Reimplementation of CMS Search for Displaced Decays to Dijets

April 1, 2023

Contents

In the following, we go over the process we used to implement the cuts that were proposed by the Cornell paper 'Phenomenology of a Long-Lived LSP with R-Parity Violation'[1], which in turn mirrors the CMS paper 'Search for long-lived neutral particles decaying to quark-antiquark pairs in proton-proton collisions at sqrt(s) = 8 TeV'[2].

We go down the pipeline, starting with the changes to Higgs Effective Field Theory imposed in Feynrules to create our model, to the event simulation in Madgraph/Pythia8/Delphes, and finally to the implementation of the analysis in Root.

1 Feynrules and Lagrangian

We use the HEFT model given by feynrules [3] as a base, and add a heavy real scalar X with interactions

$$\mathcal{L} \subset y_{np}\bar{u_i}Xu_i + y_{np}X^2|\Phi|^2$$

that allow the X to couple to a SM-like Higgs Boson and all quark antiquark pairs except the top.

2 Madgraph, Pythia 8, and Delphes

In madgraph, we first create folders for event generation by inputting the commands (in Madgraph5_aMC_NLO, version 2.5.5):

generate p p > h > x x QED <= 1
output

which creates 5 diagrams of the form g g > h > x x or q $\tilde{q}$ > h > x x, where q $\tilde{q}$ includes first or second generation quark antiquark pairs. The restriction QED <= 1 simply ensures that there are no other interfering diagrams that are possible in the HEFT.

We then go to the output folder:

2.1 Run Card

We change the run_card.dat to have $\sqrt{s} = 8$ TeV and include trigger level cuts on jets ($PT_{jets} > 40, |\eta| < 2$).

2.2 Pythia8 Card

In the pythia8 card add 'partonlevel:mpi=off' to turn off multiparton interactions.

2.3 Param Card

The changes we make in the param card allow allow us to scan over our parameter space and add decay information for the X.

To scan over our parameter space, we change the mass of the H and X, as well as the width of the X, for each event generation.

We add decay information by adding the following width and branching ratios under the width of the X (to whom we give the pdg 5000001):

DECAY 5000001 {Width}
BR NDaughters ID1 ID2
2.00000000E-01 2 1 -1 # BR(X $->$ d d)
2.00000000E-01 2 2 -2 # BR(X $->$ u u)
2.00000000E-01 2 3 -3 # BR(X $->$ s s)
2.00000000E-01 2 4 -4 # BR(X $->$ c c)
2.00000000E-01 2 5 -5 # BR(X $->$ b b)

2.4 Event Generation

Finally, in the main folder created by madgraph, we use ./bin/generate_events to create 10000 events of the form p p > h > x x (as above) at 8 TeV. Event generation includes operation of Delphes and Pythia8, with the above edits to all used cards. In addition, we use the default Delphes card for the CMS detector.The default anti-kt, $\Delta R < .5$ algorithm that is used by Delphes is in accordance with the jet finding algorithm used by CMS and Cornell.

3 ROOT

The main program we implement in root is a C++ program. It starts by loading all events from a root file and looping through them. It implements CMS/Cornell cuts in the following manner:

3.1 Jet Cuts

The first thing we do in each event is throw out all tracks with PT < 1 GeV due to their poor reconstruction efficiency.

After this step, we look at all jets which satisfy:

- PT$_{jet} > 60$ GeV
- $|\eta_{jet}| < 2$
- Number of Close Tracks ≤ 2
- Number of Prompt Tracks ≤ 1
- Σ E$_{prompt\ tracks} \leq .09$ E$_{jet,\ total}$

Here, a close track is one whose 3D impact parameter $D_{3D} < .3$ mm. We take the 3D impact parameter given by root as $\sqrt{D_0^2 + D_z^2}$, the sum of the transverse and longitudinal impact parameters of the track in quadrature[4]. Prompt tracks are defined as those with transverse impact parameter $D_0 < .5$ mm. We note here that the transverse impact parameter of each track, D_0, used here

is different from the track impact parameter used later in the cluster section: $D_0 \neq L_{xy}^{track}$, though they can confusingly go by the same name. The latter is a projection of track momentum onto the dijet momentum, as discussed in Section 3.3. This is in agreement with the methods implemented by Cornell. In addition, we assume the pion mass for each track when doing a sum of their energies.

