CLAS12 Drift Chamber Reconstruction Software Unit Test

M. Armstrong[1], V. Ziegler[2], and G.P. Gilfoyle[3]

[1] *University of Surrey, Guilford, UK*
[2] *Jefferson Lab, Newport News, VA*
[3] *University of Richmond, Richmond, VA*

July 6, 2020

Abstract

This CLAS12 NOTE describes the upgrade of a software tool called a unit test to validate that changes to the CLAS12 reconstruction code operate as intended. The unit test here is part of the Drift Chamber (DC) subsystem. The previous one consistently produced false positives so its warning had become unusable. The source of this failure was identified and the test modified for use with the latest version of COATJAVA. The test was also expanded to include the reconstructed vertex of a test track and the positions of the reconstruction crosses.

Contents

1 Introduction

1.1 CLAS12 Reconstruction Software Testing

Computer code testing and validation are essential for developing accurate, high-performance software. The goal is to catch mistakes/bugs that are introduced as the code evolves and to monitor changes in performance. These goals are especially important in large physics collaborations where many different people are working on the software so it evolves at a rapid pace.

In the CLAS collaboration we use several tools to maintain, test, and validate the main Collaboration reconstruction package called COATJAVA. (1) Version control is maintained with github. This service is built on the command-line tool Git and provides additional, web-based features. (2) The entire COATJAVA code base is recompiled nightly to check for bugs introduced into the code. (3) During the nightly builds, unit tests are run on the subsystems to test their performance. Fixing and expanding the Drift Chamber unit tests is the focus of this work.

In this work we have developed new unit test criteria for momentum, vertex position, and reconstruction cross position for a single, representative test event. The process is (1) to simulate the CLAS12 response to a 'typical' electron event defined in a narrow kinematic range. (2) Reconstruct that simulated data and extract the distribution for each quantity of interest. (3) Select a central value from each distribution. (4) Extract the reconstruction resolution by comparing the reconstruction results with the known input to the simulation. (5) Use the central value and reconstruction resolution to define the tolerance of the unit test for each quantity. (6) Add these criteria to the CLAS12 reconstruction code.

1.2 CLAS12 Design Overview

CLAS12 is positioned in the centre of Hall B's large circular underground chamber and is depicted in Figure 1 [1]. CEBAF's main beamline leads into CLAS12's Central Detector (CD) from the right side in Figure 1- the first of two main sections that make up CLAS12 and contains its target. The Forward Detector (FD) is the second one and sits downstream from the CD. The FD covers the angular range $\theta = 5° - 35°$ and houses the CLAS12 drift chambers (DC) which are the main focus of this work [1].

The Forward Detector consist of six identical sectors with a toroidal field generated by superconducting coils in each sector. An array of drift chambers, time-of-flight counters, Cerenkov counters, calorimeters, and other devices measure and identify the reaction products [1]. The Central Detector covers large angles $(35° - 125°)$and is built around a solenoid magnet and another suite of detectors. In Figure 1 the electron beam enters from the right and impinges on the production target located in the center of the solenoid

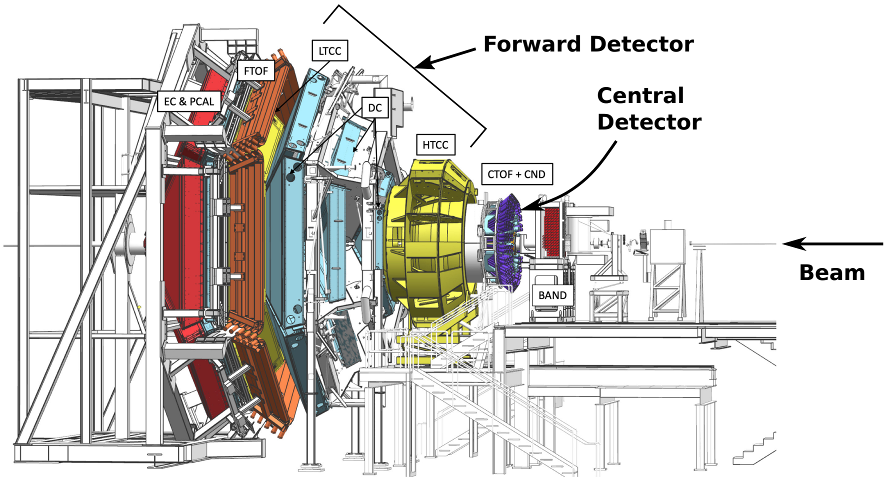


Figure 1: Representation of the CLAS12 spectrometer in Hall B at JLab. The electron beam is incident from the right side of this figure. The CLAS12 detector is roughly 13 m in scale along the beam axis. The drift chambers (DC) are marked.

