CMS DN-20-005

2023/06/08
Archive Hash: caa5f0f-D
Archive Date: 2023/06/05

The L1 Track Trigger Upgrade: Properties, Efficiencies, and Rates for Track Stubs

Reza Goldouzian[1], Mike Hildreth[1], Alexander Morton[2], and Austin Townsend[1]

[1] University of Notre Dame (US)
[2] Vrije Universiteit Brussel (US)

Abstract

A new CMS tracker detector will be installed for the High Luminosity LHC. The new tracker will read out hits at a sufficiently high rate to allow track reconstruction in real time. This will allow the inclusion of tracking information in the Level 1 (L1) trigger system for the first time, dramatically lowering the CMS L1 trigger rate. Stubs are building blocks of the L1 trigger system. Therefore, determination and measurement of the stub properties are crucial tasks. In this note, we study various features of stubs for the CMS experiment phase II track trigger upgrade including stub rates, stub construction efficiency, stub transmission efficiency, stub bend decoding, etc. Consequently, new stub window tunes are proposed which have high stub construction efficiencies with the lowest possible rates. Additionally we present a new method of stub bend decoding which can cope with changing stub window sizes. This work is based on a full simulation of LHC events with high pileup.

1 Introduction

The goal of the High-Luminosity LHC program is to collect an integrated luminosity of 3000 fb^{-1} in about ten years of operations starting in 2028 and with a peak luminosity of 7.5×10^{34} $cm^{-2}s^{-1}$. As the bunch crossing separation will stay the same as today (25 ns), the increase of instantaneous luminosity will result in up to 200 inelastic proton-proton collisions per bunch crossing (pileup). The CMS detector needs to be substantially upgraded in order to exploit the increase in luminosity provided by the HL-LHC [1].

A major new functionality of the CMS detector for the HL-LHC is the inclusion of data from the Outer Tracker (OT) in the L1 trigger, facilitated by the readout of silicon tracking information at an unprecedented 40 MHz data rate [2]. The primary function that enables this improvement is the ability to perform local transverse momentum (p_T) measurements with the detector front-end electronics. Studies have shown that 97% (99%) of the particles created in pp interactions at 14 TeV have $p_T < 2$ GeV ($p_T < 3$ GeV). The readout rate of soft interactions can be reduced by a factor of 10 via selections on the local p_T measurements [2].

New tracking modules utilize a pair of closely spaced silicon sensors ($1.6 - 4.0$ mm) and on-detector correlation logic in order to discriminate charged particle tracks with a p_T exceeding a threshold of 2-3 GeV. Two module types are foreseen for the OT:

- PS modules: A PS module is composed of one strip sensor (two rows of 960 AC-coupled strips with dimensions 2.35 cm$\times$100 μm) and a one macro-pixel sensor (30 720 DC-coupled pixels with dimensions 1.5 mm$\times$100 μm).
- 2S modules: A 2S module is composed of two strip sensors (two rows of 1016 AC-coupled strips with dimensions 5 cm$\times$90 μm per sensor).

The OT is arranged in a barrel made of two subsystems, TBPS for the innermost 3 layers and TB2S for the outermost 3 layers, and 5(x2) endcap discs (TEDD). The OT will cover a surface of about 192 m^2 for a total of 42M strip and 170M macro pixel channels. The PS modules are deployed in the first three layers of the Outer Tracker, in the radial region of $200 - 600$ mm and in rings on disc-like structure in the endcaps up to radii 700 mm [3]. The 2S modules are deployed in the outermost three layers in Barrel (in the radial region above 600 mm) and in the large radii rings in the endcaps. The proposed layout of the Tracker is shown in Fig.1.

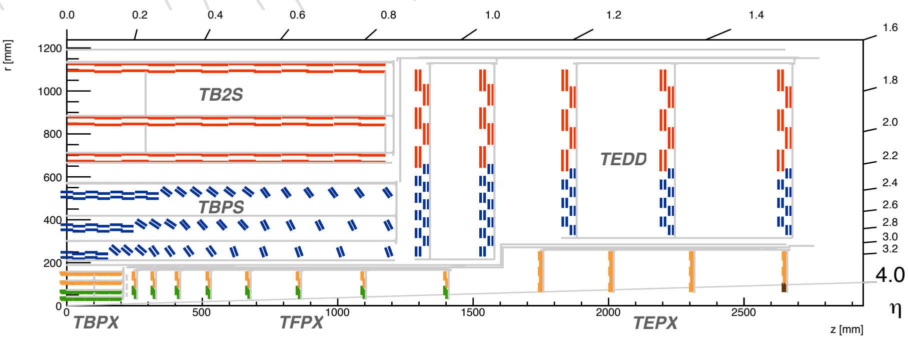


Figure 1: Sketch of one quarter of the layout of the CMS Tracker for HL-LHC in the $r - z$ view. Inner Tracker 1x2 and 2x2 readout chips modules are shown in green and yellow respectively, Outer Tracker PS and 2S modules in blue and red.

Pairs of closely spaced detector layers are inspected to see if they have pairs of clusters consistent with the passage of a high momentum particle. For each hit in the inner layer (closer to the interaction point), a window is opened on the outer layer. If a hit is found within the window, a stub is generated. Each stub consists of a position and a rough p_T measurement. Sketch of a

p_T-module showing the concept of stub is shown in Fig. 2.

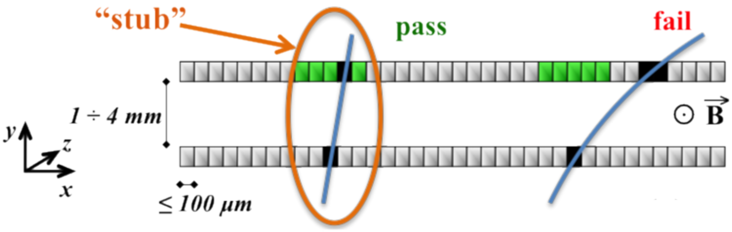


Figure 2: Sketch of a p_T-module showing the concept of stub selection.

To perform stub correlation in the 2S modules, the signals of the top and bottom sensor are routed to the same CMS binary chip (CBC), which performs the correlation logic. This is possible by folding the readout hybrids around a stiffener. In the PS modules, strip signals are processed by the strip-sensor ASIC (SSA), and macro-pixel signals by the macro-pixel ASIC (MPA). The strip data is routed from the SSA to the MPA via a folded hybrid, which then performs the cluster correlation. A detailed description of the front-end electronics can be found in [3]. One 2S front-end hybrid carries eight CBCs reading out the strips of the top and bottom sensors at one sensor end, plus the Concentrator Integrated Circuit (CIC), which serves as interface between all the CBCs of the hybrid and the readout link. The role of the CIC is mainly to aggregate and serialize the data of the readout chips and to distribute clock, trigger, and control signals to them. One PS front-end hybrid houses eight SSAs reading out the strip sensor, and the same CIC as used for 2S hybrids. All the front-end chips implement binary readout. Images of the 2S and PS modules are shown in Figure 3.

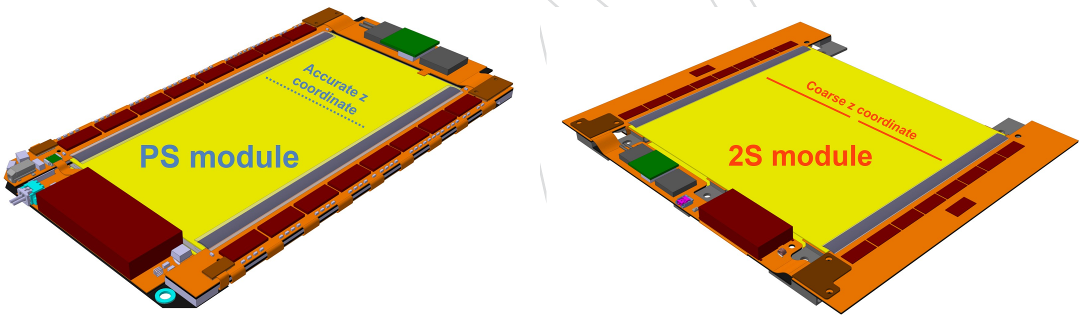


Figure 3: The 2S module (left) and PS module (right) of the Outer Tracker.

The data flow is organized in two separate paths: L1 readout (DAQ) and Trigger (TRIG). Each CBC generates high p_T stub data at bunch crossing (BX) rate. The CBCs exchange data with their neighbors to identify clusters spanning across chip boundaries. At a data transfer rate of 320 Mb/s, each CBC chip sends 1 bit of DAQ and 5 bit of TRIG data to the concentrator every 3.125 ns. This bandwidth is compatible with transferring up to three trigger stubs from each CBC every BX, and sending unsparsified readout data from each CBC pipeline up to an average L1-accept rate of 750 kHz. The transfer scheme and data formats in PS module are very similar to those used in the 2S module. The data transfer bandwidth is compatible with transferring up to five trigger stubs from each MPA every 2 BX, and sending all sparsified readout data from each MPA pipeline up to a 750 kHz L1-accept rate with negligible loss. CBCs/MPAs send out stubs to the CIC. The CIC format them into data packets containing the trigger information from eight BX and the raw data from events passing the Level 1 (L1) trigger, before transmission to the Low-power Gigabit Transceiver (lpGBT). Due to the available CIC bandwidth, additional stubs are discarded, but stubs are sorted such as to keep those with lower bending, presumably corresponding to tracks with higher p_T. The format of the CIC trigger block has been described in details in [4]. The CIC bandwidth can be adjusted to the module local occupancy by configuring the data rate (5.12 Gb/s or 10.24 Gb/s raw data rate, also referred to as 5G and 10G) and/or the forward error correction (FEC) level (FEC5 or FEC12, up to five or twelve consecutive bits in error can be corrected). This tuneability is used to increase bandwidth where expected data rates are very high, at the expense of increased power dissipation and decreased error correction level. In Figure 4, layers and rings with 5G or 10G are shown. In the current simulation code (see L1Trigger/TrackTrigger/python/TTStub_cfi.py), the following limits on number of transferred stubs are considered based on the mentioned bandwidths.

- CBClimit = cms.uint32(3), CBC chip limit (in stubs/chip/BX)
- MPAlimit = cms.uint32(5), MPA chip limit (in stubs/chip/2BX)
- SS5GCIClimit = cms.uint32(16), 2S 5G chip limit (in stubs/CIC/8BX)
- PS5GCIClimit = cms.uint32(16), PS 5G chip limit (in stubs/CIC/8BX)
- PS10GCIClimit = cms.uint32(35), PS 10G chip limit (in stubs/CIC/8BX)

If number of stubs are more than CBC/MPA/CIC transfer threshold in 1/2/8 BX, they will not transfer to the next step. The rate of stubs that are lost because of the bandwidth of CBC/MPA/CIC are called ”CBC/MPA/CIC fail” in this note.

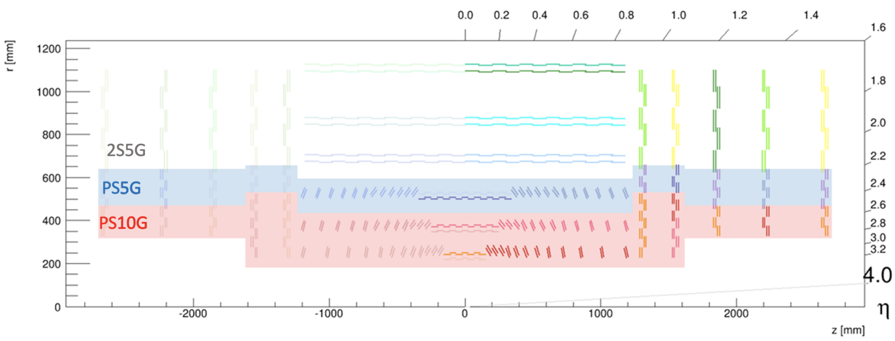


Figure 4: Modules with 5G and 10G data rate.

2 Samples

The samples used here have been produced by CMSSW 11_3_0 and are listed in table 1. The detector geometry considered is the D76 (=T21 tracker).

Table 1: List of simulated samples (see `https://twiki.cern.ch/twiki/bin/view/CMS/L1TrackMC#RelVal_MC_samples`).

sample	PU	p_T	N	DAS name
$t\bar{t}$	0	-	10k	/RelValTTbar_14TeV/CMSSW_11_3_0-113X_mcRun4_realistic_v7_2026D76noPU-v1
$t\bar{t}$	200	-	9k	/RelValTTbar_14TeV/CMSSW_11_3_0_pre6-PU_113X_mcRun4_realistic_v6_2026D76PU200-v1
Single electron	0	2-100	100k	/RelValSingleEFlatPt2To100/CMSSW_11_3_0_pre6-113X_mcRun4_realistic_v6_2026D76noPU-v2
Single electron	0	1.5-8	100k	/RelValSingleElectronFlatPt1p5To8/CMSSW_11_3_0_pre6-113X_mcRun4_realistic_v6_2026D76noPU-v1
Single muon	0	2-100	100k	/RelValSingleMuFlatPt2To100/CMSSW_11_3_0_pre6-113X_mcRun4_realistic_v6_2026D76noPU-v1
Single muon	0	1.5-8	100k	/RelValSingleMuFlatPt1p5To8/CMSSW_11_3_0_pre6-113X_mcRun_realistic_v6_2026D76noPU-v1
Displaced muon	0	2-100	100k	/RelValDisplacedMuPt2To100Dxy100/CMSSW_11_3_0_pre6-113X_mcRun4_realistic_v6_2026D76noPU-v1
Displaced muon	0	1.5-8	100k	/RelValDisplacedMuPt1p5To8Dxy100/CMSSW_11_3_0_pre6-113X_mcRun4_realistic_v6_2026D76noPU-v1

3 Stub rates

The stub rate is an important parameter for both the front-end and back-end electronics. At the front-end it is important to check that the average stub rate per module is well within the readout capacity of the detector. At the back-end, the stub multiplicity per trigger tower will impact directly the performance of the L1 tracking system, and must therefore stay under control.

Each stub is built from two clusters. Clusters are categorized into "genuine", "combinatoric" and "unknown" based on their matching condition to the tracking particles (TP). If cluster is matched to only one tracking particle or a single TP has more than 99% of the total p_{T} of all TP associated to the cluster, cluster is called genuine. If more than one TP are matched to the cluster and non of them has more than 99% of the total p_{T} of all TP, cluster is called combinatoric. If no TP is matched, cluster is called unknown. See `https://twiki.cern.ch/twiki/bin/viewauth/CMS/SLHCTrackerTriggerSWTools` for more details. Stubs that are made from these three cluster types are called "genuine", "combinatoric" and "unknown" according to the following criteria;

- If both clusters are unknown, the stub is unknown.
- If only one cluster is unknown, the stub is combinatoric.
- If both clusters are genuine, and are associated to the same (main) TP, the stub is genuine.
- If both clusters are genuine, but are associated to different (main) TP, the stub is combinatoric.
- If one cluster is combinatoric and the other is genuine/combinatoric, and they both share exactly one TP in common, then the stub is genuine. (The clusters can have other TP besides the shared one, as long as these are not shared). If instead the clusters share 0 or 2 TP in common, then the stub is combinatoric.

In Figure 5, stub rates per BX (PBX) and stub rate per BX per module (PBXPmodule) are shown for all, genuine, combinatoric, and unknown stubs for $t\bar{t}$ events with 200 PU. In Figure 6, stub rate per BX per module is shown as a function of module η, ρ and z. Two dimensional plots where each bin corresponds to a module is shown in Figures 7 and 8 for barrel layers and endcap disks respectively.

As was discussed in the previous section, stubs can be lost because of the limited bandwidth of the CBC/MPA/CIC data transfer. In Figure 9-10, fractions of all stubs that are failed by the CBC/MPA/CIC are shown. Less than 2×10^{-3} of stubs are failed by the MPAs/CBCs in the first three layers of the TB and in all TE disks. In TB2S, up to 5% of stubs are failed by the CBC which is too much. We have compared the fraction of failed stubs that are genuine to the combinatoric and unknown stubs in Figure 11. Genuine stubs that are lost in the last three layers are 1-1.5 % of total genuine stubs. In addition, most of these stubs are from low p_{T} TPs. In figure 12, fractions of genuine stubs with $p_{\mathrm{T}} > 2$ GeV (p_{T} of the matched TP) that are failed by the CBC/MPA are compared to those for all and genuine stubs. It can be seen that for genuine stubs with $p_{\mathrm{T}} > 2$ GeV, the fraction rates are around 0.2%. In addition, CIC fail rates are well bellow 0.1% in all detector regions. Therefore, the designed CBC/MPA (CIC) bandwidth lead up to 0.2% (0.1%) inefficiencies for high p_{T} genuine stubs.

It is worth mentioning that our simulation of the FE inefficiencies is not really accurate. We are summing hits from $t\bar{t}$ + 200PU over the bunch crossings, whereas we should have just pileup in the bunches previous to the ttbar event. For this high an occupancy, its probably not a big

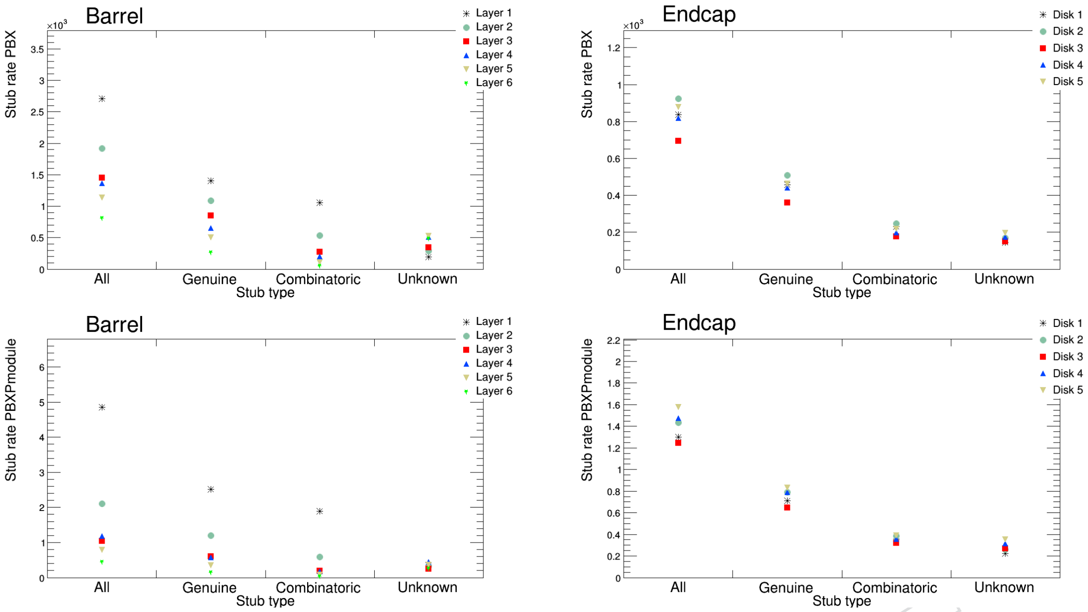


Figure 5: Stub rates per BX and stub rates per BX per module in barrel (left) and endcap (right) are shown for all, genuine, combinatoric and unknown stubs for $t\bar{t}$ events with 200 PU.

effect. But, it means these numbers are pessimistic.

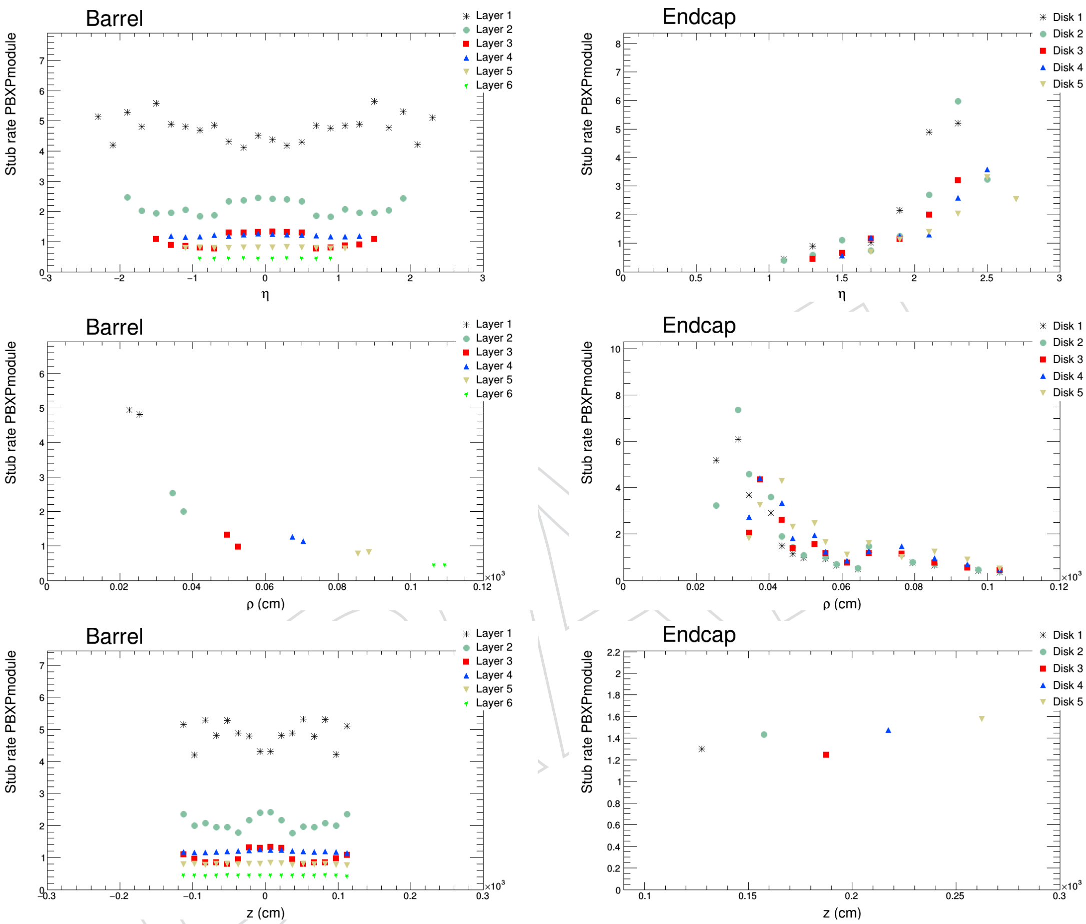


Figure 6: Stub rates per BX per module in barrel (left) and endcap (right) are shown for all stubs as a function of module η, ρ and z using $t\bar{t}$ events with 200 PU.

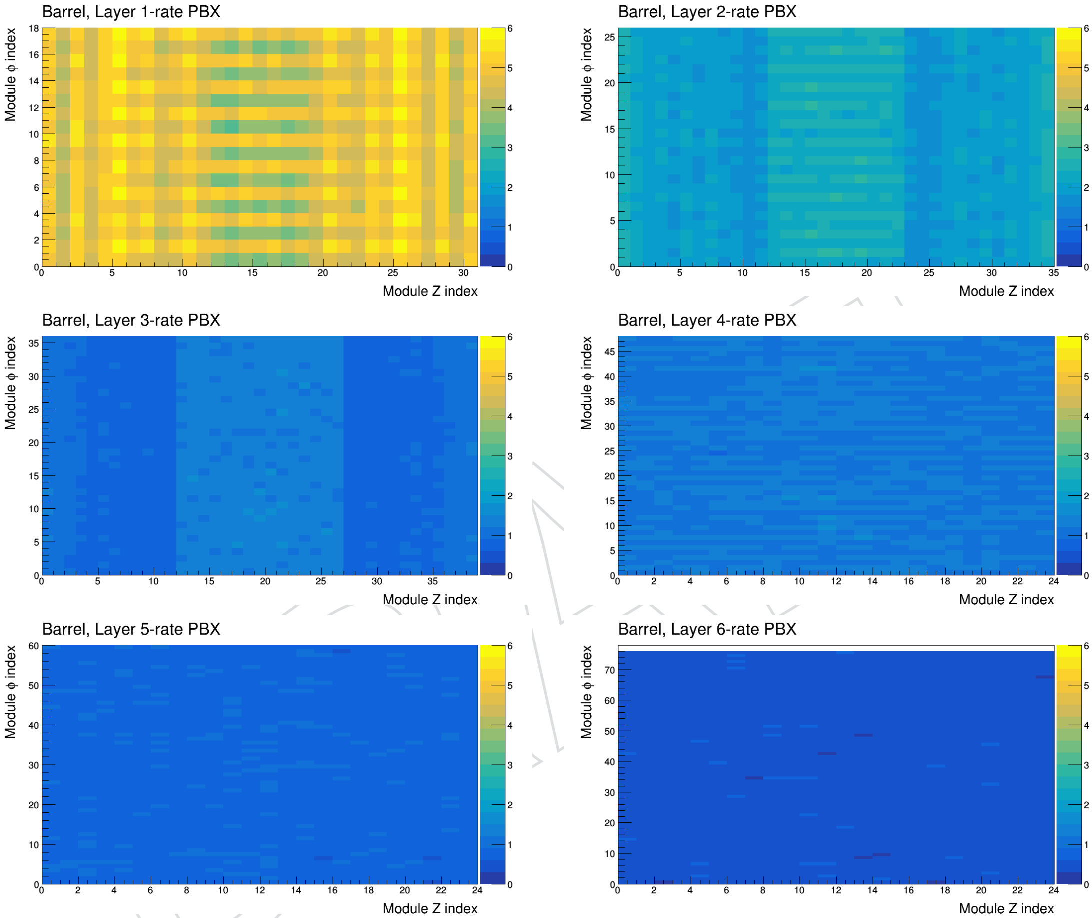


Figure 7: Two dimensional stub rates distributions in barrel layers. Each bin corresponds to a module in z, ϕ position.

