the exact dataset name and the associated details! Otherwise, you'll have to look through the data aggregation system (DAS) [browser interface](#) for your dataset. You can also perform a basic query of the DAS via the command line. To accomplish this, simply connect to the grid (discussed in Section 3.3.3), and then use the `dasgoclient` command to perform a basic query:

```
$ dasgoclient --query="file dataset=</outputPrimaryDataset/*/*> instance=prod/phys03"
```

Where the `</outputPrimaryDataset/*/*>` is the same as that which is defined in the CRAB config file. This will produce a list of all datasets under that output primary dataset tag. A working example, using the command

```
$ dasgoclient --query="dataset=/sD_model/sbutalla-crab_sD_MZD_*//* instance=prod/phys03"
```

will produce the output displayed in Fig. 17.

```
[sbutalla@lxplus7104 .globus]$ dasgoclient --query="dataset=/sD_model/sbutalla-crab_sD_MZD_*//* instance=prod/phys03"
/sD_model/sbutalla-crab_sD_MZD_110_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_110_MSD_11-1_2018_step3_cmssw_10_2_18_new_trial1-821eb7170ea52d287e431833f6c780e1/USER
/sD_model/sbutalla-crab_sD_MZD_110_MSD_8-9_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_125_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_125_MSD_11-1_2018_step3_cmssw_10_2_18_new_trial1-821eb7170ea52d287e431833f6c780e1/USER
/sD_model/sbutalla-crab_sD_MZD_125_MSD_40_2018_step2_cmssw_10_2_18-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_125_MSD_40_2018_step3_cmssw_10_2_18-821eb7170ea52d287e431833f6c780e1/USER
/sD_model/sbutalla-crab_sD_MZD_125_MSD_8-9_2018_step1_cmssw_10_2_18_new_trial1-821eb7170ea52d287e431833f6c780e1/USER
/sD_model/sbutalla-crab_sD_MZD_125_MSD_8-9_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_130_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_130_MSD_8-9_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_140_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_140_MSD_8-9_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_150_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_150_MSD_11-1_2018_step3_cmssw_10_2_18_new_trial1-821eb7170ea52d287e431833f6c780e1/USER
/sD_model/sbutalla-crab_sD_MZD_150_MSD_8-9_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_160_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_160_MSD_11-1_2018_step3_cmssw_10_2_18_new_trial1-821eb7170ea52d287e431833f6c780e1/USER
/sD_model/sbutalla-crab_sD_MZD_160_MSD_8-9_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_170_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial12-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_180_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_190_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_190_MSD_8-9_2018_step2_cmssw_10_2_18_new_trial13-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_200_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_200_MSD_11-1_2018_step3_cmssw_10_2_18_new_trial1-821eb7170ea52d287e431833f6c780e1/USER
/sD_model/sbutalla-crab_sD_MZD_200_MSD_8-9_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_200_MSD_8-9_2018_step3_cmssw_10_2_18_new_trial1-821eb7170ea52d287e431833f6c780e1/USER
/sD_model/sbutalla-crab_sD_MZD_85_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_85_MSD_8-9_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_88_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_88_MSD_8-9_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_95_MSD_11-1_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_95_MSD_11-1_2018_step3_cmssw_10_2_18_new_trial1-821eb7170ea52d287e431833f6c780e1/USER
/sD_model/sbutalla-crab_sD_MZD_95_MSD_8-9_2018_step2_cmssw_10_2_18_new_trial1-561697a7e1ccc674784d5e0d3e6ef789/USER
/sD_model/sbutalla-crab_sD_MZD_95_MSD_8-9_2018_step3_cmssw_10_2_18_new_trial1-821eb7170ea52d287e431833f6c780e1/USER
```

Figure 17: Successful output of using `dasgoclient` to list all datasets matching a certain pattern.

Listing all of the files within a dataset

Once the name of the dataset is known, one can then list the individual files in a dataset by adding the keyword `file` to the query. For example, the command

```
$ dasgoclient --query="file dataset=/sD_model/sbutalla-
cmssw_10_2_18_mc_gen_lhe_sD_2018_MZD_200_MSD_8-9_new_trial1-64
d5b857b62366c21ce655e0d424c60b/USER instance=prod/phys03"
```