magnet which supports the CD and is shown at the right (upstream) end of CLAS12, where other detector components (CTOF+CND) are also visible. Scattered electrons and forward-going particles are detected in the FD. The High Threshold Cherenkov Counter (HTCC) (yellow) with full coverage in polar angle $5^\circ \leq \theta \leq 35^\circ$ and $\Delta\phi = 2\pi$ coverage in azimuth is used for electron/pion discrimination. The HTCC is followed by the torus magnet (gray), the drift chamber tracking system (DC) (light blue), another set of Cherenkov counters (LTCC) (yellow) for pion/kaon discrimination, forward time-of-flight scintillation counters (FTOF) (brown), and electromagnetic calorimeters (EC and PCAL) (red) for energy measurements. Between the HTCC and the torus, the Forward Tagger is installed to detect electrons and photons at polar angles $2^\circ \leq \theta \leq 5^\circ$. The Central Detector (CD) consists of the Silicon Vertex Tracker (hidden), surrounded by a micromegas tracker (hidden), the Central Time-of-Flight system (CTOF), and the Central Neutron Detector (CND) (PMTs in blue). At the upstream end, a Back Angle Neutron Detector (BAND) (red) is installed. In the operational configuration the entire CLAS12 detector extends for 13 m along the beamline.

Particles that enter the FD travel through three regions containing the DCs shown in the cross sectional view of the wider detector in Figure 2. The example trajectory shows how an incident beam electron might enter, scatter off the target and continue through calorimeters, DC regions, time-of-flight scintillators and Cherenkov counters in the FD. This combination of detector subsystems enables us to reconstruct the trajectory, momentum, and identity of scattered particles [1,3]. Some of the design specifications of the FD and CD are in Table 1 [1].

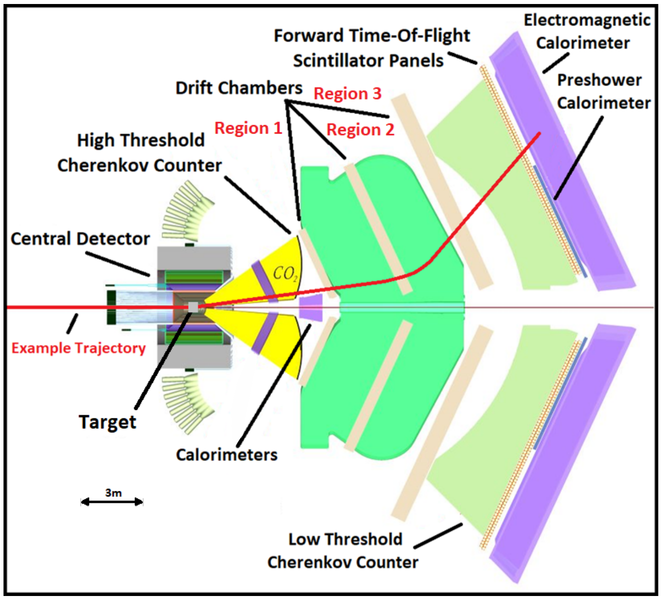


Figure 2: Cutaway view of CLAS12 detector showing example trajectory (entering left) as it travels through the CD and FD. In this case the example particle is positively charged and so bends away from the beamline due to the direction of the toroidal field (green). Adapted image from [4].

	Forward Detector	Central Detector
Angular Range	$5-35\ deg$	$35-125\ deg$
$\delta p/p$ (%)	0.5-1.5 %	¡5%
$\delta\theta$ ($mrad$)	1-2	2-5
$\delta\phi$ ($mrad/\sin\theta$)	≤ 2	3-15
Energy (MeV)	≥ 200	≥ 200

Table 1: Summary of CLAS12 detector design parameters [1].