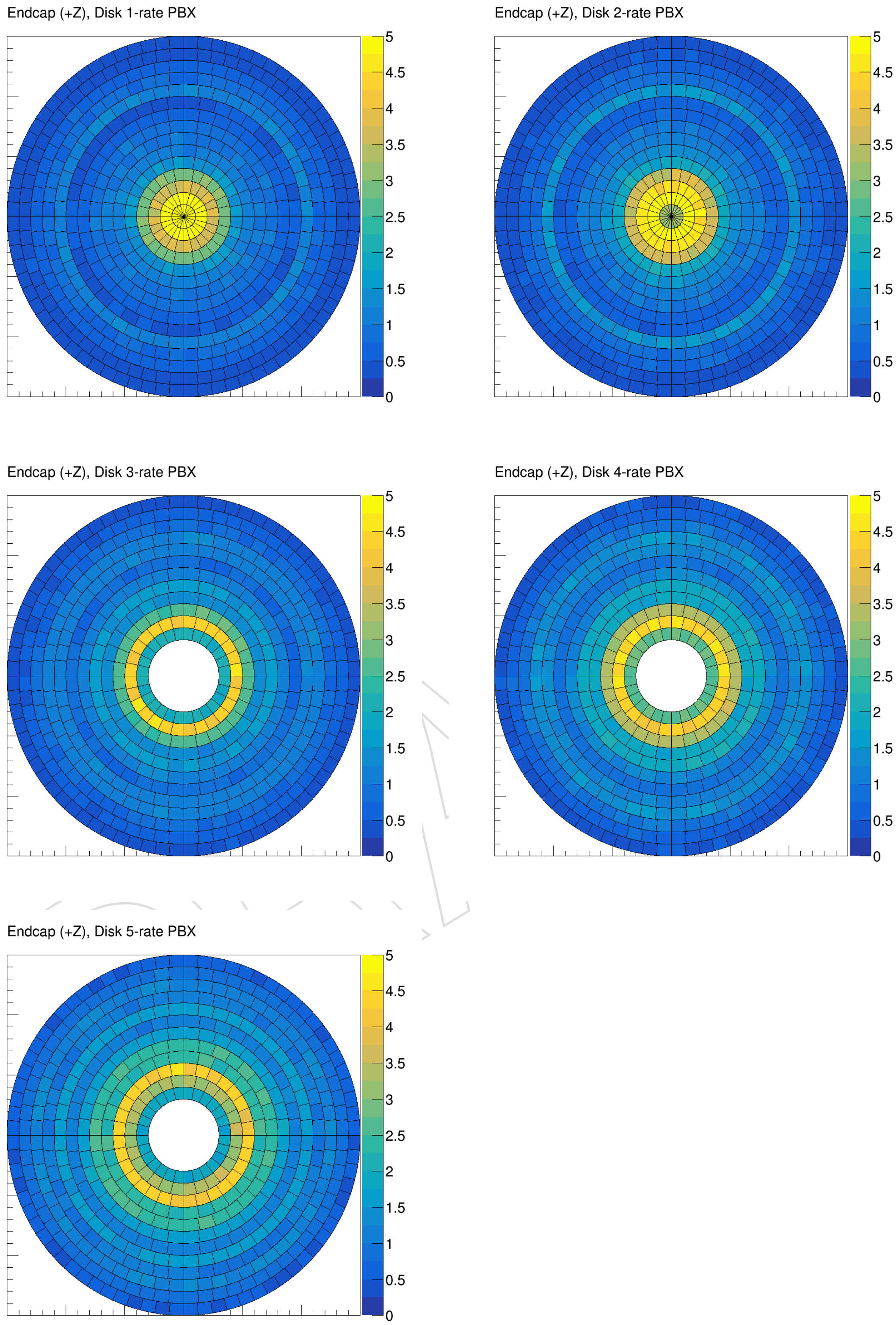


Figure 8: Two dimensional stub rates distributions in endcap disks. Each bin corresponds to a module in r, ϕ position.

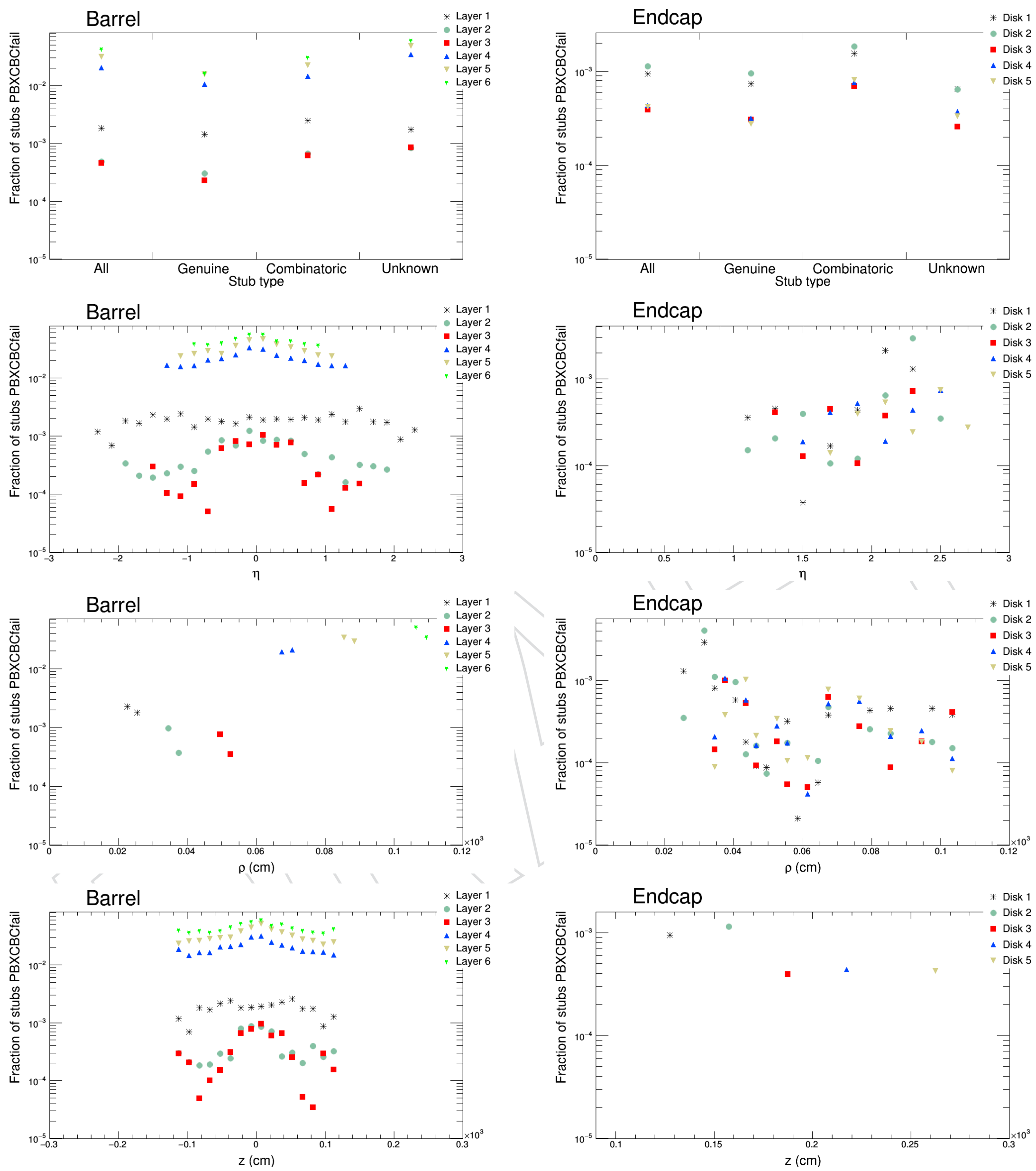


Figure 9: Fraction of stubs that are lost by the CBC/MPA in barrel (left) and endcap (right) are shown as a function of stub types, module η, ρ, and z using $t\bar{t}$ events with 200 PU.

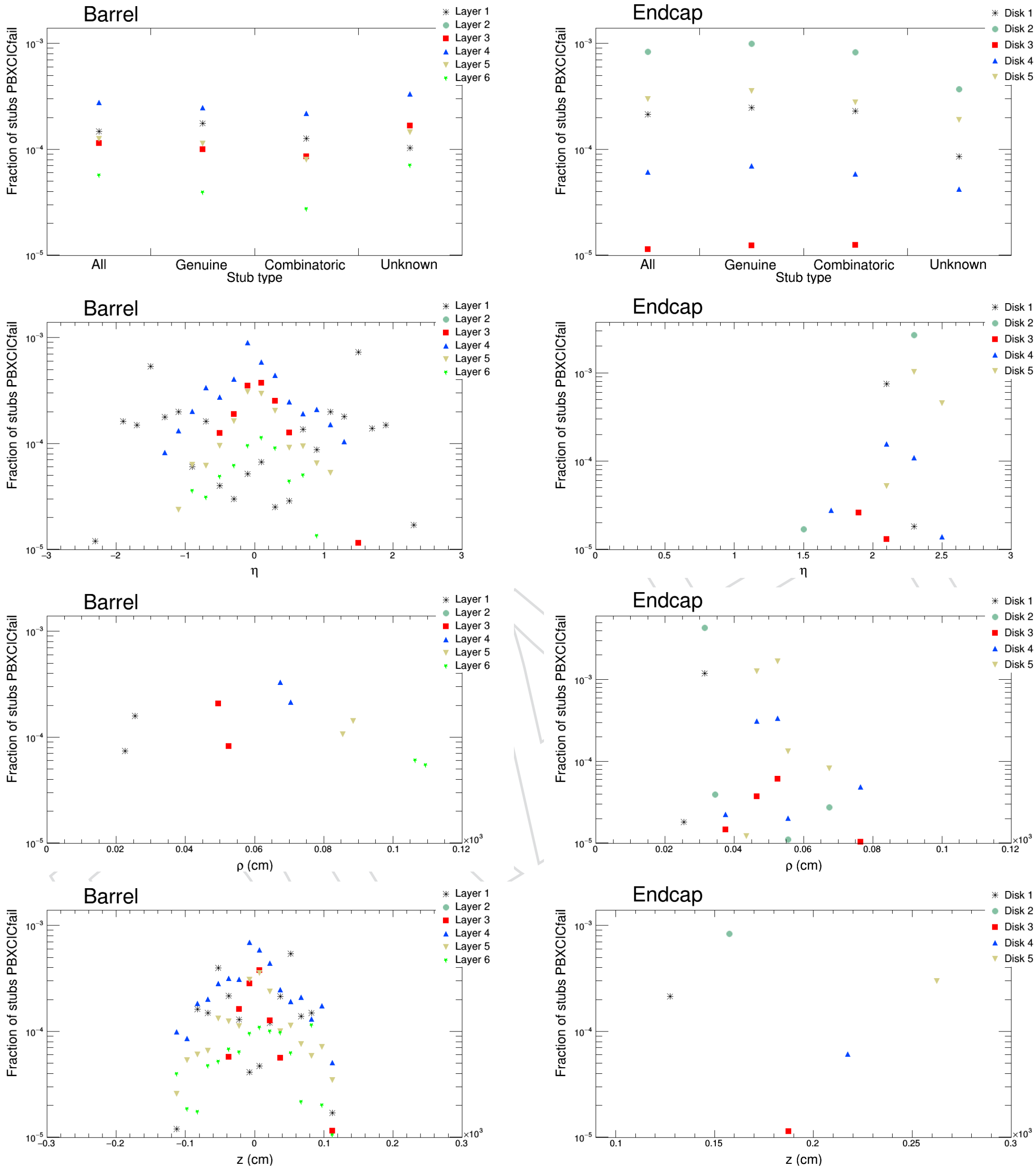


Figure 10: Fraction of stubs that are lost by the CIC in barrel (left) and endcap (right) are shown as a function of stub types, module η, ρ, and z using $t\bar{t}$ events with 200 PU.

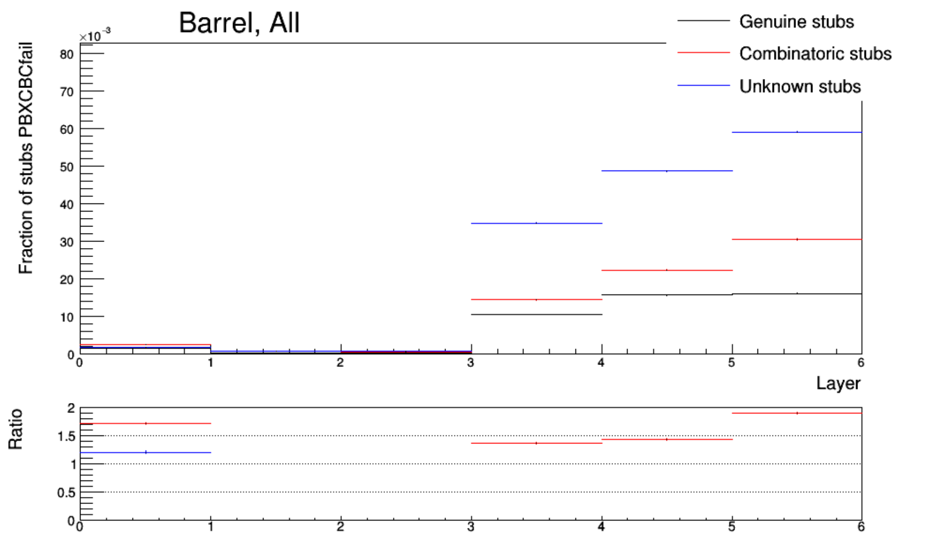


Figure 11: Fraction of stubs that are lost by the CBC/MPAs in barrel are shown for genuine, combinatoric and unknown stubs using $t\bar{t}$ events with 200 PU.

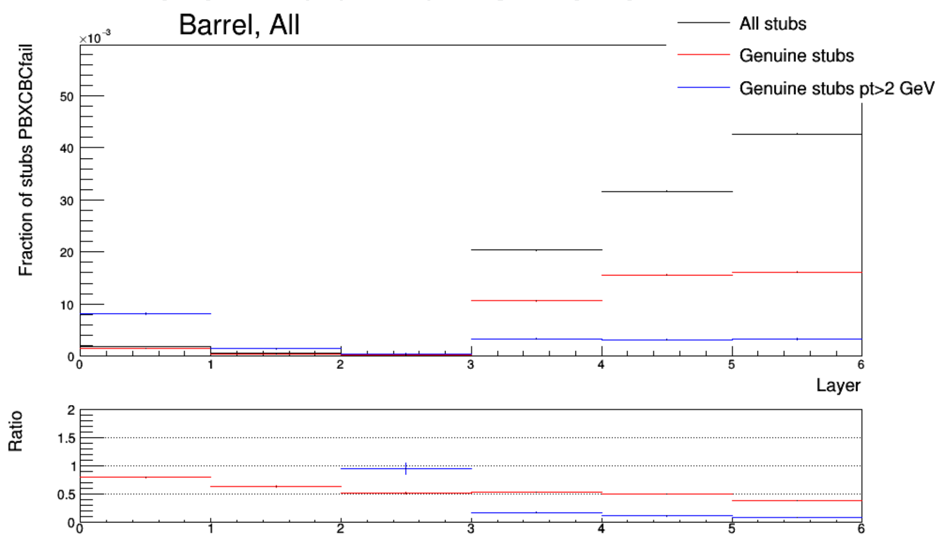


Figure 12: Fraction of stubs that are lost by the CBC/MPAs in barrel are shown for all stubs, genuine stubs and genuine stubs with TP $p_T > 2$ GeV using $t\bar{t}$ events with 200 PU.

4 Stub efficiency

Stub efficiency, defined as the ratio of the number of genuine clusters (matched to a track) in single track events that are used in a genuine stubs to the number of all genuine clusters.

$$\text{Stub efficiency (1)} = \frac{\text{Number of genuine clusters used in genuine stubs}}{\text{Number of genuine clusters}} \tag{1}$$

To measure stub efficiencies for electrons, muons, and displaced muons, we use single lepton samples with zero PU listed in table 1. We loop over TPs in lepton gun sample and find matched clusters for each TP. Then we check if the matched clusters are used in stub construction to find the numerator of the above equation.

In single lepton sample, it is enough to find at least one stub in each layer/Disk. Therefore, we define another stub efficiency;

$$\text{Stub efficiency (2)} = \frac{\text{Number of genuine clusters if a genuine stub is found in a layer/disk}}{\text{Number of genuine clusters}} \tag{2}$$

In Figure 13-14, stub reconstruction efficiencies for electron, muon and displaced muon are shown as a function of the TP p_{T} for the first and second efficiency definition. By construction, the second efficiency is higher than the first efficiency. Similar efficiency plots are shown as a function of the stub η, TP d_0 and TP d_0 in Figures 15-20,

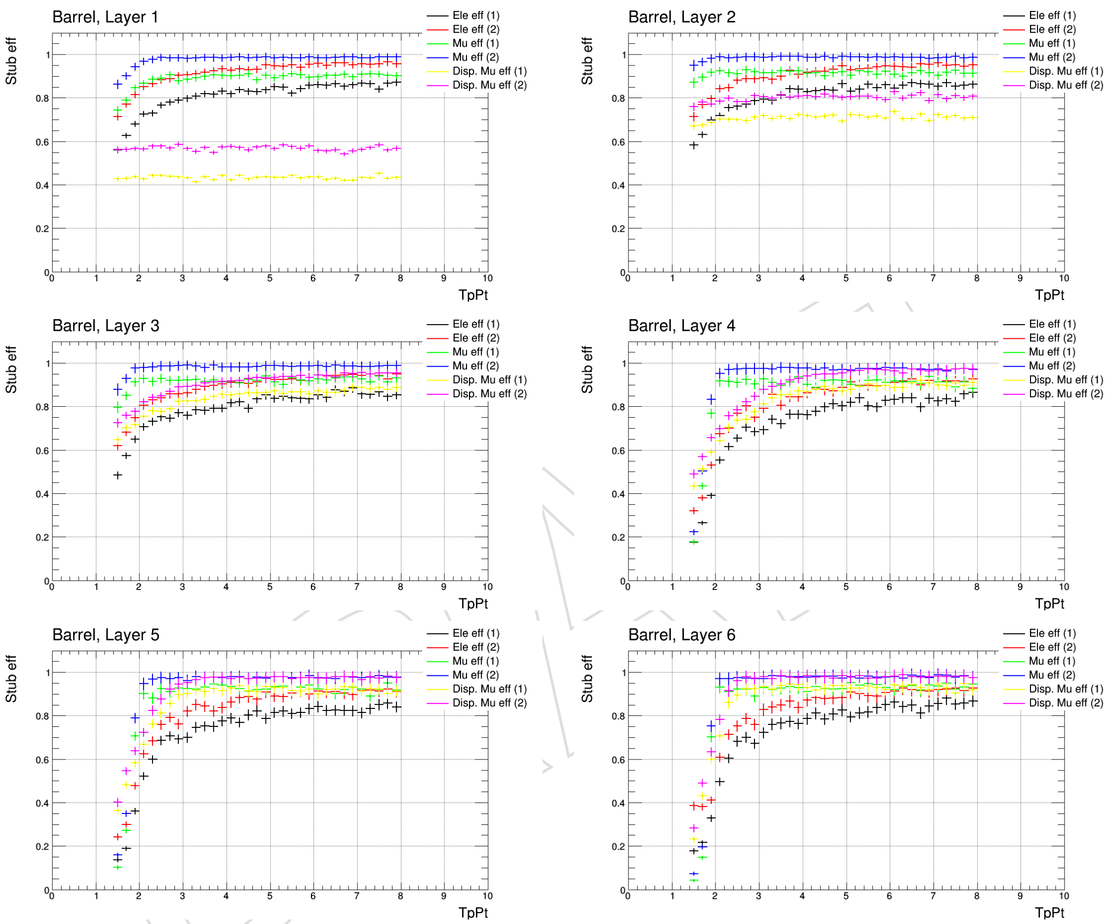


Figure 13: Stub reconstruction efficiencies in barrel layers for electron, muon and displaced muon as a function of TP p_T. Both efficiency definitions defined in the text are shown.

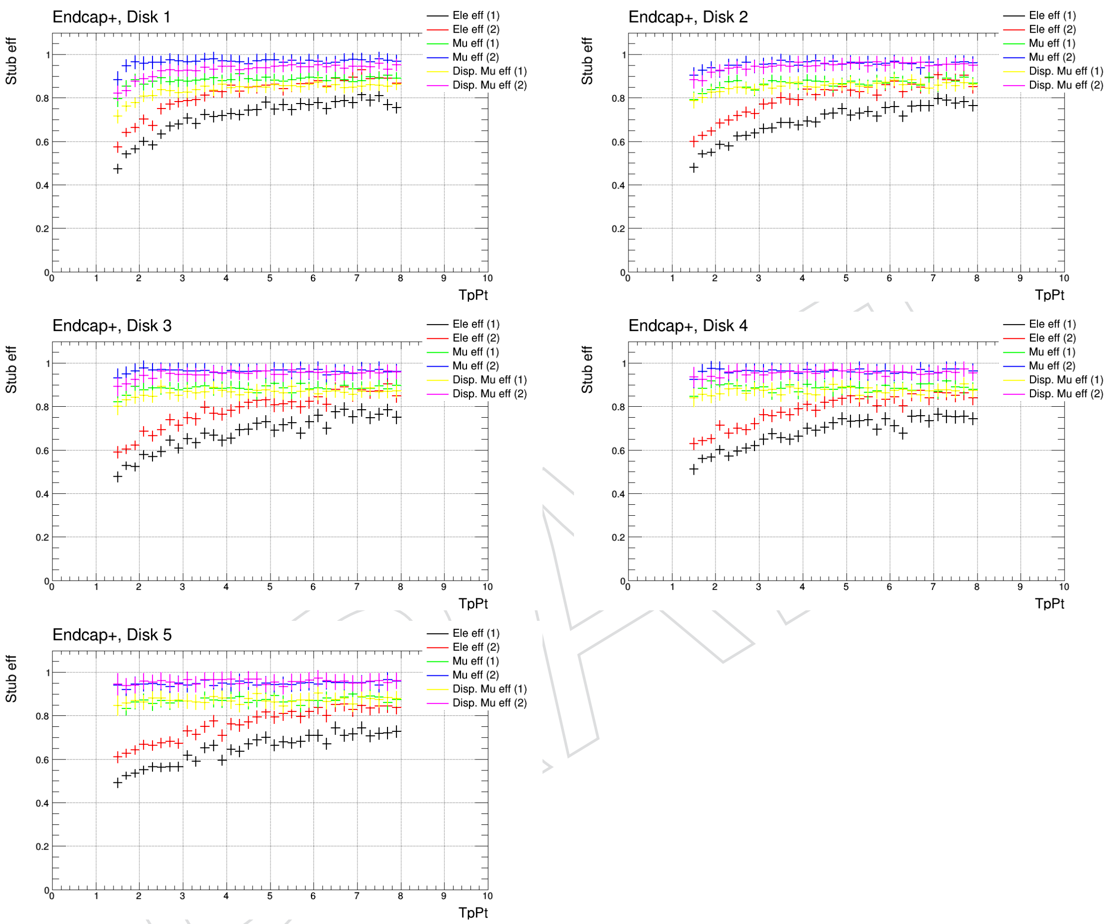


Figure 14: Stub reconstruction efficiencies in endcap disks for electron, muon and displaced muon as a function of TP p_{T}. Both efficiency definitions defined in the text are shown.

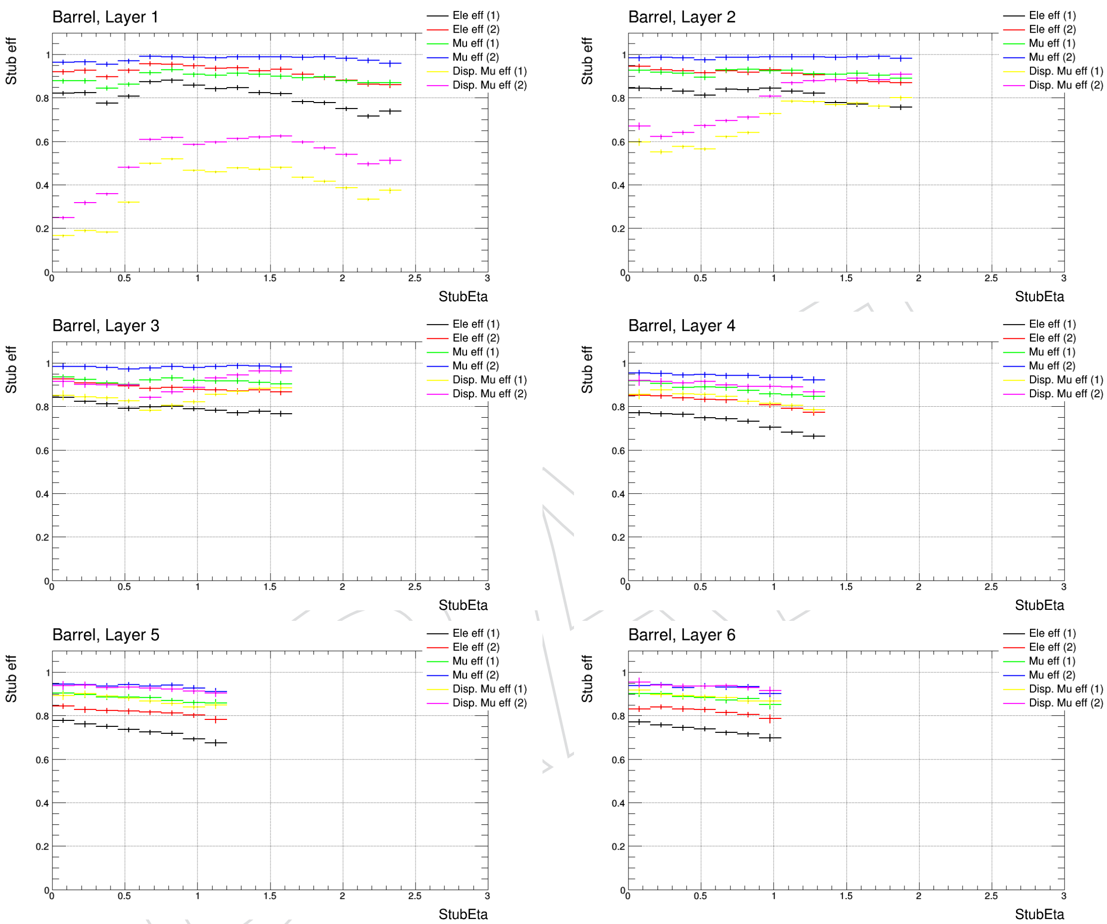


Figure 15: Stub reconstruction efficiencies in barrel layers for electron, muon and displaced muon as a function of Stub Eta. Both efficiency definitions defined in the text are shown.