will produce the output given in Fig. 18.


```
[sbutalla@lxplus7104 .globus]$ dasgoclient --query="file dataset=/sD_model/sbutalla-cmssw_10_2_18_mc_gen_lhe_sd_2018_MZD_200_MSD_8-9_new_trial1-64d5b857b62366c21ce65e0d424c60b/USER instance=prod/phys03"
/store/user/sbutalla/sD_model/cmssw_10_2_18_mc_gen_lhe_sd_2018_MZD_200_MSD_8-9_new_trial1/230322_180847/0000/GEN-SIM_4.root
/store/user/sbutalla/sD_model/cmssw_10_2_18_mc_gen_lhe_sd_2018_MZD_200_MSD_8-9_new_trial1/230322_180847/0000/GEN-SIM_7.root
/store/user/sbutalla/sD_model/cmssw_10_2_18_mc_gen_lhe_sd_2018_MZD_200_MSD_8-9_new_trial1/230322_180847/0000/GEN-SIM_8.root
/store/user/sbutalla/sD_model/cmssw_10_2_18_mc_gen_lhe_sd_2018_MZD_200_MSD_8-9_new_trial1/230322_180847/0000/GEN-SIM_10.root
/store/user/sbutalla/sD_model/cmssw_10_2_18_mc_gen_lhe_sd_2018_MZD_200_MSD_8-9_new_trial1/230322_180847/0000/GEN-SIM_1.root
/store/user/sbutalla/sD_model/cmssw_10_2_18_mc_gen_lhe_sd_2018_MZD_200_MSD_8-9_new_trial1/230322_180847/0000/GEN-SIM_6.root
/store/user/sbutalla/sD_model/cmssw_10_2_18_mc_gen_lhe_sd_2018_MZD_200_MSD_8-9_new_trial1/230322_180847/0000/GEN-SIM_5.root
/store/user/sbutalla/sD_model/cmssw_10_2_18_mc_gen_lhe_sd_2018_MZD_200_MSD_8-9_new_trial1/230322_180847/0000/GEN-SIM_2.root
/store/user/sbutalla/sD_model/cmssw_10_2_18_mc_gen_lhe_sd_2018_MZD_200_MSD_8-9_new_trial1/230322_180847/0000/GEN-SIM_9.root
/store/user/sbutalla/sD_model/cmssw_10_2_18_mc_gen_lhe_sd_2018_MZD_200_MSD_8-9_new_trial1/230322_180847/0000/GEN-SIM_3.root
```

Figure 18: Successful output of using `dasgoclient` to list all of the files in a dataset.

List `dasgoclient` examples

The aforementioned example commands should cover most of what would be needed in an analysis. However, if you do need more examples for more technical DAS queries, run the command below:

```
$ dasgoclient -examples
```

Transfer files from your EOS storage to your workspace

Transferring files is relatively easy, given you have the full path to the files you need, and the physical storage location (i.e., the FNAL LPC CAF, CERNBox/personal EOS, etc.). The service that provides this task is the Any Data, Anytime, Anywhere (AAA), which is a CMS-specific implementation of `XRootD`. With the necessary ingredients, one can use the `XRootD` copy (`xrdcp`) command like

```
$ xrdcp root://cmsxrootd.storage.site//store/user/username//path/to/root.file path/to/store/file
```

A complete, working example will resemble:

```
$ xrdcp root://cmsxrootd.fnal.gov//store/user/sbutalla/sD_model/cmssw_10_2_18_mc_gen_lhe_sd_2018_MZD_200_MSD_8-9_new_trial1/230322_180847/0000/GEN-SIM_3.root .
```

The above command will completely copy, just like `scp`, the file to the desired location. For more information, refer to [Chapter 5](#) of the CMSSW Offline Workbook and the [Wisconsin CMS Tier-2 FAQ webpage](#) for more information.

7.3.3 CRAB configuration files

Table 1 below contains the CMSSW configuration files, which are hyperlinked to the GitHub repository.

Table 1: The CMSSW configuration files, hyperlinked to the GitHub repository.

Step	Description	Year	
		2017	2018
1	GEN-SIM	GEN-SIM_step1_cfg.py	GEN-SIM_step1_cfg.py
2	GEN-SIM-RAW	GEN-SIM-RAW_step2_cfg.py	GEN-SIM-RAW_step2_cfg.py
3	DIGI I	RECO_step3_cfg.py	RECO_step3_cfg.py
4	DIGI II	MiniAOD_step4_cfg.py	MiniAOD_step4_cfg.py

7.4 Step 1: GEN-SIM

The first step, GEN-SIM, uses the hard-scattering data in the LHE file as input. Hadronization and showering (e.g., for simulating QCD background radiation, any processes that result in quarks in the final state, which

will cause jets, etc.), is provided by Pythia. A ROOT file (or several ROOT files) will be output from this step, which are then used for step 2 (reviewed in Section 7.5).