1.3 CLAS12 Software

In order to sort, store and process data from CLAS12's 111,832 detector readout channels, the CLAS12 collaboration uses a sophisticated suite of data processing tools called the CLAS Offline Analysis Tools (COATJAVA). The Collaboration uses a locally-developed file format called EVIO (Event Input Output) to store the raw signals captured by CLAS12 in a digital format. The I/O package decodes these signals, extracting and storing the relevant information (such the detector component ID or voltage peak) into High Performance Output (HIPO) data banks. The HIPO file format was developed by the Collaboration to speed access and to minimize file sizes for physics analysis.

The reconstruction package reads signal data from these data banks and combines it with information from the detector geometry package and calibration database to reconstruct the trajectories, identities and properties of the reaction products that generated the original signals. The results are stored in additional

HIPO data banks [2].

The analysis and plotting packages are used to interface with these data banks and produce particle data plots. These packages enable us to make cuts based on the properties of the data like selecting particles from well-understood regions of CLAS12.

The Geant4 Monte-Carlo (GEMC) package is a Geant4, physics-based simulation package which uses information from the geometry package and models of the subsystem responses to build simulations of the CLAS12 detector and to produce pseudo-data. GEMC is used by the Collaboration to test test and study the response of CLAS12 to known inputs.

2 Drift Chamber Reconstruction Software

The three DC regions are split into pairs of superlayers made up many hexagonal gas chambers (or cells) that are arranged into six layers each 112 cells wide. These cells (see Figure 3) contain a 9:1 mixture of argon-CO_2 gas and a central anode sense wire [5]. Particles traveling through the DC layers leave trails of electrons as they ionize gas molecules. These electrons accelerate towards the positive sense wires and create a current signal recording the passage of a charged particle through the cell. The pattern of these sense wire hits is used to reconstruct the trajectory of the charged particles.

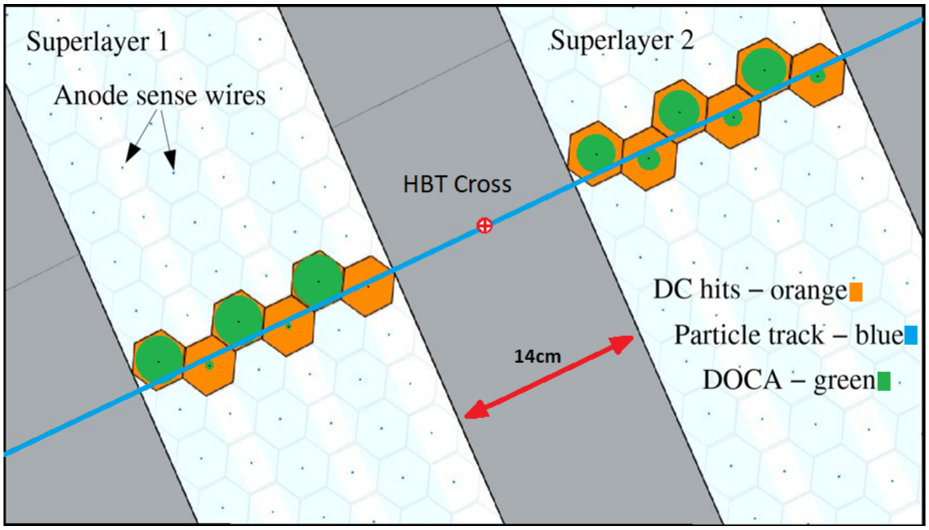


Figure 3: Illustration of particle reconstruction in DC Region 2. Activated cells in orange, Distance of Closest Approach (DOCA) in green and reconstructed track in blue. HBT reconstruction cross positioned at midpoint between superlayers separated by 14 cm.