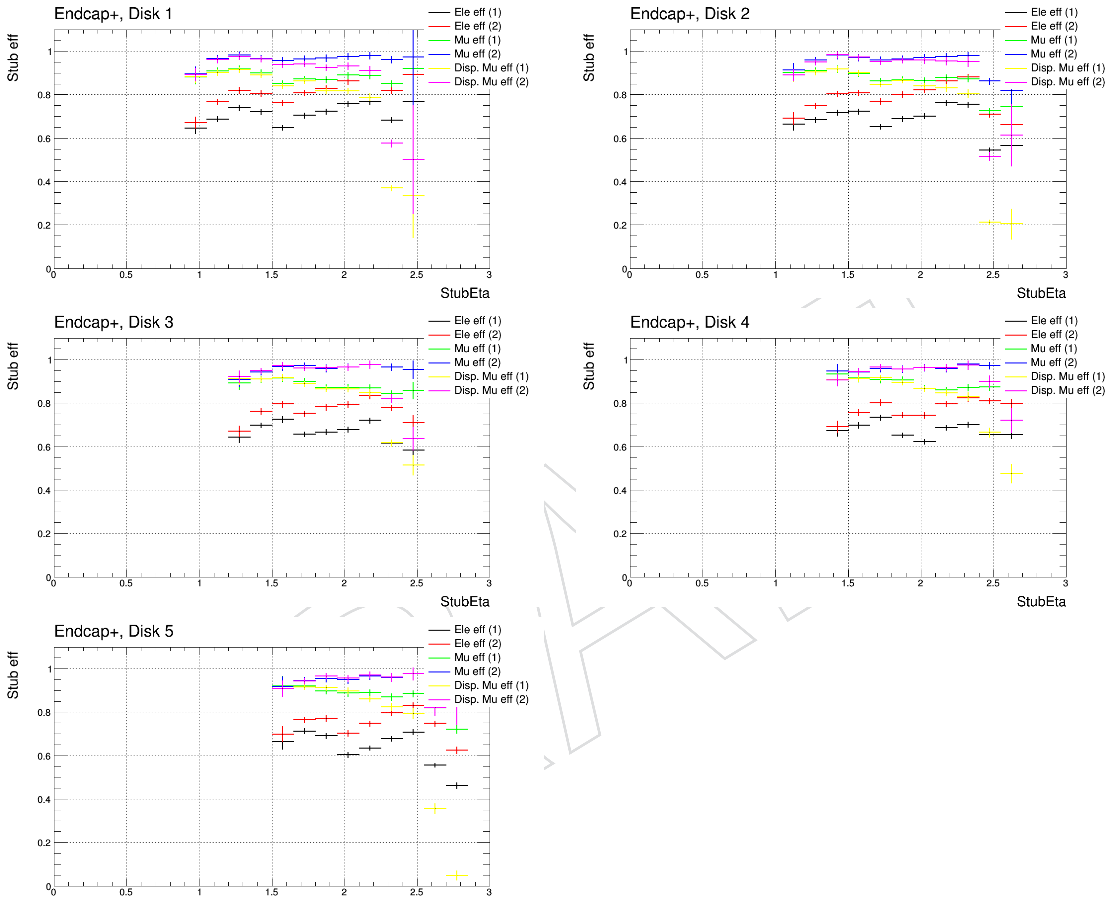


Figure 16: Stub reconstruction efficiencies in endcap disks for electron, muon and displaced muon as a function of Stub Eta. Both efficiency definitions defined in the text are shown.

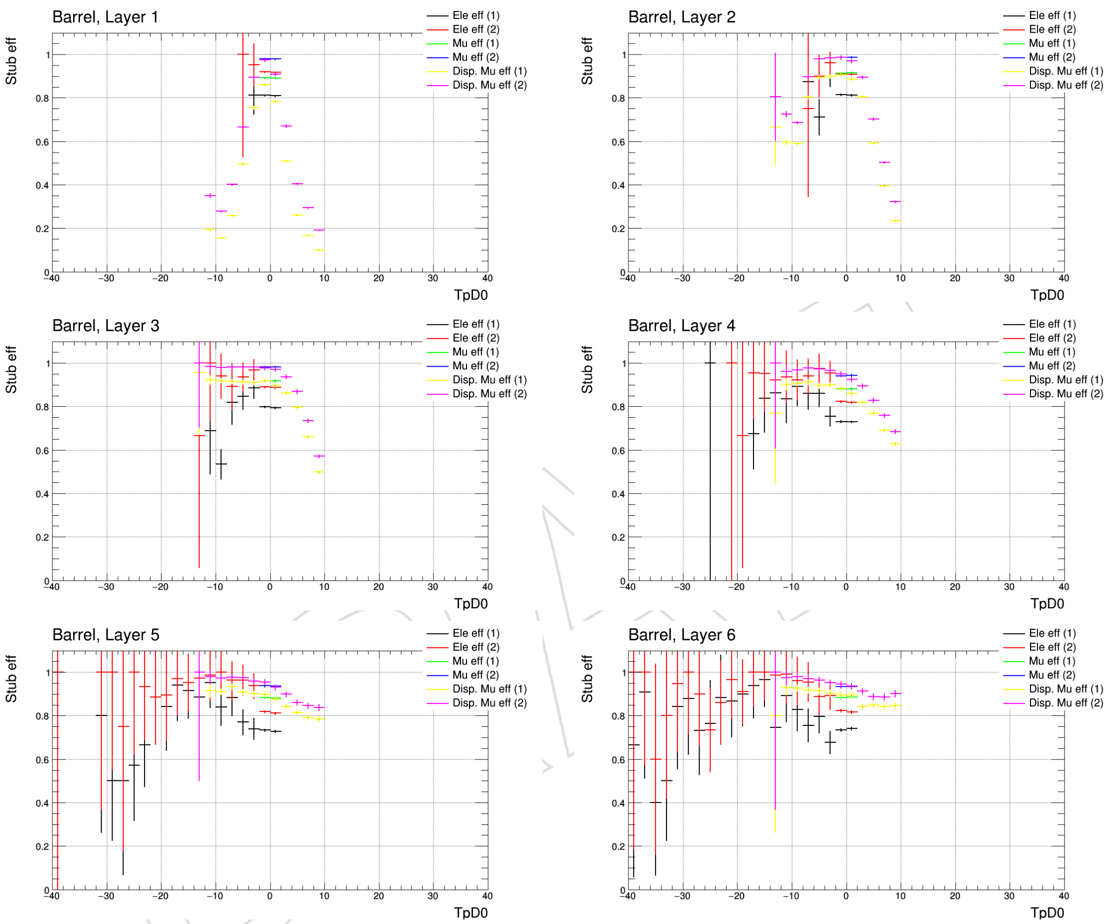


Figure 17: Stub reconstruction efficiencies in barrel layers for electron, muon and displaced muon as a function of TP d_0. Both efficiency definitions defined in the text are shown.

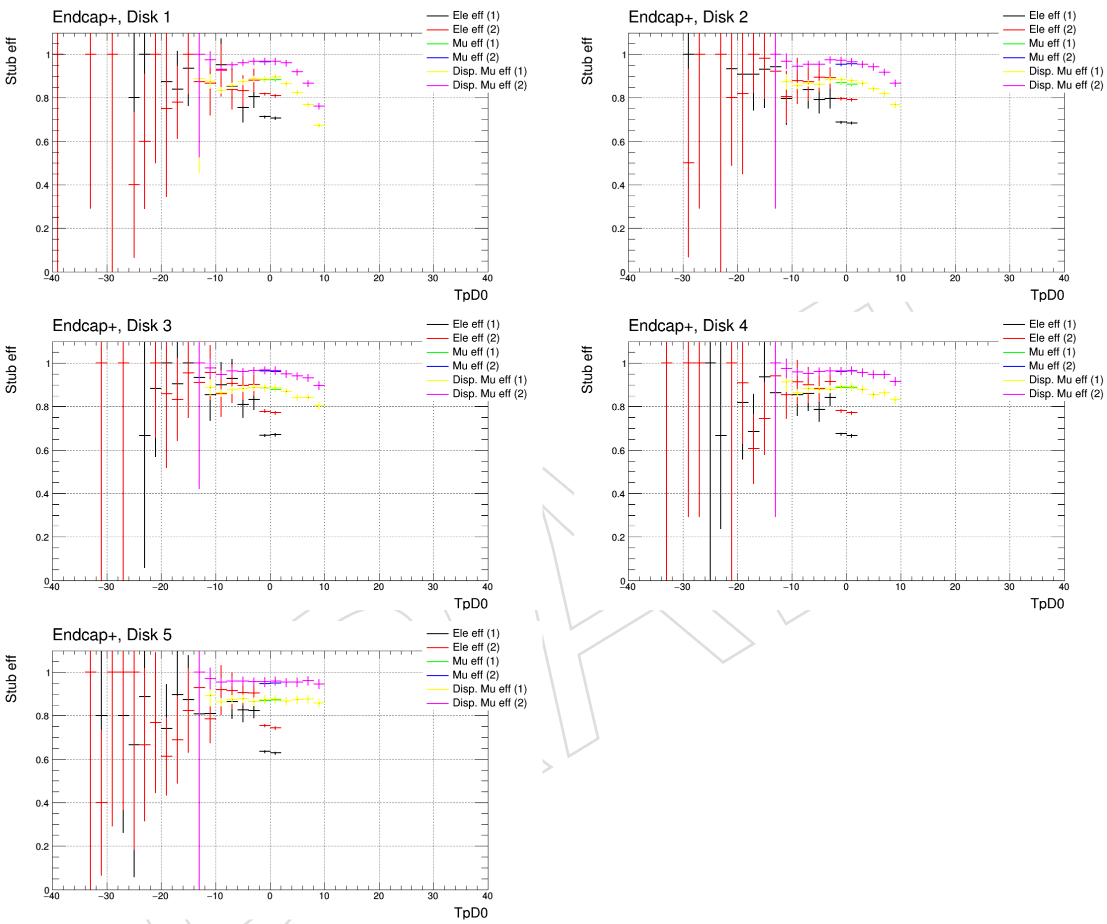


Figure 18: Stub reconstruction efficiencies in endcap disks for electron, muon and displaced muon as a function of TP d_0. Both efficiency definitions defined in the text are shown.

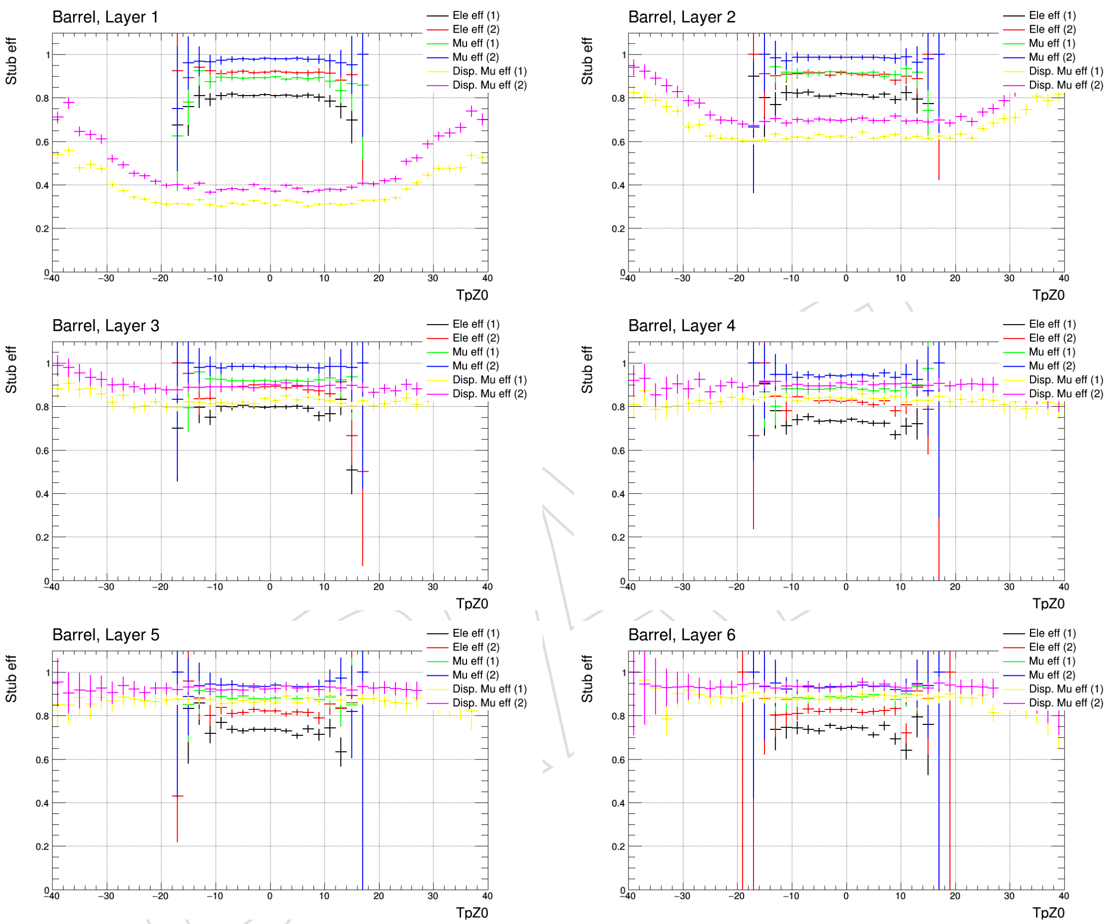


Figure 19: Stub reconstruction efficiencies in barrel layers for electron, muon and displaced muon as a function of TP z_0. Both efficiency definitions defined in the text are shown.

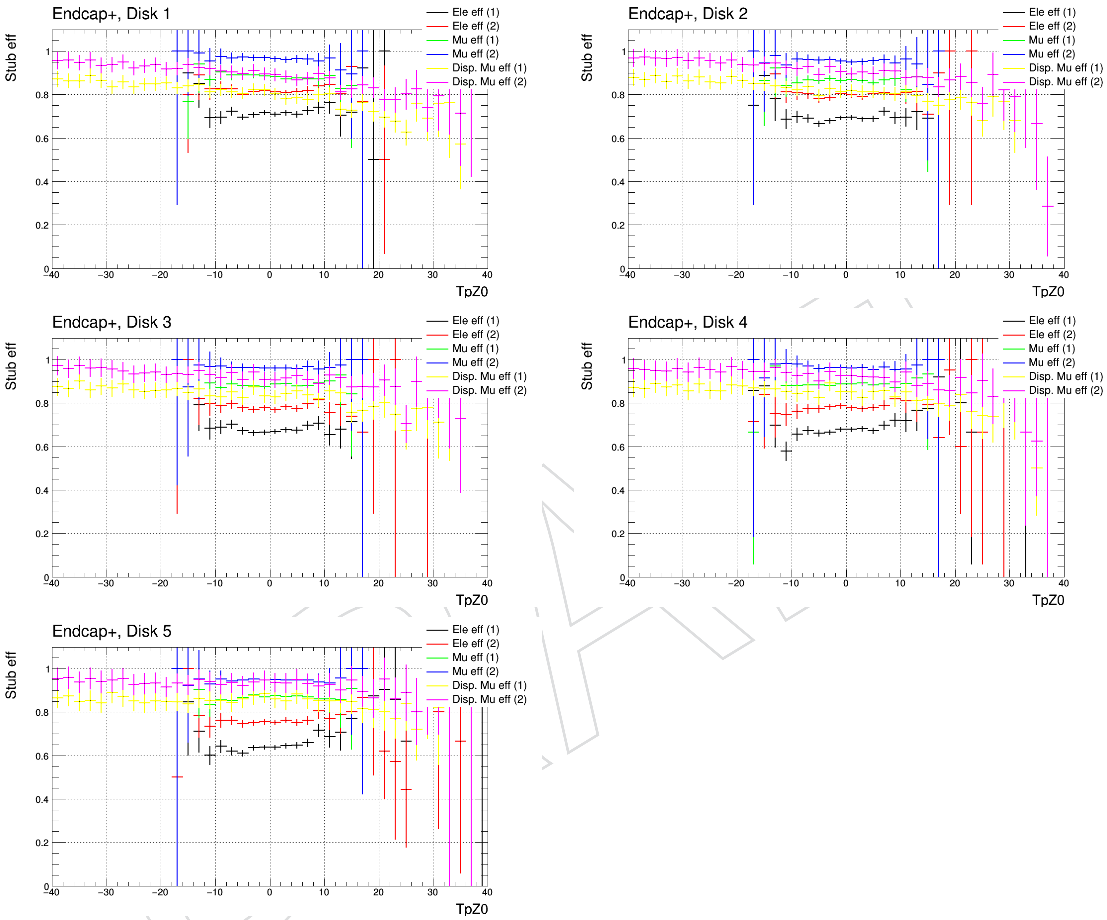


Figure 20: Stub reconstruction efficiencies in endcap disks for electron, muon and displaced muon as a function of TP z_0. Both efficiency definitions defined in the text are shown.

5 Decoding Bend

The p_{T} of a track is estimated from the cluster displacement between the top and bottom sensor of a p_{T} module. This quantity, in units of detector strips, is known as the bend. The bend can be approximated from track parameters with

$$bend = \Delta r \frac{r}{2p} rinv = \Delta r \frac{1.14qr}{2p_{\mathrm{T}}}$$

Where r is the radial position of the hit, p is the strip pitch, rinv is the inverse radius of curvature of the track, q is the charge. Δr is the radial separation of the hits in a p_{T} module given by:

$$\Delta r = \frac{s}{\sin(\alpha)\frac{z}{r} + \cos(\alpha)} \; ; \; \alpha_{flat} = 0^{\circ}, \; \alpha_{disk} = 90^{\circ}$$

.

The bend, measured in half-strip bins, is encoded into 3 or 4 bits (PS/2S) which represent the range of bends accepted for that module. In general the bend bins are merged starting with high bend or low p_{T} bins. This loss of resolution only occurs for modules with stub window size larger than 1.5/3.5 (PS/2S). To use the encoded bend for a set of modules we define a map from encoded bend to some bend parameter, which is then used to estimate p_{T}. This map is what we call the bend decoding.

The stub-to-stub or stub-to-track bend correlations are quickly assessed via the use of pre-computed lookup tables (LUT) that are indexed by stub positions and encoded bends. Hardware constraints limit the granularity of lookup table bins, in particular this means that any single r/z bin in the disks and PS layers may contain several modules for which the same encoded bend represents different values of p_{T}. The bend decoding must account for this variation for optimal performance. The bend decoding for some region is constructed by considering all modules and their bends which map to the encoded bend. Using this we can construct a range in our bend parameter that covers the variation of all the modules in that LUT region.

In the legacy algorithm the bend decoding is defined using simulated events. This was estimated by optimizing the efficiency in single muon events and then adjusting the rate in some areas with ttbar+200PU events. This method does not require knowledge of module tilt, window sizes, or encoding scheme. The drawback is that in general any change to the bend window sizes, encoding scheme, bend parameter, or LUT r/z bins requires the user to redefine the bend decoding. Here we introduce a process for which the bend decoding can be calculated as a function of the stub window sizes and LUT r/z bins. These changes can enable future studies involving alternative bend parameters, LUT r/z bins, or encoding schemes.

In practice the bend is decoded in terms of a quantity called bendstrip which is proportional to the inverse radius of curvature (rinv) and is inversely proportional to p_{T}. The bendstrip is calculated from track parameters using:

$$bendstrip = 0.18 \frac{r}{2} rinv$$

Where 0.18 represents the sensor spacing for the 2S modules and r is the radial position of the stub. This bend parameter works well for the PS/2S layers due to a single layer varying only a little in r however this introduces issues when decoding the bend in the disks. The mean bendstrip for a particular encoded bend, depends on the radial position of the stub as shown in

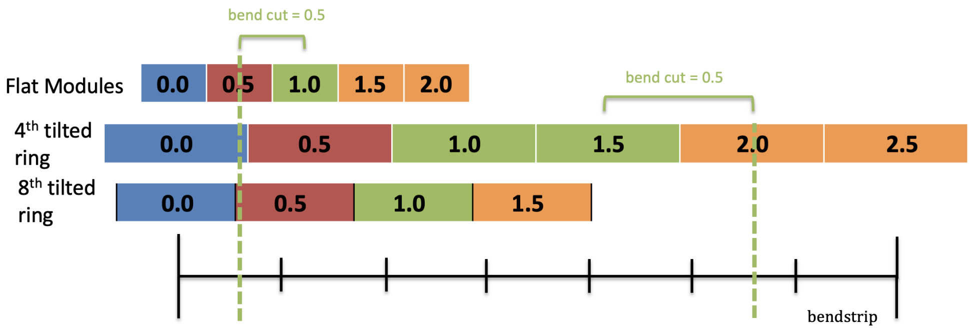


Figure 21: Representation of the encoding and proposed decoding algorithm for three rings in layer 1. Each row shows the encoding of a specific ring and their lengths represent the range in bendstrip that is covered by their stub window size. The half integer numbers are the possible (positive) bends, and the colors represent their mapping to the encoded bend. The green dashed lines represent the decoded bend for an encoded bend of 2. The encoded bend is a representation of the detector bend in 3 bits, an encoded bend of 2 does not correspond to a detector bend of 2. The bend cut is a tunable parameter.

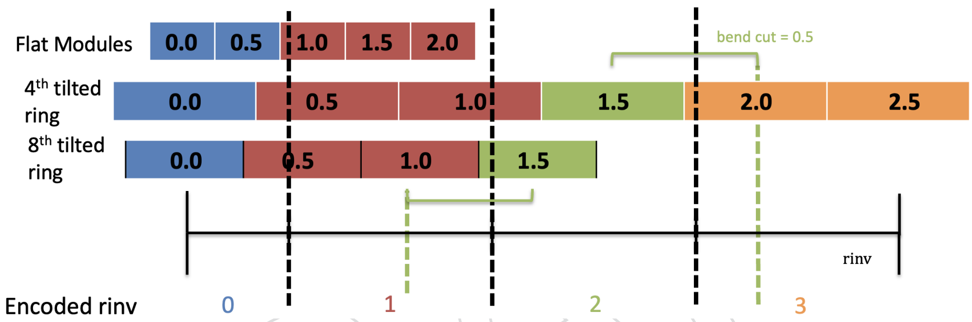


Figure 22: Representation of a potential rinv encoding/decoding algorithm for three rings in layer 1. Each row shows the encoding of a specific ring and their lengths represent the range in rinv that is covered by their stub window size. The half integer numbers are the possible (positive) bends, and the colors represent their mapping to the encoded rinv. The black dashed lines are the rinv bins, the green dashed lines represent the decoded rinv for an encoded rinv of 2.

figure 23 . To account for this additional r bins were added to the lookup tables used during the track seeding stage (TrackletEngine). It may be beneficial to also add additional r bins to the lookup tables in the track-to-stub association stage (MatchEngine). A different choice of bend parameter (rinv) or encoding scheme may also fix this issue.

This improvement to the bend decoding process is enabled in part by the use of module information obtained with the SensorModule class implemented in L1Trigger/TrackTrigger. Without the module information (position, tilt, stub window size) we would not be able to properly represent a modules bend in whatever bend parameter space we are using. To get this information where we need it we are required to pass the Setup class (L1Trigger/TrackTrigger) through to the TrackletLUT. In the TrackletLUT we define new methods which are used to decode the bend in LUT creation.

The bend decoding process uses the methods getSensorModules, getBendCut, and sometimes, getTanRange which are defined in the TrackletLUT class. To decode the bend we first find a set of all modules for which we want the decoding to be valid. This is done using the getSensorModules method. There are two ways that this method is used in the track trigger code. The

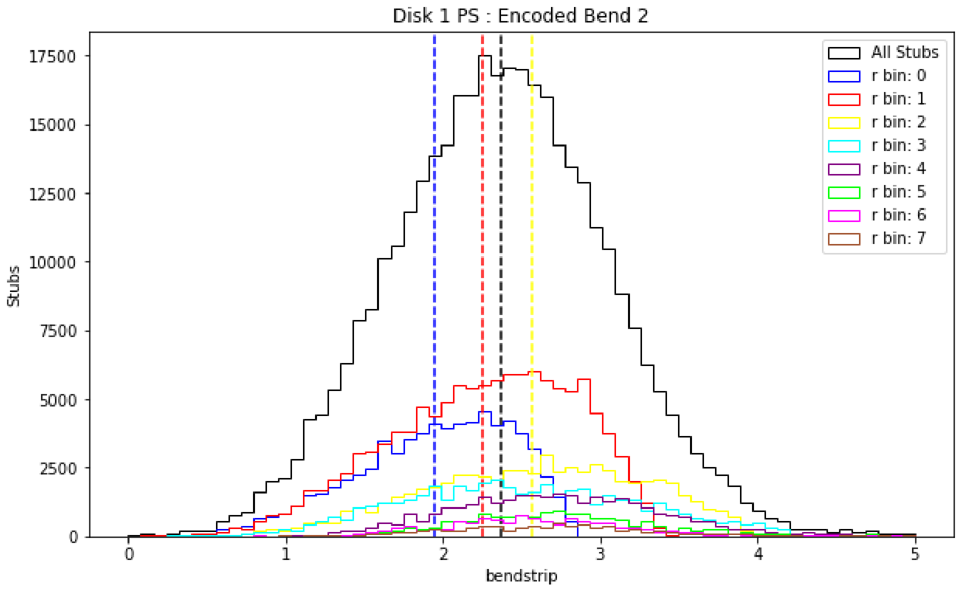


Figure 23: The dashed lines represent the mean bendstrip for an ensemble of stubs, with an encoded bend of 2, in a particular r-bin of disk 1 (PS). This shows that the bendstrip for a particular encoded bend varies as a function of r in the disks. To recover bend resolution one can either bin the disks in r or use a bend parameter which is independent of r. These stubs are taken from a sample of 5000 TTbar 200PU events.

default is to return all modules, unique in $|z|$ for a given layer/disk and module type (PS/2S); this is used mostly in the MatchEngine. For the TrackletEngine lookup tables we can reduce the number of modules by passing the getSensorModules method a range in $\tan(\theta)$ that will cover only the relevant modules. Theta here is measured with respect to the radial axis. To account for displaced tracks the max/min $\tan(\theta)$ are measured from ± 15 cm along the z axis.