7.4.1 Running locally

- The first step is to produce the CMSSW configuration file (proceed to step 2 if you already have one and have made the appropriate modifications):

```
$ cmsDriver.py\
Configuration/GenProduction/python/Pythia8HadronizerFilter_13TeV_cfi.py\
--fileout file:<step1>.root\
--mc\
--eventcontent RAWSIM\
--datatier GEN-SIM\
--conditions 93X_mc2017_realistic_v3\
--beamspace Realistic25ns13TeVEarly2017Collision\
--step GEN,SIM\
--geometry DB:Extended\
--era Run2_2017\
--filetype LHE\
--filein file:unweighted_events.lhe.gz\
--python_filename <cmssw_step1_config>.py\
--no_exec\
-n -1
```

Here, you'll need to make the appropriate substitutions for the input and output file names, the conditions, era, etc., replacing the text within the angled brackets < and >, based on your specific analysis. After running this command, the output file <step1>.root will be produced in your current working directory.

Alternatively, refer to my companion repository [analysis tutorial](#) for the pre-made CMSSW configuration file for 2017 and 2018. If running locally, you'll need to update the file name within the file appropriately.

- Next, we use cmsRun to perform this reconstruction step:

```
$ cmsRun gen_sim_cfg.py
```

- The output file (<step1>.root) will be produced and stored in the directory where this command is executed from.

7.4.2 Running with CRAB

Submitting a job

1. First, make sure the configuration file `crabConfStep1.py` is located in the `src` directory of your CMSSW installation (e.g., `/path/to/CMSSW/src`). Modify the CRAB configuration file with the appropriate information [work area name, request name, input file(s), etc.].
2. Now, launch the CRAB job using the configuration file `crabConfStep1.py`:

```
$ crab submit crabConfStep1.py
```

3. For a successful submission, the output produced will resemble the following:

```
[sbutalla@lxplus732 src]$ crab submit -c crabConfStep1_2018_temp.py
Will use CRAB configuration file crabConfStep1_2018_temp.py
Importing CMSSW configuration GEN-SIM_step1_cfg_2018_edited.py
Finished importing CMSSW configuration GEN-SIM_step1_cfg_2018_edited.py
Sending the request to the server at cmsweb.cern.ch
Success: Your task has been delivered to the prod CRAB3 server.
Task name: 230322_180847:sbutalla_crab_sD_MZD_200_MSD_8-9
         _2018_step1_cmssw_10_2_18_new_trial1
Project dir: sD_model_2018_step1_cmssw_10_2_18_new_trial1/crab_sD_MZD_200_MSD_8-9
         _2018_step1_cmssw_10_2_18_new_trial1
Please use 'crab status -d sD_model_2018_step1_cmssw_10_2_18_new_trial1/
         crab_sD_MZD_200_MSD_8-9_2018_step1_cmssw_10_2_18_new_trial1' to check how the
         submission process proceeds.
Log file is /afs/cern.ch/user/s/sbutalla/analysis/CMSSW/CMSSW_10_2_18/src/
         sD_model_2018_step1_cmssw_10_2_18_new_trial1/crab_sD_MZD_200_MSD_8-9
         _2018_step1_cmssw_10_2_18_new_trial1/crab.log
```

4. **Copy this information to a log**; the most important piece of information here is of course the `crab status` command. To do this automatically, use the function I wrote here:

```
function crabsubmit () {
  crab submit -c "${1}" 2>&1 | tee -a crab_step1.log
}
```

This will append the output to a single log file for all of your step 1 submissions.