Each jet that satisfies the above cuts is a "good" jet. As long as there are two or more good jets, we continue, as we have one or more dijet candidates. In the fudge factor pipeline, we implement these cuts along with the dijet cuts below.

3.2 Dijet Cuts

Next, we take a look at all pairs of good jets. Each pair constitutes a dijet candidate. To reconstruct the secondary vertex of the event, we look at all tracks associated with both jets in the dijet candidate.

We cluster all tracks according to a depth first search for all connected components [5][6]. In particular, we create clusters of all tracks associated with the dijet candidate, such that each track in the network is less than 1 mm away from at least one other track in the cluster – this method is detailed in the appendix. Each cluster can now be used to reconstruct a possible secondary vertex.

Once we have these clusters, we reconstruct the secondary vertex of the dijet by finding the cluster, calculating the average position of the origins of all tracks in the cluster, and then choosing the cluster with the highest track multiplicity. We require additionally that the cluster has tracks associated with each jet and its average vertex position has a transverse displacement $L_{xy} > 2.4$ mm.

The value 2.4 mm comes from the Cornell assumption of a transverse detector resolution of .3 mm in the main part of the CMS detector, along with the CMS requirement that the secondary vertex transverse displacement is greater than 8 times its uncertainty.

If we end up with a cluster that satisfies these conditions, we use its average vertex position as the reconstructed vertex of the dijet, and continue.

3.3 Cuts on Clusters

Next, we create clusters of tracks (unrelated to the clusters above) by looking at all tracks associated with the dijet. For each track, we look at the track transverse impact parameter L_{xy}^{track}, defined as the intersection of the track trajectory with the dijet momentum projected from the primary vertex in the plane transverse to the beam axis. We find the dijet momentum by taking a vector sum of the two constituent jet momenta.

To find L_{xy}^{track}, we solve

$$(x(t), y(t))_{track} = u \cdot (P_{x,dijet}, P_{y,dijet})$$

for t and u, to find where the track trajectory intersects the dijet momentum when both are projected into the transverse plane. Then $L_{xy}^{track} = \sqrt{x(t)^2 + y(t)^2}$. Using the classic $\vec{F} = q\vec{v} \times \vec{B}$, we arrive at the two equations:

$$\frac{P_{dy}}{P_{dx}}\big[x_0 + \frac{P_{tx}}{m\gamma\omega} sin(\omega t) + \frac{P_{tz}}{m\gamma\omega}(cos(\omega t) - 1)\big] - y_0 - \frac{P_{ty}}{m\gamma} t = 0$$

and

$$L_{xy}^{track} = \big[x_0 + \frac{P_{tx}}{m\gamma\omega} sin(\omega t) + \frac{P_{tz}}{m\gamma\omega}(cos(\omega t) - 1)\big] \sqrt{1 + (\frac{P_+ dy}{P_{dx}})^2}$$

Where m is the pion mass, $\vec{P}_{t/d}$ is the track/dijet momentum, $\omega = qB/m$ and $\vec{r}_0$ is the initial position/vertex of the track. We use $B = 4$ T for the strength of the CMS magnetic field.

We solve these equations in root for L_{xy}^{track}. We now create the entirely new clusters of tracks:
Recalling that L_{xy} is the transverse displacement of the reconstructed secondary vertex, we find the value of L_0 for which the there are the most tracks with $L_0 < L_{xy}^{track} < L_0 + .15 \cdot L_{xy}$, and put all such tracks into a cluster (emulating the 'hierarchical clustering algorithm, with a size parameter which is set to 15% of the distance L_{xy}' used by CMS). We find the vector sum of the momenta of all tracks in the cluster, and if the cluster satisfies $PT_{Cluster} > 8$ GeV and $M_{cluster} > 4$ GeV, the cluster passes cuts.