Once we find our sensor modules we pass that information to the getBendCut method. To start we initialize a few data structures to store our bend decoding information and then loop over all the sensor modules. For each module we loop over all possible bend values, which are determined by the stub window size. For each bend in a given module we find its encoded bend value and calculate the corresponding bend max/min, defined as the bend plus/minus a bend cut. Finally we transform the bend max/min into the chosen bend parameter space, bendstrip in this case, and update the bend decoding to cover this range for this particular encoded bend value. This process provides a different bend decoding for every r/z bin in the Tracklet/MatchEngine lookup tables.

The bend cuts are used to balance the rates of real (true positive) and fake (false positive) stub combinations or track projections, while keeping the total pass rate below the threshold set by truncation. A real combination will pass in the LUT and also has matching tracking particles, while a fake combination will pass in the LUT but not have any matching tracking particles. Bend cuts which are too small will miss too many real tracks while bend cuts which are too large will lead to real tracks being truncated. The tuning strategy used here is just to match the true positive rate of the legacy algorithm. In general this method of decoding the bend results in a lower false positive and therefore truncation rate while maintaining or improving the true positive rate for the samples studied.

The bend cuts, defined in TrackFindingTracklet/Interface/Settings.h, are represented as multiples of the approximate bend resolution:

$$\sigma_{bend} \approx \frac{1}{\sqrt{6}}[5]$$

For the TrackletEngine the bend cuts are all between 2-2.6; the MatchEngine however requires bend cuts up to 4 to maintain efficiency because the disks are not binned in r. The true negative, true positive, truncation, and false positive rates are compared in figure 24.

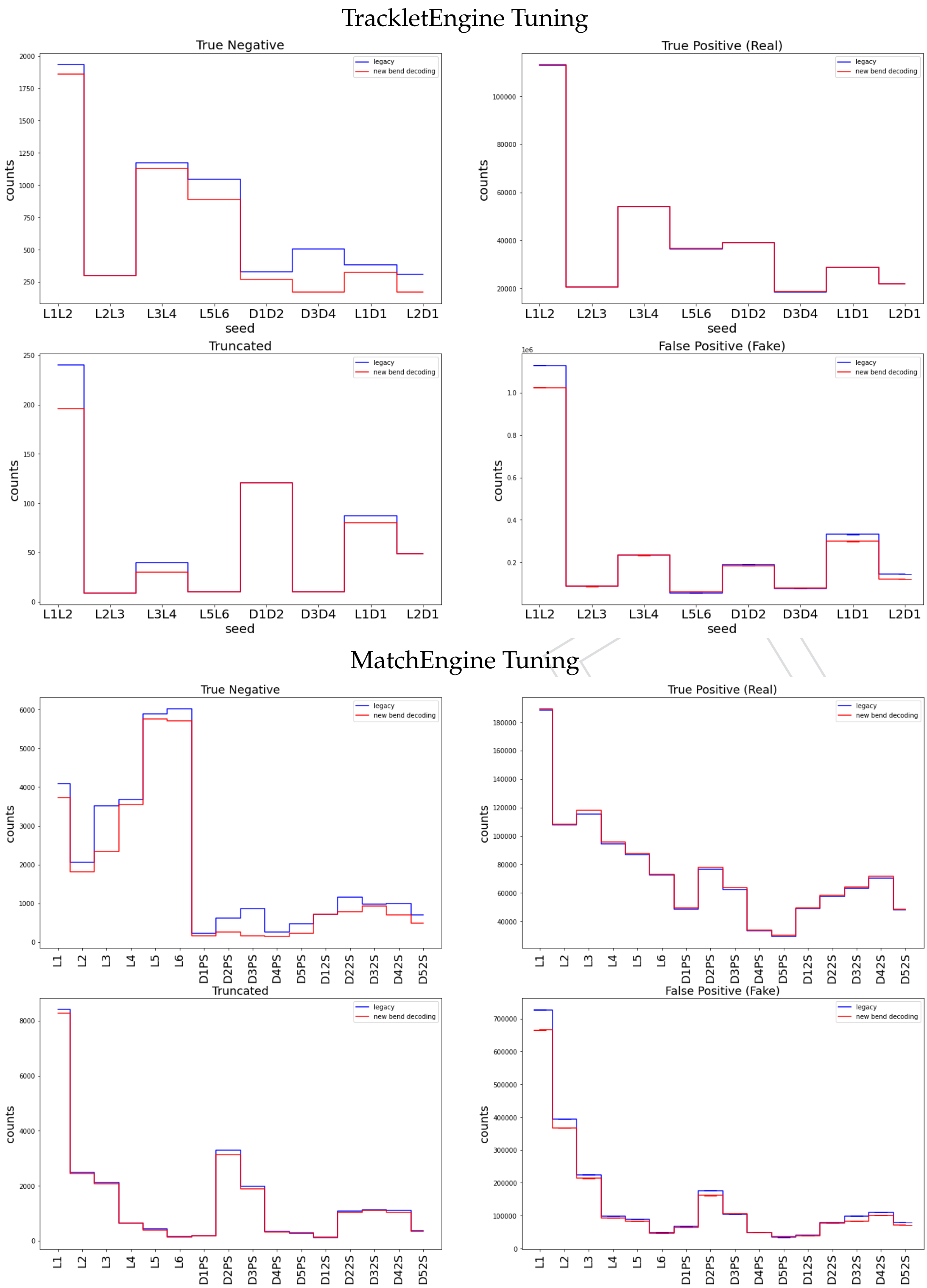


Figure 24: TrackletEngine (top) and MatchEngine (bottom) tuning. The Tracklet/MatchEngine associate stubs using their bend information. A "True" association is one in which all stubs share a matching tracking particle. A positive association is one that passes their respective LUT. The bend cuts are set such that the True Positive (Real) rate is larger than the legacy algorithm (equivalently a smaller True Negative rate). The bend cuts are parameterized by seed in the TrackletEngine and by layer/disk in the MatchEngine. These plots are made using 500 TTbar 200PU events.

6 Stub window tuning

Stubs are reconstructed by the CBC and MPA chips each time two correlated clusters will be found in the two layers (see Figure 2). Stub rates and reconstruction efficiencies depend on the stub acceptance window size. Establishing this window size, known as stub window (SW), in all the tracker regions is a fundamental step requiring a careful optimization. Indeed, small stub windows will ensure very good data reduction with a cost in efficiency, whereas large stub windows will provide good efficiency with a cost in rate [5].

As the signal information is binary, a group of adjacent hits is considered a cluster. The cluster width w is defined as the number of strips in the cluster

$$w = n_{last} - n_{first} + 1 \tag{3}$$

Where n_{first} and n_{last} are the indices of the first and last strips in the cluster, respectively, in accordance with the geometric order of the strips on the sensors. The cluster position X is defined as the mean value

$$X = \frac{n_{last} + n_{first}}{2} \tag{4}$$

The cluster width is given in steps of full-strips, while the position is in half-strips.

The stub position μ is defined as the mean cluster position per module

$$\mu = \frac{X_{top} + X_{bottom}}{2} \tag{5}$$

Where X_{top} and X_{bottom} are the cluster positions in the top and bottom sensor of the module, respectively. The stub window ΔX is defined as the difference between the cluster positions in the top and bottom sensor,

$$\Delta X = X_{top} - X_{bottom} \tag{6}$$

the stub window size is given in steps of half-strips, while the stub position is in quarter-strips. This window size varies from as little as two strips in the PS modules at the lowest radii in the forward region to nine strips in the current version of SW tune [5]. For the 2S modules the acceptance window varies between 6-15 strips. These acceptance windows are configurable and can be tuned to manage the rate for the trigger data [6].

The latest SW tune which is used in simulated samples are called "tight tune" which is optimized to ensure a good efficiency for muon stubs. There is another tune called "loose tune" in which electron efficiencies are also included (see `https://indico.cern.ch/event/681577/contributions/2816628/attachments/1572998/2482715/UpgradeSim_111217.pdf`). Stub windows should vary in different geometrical regions of the tracker to control rates and FE losses. In the current tune, SW is defined for 108 geometrical regions as are listed bellow and are shown in Figure 25;

- Flat modules in barrel layers (6 regions)
- Tilted modules in the first three layers (3×12 = 36 regions)
- Rings in the first two disks (2×15 = 30 regions)
- Rings in the last three disks (3×12 = 36 regions)

In order to update the stub tune, we need to know the following variables in 108 geometrical regions of the tracker

- Stub rate as a function of stub window size

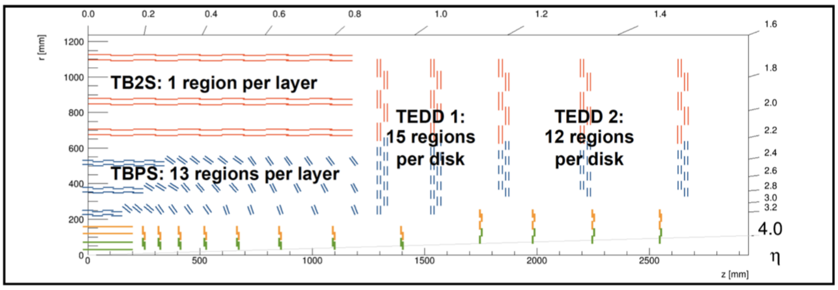


Figure 25: Tracker regions requiring a specific SW tuning. TB2S: Tracker Barrel with 2S modules. TBPS: Tracker Barrel with PS modules. TEDD: Tracker Endcap Double Disks [5].

- FE losses as a function of stub window size
- Stub efficiency for electron, muon as a function of stub window size

As was discussed in the previous sections, we use $t\bar{t}$ events with PU=200 for finding the stub-rate/FE-losses and single lepton samples for measuring the stub reconstruction efficiencies. For SW tuning, we find efficiencies for muon in [2,8] GeV and electron in [4,8] GeV p_T range to be in the plateau of the efficiency turn on curve. We run the same chain of stub reconstruction using a fix value for the SW size in all 108 geometrical regions and measure the above variables. In the following, the input SW sizes for 108 geometrical regions of the tracker are printed for tight and loose tunes (see L1Trigger/TrackTrigger/python/TTStubAlgorithmRegister_cfi.py).

```
# PU200 tight tuning, optimized for muons
BarrelCut    = cms.vdouble( 0, 2, 2.5, 3.5, 4.5, 5.5, 7),
TiltedBarrelCutSet = cms.VPSet(
     cms.PSet( TiltedCut = cms.vdouble( 0 ) ),
     cms.PSet( TiltedCut = cms.vdouble( 0, 3, 3, 2.5, 3, 3, 2.5, 2.5, 2, 1.5, 1.5, 1, 1) ),
     cms.PSet( TiltedCut = cms.vdouble( 0, 3.5, 3, 3, 3, 3, 2.5, 2.5, 3, 3, 2.5, 2.5, 2.5) ),
     cms.PSet( TiltedCut = cms.vdouble( 0, 4, 4, 4, 3.5, 3.5, 3.5, 3.5, 3, 3, 3, 3, 3) ),
     ),
EndcapCutSet = cms.VPSet(
     cms.PSet( EndcapCut = cms.vdouble( 0 ) ),
     cms.PSet( EndcapCut = cms.vdouble( 0, 1, 2.5, 2.5, 3, 2.5, 3, 3.5, 4, 4, 4.5, 3.5, 4, 4.5, 5, 5.5) ),
     cms.PSet( EndcapCut = cms.vdouble( 0, 0.5, 2.5, 2.5, 3, 2.5, 3, 3, 3.5, 3.5, 4, 3.5, 3.5, 4, 4.5, 5) ),
     cms.PSet( EndcapCut = cms.vdouble( 0, 1, 3, 3, 2.5, 3.5, 3.5, 3.5, 4, 3.5, 3.5, 4, 4.5) ),
     cms.PSet( EndcapCut = cms.vdouble( 0, 1, 2.5, 3, 2.5, 3.5, 3, 3, 3.5, 3.5, 3.5, 4, 4) ),
     cms.PSet( EndcapCut = cms.vdouble( 0, 0.5, 1.5, 3, 2.5, 3.5, 3, 3, 3.5, 4, 3.5, 4, 3.5) ),
     )
```

```
    # PU200 loose tuning, optimized for robustness
    BarrelCut    = cms.vdouble( 0, 2.0, 3, 4.5, 6, 6.5, 7.0),
    TiltedBarrelCutSet = cms.VPSet(
         cms.PSet( TiltedCut = cms.vdouble( 0 ) ),
         cms.PSet( TiltedCut = cms.vdouble( 0, 3, 3, 2.5, 3, 3, 2.5, 2.5, 2, 1.5, 1.5, 1, 1) ),
         cms.PSet( TiltedCut = cms.vdouble( 0, 4., 4, 4, 4, 4, 4.5, 5, 4, 3.5, 3.5, 3) ),
         cms.PSet( TiltedCut = cms.vdouble( 0, 5, 5, 5, 5, 5, 5, 5.5, 5, 5, 5.5, 5.5, 5.5) ),
       ),
    EndcapCutSet = cms.VPSet(
         cms.PSet( EndcapCut = cms.vdouble( 0 ) ),
         cms.PSet( EndcapCut = cms.vdouble( 0, 1, 2.5, 2.5, 3.5, 5.5, 5.5, 6, 6.5, 6.5, 6.5, 6.5, 6.5, 6.5, 7, 7) ),
         cms.PSet( EndcapCut = cms.vdouble( 0, 0.5, 2.5, 2.5, 3, 5, 6, 6, 6.5, 6.5, 6.5, 6.5, 7, 7) ),
         cms.PSet( EndcapCut = cms.vdouble( 0, 1, 3., 4.5, 6., 6.5, 6.5, 6.5, 7, 7, 7, 7, 7) ),
         cms.PSet( EndcapCut = cms.vdouble( 0, 1., 2.5, 3.5, 6., 6.5, 6.5, 6.5, 6.5, 7, 7, 7, 7) ),
         cms.PSet( EndcapCut = cms.vdouble( 0, 0.5, 1.5, 3., 4.5, 6.5, 6.5, 7, 7, 7, 7, 7, 7) ),
         )
```

We have calculated stub rates, FE losses, and stub efficiencies for the following fix SWs; 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0. For example,

```
    #Fixed tune
    BarrelCut    = cms.vdouble( 0, 1, 1, 1, 1, 1, 1),
    TiltedBarrelCutSet = cms.VPSet(
         cms.PSet( TiltedCut = cms.vdouble( 0 ) ),
         cms.PSet( TiltedCut = cms.vdouble( 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) ),
         cms.PSet( TiltedCut = cms.vdouble( 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) ),
         cms.PSet( TiltedCut = cms.vdouble( 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) ),
         ),
    EndcapCutSet = cms.VPSet(
         cms.PSet( EndcapCut = cms.vdouble( 0 ) ),
         cms.PSet( EndcapCut = cms.vdouble( 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) ),
         cms.PSet( EndcapCut = cms.vdouble( 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) ),
```

```
        cms.PSet( EndcapCut = cms.vdouble( 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) ),
        cms.PSet( EndcapCut = cms.vdouble( 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) ),
        cms.PSet( EndcapCut = cms.vdouble( 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) ),
        )
```

The following algorithm is used to choose the optimum window size with high-efficiency and low-rate for each geometrical region:

- The CBC/MPA stub fail fraction should be less than 1%
- The CIC stub fail fraction should be less than 0.5%
- Normalize the stub reconstruction efficiencies for electron and muon to the efficiency obtained with the largest possible window size (7.0)
- The normalized stub reconstruction efficiencies for electron and muon should be greater than 60%
- Choose the smallest window size where the efficiencies of the electron or muon do not increase more than 0.005 with respect to the closest larger window size

The stub window tune which is optimized based on the stub reconstruction efficiency for muon (muon + electron) is called “New tight” (“New loose”). New tunes are;

```
# New tight tune based on simulated events CMSSW_11_3_0_pre3, D76
    BarrelCut    = cms.vdouble(0, 2.0, 2.0, 3.0, 4.5, 5.5, 6.5),
    TiltedBarrelCutSet = cms.VPSet(
        cms.PSet( TiltedCut = cms.vdouble( 0 ) ),
        cms.PSet( TiltedCut = cms.vdouble( 0, 2.0, 2.0, 2.0, 2.5, 2.0, 2.0, 2.0, 1.5, 1.5, 1.5, 1.0, 1.0) ),
        cms.PSet( TiltedCut = cms.vdouble( 0, 2.5, 2.5, 2.5, 2.5, 2.0, 2.0, 2.0, 2.5, 2.5, 2.0, 2.0, 2.0) ),
        cms.PSet( TiltedCut = cms.vdouble(0, 3.5, 3.5, 3.0, 3.0, 3.0, 3.0, 2.5, 2.5, 2.5, 2.5, 2.5, 2.0) ),
        ),
    EndcapCutSet = cms.VPSet(
        cms.PSet( EndcapCut = cms.vdouble( 0 ) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.0, 1.5, 1.5, 2.0, 2.0, 2.5, 2.5, 3.0, 3.0, 4.0, 2.5, 3.0, 3.5, 4.0, 5.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 0.5, 1.5, 1.5, 2.0, 2.0, 2.0, 2.5, 2.5, 3.0, 3.5, 2.0, 2.5, 3.0, 4.0, 4.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.5, 2.0, 2.0, 2.0, 2.0, 2.5, 2.5, 3.5, 2.0, 2.5, 3.5, 3.5) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.0, 1.5, 1.5, 2.0, 2.0, 2.0, 2.0, 3.0, 2.0, 2.0, 3.0, 3.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.0, 1.5, 1.5, 2.0, 2.0, 2.0, 2.0, 2.5, 3.0, 2.0, 2.0, 2.5) ),
        )
)
\


# New loose tune based on simulated events CMSSW_11_3_0_pre3, D76
    BarrelCut    = cms.vdouble(0, 2.0, 3.0, 4.0, 5.0, 7.0, 7.0),
    TiltedBarrelCutSet = cms.VPSet(
        cms.PSet( TiltedCut = cms.vdouble( 0 ) ),
        cms.PSet( TiltedCut = cms.vdouble( 0, 3.0, 2.0, 2.0, 3.0, 2.5, 3.0, 2.0, 1.5, 1.5, 1.5, 1.0, 1.0) ),
        cms.PSet( TiltedCut = cms.vdouble( 0, 4.0, 3.5, 3.5, 3.5, 4.0, 4.0, 3.5, 5.0, 4.5, 4.0, 4.5, 3.0) ),
        cms.PSet( TiltedCut = cms.vdouble(0, 5.0, 5.0, 5.0, 4.0, 4.0, 4.0, 4.0, 4.0, 5.0, 6.0, 5.0, 5.5) ),
        ),
    EndcapCutSet = cms.VPSet(
        cms.PSet( EndcapCut = cms.vdouble( 0 ) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.0, 2.5, 5.0, 3.0, 4.5, 6.0, 6.0, 6.0, 6.0, 7.0, 6.0, 7.0, 6.0, 7.0, 7.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 0.5, 2.5, 4.0, 5.0, 5.0, 4.0, 4.0, 6.0, 6.0, 7.0, 5.5, 6.0, 5.5, 6.5, 7.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 3.5, 5.5, 6.5, 4.5, 6.0, 6.5, 7.0, 7.0, 4.5, 7.0, 7.0, 7.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 2.0, 5.5, 6.0, 6.0, 6.0, 6.5, 6.0, 7.0, 7.0, 6.0, 7.0, 7.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.5, 4.5, 5.5, 5.0, 7.0, 7.0, 6.0, 7.0, 7.0, 7.0, 7.0, 7.0) ),
        )
)
\
```


In figures 26-36, rates, CBC/MPA fail fractions, CIC fail fractions, normalized muon and electron efficiencies are shown in the barrel layers and endcap disks for above tunes. In figures 37-40, nominal stub reconstruction efficiencies (not normalized) for muon and electron are shown. Each region of the mentioned 108 geometrical regions of the tracker are shown as an independent bin in x-axis. We have also divided the flat barrel part, although they are counted as one region in stub tuning procedure. In the barrel histograms, first bin is related to the leftmost module and last bin is related to the rightmost module in z-axis. In the endcap histograms, first bin is related to the innermost ring and last bin is related to the outermost ring in r-axis.

In table 2, the total number of stubs are shown for the four available tunes. As it is clear from the last column of the table 2, the new tight tune leads to around 20% less stubs in the first three layers in the barrel compared to the tight tune. Moreover, the stub rate reduction is more significant in the endcap disks (around 30% in all disks).

Table 2: Stub rate for the tight, loose, new tight, and new loose tunes

Barrel layer	Tight tune	loose Tune	New tight tune	New loose tune	new tight / tight
1	2.66e+03	2.66e+03	2.35e+03	2.55e+03	0.884
2	1.9e+03	2.62e+03	1.49e+03	2.72e+03	0.784
3	1.46e+03	2.16e+03	1.22e+03	1.91e+03	0.839
4	1.37e+03	1.78e+03	1.37e+03	1.51e+03	1.0
5	1.14e+03	1.31e+03	1.14e+03	1.39e+03	1.0
6	8.02e+02	8.02e+02	7.54e+02	8.01e+02	0.94
Endcap disks					
1	8.32e+02	1.18e+03	6.03e+02	1.19e+03	0.724
2	9.21e+02	1.38e+03	6.35e+02	1.4e+03	0.689
3	6.89e+02	1.19e+03	5.16e+02	1.36e+03	0.75
4	8.14e+02	1.42e+03	5.31e+02	1.64e+03	0.652
5	8.74e+02	1.51e+03	6.37e+02	1.89e+03	0.729

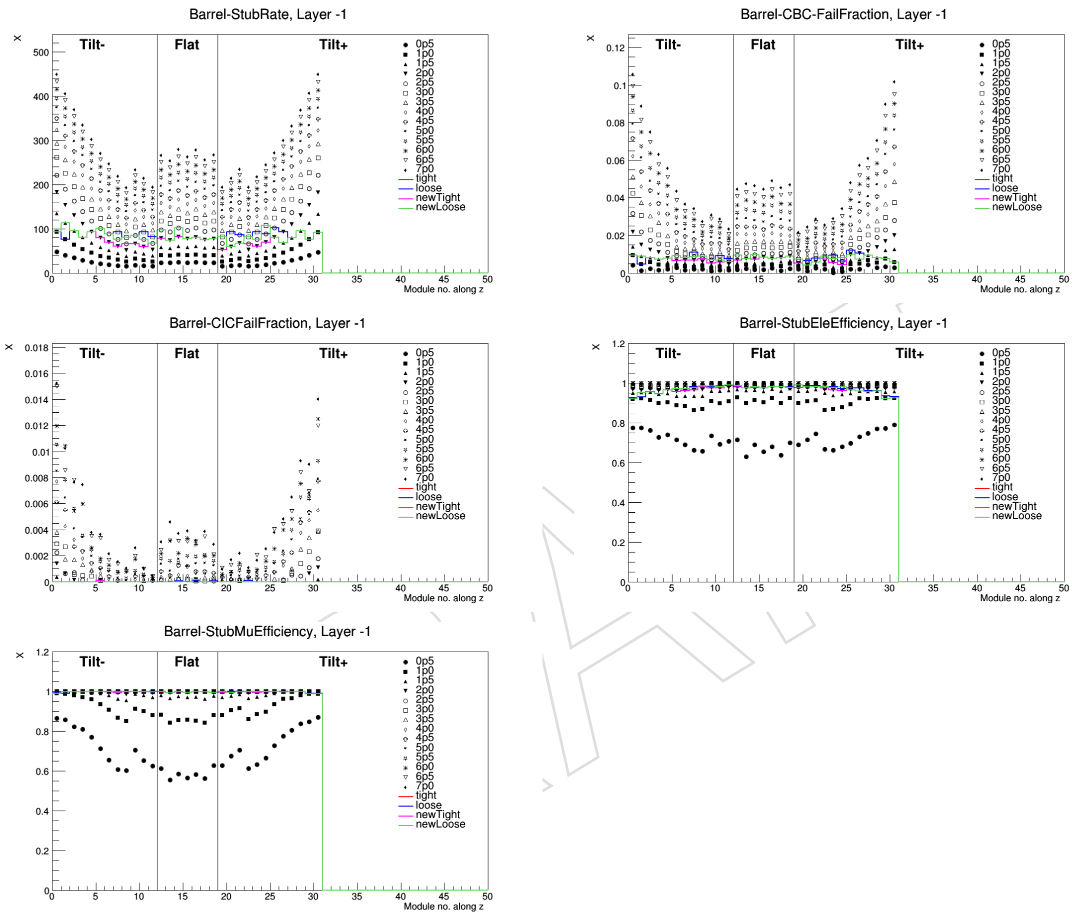


Figure 26: Distributions of stub rate (top left), CBC/MPA fail fraction (top right), CIC fail fraction (middle left), stub efficiency for electron (middle right), and stub efficiency for muon in barrel layer 1.