Getting the output file information

1. Check the status of the process using `crab status`:

```
$ crab status -d <workArea>/<requestName>/
```

2. If the process is completed, you will see output similar to that below:

```
[sbutalla@lxplus764 src]$ crab status -d sD_model_2018_step1_cmssw_10_2_18_new_trial1/
         crab_sD_MZD_190_MSD_8-9_2018_step1_cmssw_10_2_18_new_trial1
CRAB project directory: /afs/cern.ch/user/s/sbutalla/analysis/CMSSW/CMSSW_10_2_18/src
         /sD_model_2018_step1_cmssw_10_2_18_new_trial1/crab_sD_MZD_190_MSD_8-9
         _2018_step1_cmssw_10_2_18_new_trial1
Task name: 230323_172605:sbutalla_crab_sD_MZD_190_MSD_8-9
         _2018_step1_cmssw_10_2_18_new_trial1
Grid scheduler - Task Worker: crab3@vocms0198.cern.ch - crab-prod-tw01
Status on the CRAB server: SUBMITTED
Task URL to use for HELP: https://cmsweb.cern.ch/crabserver/ui/task/230323_172605%3
         Asbutalla_crab_sD_MZD_190_MSD_8-9_2018_step1_cmssw_10_2_18_new_trial1
Dashboard monitoring URL: https://monit-grafana.cern.ch/d/cmsTMDetail/cms-task-
         monitoring-task-view?orgId=11&var-user=sbutalla&var-task=230323_172605%3
         Asbutalla_crab_sD_MZD_190_MSD_8-9_2018_step1_cmssw_10_2_18_new_trial1&from=
         1679588765000&to=now
Status on the scheduler: COMPLETED

Jobs status: finished 100.0% (10/10)

Publication status of 1 dataset(s): new 100.0% (10/10)
(from CRAB internal bookkeeping in transferdb)

Output dataset: /sD_model/sbutalla-cmssw_10_2_18_mc_gen_lhe_sD_2018_MZD_190_MSD_8-9
         _new_trial1-64d5b857b62366c21ce655e0d424c60b/USER
Output dataset DAS URL: https://cmsweb.cern.ch/das/request?input=%2FsD_model%2
         Fsbutalla-cmssw_10_2_18_mc_gen_lhe_sD_2018_MZD_190_MSD_8-9_new_trial1-64
         d5b857b62366c21ce655e0d424c60b%2FUSER&instance=prod%2Fphys03
```



```

Summary of run jobs:
* Memory: 880MB min, 903MB max, 869MB ave
* Runtime: 3:37:42 min, 6:18:24 max, 4:30:04 ave
* CPU eff: 86% min, 98% max, 95% ave
* Waste: 0:50:53 (2% of total)

Log file is /afs/cern.ch/user/s/sbutalla/analysis/CMSSW/CMSSW_10_2_18/src/
sD_model_2018_step1_cmssw_10_2_18_new_trial1/crab_sD_MZD_190_MSD_8-9
_2018_step1_cmssw_10_2_18_new_trial1/crab.log

```

3. Add this output to your log file. Be sure to record the output dataset name, which takes the form
`<outputPrimaryDataset>/<username>-<requestName>-<requestID>/USER.`

In the `crab status` example output above, the dataset is

```
/sD_model/sbutalla-cmssw_10_2_18_mc_gen_lhe_sD_2018_MZD_190_MSD_8-9_new_trial1-64d5b857b62366c21ce655e0d424c60b/USER
```

This dataset will be used as the input dataset for [Step 2](#).

7.5 Step 2: GEN-SIM-RAW

In this reconstruction step pre-simulated pile-up stored in the CMS database is overlaid on each event. Preliminary detector simulation with GEANT4 is also performed, including signals from the Level 1 trigger (L1) and the High Level Trigger (HLT).

7.5.1 Running locally

1. The following `cmsDriver` command is an example for the GEN-SIM-RAW step of reconstruction. Note, again, that you'll need to update the command below with the appropriate file names and any other additional settings you'll need.

```

$ cmsDriver.py \
--filein file:<step1>.root \
--fileout file:<step2>.root \
--pileup_input "dbs://Neutrino_E-10_gun/RunIISummer17PrePremix-
PUAutumn18_102X_upgrade2018_realistic_v15-v1/GEN-SIM-DIGI-RAW" \
--mc \
--eventcontent PREMIXRAW \
--datatier GEN-SIM-RAW \
--conditions 102X_upgrade2018_realistic_v15 \
--step DIGI,DATAMIX,L1,DIGI2RAW,HLT:@relval2018 \
--procModifiers premix_stage2 \
--geometry DB:Extended \
--datamix PreMix \
--era Run2_2018 \
--python_filename <cmssw_step2_config>.py \
--no_exec \
-n -1

```

2. The output Python file will then be used as input to the `cmsRun` command:

```
$ cmsRun <cmssw_step2_config>.py
```

7.5.2 Running with CRAB

1. Update the the information in the step 2 CRAB configuration file, `crabConfStep2.py`, with the appropriate information. Be sure to add the output dataset from [Step 1](#) (obtained using the `crab status` command).