3.4 H_T

Finally, we take require that the total hadronic transverse energy of the event satisfies $H_T > 325$ GeV.

4 Confidence Limits

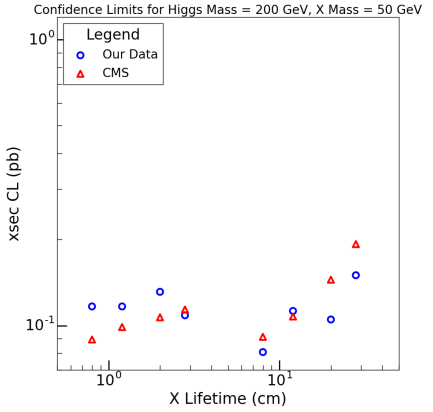
Efficiencies for each data set are given by: $\epsilon = \frac{\text{Number of Passed Events}}{10000}$, where 10000 is the total number of generated events for each point in our parameter space.

Next, we compute the confidence limits with:

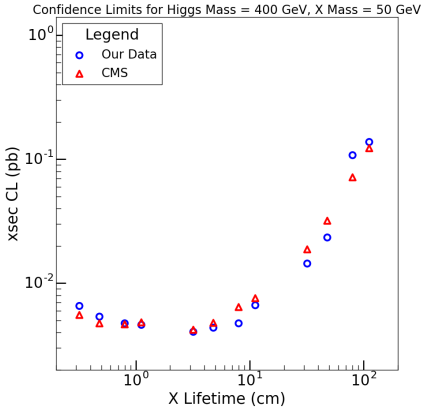
$$BR \cdot \sigma = \frac{\mathcal{N}_{sig}}{\epsilon \mathcal{L}}$$

with $\mathcal{N}_{sig} = 4.167$ by Poisson statistics, and $\mathcal{L} = 18.5$ fb^{-1}. We then plot these confidence limits in our parameter space (mass of the X, mass of the Higgs-like Scalar, and width of the X).

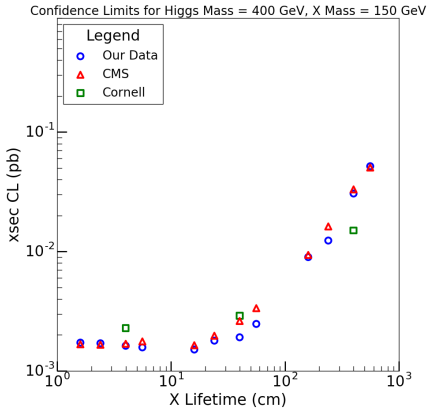
While the shapes of the this raw output agree well with the shape of CMS confidence limits, the data points are off by a constant numerical factor for each $(m_h,\ m_x)$ point. To fix this, we put in a fudge factor by hand for each point in our parameter space, after which the values of our efficiencies match the values given by CMS. It turns out that these fudge factors can be fit fairly well to a linear function in $1/(m_h + m_x)$. When we fit the fudge factors to a cubic function in $1/(m_h + m_x)$, we get even better agreement (especially above $m_h = 200$ GeV), as shown below.



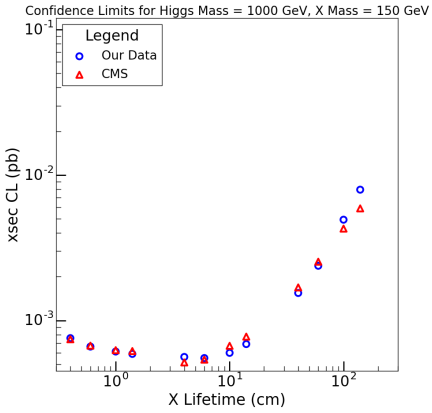
(a)



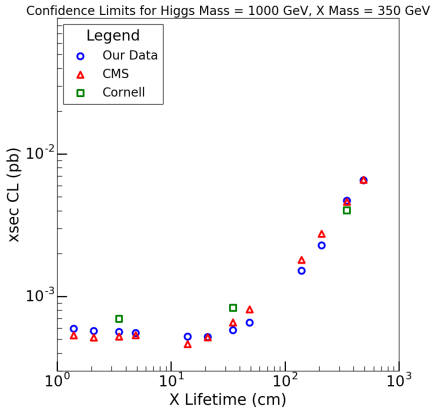
(b)