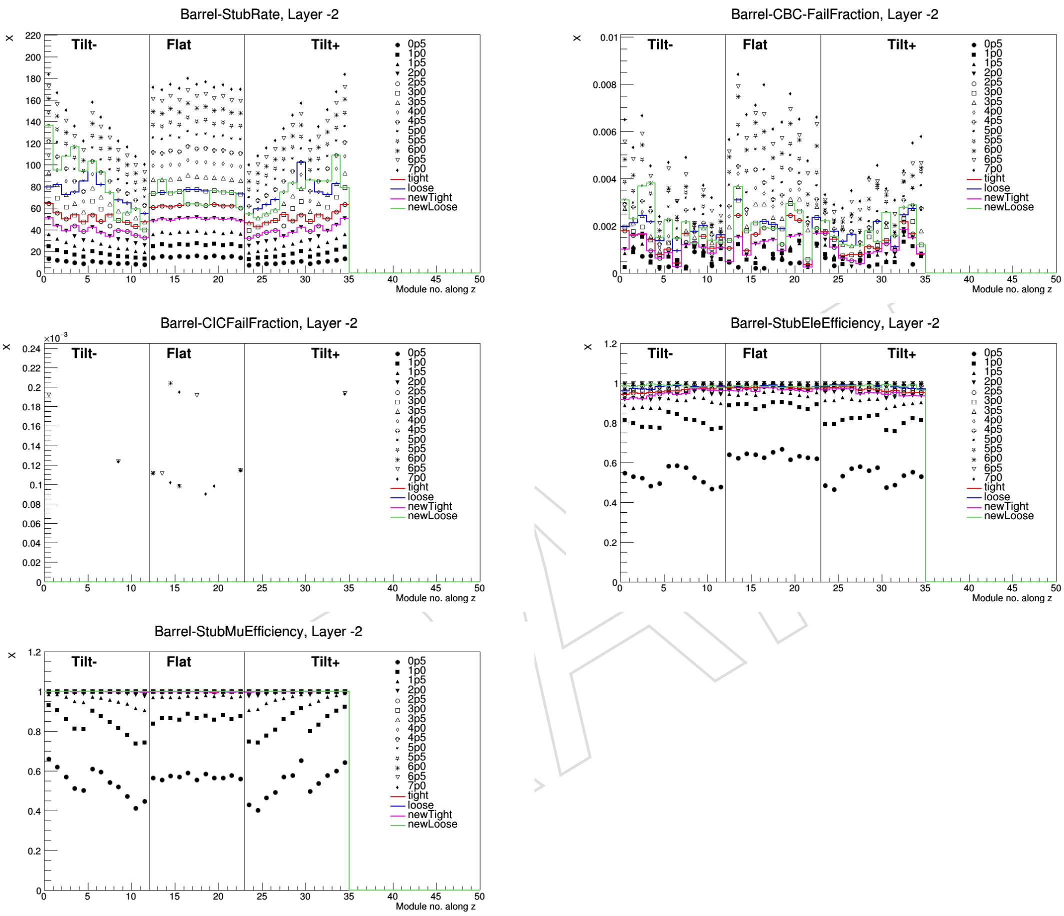


Figure 27: Distributions of stub rate (top left), CBC/MPA fail fraction (top right), CIC fail fraction (middle left), stub efficiency for electron (middle right), and stub efficiency for muon in barrel layer 2.

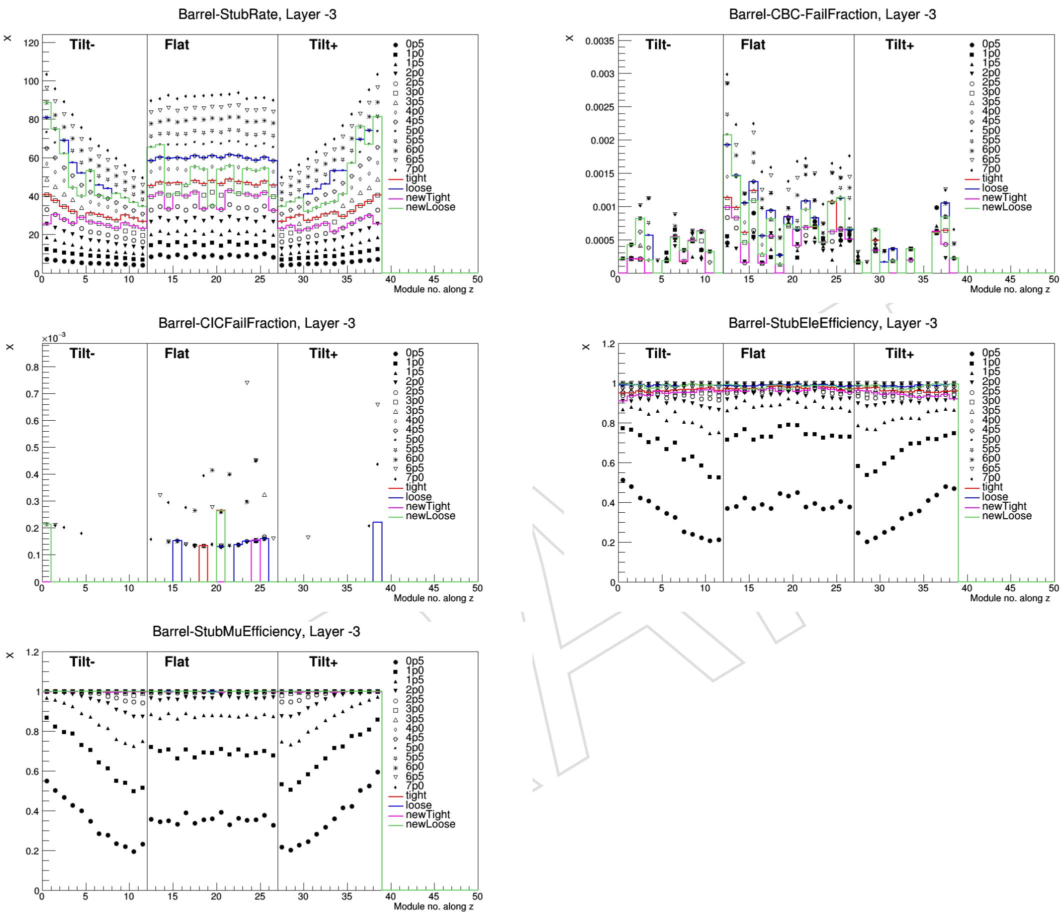


Figure 28: Distributions of stub rate (top left), CBC/MPA fail fraction (top right), CIC fail fraction (middle left), stub efficiency for electron (middle right), and stub efficiency for muon in barrel layer 3.

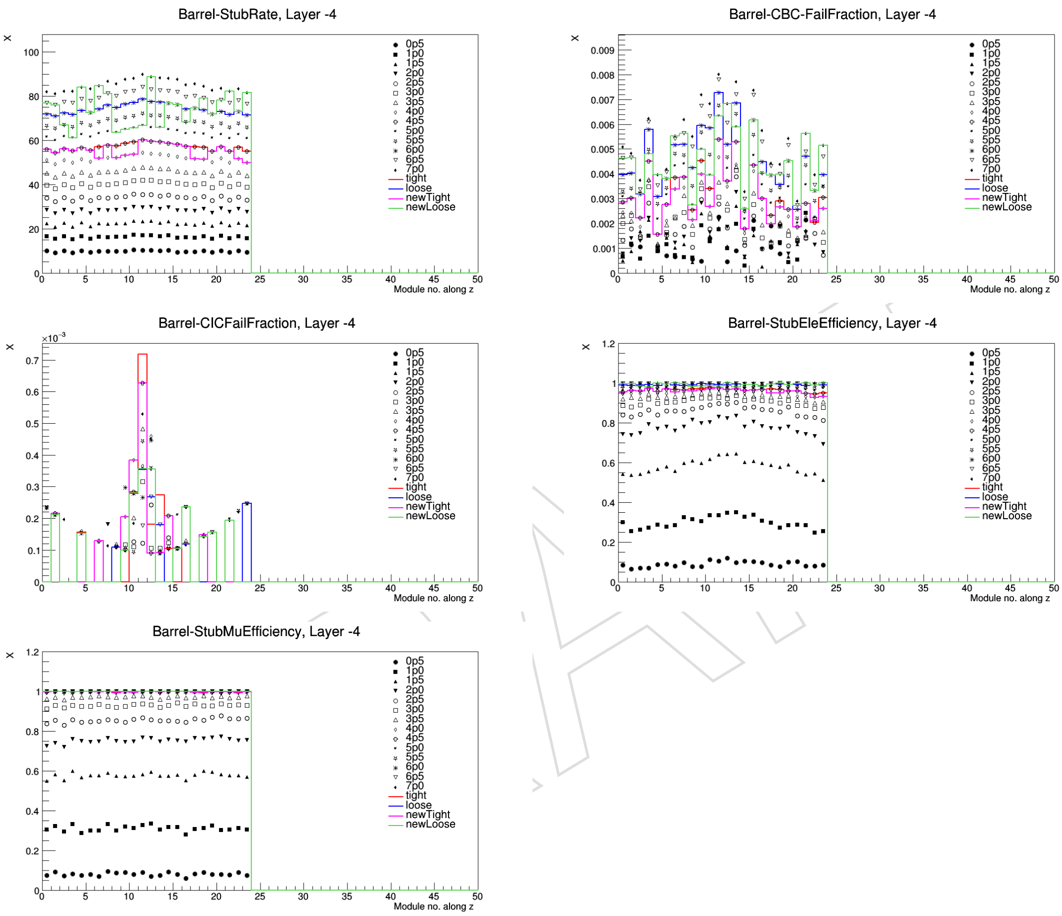


Figure 29: Distributions of stub rate (top left), CBC/MPA fail fraction (top right), CIC fail fraction (middle left), stub efficiency for electron (middle right), and stub efficiency for muon in barrel layer 4.

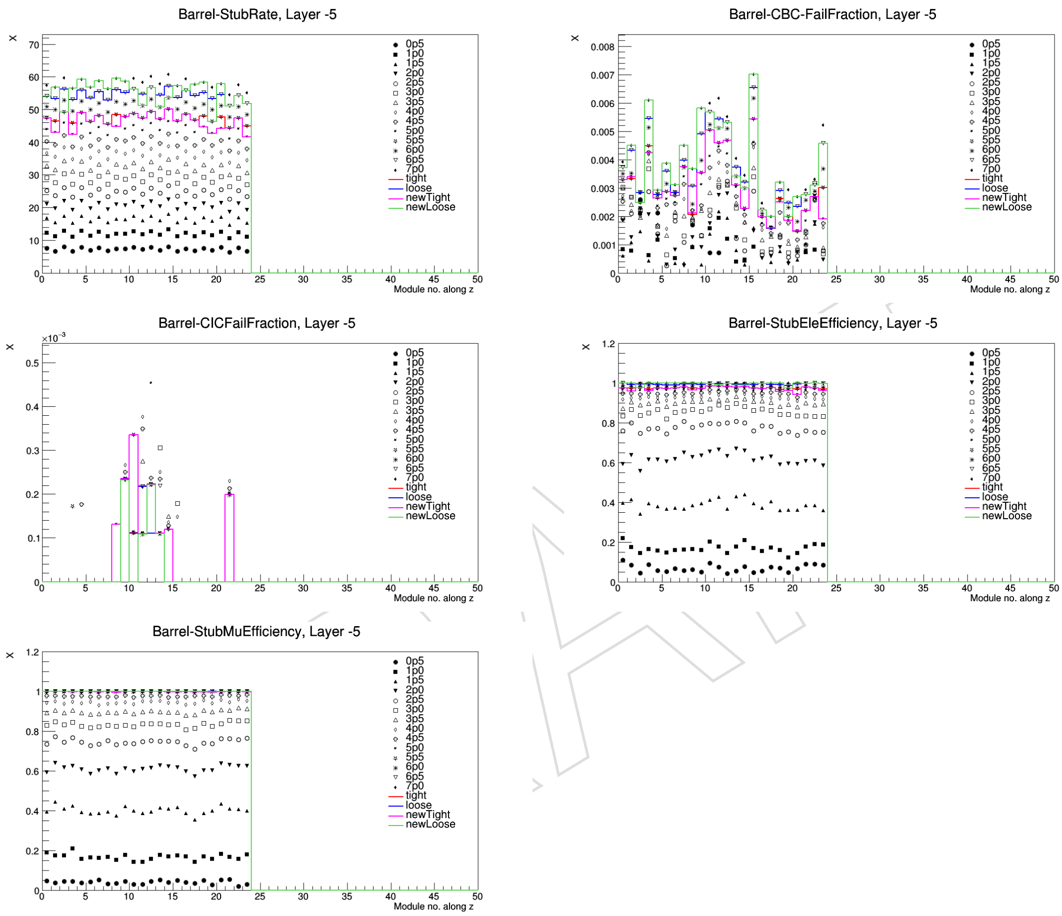


Figure 30: Distributions of stub rate (top left), CBC/MPA fail fraction (top right), CIC fail fraction (middle left), stub efficiency for electron (middle right), and stub efficiency for muon in barrel layer 5.

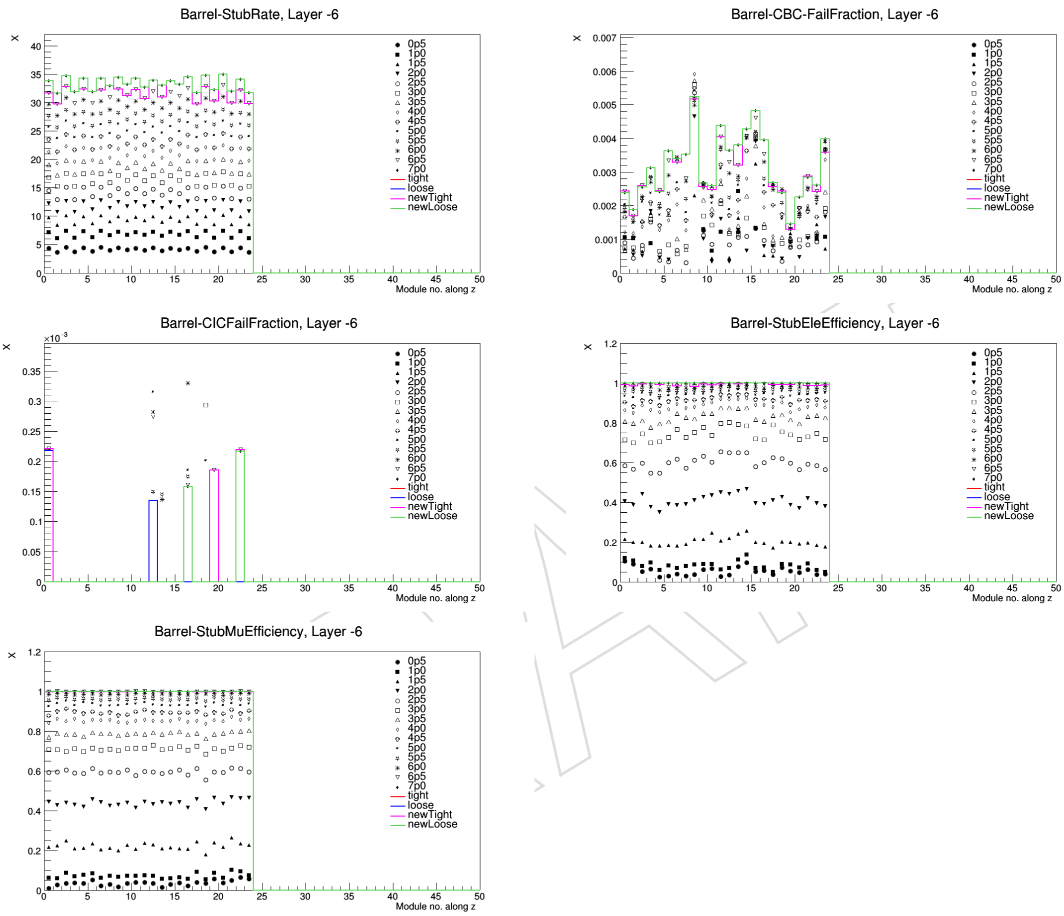


Figure 31: Distributions of stub rate (top left), CBC/MPA fail fraction (top right), CIC fail fraction (middle left), stub efficiency for electron (middle right), and stub efficiency for muon in barrel layer 6.

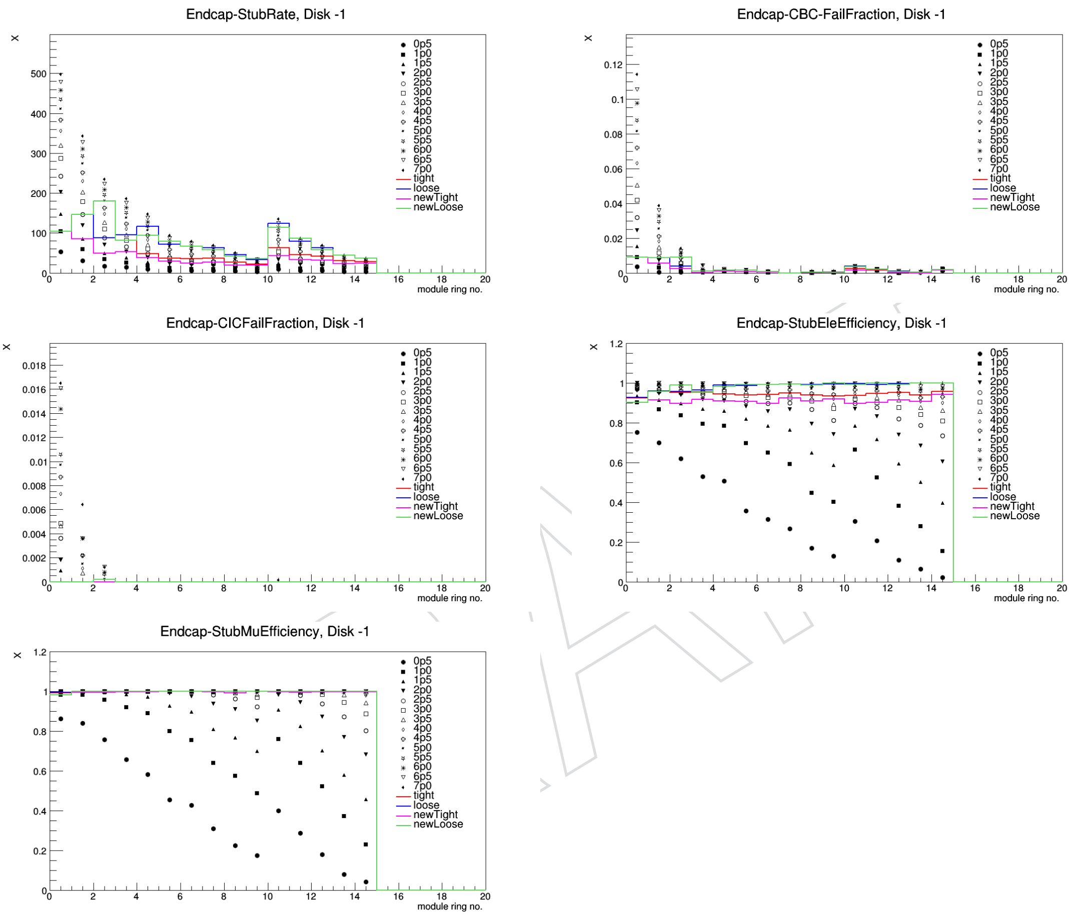


Figure 32: Distributions of stub rate (top left), CBC/MPA fail fraction (top right), CIC fail fraction (middle left), stub efficiency for electron (middle right), and stub efficiency for muon in endcap disk 1.

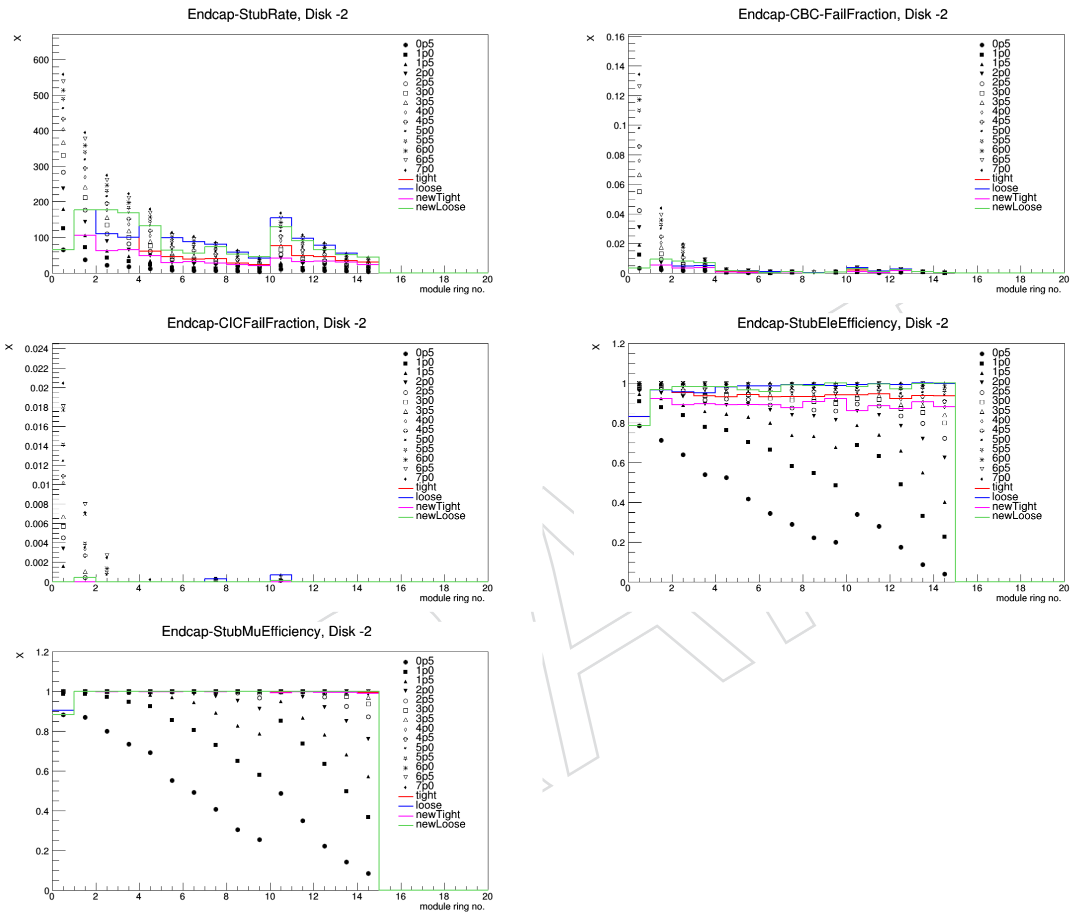


Figure 33: Distributions of stub rate (top left), CBC/MPA fail fraction (top right), CIC fail fraction (middle left), stub efficiency for electron (middle right), and stub efficiency for muon in endcap disk 2.

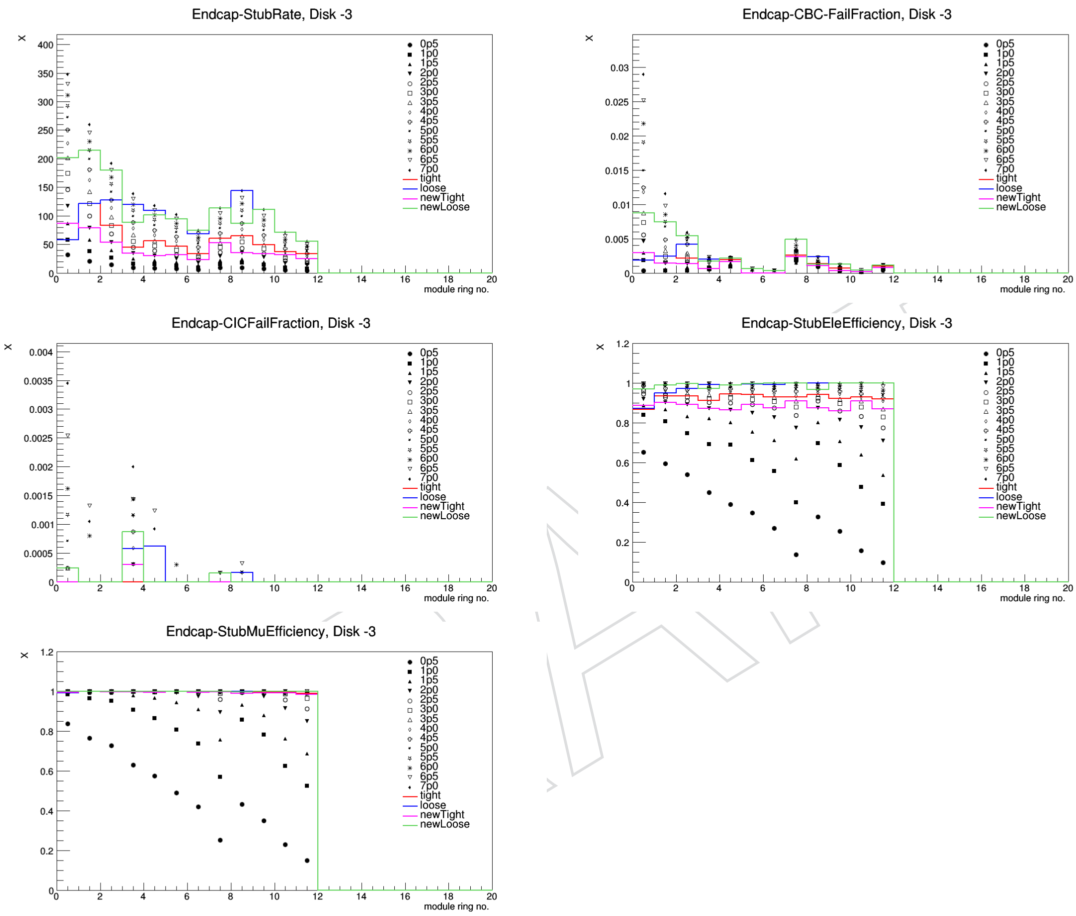


Figure 34: Distributions of stub rate (top left), CBC/MPA fail fraction (top right), CIC fail fraction (middle left), stub efficiency for electron (middle right), and stub efficiency for muon in endcap disk 3.