7.7.2 Running with CRAB

Follow the instructions in Section 7.4.2, replacing the step 1 CRAB configuration file with `crabConfStep4.py`. To submit, execute:

```
$ crab submit -c crabConfStep3.py
```

8 n -Tuple production with MuJetAnalysis

To be completed

References

- [1] A. Alloul, et al., “FeynRules 2.0 – A complete toolbox for tree-level phenomenology,” 2014, [arXiv:1310.1921](#).
- [2] E. Boos, et al., “Generic User Process Interface for Event Generators,” 2001, [arXiv:hep-ph/0109068](#).
- [3] J. Alwall, et al., “A standard format for Les Houches Event Files,” 2006, [arXiv:hep-ph/0609017](#).
- [4] J. M. Campbell, et al., “Event Generators for High-Energy Physics Experiments,” 2022, [arXiv:2203.11110](#).

A List of acronyms

Table 2: List of acronyms

AOD	A nalysis O bject D ata
CAF	C omputing A nalysis F acility
CMS	C ompact M uon S olenoid
CMSSW	C ompact M uon S olenoid S oftware
CRAB	C MS R emote A nalysis B uilder
DM	D ark M atter
DAS	D ata A ggregation S ystem
EM	E lectromagnetic
FNAL	F ermilab N ational A ccelerator L aboratory
HEP	H igh E nergy P hysics
HLT	H igh L evel T rigger
LFN	L ogical F ile N ame
LHC	L arge H adron C ollider
LHE	L es H ouches E vent
LPC	L HC P hysics C enter
L1	L evel 1 (trigger)
MC	M onte- C arlo
QCD	Q uantum C hromodynamic
SM	S tandard M odel
T1	T ier 1
T2	T ier 2
VO	V irtual O rganization
VOMS	V irtual O rganization M embership S ervice

B Successful output of the vertices calculation using FeynRules

Below are the remaining screenshots of the vertex calculations:

```

Vertex 3
Particle 1 : Majorana , Fd11
Particle 2 : Majorana , Fd11
Particle 3 : Vector , Zd
Vertex:
-i gAffd1 (PRIVATE`CGa[5].\gamma^3 s1,s2 + \gamma^3.PRIVATE`CGa[5]s1,s2 - \gamma^3.5s1,s2 - 5.\gamma^3 s1,s2)
(* * * * * *)
Vertex 4
Particle 1 : Majorana , Fd11
Particle 2 : Majorana , Fd21
Particle 3 : Vector , Zd
Vertex:
i gFd1Fd21 \gamma^3.5s1,s2 + i gFd1Fd21 5.\gamma^3 s1,s2
(* * * * * *)
Vertex 5
Particle 1 : Dirac , dq
Particle 2 : Dirac , dq
Particle 3 : Vector , A
Vertex:
\frac{1}{6} i e \delta_{m1,m2} \delta_{f1,f2} \gamma^3.P_{-s1,s2} - \frac{1}{3} i e \delta_{m1,m2} \delta_{f1,f2} \gamma^3.P_{+s1,s2}
(* * * * * *)
Vertex 6
Particle 1 : Dirac , dq
Particle 2 : Dirac , dq
Particle 3 : Vector , Z
Vertex:
\frac{i e \text{Eta } s_0 \delta_{m1,m2} \delta_{f1,f2} \gamma^3.P_{-s1,s2}}{6 c_w} - \frac{i c_0 e s_w \delta_{m1,m2} \delta_{f1,f2} \gamma^3.P_{-s1,s2}}{6 c_w} - \frac{i e \text{Eta } s_0 \delta_{m1,m2} \delta_{f1,f2} \gamma^3.P_{+s1,s2}}{3 c_w} + \frac{i c_0 e s_w \delta_{m1,m2} \delta_{f1,f2} \gamma^3.P_{+s1,s2}}{3 c_w}
(* * * * * *)
Vertex 7
Particle 1 : Dirac , dq
Particle 2 : Dirac , dq
Particle 3 : Vector , Zd
Vertex:
\frac{i c_0 e \text{Eta } \delta_{m1,m2} \delta_{f1,f2} \gamma^3.P_{-s1,s2}}{6 c_w} + \frac{i e s_0 s_w \delta_{m1,m2} \delta_{f1,f2} \gamma^3.P_{-s1,s2}}{6 c_w} - \frac{i c_0 e \text{Eta } \delta_{m1,m2} \delta_{f1,f2} \gamma^3.P_{+s1,s2}}{3 c_w} - \frac{i e s_0 s_w \delta_{m1,m2} \delta_{f1,f2} \gamma^3.P_{+s1,s2}}{3 c_w}
(* * * * * *)
Vertex 8
Particle 1 : Dirac , l
Particle 2 : Dirac , l
Particle 3 : Vector , A
Vertex:
-\frac{1}{2} i e \delta_{f1,f2} \gamma^3.P_{-s1,s2} - i e \delta_{f1,f2} \gamma^3.P_{+s1,s2}
(* * * * * *)
Vertex 9
Particle 1 : Dirac , l
Particle 2 : Dirac , l
Particle 3 : Vector , Z
Vertex:
\frac{i e \text{Eta } s_0 \delta_{f1,f2} \gamma^3.P_{-s1,s2}}{2 c_w} + \frac{i c_0 e s_w \delta_{f1,f2} \gamma^3.P_{-s1,s2}}{2 c_w} - \frac{i e \text{Eta } s_0 \delta_{f1,f2} \gamma^3.P_{+s1,s2}}{c_w} + \frac{i c_0 e s_w \delta_{f1,f2} \gamma^3.P_{+s1,s2}}{c_w}
(* * * * * *)
Vertex 10
Particle 1 : Dirac , l
Particle 2 : Dirac , l
Particle 3 : Vector , Zd
Vertex:
\frac{i c_0 e \text{Eta } \delta_{f1,f2} \gamma^3.P_{-s1,s2}}{2 c_w} - \frac{i e s_0 s_w \delta_{f1,f2} \gamma^3.P_{-s1,s2}}{2 c_w} - \frac{i c_0 e \text{Eta } \delta_{f1,f2} \gamma^3.P_{+s1,s2}}{c_w} - \frac{i e s_0 s_w \delta_{f1,f2} \gamma^3.P_{+s1,s2}}{c_w}