(c)



(d)



(e)

Figure 1: Plots of the efficiencies generated by the pipeline detailed above, after adding in fudge factors in $1/(m_h + m_x)$. Though agreement is more rough for the low mass points of $m_h = 200$ GeV, $m_x = 50$ GeV (in Figure 1a), we see good agreement for the higher mass points, and in most cases reported in [1], we come closer than the reimplementation by Cornell to the Confidence Limits of original CMS search.

5 Appendix: Clustering Algorithm Code

Below, we detail our algorithm used to reconstruct secondary vertices in processes with displaced decays into dijets.

In a depth first search, we use a recursive algorithm to find clusters of tracks that have the property that every track in the cluster is closer than 1 mm to at least one other track in the cluster. We use a notion of 'coloring' to keep track of which tracks we have done certain operations on:

A **green** track is one that we haven't touched with our recursive algorithm yet. All tracks start green.

A **red** track is one which has been touched by the recursive algorithm. If a track is red, the recursive algorithm will not be run on it again – in this way, we avoid any infinite looping.

We first look at all jets in an event. For each possible pair of jets, we collect all the associated tracks (as presented by Delphes, presumably in a cone of $\Delta R < .5$), and perform the following procedure for that set of tracks.

Steps

1. Our first step is to select a track from our list of all tracks. If this track is green, we continue. If it is red, we select another track and repeat.

2. We label the track we have selected in part one with an index. We call this index i. We then color the track red, so no recursion will applied on it.

3. In this step we loop over all tracks that are not the track we have selected in part one. If any track is green, and the origin of this track is within 1 mm of the origin of the track from part one, we label it with the same index, i, and color it red.

4. For every track that we selected in part three, we repeat the same procedure: we look at all the green tracks whose origins are within 1mm of each of the tracks selected in part three, color them red, and mark them with the index i.

5. Continue the procedure of part four recursively.

6. In this way, we cluster all tracks whose origins are within 1mm of at least one other track in the cluster. This cluster is then marked with the index i.

7. We select another green track and start again with the procedure from part one, this time labeling all the tracks we consider with the index i+1.

8. We now have a collection of clusters of tracks, with no overlap.

We take a look at all the clusters generated in this way, from all possible jet pairs. We select the cluster of maximal track multiplicity, and average the positions of all tracks in the cluster. This average position is what we assign to our secondary vertex.

We then perform all the appropriate cuts based on secondary vertex positon. We note that we prioritize track multiplicity in our clusters, rather than displacement. To clarify, if our cluster of maximal track multiplicity has N tracks, but does not satisfy cuts on the reconstructed secondary vertex displacement, and there is a cluster of N-1 tracks that **does** satisfy displacement cuts, we still select the cluster of N tracks when performing cuts. Our hope is that this will decrease the number of background events passing cuts.

References

[1] Phenomenology of a Long-Lived LSP with R-Parity Violation. Csaba Csáki, et. al. https://arxiv.org/pdf/1505.00784.pdf.

[2] Search for long-lived neutral particles decaying to quark-antiquark pairs in proton-proton collisions at sqrt(s) = 8 TeV. CMS Collaboration. https://arxiv.org/pdf/1411.6530v2.pdf

[3] Higgs Effective Lagrangian. Feynrules. http://feynrules.irmp.ucl.ac.be/wiki/HEL

[4] Root Tree Description. https://cp3.irmp.ucl.ac.be/projects/delphes/wiki/WorkBook/RootTreeDescription.

[5] "Fundamental Graph Algorithms, Part II". CS161. web.stanford.edu. https://web.stanford.edu/class/archive/cs/cs161/cs161.1138/lectures/02/Small02.pdf

[6] "Fundamental Graph Algorithms, Part Three". CS161. web.stanford.edu. https://web.stanford.edu/class/archive/cs/cs161/cs161.1138/lectures/03/Small03.pdf