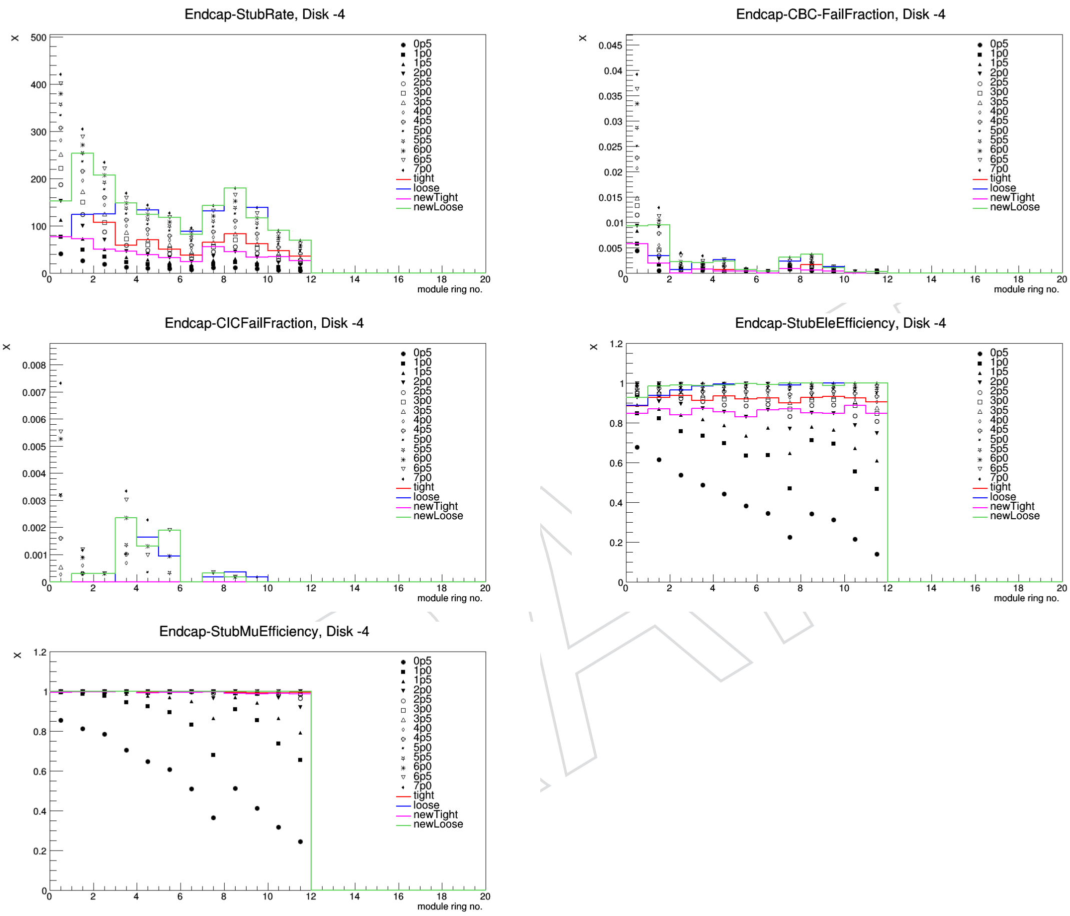


Figure 35: Distributions of stub rate (top left), CBC/MPA fail fraction (top right), CIC fail fraction (middle left), stub efficiency for electron (middle right), and stub efficiency for muon in endcap disk 4.

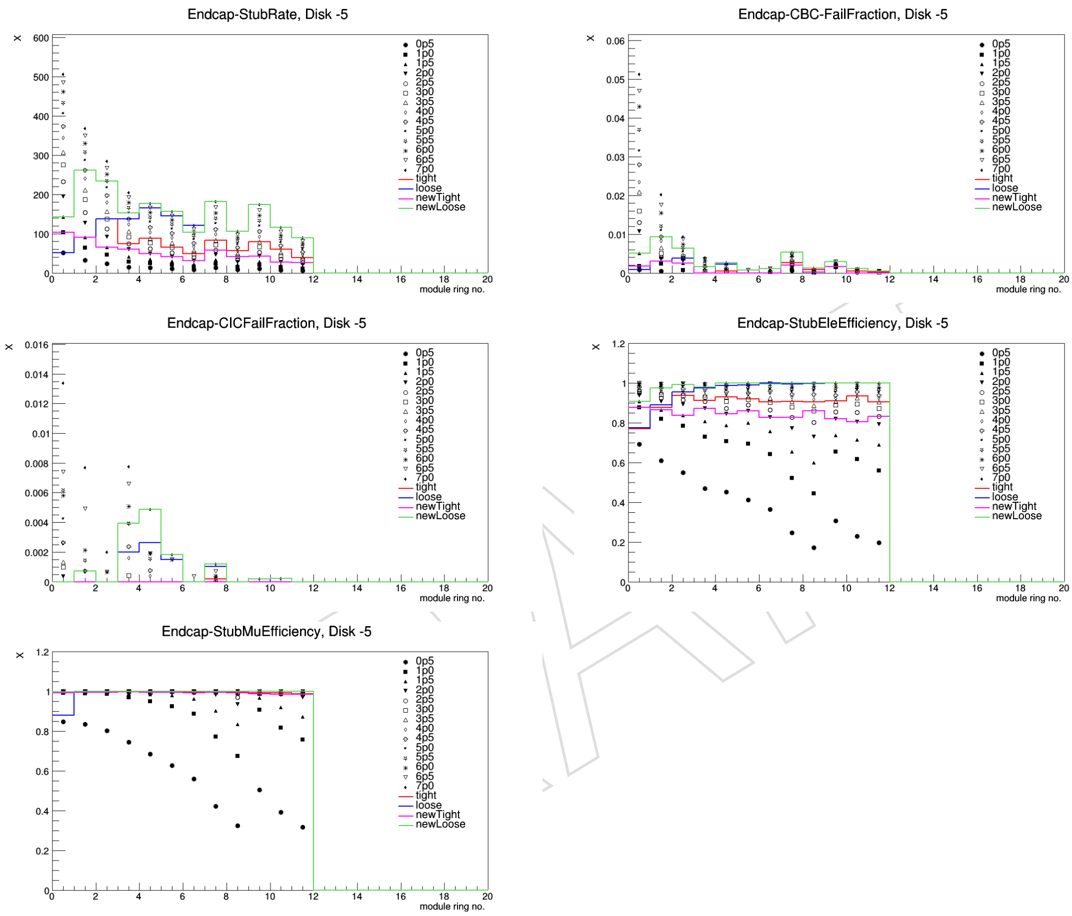


Figure 36: Distributions of stub rate (top left), CBC/MPA fail fraction (top right), CIC fail fraction (middle left), stub efficiency for electron (middle right), and stub efficiency for muon in endcap disk 5.

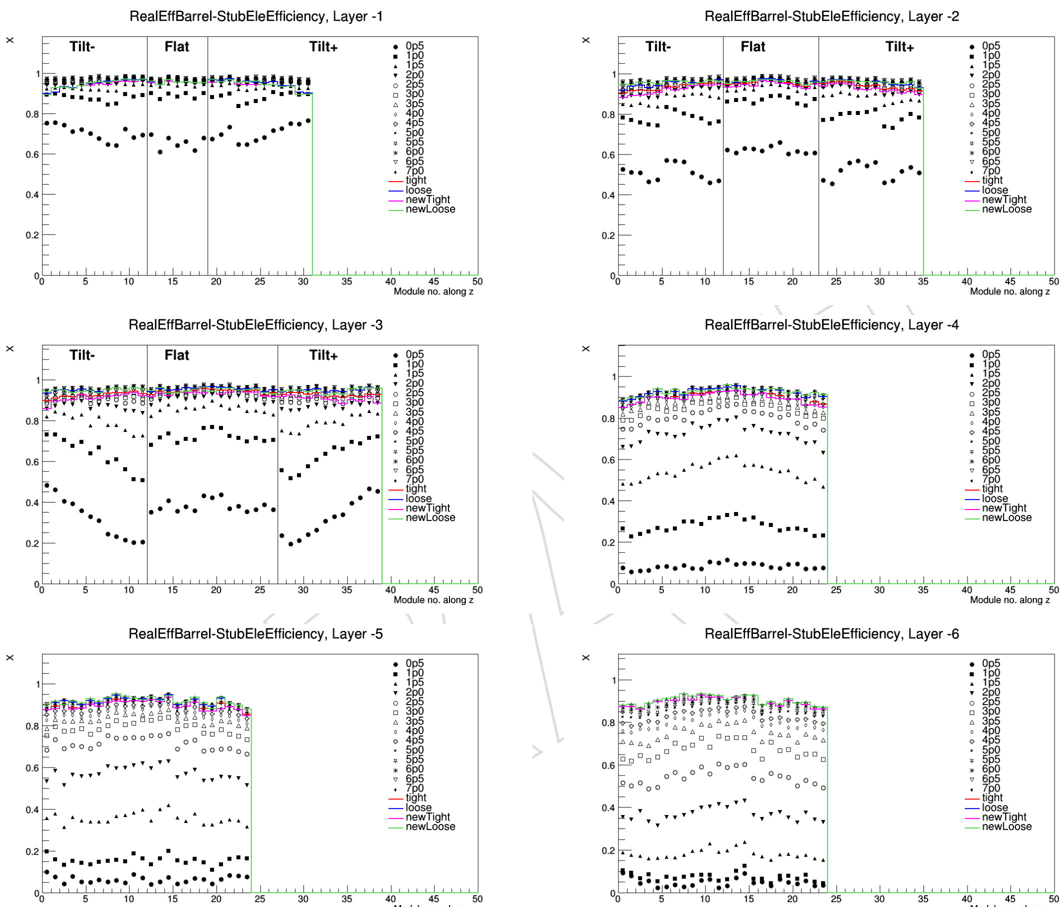


Figure 37: Distributions of stub efficiency for electron in Barrel layers.

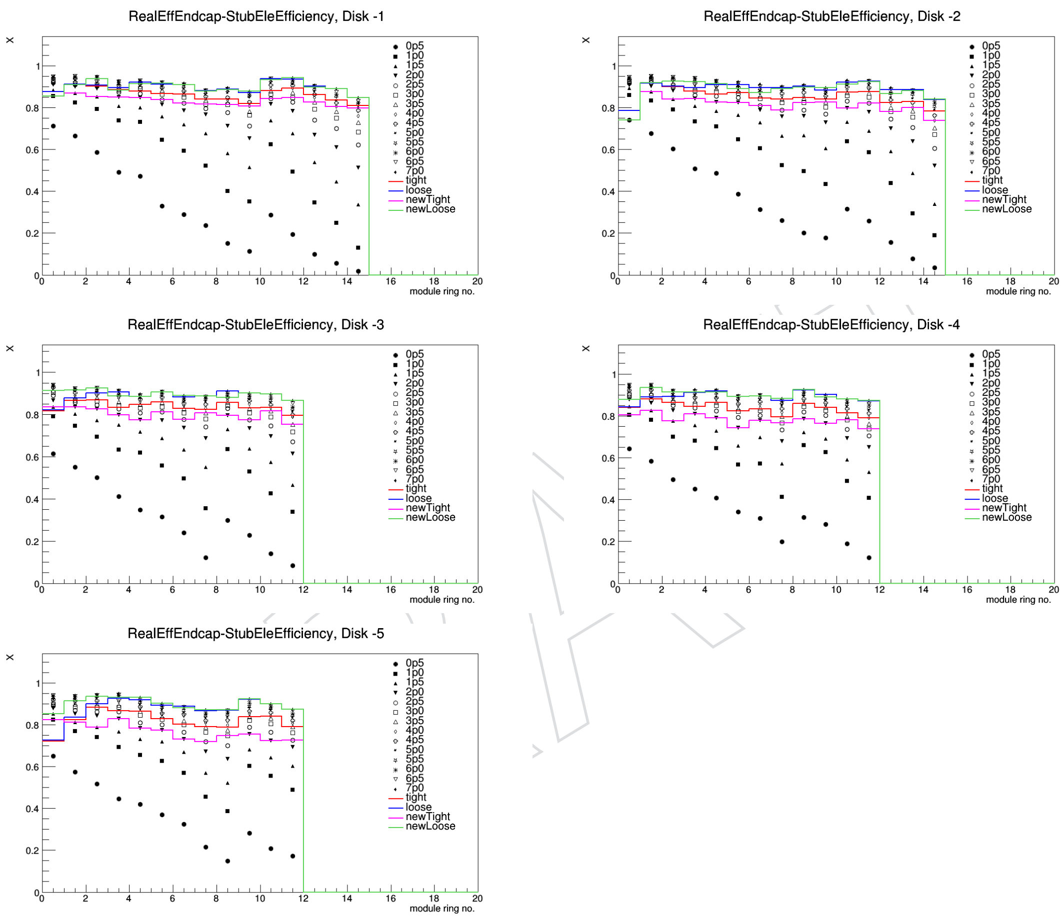


Figure 38: Distributions of stub efficiency for electron in endcaps layers.

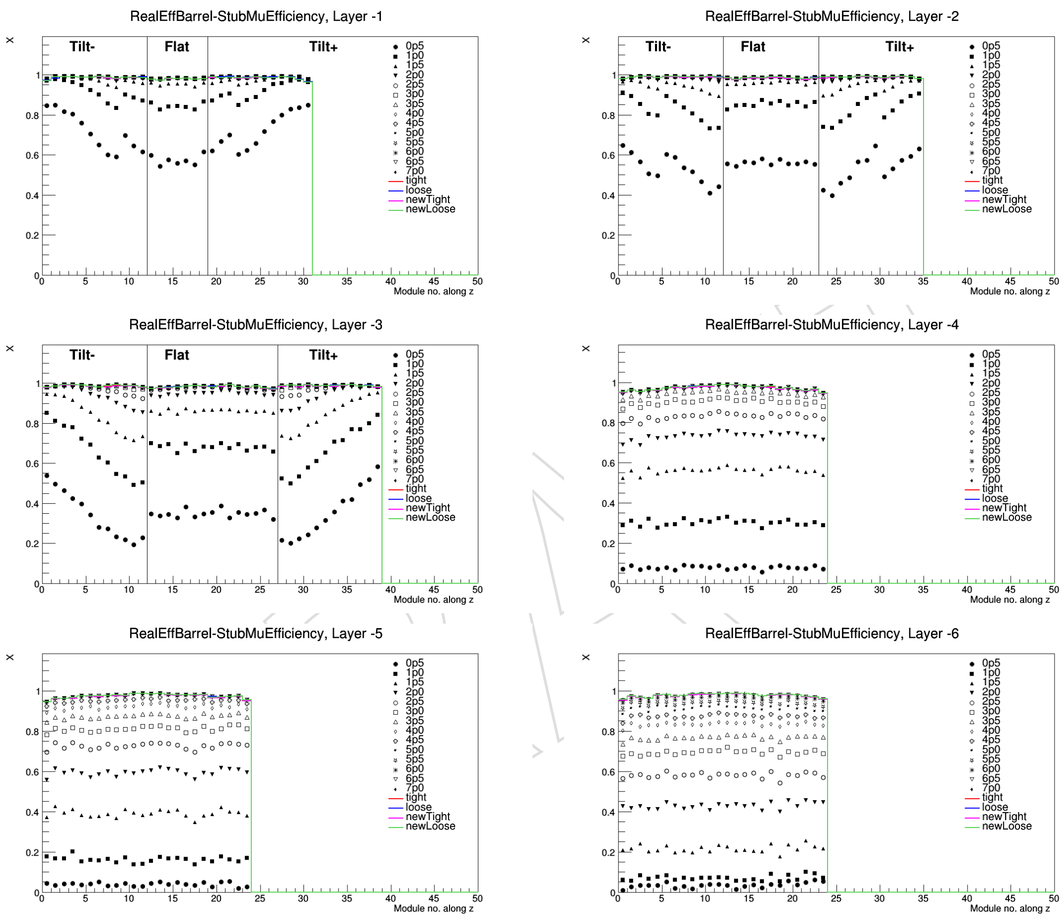


Figure 39: Distributions of stub efficiency for muon in Barrel layers.

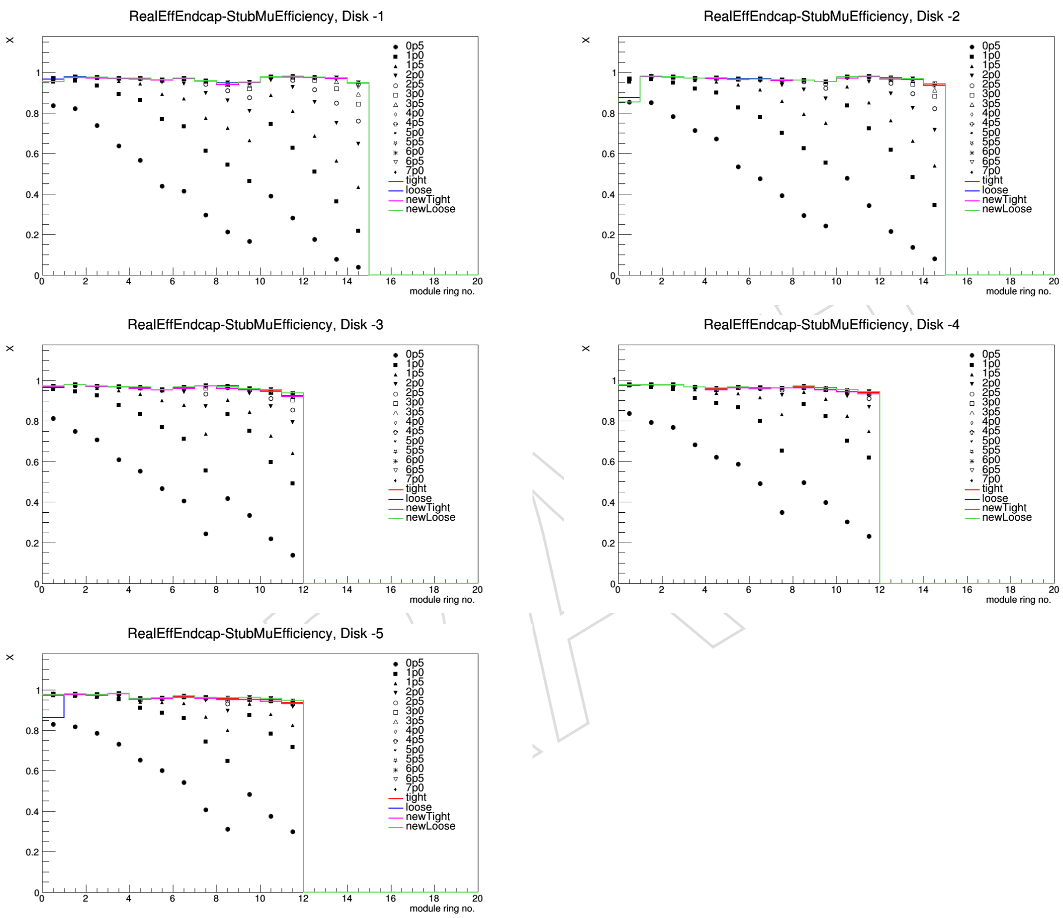


Figure 40: Distributions of stub efficiency for muon in endcaps layers.

7 Validation of the new stub window tunes

As shown in table 2, the new tight tune leads to the lowest stub rates while keeping the fail rates small enough. So it could be used instead of the current tune. Before using the new tune, we need to check the effect of the new tunes on the L1 track reconstruction efficiencies. In addition, since we have not included the displaced muon stub efficiency in the stub window optimisation procedure, we need to make sure that the L1 track efficiency is not degraded for displaced tracks.

In order to measure the track reconstruction efficiencies for different stub window tunes we have used the nominal L1 tracking Ntuple maker [1] with the following changes:

```
in the L1Trigger/TrackFindingTracklet/test/L1TrackNtupleMaker_cfg.py file
     TP_minNStub = cms.int32(0), # require TP to have >= X number of stubs associated with it
     TP_minNStubLayer = cms.int32(0), # require TP to have stubs in >= X layers/disks
     MCTruthClusterInputTag = cms.InputTag("TTClusterAssociatorFromPixelDigis", "ClusterInclusive")
```

In figures 41-43, track reconstruction efficiencies are shown for electron, muon, and ttbar general tracks. All four available tunes are shown for two cases, with or without the 'calcbendcut' (see section 5). In general, the old and new tunes show similar tracking efficiencies. Although the tight tune shows a bit higher efficiency (0.2%) compared to the new tight tune. It is worth mentioning that the 'calcbendfix' works since without the calcbend fix, the loose tunes have higher tracking efficiencies compared to the tight tunes.

In order to measure the L1 track reconstruction efficiency for displaced muon samples, we need to apply the following modifications in the L1 code;

```
in the L1Trigger/TrackFindingTracklet/test/L1TrackNtupleMaker_cfg.py file
      L1TRKALGO = 'HYBRID_DISPLACED'
in the L1Trigger/TrackFindingTracklet/test/L1TrackNtuplePlot.C file
      TP_minPt = 3.0,  TP_maxEta = 2, TP_maxDxy = 10.0,  TP_maxD0 = 10.0
```

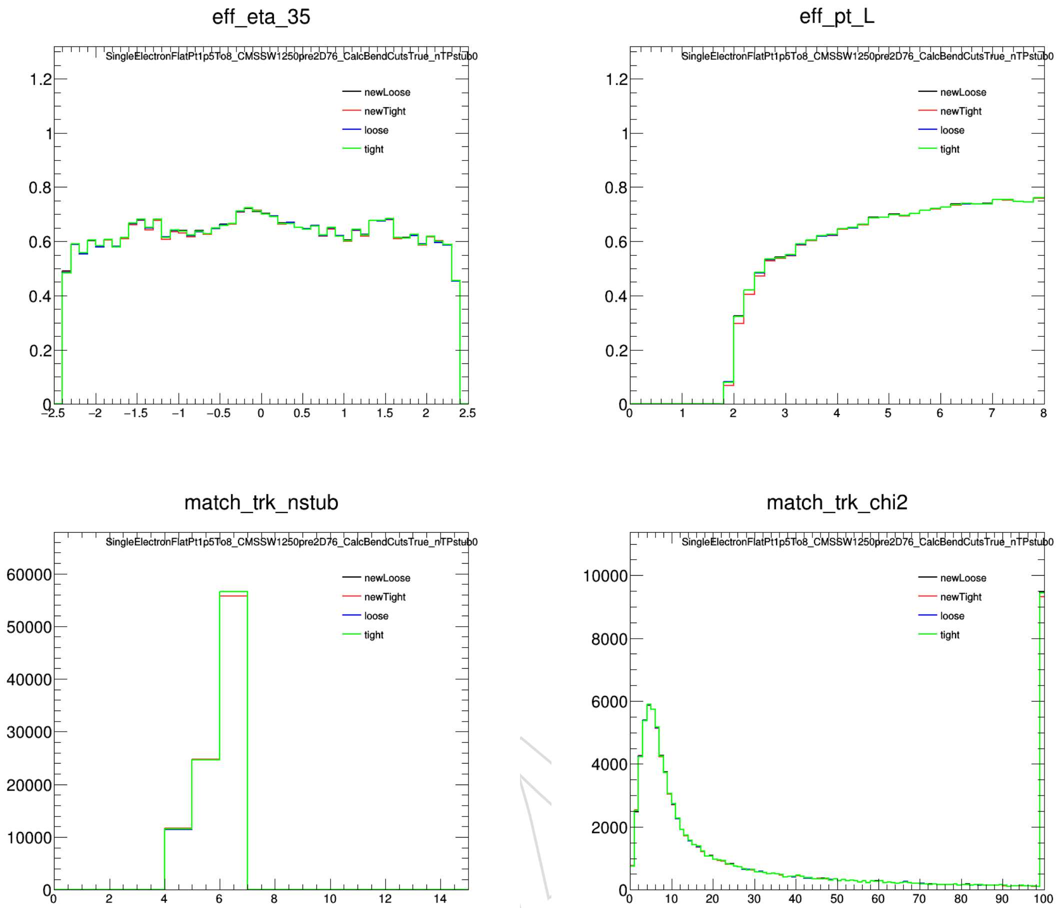
In figure 45, track reconstruction efficiencies are shown for displaced muon tracks. All four available tunes are shown for two cases, with or without the 'calcbendcut' (see section 5). The new tight tune shows lower efficiency ($\approx$8%) compared to the tight tune. In figure 45, track reconstruction efficiencies are shown as a function of the track impact parameters (d_0 and z_0). The observed inefficiencies are related to the tracks with large d_0 ($>$ 3 cm). In order to find the origin of the observed inefficiencies, we looked at the stub reconstruction efficiencies for displaced muons (see section 4). The stub reconstruction efficiencies for all displaced muon and displaced muon with $d_0 < 3$ cm are shown in figures 46 and 46 for barrel layers and endcap disks. The most affected regions are barrel layers 2 and 3 the most inner rings in the endcap disks. The L1 tracking efficiency for displaced muon as a function of η (see figure 44) shows that the L1 tracking efficiency at high η are not highly affected. Therefore the L1 track efficiencies for displaced muons are affected by tighter window sizes in barrel layers 2 and 3 in the new tight tune compared to the tight tune. We found that by widening the stub window sizes in barrel layers 2 and 3 of the new tight tune, we can restore the efficiency. Therefore, we made a new tune called "Modified tight tune" as the following

```
# New modified tight tune based on simulated events CMSSW_11_3_0_pre3, D76
    BarrelCut    = cms.vdouble(0, 2.0, 2.5, 3.5, 4.0, 5.5, 6.5),
    TiltedBarrelCutSet = cms.VPSet(
        cms.PSet( TiltedCut = cms.vdouble( 0 ) ),
        cms.PSet( TiltedCut = cms.vdouble( 0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 1.5, 1.5, 1.5, 1.0, 1.0) ),
        cms.PSet( TiltedCut = cms.vdouble( 0, 3.0, 3.0, 3.0, 3.0, 3.0, 2.5, 2.5, 3.0, 3.0, 2.5, 2.5, 2.5) ),
        cms.PSet( TiltedCut = cms.vdouble(0, 4.0, 4.0, 4.0, 3.5, 3.5, 3.5, 3.0, 3.0, 2.5, 2.5, 2.5, 2.5) ),
        ),
    EndcapCutSet = cms.VPSet(
```

[1] https://twiki.cern.ch/twiki/bin/view/CMS/L1TrackSoftware

```
        cms.PSet( EndcapCut = cms.vdouble( 0 ) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.0, 1.5, 1.5, 2.0, 2.0, 2.5, 2.5, 3.0, 4.0, 4.0, 2.5, 3.0, 3.5, 4.0, 5.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 0.5, 1.5, 1.5, 2.0, 2.0, 2.0, 2.5, 2.5, 3.0, 3.5, 2.0, 2.5, 3.0, 4.0, 4.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.5, 2.0, 2.0, 2.0, 2.0, 2.5, 3.0, 3.5, 2.5, 2.5, 3.0, 3.5) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.0, 1.5, 1.5, 2.0, 2.0, 2.0, 2.0, 3.0, 2.0, 2.0, 3.0, 3.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.0, 1.5, 1.5, 2.0, 2.0, 2.0, 2.0, 2.5, 3.0, 2.0, 2.0, 2.5) ),
        )
\
```

In figure 48, L1 track reconstruction efficiencies are shown as a function of the track parameter including the new modified tight tune. The efficiency loss at high d_0 are restored. So the new modified tight tune passes all validation tests.