```

Vertex 7
 Particle 1 : Dirac , $\bar{d}q$
 Particle 2 : Dirac , dq
 Particle 3 : Vector , Zd
 Vertex:

$$\frac{i c_u e \text{Eta} \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2}}{6 c_w} + \frac{i e s_u s_w \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2}}{6 c_w} - \frac{i c_u e \text{Eta} \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{+s1, s2}}{3 c_w} - \frac{i e s_u s_w \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{+s1, s2}}{3 c_w}$$
 (* * * * *)
 Vertex 8
 Particle 1 : Dirac , \bar{l}
 Particle 2 : Dirac , l
 Particle 3 : Vector , A
 Vertex:

$$-\frac{1}{2} i e \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2} - i e \delta_{f1, f2} \gamma^{i3} \cdot P_{+s1, s2}$$
 (* * * * *)
 Vertex 9
 Particle 1 : Dirac , \bar{l}
 Particle 2 : Dirac , l
 Particle 3 : Vector , Z
 Vertex:

$$-\frac{i e \text{Eta} s_u \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2}}{2 c_w} + \frac{i c_u e s_w \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2}}{2 c_w} - \frac{i e \text{Eta} s_u \delta_{f1, f2} \gamma^{i3} \cdot P_{+s1, s2}}{c_w} + \frac{i c_u e s_w \delta_{f1, f2} \gamma^{i3} \cdot P_{+s1, s2}}{c_w}$$
 (* * * * *)
 Vertex 10
 Particle 1 : Dirac , \bar{l}
 Particle 2 : Dirac , l
 Particle 3 : Vector , Zd
 Vertex:

$$\frac{i c_u e \text{Eta} \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2}}{2 c_w} - \frac{i e s_u s_w \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2}}{2 c_w} - \frac{i c_u e \text{Eta} \delta_{f1, f2} \gamma^{i3} \cdot P_{+s1, s2}}{c_w} - \frac{i e s_u s_w \delta_{f1, f2} \gamma^{i3} \cdot P_{+s1, s2}}{c_w}$$
 Vertex 11
 Particle 1 : Dirac , $\bar{u}q$
 Particle 2 : Dirac , uq
 Particle 3 : Vector , A
 Vertex:

$$\frac{1}{6} i e \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2} + \frac{2}{3} i e \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{+s1, s2}$$
 (* * * * *)
 Vertex 12
 Particle 1 : Dirac , $\bar{u}q$
 Particle 2 : Dirac , uq
 Particle 3 : Vector , Z
 Vertex:

$$\frac{i e \text{Eta} s_u \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2}}{6 c_w} - \frac{i c_u e s_w \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2}}{6 c_w} + \frac{2 i e \text{Eta} s_u \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{+s1, s2}}{3 c_w} - \frac{2 i c_u e s_w \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{+s1, s2}}{3 c_w}$$
 (* * * * *)
 Vertex 13
 Particle 1 : Dirac , $\bar{u}q$
 Particle 2 : Dirac , uq
 Particle 3 : Vector , Zd
 Vertex:

$$\frac{i c_u e \text{Eta} \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2}}{6 c_w} + \frac{i e s_u s_w \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2}}{6 c_w} + \frac{2 i c_u e \text{Eta} \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{+s1, s2}}{3 c_w} + \frac{2 i e s_u s_w \delta_{n1, m2} \delta_{f1, f2} \gamma^{i3} \cdot P_{+s1, s2}}{3 c_w}$$
 (* * * * *)
 Vertex 14
 Particle 1 : Dirac , $\bar{\nu}l$
 Particle 2 : Dirac , νl
 Particle 3 : Vector , A
 Vertex:

$$-\frac{1}{2} i e \delta_{f1, f2} \gamma^{i3} \cdot P_{-s1, s2}$$

D Example of a generated CMSSW configuration file

```

# Auto generated configuration file
# using:
# Revision: 1.19
# Source: /local/repos/CMSSW/CMSSW/Configuration/Applications/python/ConfigBuilder.py,v
# with command line options: Configuration/GenProduction/python/HIG-RunIIFall17wmLHEGS
-02961-fragment.py --python_filename HIG-RunIIFall17wmLHEGS-02961_1_cfg.py --
eventcontent RAWSIM,LHE --customise Configuration/DataProcessing/Utils.addMonitoring --
datatier GEN-SIM,LHE --fileout file:HIG-RunIIFall17wmLHEGS-02961.root --conditions 93
X_mc2017_realistic_v3 --beamspot Realistic25ns13TeVEarly2017Collision --
customise_commands process.RandomNumberGeneratorService.externalLHEProducer.initialSeed=
int(96) --step LHE,GEN,SIM --geometry DB:Extended --era Run2_2017 --no_exec --mc -n 1000
import FWCore.ParameterSet.Config as cms

from Configuration.StandardSequences.Eras import eras

process = cms.Process('SIM',eras.Run2_2017)

# import of standard configurations
process.load('Configuration.StandardSequences.Services_cff')
process.load('SimGeneral.HepPDTESource.pythiapdt_cfi')
process.load('FWCore.MessageService.MessageLogger_cfi')
process.load('Configuration.EventContent.EventContent_cff')
process.load('SimGeneral.MixingModule.mixNoPU_cfi')
process.load('Configuration.StandardSequences.GeometryRecoDB_cff')
process.load('Configuration.StandardSequences.GeometrySimDB_cff')
process.load('Configuration.StandardSequences.MagneticField_cff')
process.load('Configuration.StandardSequences.Generator_cff')
process.load('IOMC.EventVertexGenerators.VtxSmearedRealistic25ns13TeVEarly2017Collision_cfi'
)
process.load('GeneratorInterface.Core.genFilterSummary_cff')
process.load('Configuration.StandardSequences.SimIdeal_cff')
process.load('Configuration.StandardSequences.EndOfProcess_cff')
process.load('Configuration.StandardSequences.FrontierConditions_GlobalTag_cff')

process.maxEvents = cms.untracked.PSet(
    input = cms.untracked.int32(-1)
)

# Input source
process.source = cms.Source("LHESource",
    fileNames = cms.untracked.vstring(
        ['file:XXFILEXX']
    )
)

process.options = cms.untracked.PSet(
)

# Production Info
process.configurationMetadata = cms.untracked.PSet(
    annotation = cms.untracked.string('Configuration/GenProduction/python/HIG-
RunIIFall17wmLHEGS-02961-fragment.py nevt:1000'),
    name = cms.untracked.string('Applications'),
    version = cms.untracked.string('$Revision: 1.19 $')
)