COATJAVA begins this reconstruction with a process called Hit Based Tracking (HBT). In HBT the positions of activated sense wires are readout as points in 3D Cartesian space. A simplified 2D version of the process is illustrated in Figure 3. The process proceeds by fitting trajectories through the sense wire hits, (orange in Figure 3) each with an associated covariance matrix characterizing the fitting uncertainties. The widths of the gaps between each pair of superlayers are comparable to the width of superlayers themselves so virtual points called reconstruction crosses are interpolated at the midpoint between the superlayers. These crosses shown in Figure 3 are used to better constrain possible particle trajectories between superlayers in the subsequent Time Based Tracking (TBT) process.

When an incident particle ionizes DC gas molecules, the ionized electrons take time to drift to the positive anode sense wire. During this drift time, the incident particle continues traveling through the detector where the timing of its passage through the Forward Time-Of-Flight scintillators (FTOF) is recorded (see Figure 2). This TOF information and the path length from the target to the FTOF (extracted from the HBT trajectory) is used to determine when the scattered particle left the target and when it later enters each DC cell along its HBT trajectory. The difference between the time when the scattered particle reaches a DC cell t_{cell} and when ionized electrons are detected at the anode t_{anode} is taken as the time for the ionized electrons to drift from the ionization site to the anode $\tau_{drift} = t_{anode} - t_{cell}$. The Distance-Of-Closest-Approach (DOCA) (shown in Figure 4) of a particle path to the sense wire is related to τ_{drift} by $DOCA = f(\tau_{drift})$ and the time-to-distance function $f(\tau_{drift})$ is a measured calibration function. The DOCA for each chamber hit further constrains the fitted trajectory and is used to more precisely reconstruct the particle's trajectory.

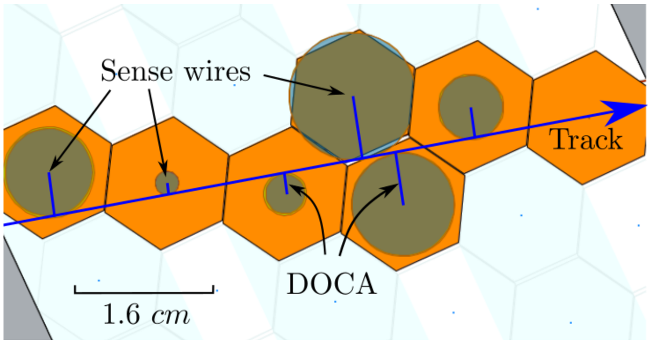


Figure 4: DC particle track (blue) and TBT DOCA illustration: activated cell is in orange and incident particle passes tangent to dark green circle.

The FD makes use of a powerful superconducting toroidal magnet to analyze the momentum of charged particles. See the dark green regions in Figure 2. The toroid is made up of six, identical coils and encloses the DC Region 2 with a peak field strength of 3.6 T [6]. As shown with the example trajectory in Figure 2 when charged particles approach this field they accelerate and their trajectories bend. The curvature of a charged particles track is used to measure its momentum.

3 DC Software Unit Test

In order to make development more easily manageable, COATJAVA's 84,000 lines of executable code are split into modules. Each module comes with a software unit test designed to check individual parts of the source code and assess its performance. When a developer updates the source code associated with the DC reconstruction module, the DC unit test takes preprogrammed detector data representing a single test particle and reconstructs its momentum from the best fitting TBT trajectory. If the momentum is sufficiently different from expectation (*i.e.* lays outside the tolerance range), the test signals that a software failure occurred. As of March 2019 the previous unit test was found to be falsely signaling software failures every time it was run. The test needed to be updated before it could continue to be used as a tool to validate the DC reconstruction module.

3.1 Simulation of Data for DC Software Validation

In order to understand why the unit test was producing false positives an analysis of the DC reconstruction was undertaken. Simulated detector data representing 20,000 electrons scattered from the target through the FD was generated using GEMC. These electrons were generated with the same initial starting position and momentum as the particles in the original simulation of the previous test particle. These initial conditions were a scattered electron originating at the CLAS12 origin with a momentum of 2.5 ± 0.5 GeV/c through a polar angle range $\theta = 25° \pm 10°$ and an azimuthal angle range $\phi = 0° \pm 5°$ from the beamline. See Figure 5 for detector co-ordinate system definitions.