<table>
<tr><th rowspan="2">Calc Bend Cuts</th><th colspan="4">efficiency</th></tr>
<tr><th>newLoose</th><th>newTight</th><th>loose</th><th>tight</th></tr>
<tr><td>True</td><td>efficiency for |eta| < 1.0 = 68.3383 +- 0.18996
efficiency for 1.0 < |eta| < 1.75 = 66.2133 +- 0.225373
efficiency for 1.75 < |eta| < 2.4 = 59.8161 +- 0.252371
combined efficiency for |eta| < 2.4 = 65.409 +- 0.126342 = 92714/141745

efficiency for pt > 2 = 65.409 +- 0.126342
efficiency for 2 < pt < 8.0 = 65.409 +- 0.126342
TP/event (pt > 2) = 1.41745
TP/event (pt > 3.0) = 1.18038
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 0.9725
tracks/event (pt > 3.0) = 0.75781
tracks/event (pt > 10.0) = 0.00027</td><td>efficiency for |eta| < 1.0 = 67.818 +- 0.190784
efficiency for 1.0 < |eta| < 1.75 = 65.7365 +- 0.226139
efficiency for 1.75 < |eta| < 2.4 = 59.7949 +- 0.252393
combined efficiency for |eta| < 2.4 = 65.0351 +- 0.126659 = 92184/141745

efficiency for pt > 2 = 65.0351 +- 0.126659
efficiency for 2 < pt < 8.0 = 65.0351 +- 0.126659
TP/event (pt > 2) = 1.41745
TP/event (pt > 3.0) = 1.18038
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 0.9675
tracks/event (pt > 3.0) = 0.7566
tracks/event (pt > 10.0) = 0.00023</td><td>efficiency for |eta| < 1.0 = 68.3049 +- 0.190013
efficiency for 1.0 < |eta| < 1.75 = 66.1997 +- 0.225395
efficiency for 1.75 < |eta| < 2.4 = 59.7075 +- 0.252483
combined efficiency for |eta| < 2.4 = 65.3617 +- 0.126382 = 92647/141745

efficiency for pt > 2 = 65.3617 +- 0.126382
efficiency for 2 < pt < 8.0 = 65.3617 +- 0.126382
TP/event (pt > 2) = 1.41745
TP/event (pt > 3.0) = 1.18038
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 0.97177
tracks/event (pt > 3.0) = 0.75756
tracks/event (pt > 10.0) = 0.00025</td><td>efficiency for |eta| < 1.0 = 68.2599 +- 0.190085
efficiency for 1.0 < |eta| < 1.75 = 66.2065 +- 0.225384
efficiency for 1.75 < |eta| < 2.4 = 59.8585 +- 0.252327
combined efficiency for |eta| < 2.4 = 65.385 +- 0.126362 = 92680/141745

efficiency for pt > 2 = 65.385 +- 0.126362
efficiency for 2 < pt < 8.0 = 65.385 +- 0.126362
TP/event (pt > 2) = 1.41745
TP/event (pt > 3.0) = 1.18038
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 0.97218
tracks/event (pt > 3.0) = 0.75738
tracks/event (pt > 10.0) = 0.00025</td></tr>
<tr><td>False</td><td>efficiency for |eta| < 1.0 = 67.3527 +- 0.191498
efficiency for 1.0 < |eta| < 1.75 = 65.9704 +- 0.225767
efficiency for 1.75 < |eta| < 2.4 = 59.8029 +- 0.252385
combined efficiency for |eta| < 2.4 = 64.913 +- 0.126761 = 92011/141745

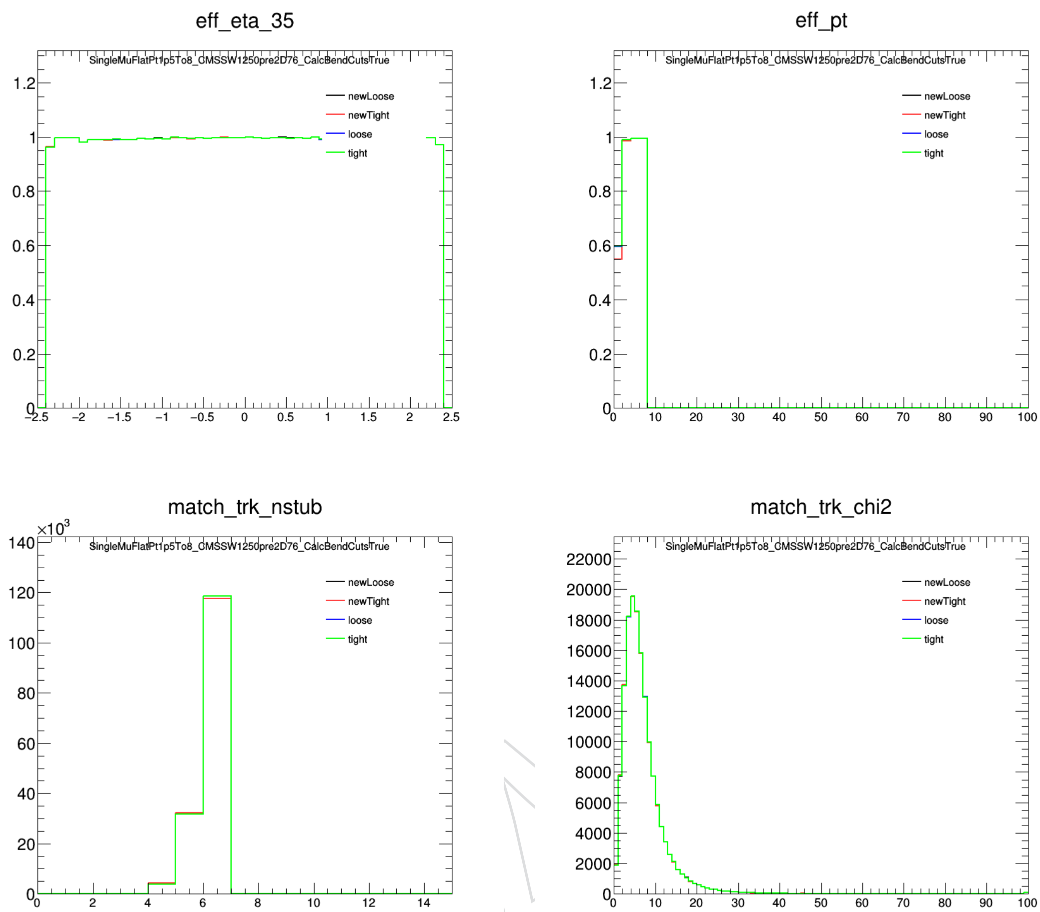
efficiency for pt > 2 = 64.913 +- 0.126761
efficiency for 2 < pt < 8.0 = 64.913 +- 0.126761
TP/event (pt > 2) = 1.41745
TP/event (pt > 3.0) = 1.18038
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 0.96595
tracks/event (pt > 3.0) = 0.75645
tracks/event (pt > 10.0) = 0.00028</td><td>efficiency for |eta| < 1.0 = 67.8063 +- 0.190802
efficiency for 1.0 < |eta| < 1.75 = 65.6934 +- 0.226207
efficiency for 1.75 < |eta| < 2.4 = 59.726 +- 0.252464
combined efficiency for |eta| < 2.4 = 64.9984 +- 0.12669 = 92132/141745

efficiency for pt > 2 = 64.9984 +- 0.12669
efficiency for 2 < pt < 8.0 = 64.9984 +- 0.12669
TP/event (pt > 2) = 1.41745
TP/event (pt > 3.0) = 1.18038
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 0.96663
tracks/event (pt > 3.0) = 0.75567
tracks/event (pt > 10.0) = 0.00026</td><td>efficiency for |eta| < 1.0 = 67.6845 +- 0.190991
efficiency for 1.0 < |eta| < 1.75 = 65.9749 +- 0.225759
efficiency for 1.75 < |eta| < 2.4 = 59.6465 +- 0.252544
combined efficiency for |eta| < 2.4 = 65.0132 +- 0.126677 = 92153/141745

efficiency for pt > 2 = 65.0132 +- 0.126677
efficiency for 2 < pt < 8.0 = 65.0132 +- 0.126677
TP/event (pt > 2) = 1.41745
TP/event (pt > 3.0) = 1.18038
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 0.96815
tracks/event (pt > 3.0) = 0.75831
tracks/event (pt > 10.0) = 0.00026</td><td>efficiency for |eta| < 1.0 = 68.2866 +- 0.190043
efficiency for 1.0 < |eta| < 1.75 = 66.1838 +- 0.225421
efficiency for 1.75 < |eta| < 2.4 = 59.7896 +- 0.252399
combined efficiency for |eta| < 2.4 = 65.3709 +- 0.126374 = 92660/141745

efficiency for pt > 2 = 65.3709 +- 0.126374
efficiency for 2 < pt < 8.0 = 65.3709 +- 0.126374
TP/event (pt > 2) = 1.41745
TP/event (pt > 3.0) = 1.18038
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 0.97179
tracks/event (pt > 3.0) = 0.75686
tracks/event (pt > 10.0) = 0.00027</td></tr>
</table>

Figure 41: L1 track reconstruction efficiency for electrons with four available tunes and the ‘calcBendCut’ variable true or false .



<table>
<tr><th rowspan="2">Calc Bend Cuts</th><th colspan="4">efficiency</th></tr>
<tr><th>newLoose</th><th>newTight</th><th>loose</th><th>tight</th></tr>
<tr><td>True</td>
<td>efficiency for |eta| < 1.0 = 99.5461 +- 0.0263692
efficiency for 1.0 < |eta| < 1.75 = 99.2919 +- 0.0379305
efficiency for 1.75 < |eta| < 2.4 = 98.9149 +- 0.0509313
combined efficiency for |eta| < 2.4 = 99.2978 +- 0.0211936 = 154141/155231

efficiency for pt > 2 = 99.2978 +- 0.0211936
efficiency for 2 < pt < 8.0 = 99.2978 +- 0.0211936
TP/event (pt > 2) = 1.55231
TP/event (pt > 3.0) = 1.29356
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 1.56939
tracks/event (pt > 3.0) = 1.31165
tracks/event (pt > 10.0) = 0</td>
<td>efficiency for |eta| < 1.0 = 99.3429 +- 0.0316924
efficiency for 1.0 < |eta| < 1.75 = 99.0853 +- 0.0430677
efficiency for 1.75 < |eta| < 2.4 = 98.8835 +- 0.0516552
combined efficiency for |eta| < 2.4 = 99.1393 +- 0.0234449 = 153895/155231

efficiency for pt > 2 = 99.1393 +- 0.0234449
efficiency for 2 < pt < 8.0 = 99.1393 +- 0.0234449
TP/event (pt > 2) = 1.55231
TP/event (pt > 3.0) = 1.29356
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 1.56722
tracks/event (pt > 3.0) = 1.31123
tracks/event (pt > 10.0) = 0</td>
<td>efficiency for |eta| < 1.0 = 99.563 +- 0.025875
efficiency for 1.0 < |eta| < 1.75 = 99.3042 +- 0.0376026
efficiency for 1.75 < |eta| < 2.4 = 98.9149 +- 0.0509313
combined efficiency for |eta| < 2.4 = 99.3088 +- 0.0210289 = 154158/155231

efficiency for pt > 2 = 99.3088 +- 0.0210289
efficiency for 2 < pt < 8.0 = 99.3088 +- 0.0210289
TP/event (pt > 2) = 1.55231
TP/event (pt > 3.0) = 1.29356
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 1.56939
tracks/event (pt > 3.0) = 1.3113
tracks/event (pt > 10.0) = 0</td>
<td>efficiency for |eta| < 1.0 = 99.5568 +- 0.0260559
efficiency for 1.0 < |eta| < 1.75 = 99.2878 +- 0.0380391
efficiency for 1.75 < |eta| < 2.4 = 98.9342 +- 0.0504805
combined efficiency for |eta| < 2.4 = 99.3062 +- 0.0210677 = 154154/155231

efficiency for pt > 2 = 99.3062 +- 0.0210677
efficiency for 2 < pt < 8.0 = 99.3062 +- 0.0210677
TP/event (pt > 2) = 1.55231
TP/event (pt > 3.0) = 1.29356
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 1.56983
tracks/event (pt > 3.0) = 1.31159
tracks/event (pt > 10.0) = 0</td></tr>
<tr><td>False</td>
<td>efficiency for |eta| < 1.0 = 99.0983 +- 0.0370813
efficiency for 1.0 < |eta| < 1.75 = 99.1139 +- 0.042394
efficiency for 1.75 < |eta| < 2.4 = 98.8786 +- 0.0517655
combined efficiency for |eta| < 2.4 = 99.0447 +- 0.0246892 = 153748/155231

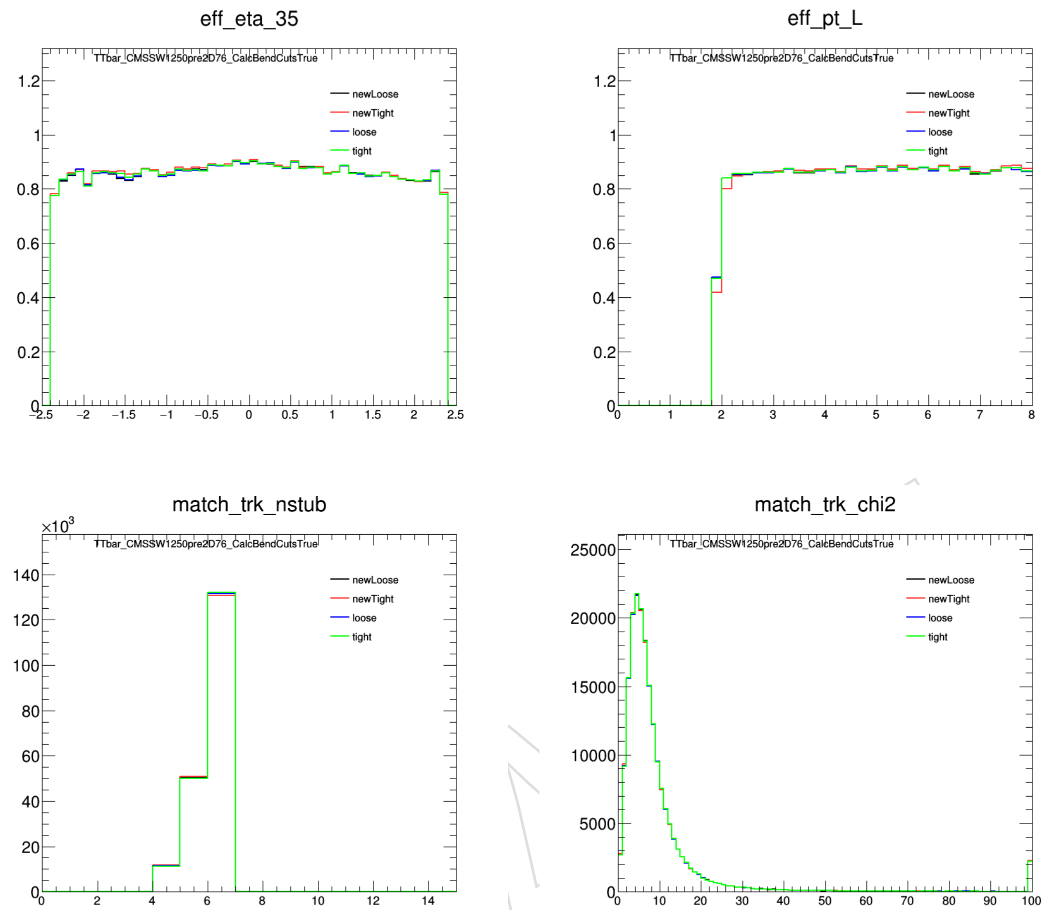
efficiency for pt > 2 = 99.0447 +- 0.0246892
efficiency for 2 < pt < 8.0 = 99.0447 +- 0.0246892
TP/event (pt > 2) = 1.55231
TP/event (pt > 3.0) = 1.29356
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 1.56565
tracks/event (pt > 3.0) = 1.31119
tracks/event (pt > 10.0) = 0</td>
<td>efficiency for |eta| < 1.0 = 99.3229 +- 0.0321681
efficiency for 1.0 < |eta| < 1.75 = 99.0382 +- 0.0441514
efficiency for 1.75 < |eta| < 2.4 = 98.7167 +- 0.0553315
combined efficiency for |eta| < 2.4 = 99.0717 +- 0.0243404 = 153790/155231

efficiency for pt > 2 = 99.0717 +- 0.0243404
efficiency for 2 < pt < 8.0 = 99.0717 +- 0.0243404
TP/event (pt > 2) = 1.55231
TP/event (pt > 3.0) = 1.29356
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 1.56622
tracks/event (pt > 3.0) = 1.31025
tracks/event (pt > 10.0) = 0</td>
<td>efficiency for |eta| < 1.0 = 99.126 +- 0.0365124
efficiency for 1.0 < |eta| < 1.75 = 99.0771 +- 0.0432581
efficiency for 1.75 < |eta| < 2.4 = 98.7771 +- 0.0540299
combined efficiency for |eta| < 2.4 = 99.0176 +- 0.025033 = 153706/155231

efficiency for pt > 2 = 99.0176 +- 0.025033
efficiency for 2 < pt < 8.0 = 99.0176 +- 0.025033
TP/event (pt > 2) = 1.55231
TP/event (pt > 3.0) = 1.29356
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 1.56522
tracks/event (pt > 3.0) = 1.31104
tracks/event (pt > 10.0) = 0</td>
<td>efficiency for |eta| < 1.0 = 99.5722 +- 0.0256014
efficiency for 1.0 < |eta| < 1.75 = 99.2674 +- 0.0385779
efficiency for 1.75 < |eta| < 2.4 = 98.7554 +- 0.0545023
combined efficiency for |eta| < 2.4 = 99.2585 +- 0.0217742 = 154080/155231

efficiency for pt > 2 = 99.2585 +- 0.0217742
efficiency for 2 < pt < 8.0 = 99.2585 +- 0.0217742
TP/event (pt > 2) = 1.55231
TP/event (pt > 3.0) = 1.29356
TP/event (pt > 10.0) = 0
tracks/event (pt > 2) = 1.56916
tracks/event (pt > 3.0) = 1.31092
tracks/event (pt > 10.0) = 0</td></tr>
</table>

Figure 42: L1 track reconstruction efficiency for muons with four available tunes and the ‘calcBendCut’ variable true or false.



<table>
<tr><th rowspan="2">Calc Bend Cuts</th><th colspan="4">efficiency</th></tr>
<tr><th>newLoose</th><th>newTight</th><th>loose</th><th>tight</th></tr>
<tr><td>True</td>
<td>efficiency for |eta| < 1.0 = 87.6161 +- 0.0949411
efficiency for 1.0 < |eta| < 1.75 = 85.2469 +- 0.137576
efficiency for 1.75 < |eta| < 2.4 = 83.4776 +- 0.193125
combined efficiency for |eta| < 2.4 = 86.2288 +- 0.0728418 = 192981/223801

efficiency for pt > 2 = 86.2288 +- 0.0728418
efficiency for 2 < pt < 8.0 = 86.1797 +- 0.0840724
efficiency for pt > 8.0 = 86.3785 +- 0.145872
efficiency for pt > 40.0 = 87.9521 +- 0.508875

TP/event (pt > 2) = 175.946
TP/event (pt > 3.0) = 58.3381
TP/event (pt > 10.0) = 5.076
tracks/event (pt > 2) = 167.025
tracks/event (pt > 3.0) = 58.7362
tracks/event (pt > 10.0) = 6.03033</td>
<td>efficiency for |eta| < 1.0 = 87.6086 +- 0.0949657
efficiency for 1.0 < |eta| < 1.75 = 85.5178 +- 0.136524
efficiency for 1.75 < |eta| < 2.4 = 83.8588 +- 0.191319
combined efficiency for |eta| < 2.4 = 86.3682 +- 0.0725307 = 193293/223801

efficiency for pt > 2 = 86.3682 +- 0.0725307
efficiency for 2 < pt < 8.0 = 86.0693 +- 0.0843534
efficiency for pt > 8.0 = 87.2791 +- 0.1417
efficiency for pt > 40.0 = 88.9052 +- 0.490971

TP/event (pt > 2) = 175.946
TP/event (pt > 3.0) = 58.3381
TP/event (pt > 10.0) = 5.076
tracks/event (pt > 2) = 164.54
tracks/event (pt > 3.0) = 58.3723
tracks/event (pt > 10.0) = 6.02678</td>
<td>efficiency for |eta| < 1.0 = 87.5621 +- 0.0951186
efficiency for 1.0 < |eta| < 1.75 = 85.2574 +- 0.137536
efficiency for 1.75 < |eta| < 2.4 = 83.5316 +- 0.192871
combined efficiency for |eta| < 2.4 = 86.2119 +- 0.0728795 = 192943/223801

efficiency for pt > 2 = 86.2119 +- 0.0728795
efficiency for 2 < pt < 8.0 = 86.1708 +- 0.0840951
efficiency for pt > 8.0 = 86.3369 +- 0.146059
efficiency for pt > 40.0 = 87.8788 +- 0.510208

TP/event (pt > 2) = 175.946
TP/event (pt > 3.0) = 58.3381
TP/event (pt > 10.0) = 5.076
tracks/event (pt > 2) = 167.218
tracks/event (pt > 3.0) = 58.7973
tracks/event (pt > 10.0) = 6.02222</td>
<td>efficiency for |eta| < 1.0 = 87.7208 +- 0.0945955
efficiency for 1.0 < |eta| < 1.75 = 85.6066 +- 0.136175
efficiency for 1.75 < |eta| < 2.4 = 83.6019 +- 0.19254
combined efficiency for |eta| < 2.4 = 86.4125 +- 0.0724315 = 193392/223801

efficiency for pt > 2 = 86.4125 +- 0.0724315
efficiency for 2 < pt < 8.0 = 86.3334 +- 0.0836781
efficiency for pt > 8.0 = 86.6534 +- 0.144622
efficiency for pt > 40.0 = 88.001 +- 0.507983

TP/event (pt > 2) = 175.946
TP/event (pt > 3.0) = 58.3381
TP/event (pt > 10.0) = 5.076
tracks/event (pt > 2) = 167.046
tracks/event (pt > 3.0) = 58.5893
tracks/event (pt > 10.0) = 6.00711</td></tr>
<tr><td>False</td>
<td>efficiency for |eta| < 1.0 = 85.7029 +- 0.100892
efficiency for 1.0 < |eta| < 1.75 = 84.5877 +- 0.140071
efficiency for 1.75 < |eta| < 2.4 = 83.4181 +- 0.193403
combined efficiency for |eta| < 2.4 = 84.9943 +- 0.0754905 = 190218/223801

efficiency for pt > 2 = 84.9943 +- 0.0754905
efficiency for 2 < pt < 8.0 = 84.7264 +- 0.0876341
efficiency for pt > 8.0 = 85.8106 +- 0.148391
efficiency for pt > 40.0 = 86.9746 +- 0.526168

TP/event (pt > 2) = 175.946
TP/event (pt > 3.0) = 58.3381
TP/event (pt > 10.0) = 5.076
tracks/event (pt > 2) = 164.019
tracks/event (pt > 3.0) = 58.7816
tracks/event (pt > 10.0) = 6.04567</td>
<td>efficiency for |eta| < 1.0 = 87.6136 +- 0.0949493
efficiency for 1.0 < |eta| < 1.75 = 85.3477 +- 0.137186
efficiency for 1.75 < |eta| < 2.4 = 83.6695 +- 0.19222
combined efficiency for |eta| < 2.4 = 86.2892 +- 0.0727075 = 193116/223801

efficiency for pt > 2 = 86.2892 +- 0.0727075
efficiency for 2 < pt < 8.0 = 86.0848 +- 0.0843142
efficiency for pt > 8.0 = 86.912 +- 0.143428
efficiency for pt > 40.0 = 88.2698 +- 0.503027