```



```
# Output definition

process.RAWSIMoutput = cms.OutputModule("PoolOutputModule",
    SelectEvents = cms.untracked.PSet(
        SelectEvents = cms.vstring('generation_step')
    ),
    compressionAlgorithm = cms.untracked.string('LZMA'),
    compressionLevel = cms.untracked.int32(9),
    dataset = cms.untracked.PSet(
        dataTier = cms.untracked.string('GEN-SIM'),
        filterName = cms.untracked.string('')
    ),
    eventAutoFlushCompressedSize = cms.untracked.int32(20971520),
    fileName = cms.untracked.string('file:HIG-RunIIFall17wmLHEGS-02961.root'),
    outputCommands = process.RAWSIMEventContent.outputCommands,
    splitLevel = cms.untracked.int32(0)
)

process.LHEoutput = cms.OutputModule("PoolOutputModule",
    dataset = cms.untracked.PSet(
        dataTier = cms.untracked.string('LHE'),
        filterName = cms.untracked.string('')
    ),
    fileName = cms.untracked.string('file:HIG-RunIIFall17wmLHEGS-02961_inLHE.root'),
    outputCommands = process.LHEEventContent.outputCommands,
    splitLevel = cms.untracked.int32(0)
)

# Additional output definition

# Other statements
process.XMLFromDBSource.label = cms.string("Extended")
process.genstepfilter.triggerConditions=cms.vstring("generation_step")
from Configuration.AlCa.GlobalTag import GlobalTag
process.GlobalTag = GlobalTag(process.GlobalTag, '93X_mc2017_realistic_v3', '')
```

```

process.generator = cms.EDFilter("Pythia8HadronizerFilter",
    PythiaParameters = cms.PSet(
        parameterSets = cms.vstring('pythia8CommonSettings',
            'pythia8CP5Settings'),
        pythia8CP5Settings = cms.vstring('Tune:pp 14',
            'Tune:ee 7',
            'MultipartonInteractions:ecmPow=0.03344',
            'PDF:pSet=20',
            'MultipartonInteractions:bProfile=2',
            'MultipartonInteractions:pT0Ref=1.41',
            'MultipartonInteractions:coreRadius=0.7634',
            'MultipartonInteractions:coreFraction=0.63',
            'ColourReconnection:range=5.176',
            'SigmaTotal:zeroAXB=off',
            'SpaceShower:alphaSOrder=2',
            'SpaceShower:alphaSValue=0.118',
            'SigmaProcess:alphaSValue=0.118',
            'SigmaProcess:alphaSOrder=2',
            'MultipartonInteractions:alphaSValue=0.118',
            'MultipartonInteractions:alphaSOrder=2',
            'TimeShower:alphaSOrder=2',
            'TimeShower:alphaSValue=0.118'),
        pythia8CommonSettings = cms.vstring('Tune:preferLHAPDF = 2',
            'Main:timesAllowErrors = 10000',
            'Check:epTolErr = 0.01',
            'Beams:setProductionScalesFromLHEF = off',
            'SLHA:keepSM = on',
            'SLHA:minMassSM = 1000.',
            'ParticleDecays:limitTau0 = on',
            'ParticleDecays:tau0Max = 10',
            'ParticleDecays:allowPhotonRadiation = on')
    ),
    comEnergy = cms.double(13000.0),
    crossSection = cms.untracked.double(0.0),
    filterEfficiency = cms.untracked.double(1.0),
    maxEventsToPrint = cms.untracked.int32(0),
    pythiaHepMCVerbosity = cms.untracked.bool(False),
    pythiaPylistVerbosity = cms.untracked.int32(1)
)

# Path and EndPath definitions
process.generation_step = cms.Path(process.pgen)
process.simulation_step = cms.Path(process.psim)
process.genfiltersummary_step = cms.EndPath(process.genFilterSummary)
process.endjob_step = cms.EndPath(process.endOfProcess)
process.RAWSIMoutput_step = cms.EndPath(process.RAWSIMoutput)

# Schedule definition
process.schedule = cms.Schedule(process.generation_step,process.genfiltersummary_step,
    process.simulation_step,process.endjob_step,process.RAWSIMoutput_step)
from PhysicsTools.PatAlgos.tools.helpers import associatePatAlgosToolsTask
associatePatAlgosToolsTask(process)
# filter all path with the production filter sequence
for path in process.paths:
    if path in ['lhe_step']: continue
    getattr(process,path)._seq = process.generator * getattr(process,path)._seq

# customisation of the process.

# Automatic addition of the customisation function from Configuration.DataProcessing.Utils
from Configuration.DataProcessing.Utils import addMonitoring

#call to customisation function addMonitoring imported from Configuration.DataProcessing.
    Utils
process = addMonitoring(process)

```

```
# End of customisation functions  
  
# Customisation from command line  
  
# Add early deletion of temporary data products to reduce peak memory need  
from Configuration.StandardSequences.earlyDeleteSettings_cff import customiseEarlyDelete  
process = customiseEarlyDelete(process)  
# End adding early deletion
```

DRAFT