The CLAS12 coordinate system is shown in Figure 5 which shows the orientation of each sector. The beamline is going into the page and ideally forms the z axis. The azimuthal angle ϕ is measured relative to the x axis.

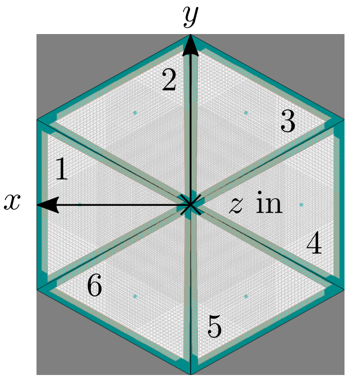


Figure 5: Representation of the CLAS12 coordinate system's xy plane with view facing down the beamline. The numbering scheme of the sectors is shown.

3.2 DC Reconstruction Analysis

The simulated data were reconstructed using the latest version of COATJAVA (version 6.3.1) to produce histograms showing the frequency distributions of the reconstructed particle events' x, y and z momenta and their vertices. The vertex is a quantity used to characterize the starting point of a particle's trajectory. By extrapolating a reconstructed trajectory back to the target region, the vertex is found as the closest point along that trajectory to the beamline. See Figure 6. It shows the extrapolated, reconstructed track (light blue), the vertex (green point), and a line segment (red) marking the DOCA between the track and the beamline.

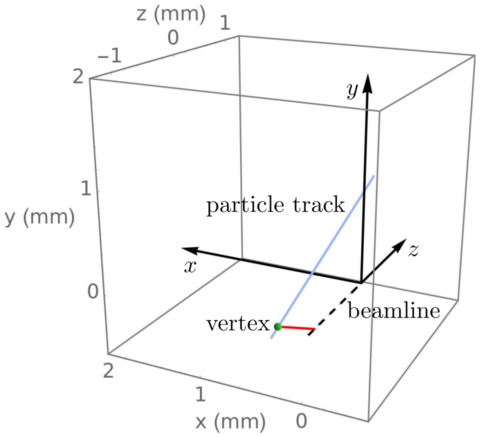


Figure 6: Close-up of CLAS12 coordinate system showing sample particle track (blue) and line segment (red) at the point of closest approach between the track and the beamline (z axis). The vertex (green point) is the point on the track at the distance-of-closest approach (DOCA) to the beamline (z axis).

3.3 Momentum and Vertex Simulation Results

Here we define the procedures to select the central value and tolerance limits for the new test event and discuss the results. The histogram in Figure 7 shows the frequency distribution of the reconstructed P_z for the 20,000 simulated particle events. The green dot is where the previous unit test event was reconstructed (P_z = 2.41 GeV/c) and the green lines mark the previous test event tolerance limits of $2.16 < P_z < 2.39$ GeV/c. The previous unit test event P_z is reconstructed outside its tolerance limits when using the latest version of COATJAVA. Recent versions of COATJAVA reconstruct P_z differently and so the previous limits do not match the current reconstruction. The unit test limits need to be updated to be consistent with current versions of the code.

To characterize the momentum distribution in Figure 7 we fit the central region with a Gaussian and avoided the low-momentum tail. This tail is the product of multiple scattering and energy loss from