TP/event (pt > 2) = 175.946
TP/event (pt > 3.0) = 58.3381
TP/event (pt > 10.0) = 5.076
tracks/event (pt > 2) = 164.425
tracks/event (pt > 3.0) = 58.3044
tracks/event (pt > 10.0) = 5.99167</td>
<td>efficiency for |eta| < 1.0 = 85.6165 +- 0.101145
efficiency for 1.0 < |eta| < 1.75 = 84.3876 +- 0.140811
efficiency for 1.75 < |eta| < 2.4 = 83.3829 +- 0.193568
combined efficiency for |eta| < 2.4 = 84.8826 +- 0.0757212 = 189968/223801

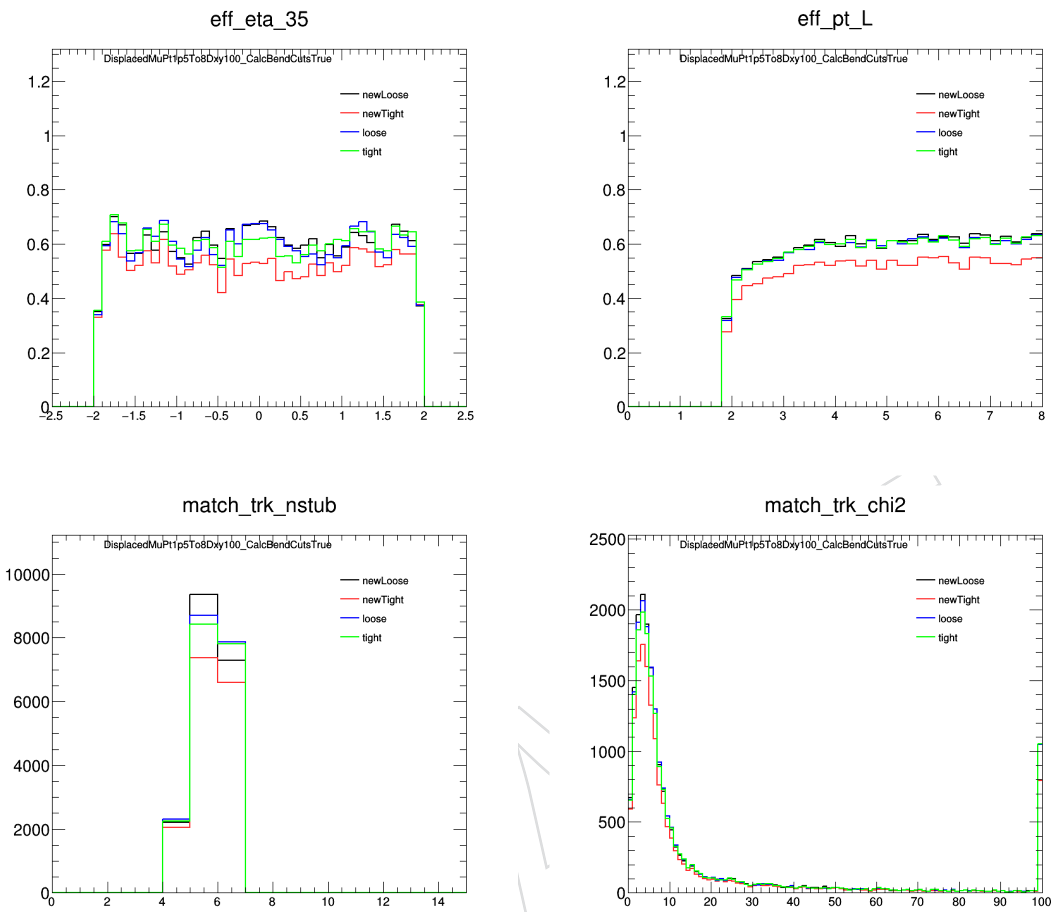
efficiency for pt > 2 = 84.8826 +- 0.0757212
efficiency for 2 < pt < 8.0 = 84.6189 +- 0.0878859
efficiency for pt > 8.0 = 85.6859 +- 0.148934
efficiency for pt > 40.0 = 86.9013 +- 0.527424

TP/event (pt > 2) = 175.946
TP/event (pt > 3.0) = 58.3381
TP/event (pt > 10.0) = 5.076
tracks/event (pt > 2) = 163.895
tracks/event (pt > 3.0) = 58.996
tracks/event (pt > 10.0) = 6.07344</td>
<td>efficiency for |eta| < 1.0 = 87.5579 +- 0.0951322
efficiency for 1.0 < |eta| < 1.75 = 85.4486 +- 0.136794
efficiency for 1.75 < |eta| < 2.4 = 83.41 +- 0.193441
combined efficiency for |eta| < 2.4 = 86.2463 +- 0.072803 = 193020/223801

efficiency for pt > 2 = 86.2463 +- 0.072803
efficiency for 2 < pt < 8.0 = 86.2444 +- 0.0839069
efficiency for pt > 8.0 = 86.2519 +- 0.146441
efficiency for pt > 40.0 = 87.4878 +- 0.517217

TP/event (pt > 2) = 175.946
TP/event (pt > 3.0) = 58.3381
TP/event (pt > 10.0) = 5.076
tracks/event (pt > 2) = 166.983
tracks/event (pt > 3.0) = 58.6621
tracks/event (pt > 10.0) = 6.00222</td></tr>
</table>

Figure 43: L1 track reconstruction efficiency for ttbar tracks with four available tunes and the ‘calcBendCut’ variable true or false.



<table>
<tr><th rowspan="2">Calc Bend Cuts</th><th colspan="4">efficiency</th></tr>
<tr><th>newLoose</th><th>newTight</th><th>loose</th><th>tight</th></tr>
<tr><td>True</td>
<td>efficiency for |eta| < 1.0 = 62.3722 +- 0.392168
efficiency for 1.0 < |eta| < 1.75 = 63.2882 +- 0.446161
efficiency for 1.75 < |eta| < 2 = 51.0649 +- 0.80564
combined efficiency for |eta| < 2 = 61.3053 +- 0.277604 = 18871/30782

efficiency for pt > 3 = 61.3053 +- 0.277604
efficiency for 3 < pt < 8.0 = 61.3053 +- 0.277604
TP/event (pt > 3) = 0.310929
TP/event (pt > 3.0) = 0.310929
TP/event (pt > 10.0) = 0
tracks/event (pt > 3) = 0.575768
tracks/event (pt > 3.0) = 0.575768
tracks/event (pt > 10.0) = 0.000909091</td>
<td>efficiency for |eta| < 1.0 = 51.6631 +- 0.408404
efficiency for 1.0 < |eta| < 1.75 = 56.7248 +- 0.463774
efficiency for 1.75 < |eta| < 2 = 47.779 +- 0.812232
combined efficiency for |eta| < 2 = 53.0911 +- 0.287324 = 16016/30167

efficiency for pt > 3 = 53.0911 +- 0.287324
efficiency for 3 < pt < 8.0 = 53.0911 +- 0.287324
TP/event (pt > 3) = 0.311
TP/event (pt > 3.0) = 0.311
TP/event (pt > 10.0) = 0
tracks/event (pt > 3) = 0.473093
tracks/event (pt > 3.0) = 0.473093
tracks/event (pt > 10.0) = 0.000525773</td>
<td>efficiency for |eta| < 1.0 = 60.6292 +- 0.39309
efficiency for 1.0 < |eta| < 1.75 = 63.9763 +- 0.441565
efficiency for 1.75 < |eta| < 2 = 49.6399 +- 0.801855
combined efficiency for |eta| < 2 = 60.5277 +- 0.276919 = 18858/31156

efficiency for pt > 3 = 60.5277 +- 0.276919
efficiency for 3 < pt < 8.0 = 60.5277 +- 0.276919
TP/event (pt > 3) = 0.31156
TP/event (pt > 3.0) = 0.31156
TP/event (pt > 10.0) = 0
tracks/event (pt > 3) = 0.56356
tracks/event (pt > 3.0) = 0.56356
tracks/event (pt > 10.0) = 0.00078</td>
<td>efficiency for |eta| < 1.0 = 59.7209 +- 0.39884
efficiency for 1.0 < |eta| < 1.75 = 64.5094 +- 0.445068
efficiency for 1.75 < |eta| < 2 = 52.6247 +- 0.808925
combined efficiency for |eta| < 2 = 60.6494 +- 0.279776 = 18492/30490

efficiency for pt > 3 = 60.6494 +- 0.279776
efficiency for 3 < pt < 8.0 = 60.6494 +- 0.279776
TP/event (pt > 3) = 0.311094
TP/event (pt > 3.0) = 0.311094
TP/event (pt > 10.0) = 0
tracks/event (pt > 3) = 0.552092
tracks/event (pt > 3.0) = 0.552092
tracks/event (pt > 10.0) = 0.000897877</td></tr>
<tr><td>False</td>
<td>efficiency for |eta| < 1.0 = 60.368 +- 0.397932
efficiency for 1.0 < |eta| < 1.75 = 62.9342 +- 0.44894
efficiency for 1.75 < |eta| < 2 = 50.9995 +- 0.810732
combined efficiency for |eta| < 2 = 60.1739 +- 0.280379 = 18344/30485

efficiency for pt > 3 = 60.1739 +- 0.280379
efficiency for 3 < pt < 8.0 = 60.1739 +- 0.280379
TP/event (pt > 3) = 0.311043
TP/event (pt > 3.0) = 0.311043
TP/event (pt > 10.0) = 0
tracks/event (pt > 3) = 0.561316
tracks/event (pt > 3.0) = 0.561316
tracks/event (pt > 10.0) = 0.000989705</td>
<td>efficiency for |eta| < 1.0 = 50.9101 +- 0.406706
efficiency for 1.0 < |eta| < 1.75 = 56.4282 +- 0.460903
efficiency for 1.75 < |eta| < 2 = 47.8958 +- 0.810175
combined efficiency for |eta| < 2 = 52.6292 +- 0.285973 = 16044/30485

efficiency for pt > 3 = 52.6292 +- 0.285973
efficiency for 3 < pt < 8.0 = 52.6292 +- 0.285973
TP/event (pt > 3) = 0.311043
TP/event (pt > 3.0) = 0.311043
TP/event (pt > 10.0) = 0
tracks/event (pt > 3) = 0.468549
tracks/event (pt > 3.0) = 0.468549
tracks/event (pt > 10.0) = 0.000622392</td>
<td>efficiency for |eta| < 1.0 = 58.9715 +- 0.400171
efficiency for 1.0 < |eta| < 1.75 = 63.6081 +- 0.447216
efficiency for 1.75 < |eta| < 2 = 49.5266 +- 0.810857
combined efficiency for |eta| < 2 = 59.5539 +- 0.281093 = 18155/30485

efficiency for pt > 3 = 59.5539 +- 0.281093
efficiency for 3 < pt < 8.0 = 59.5539 +- 0.281093
TP/event (pt > 3) = 0.311043
TP/event (pt > 3.0) = 0.311043
TP/event (pt > 10.0) = 0
tracks/event (pt > 3) = 0.550827
tracks/event (pt > 3.0) = 0.550827
tracks/event (pt > 10.0) = 0.000928486</td>
<td>efficiency for |eta| < 1.0 = 58.4797 +- 0.398892
efficiency for 1.0 < |eta| < 1.75 = 64.0936 +- 0.444039
efficiency for 1.75 < |eta| < 2 = 52.6234 +- 0.804713
combined efficiency for |eta| < 2 = 59.8759 +- 0.27937 = 18431/30782

efficiency for pt > 3 = 59.8759 +- 0.27937
efficiency for 3 < pt < 8.0 = 59.8759 +- 0.27937
TP/event (pt > 3) = 0.310929
TP/event (pt > 3.0) = 0.310929
TP/event (pt > 10.0) = 0
tracks/event (pt > 3) = 0.544222
tracks/event (pt > 3.0) = 0.544222
tracks/event (pt > 10.0) = 0.00106061</td></tr>
</table>

Figure 44: L1 track reconstruction efficiency for displaced muons with four available tunes and the ‘calcBendCut’ variable true or false.

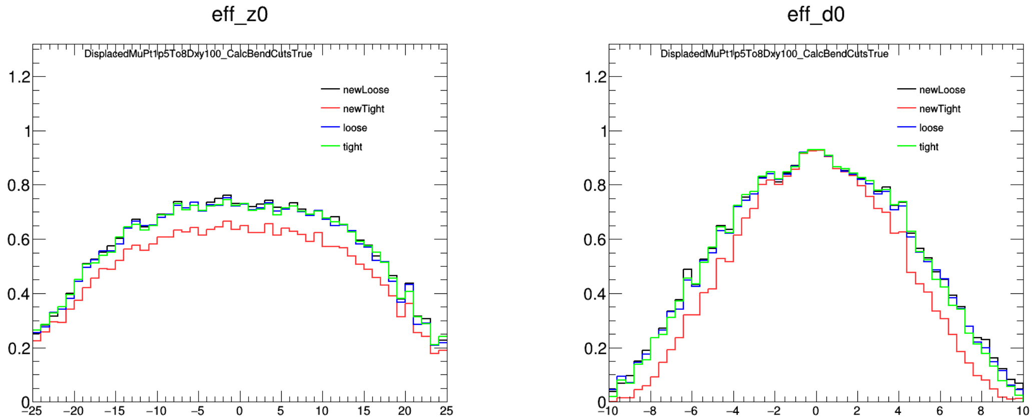


Figure 45: L1 track reconstruction efficiency for displaced muons as a function of the track impact parameters with four available tunes.

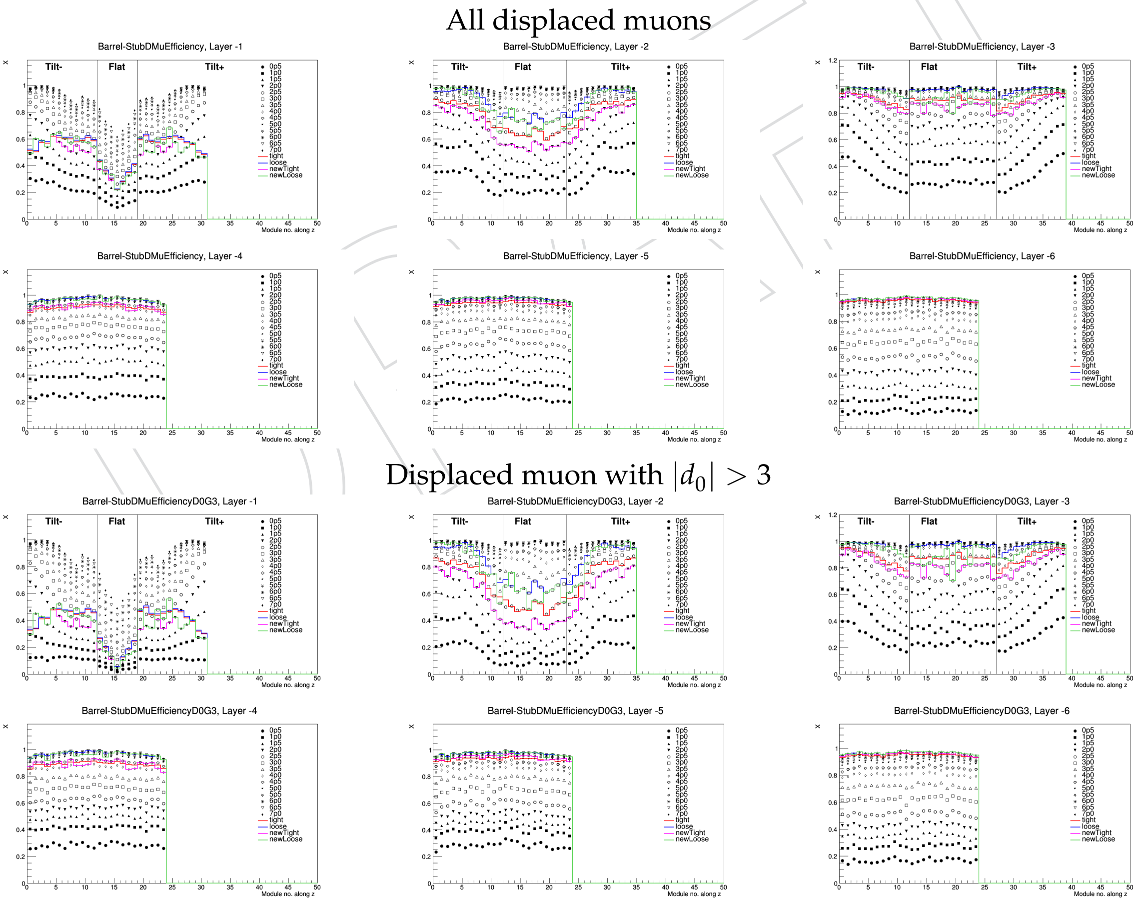


Figure 46: Stub reconstruction efficiencies in barrel layers for all displaced muon (top) and displaced muon with $|d_0| > 3$ cm (bottom) in barrel layers.

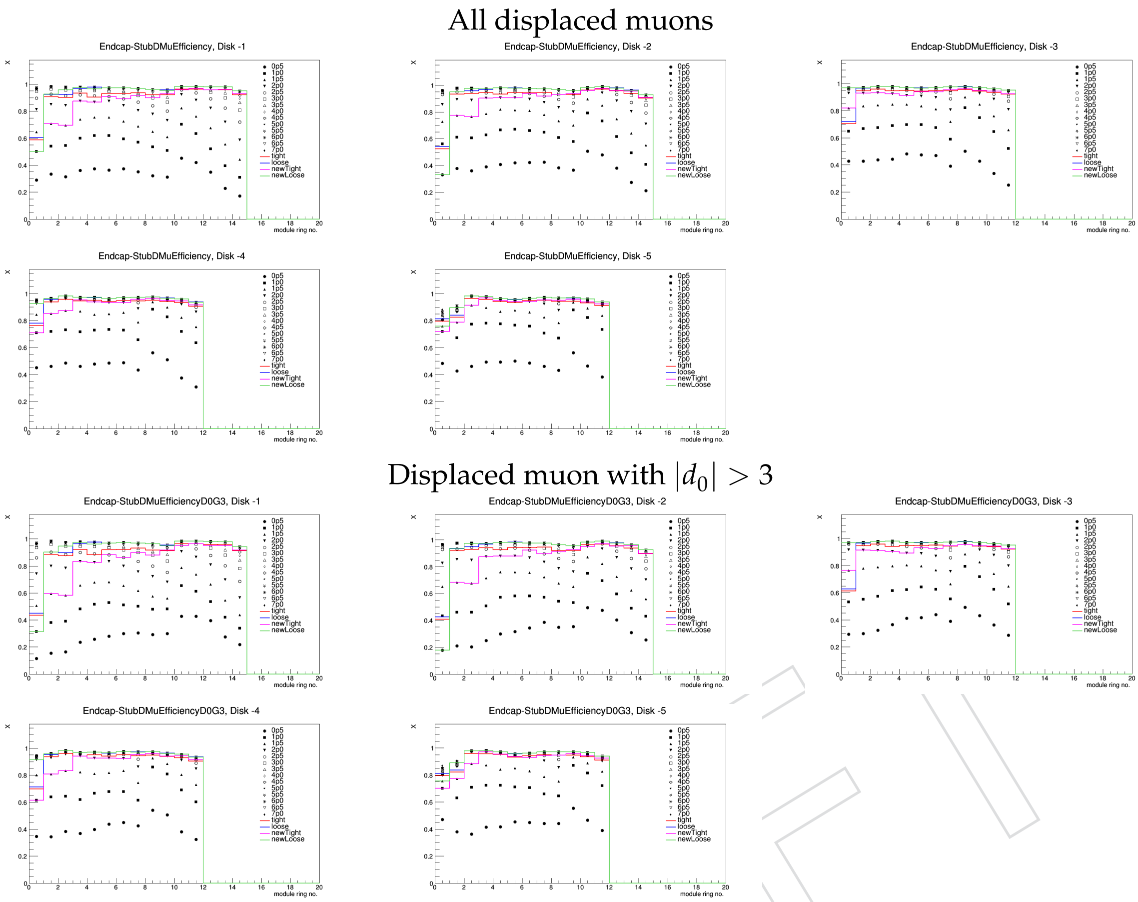


Figure 47: Stub reconstruction efficiencies in barrel layers for all displaced muon (top) and displaced muon with $|d_0| > 3$ cm (bottom) in endcap disks.

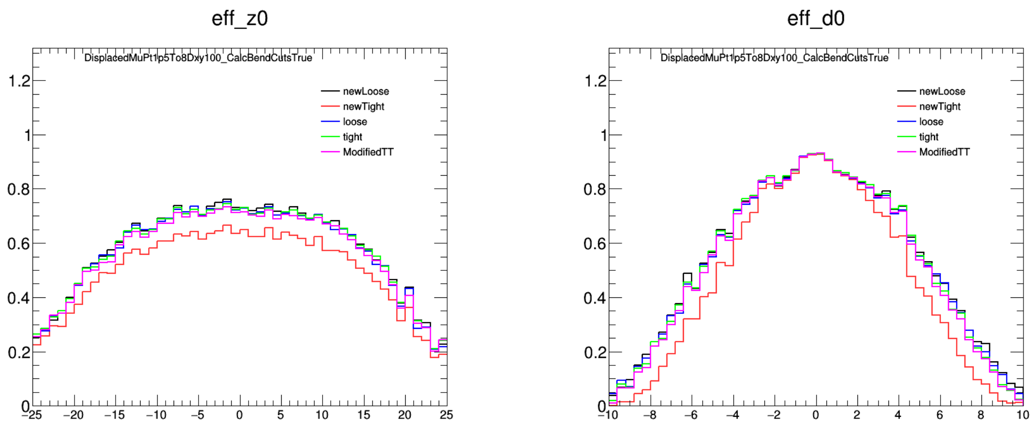


Figure 48: L1 track reconstruction efficiency for displaced muons as a function of the track impact parameters with four available tunes.

8 Summary

A new CMS Tracker is under development for operation at the High Luminosity LHC. It includes an outer tracker based on dedicated modules that will reconstruct short track segments, called stubs, using spatially coincident clusters in two closely spaced silicon sensor layers. In this detector note, we studied various properties of the stubs using the latest simulated samples. We have measured the stub rates, the fraction of stubs that are failed in data transmission and stub reconstruction efficiencies. These stub properties depend directly on the stub window size. We showed this dependency in various region of the tracker. We have developed an algorithm to combine these properties and extract optimum widow sizes in all detector regions. The final stub window tune that we propose to be used in the main L1 track trigger code is the following;

```
# New modified tight tune based on simulated events CMSSW_11_3_0_pre3, D76
    BarrelCut    = cms.vdouble(0, 2.0, 2.5, 3.5, 4.0, 5.5, 6.5),
    TiltedBarrelCutSet = cms.VPSet(
        cms.PSet( TiltedCut = cms.vdouble( 0 ) ),
        cms.PSet( TiltedCut = cms.vdouble( 0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 1.5, 1.5, 1.5, 1.0, 1.0) ),
        cms.PSet( TiltedCut = cms.vdouble( 0, 3.0, 3.0, 3.0, 3.0, 3.0, 2.5, 2.5, 3.0, 3.0, 2.5, 2.5, 2.5) ),
        cms.PSet( TiltedCut = cms.vdouble(0, 4.0, 4.0, 4.0, 3.5, 3.5, 3.5, 3.0, 3.0, 2.5, 2.5, 2.5, 2.5) ),
        ),
    EndcapCutSet = cms.VPSet(
        cms.PSet( EndcapCut = cms.vdouble( 0 ) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.0, 1.5, 1.5, 2.0, 2.0, 2.5, 2.5, 3.0, 4.0, 4.0, 2.5, 3.0, 3.5, 4.0, 5.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 0.5, 1.5, 1.5, 2.0, 2.0, 2.0, 2.5, 2.5, 3.0, 3.5, 2.0, 2.5, 3.0, 4.0, 4.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.5, 2.0, 2.0, 2.0, 2.0, 2.5, 3.0, 3.5, 2.5, 2.5, 3.0, 3.5) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.0, 1.5, 1.5, 2.0, 2.0, 2.0, 2.0, 3.0, 2.0, 2.0, 3.0, 3.0) ),
        cms.PSet( EndcapCut = cms.vdouble(0, 1.0, 1.5, 1.5, 2.0, 2.0, 2.0, 2.0, 2.5, 3.0, 2.0, 2.0, 2.5) ),
        )
\
```

References

[1] D. Contardo et al., “Technical Proposal for the Phase-II Upgrade of the CMS Detector”, Technical Report CERN-LHCC-2015-010. LHCC-P-008. CMS-TDR-15-02, Geneva, Jun, 2015. Upgrade Project Leader Deputies: Lucia Silvestris (INFN-Bari), Jeremy Mans (University of Minnesota) Additional contacts: Lucia.Silvestris@cern.ch, Jeremy.Mans@cern.ch.

[2] CMS Collaboration Collaboration, “The Phase-2 Upgrade of the CMS L1 Trigger Interim Technical Design Report”, Technical Report CERN-LHCC-2017-013. CMS-TDR-017, CERN, Geneva, Sep, 2017. This is the CMS Interim TDR devoted to the upgrade of the CMS L1 trigger in view of the HL-LHC running, as approved by the LHCC.

[3] CMS Collaboration Collaboration, “The Phase-2 Upgrade of the CMS Tracker”, Technical Report CERN-LHCC-2017-009. CMS-TDR-014, CERN, Geneva, Jun, 2017.

[4] L. Caponetto, S. Viret, and Y. Zoccarato, “I/O data formats for the Concentrator Integrated Circuit”,.

[5] S. Viret, “Stub definition and properties”, *CMS DN -2018/003* (2018).

[6] A. Ryd and L. Skinnari, “Tracking Triggers for the HL-LHC”, *Ann. Rev. Nucl. Part. Sci.* **70** (2020) 171–195, `doi:10.1146/annurev-nucl-020420-093547`, `arXiv:2010.13557`.