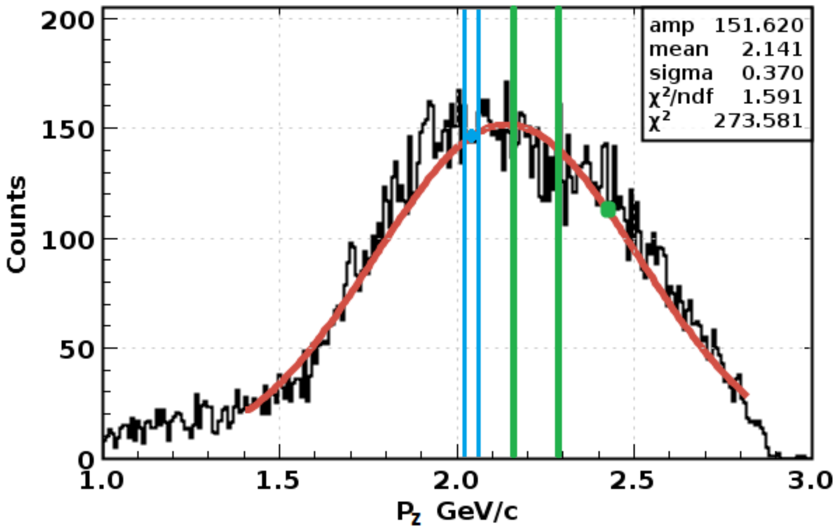
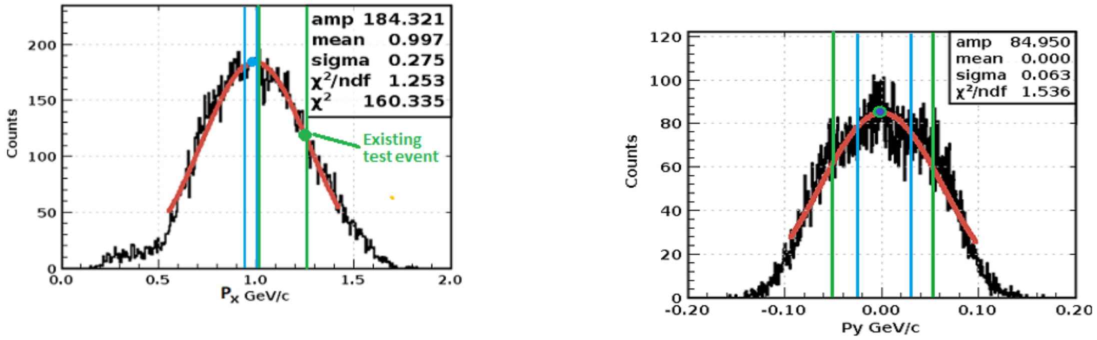


Figure 7: Simulated data P_z histogram, with previous unit test event reconstruction (green dot) and previous test limits (green lines). Replacement test event reconstructed at 2.04 GeV/c (blue dot) with new tolerance limits $2.27 < P_z < 2.53$ GeV/c (blue lines)No! No! No!.

interactions with the material in the detector and from radiative effects. By fitting the peak of the P_z distribution with a Gaussian function (shown in red) the mean P_z was found to be 2.14 ± 0.37 GeV/c. This momentum distribution extends beyond the previous unit test limits shown by the green lines in Figure 7. With the new reconstruction code that previous test event lay one standard deviation from the mean (generated under the same conditions) at 2.51 GeV/c. This suggested it is not a suitable candidate for the unit test.

A new test event was selected from the GEMC simulation data with reconstructed momentum P_z within 0.5 σ of the mean of the distribution to be sure a typical event is selected. The blue point in Figure 7 shows where the central value of the replacement test event was selected at $P_z = 2.04$ GeV/c. The next step in the procedure for the z component of the momentum is to determine the tolerance limits. We did that by measuring the reconstruction resolution and which will be presented in Section 3.4. Here we examine the results for the other momentum components and for the vertex position.

The momentum distributions for the x and y components are shown in Figure 8. The x component in Figure 8a has a low-momentum tail smaller than the tail in the P_z distribution. The P_z distribution also has a higher central value of the momentum than the P_x distribution. We fitted the central peak with a Gaussian avoiding the low-momentum tail and obtained a mean of 0.997 GeV/c. The previous test event was reconstructed with $P_x = 1.24$ GeV/c (green point) near the edge of the previous test limits at $1.02 < P_x < 1.26$ GeV/c (green vertical lines in Figure 8a). We placed the central value of the replacement test event at the mean value of 0.997 GeV/c.



(a) Reconstructed P_x histogram with previous test event (green dot) and limits (green lines). Replacement test event at $P_x = 0.997$ GeV/c (blue dot) with newly implemented tolerance limits $0.993 < P_x < 1.00$ cm (blue lines). Note the lower limit of the previous test limits overlaps with the upper limit of the new test range.

(b) Reconstructed P_y histogram with previous and replacement test event both at zero (overlapping green and blue dots), previous limits (green lines) and new limits at $-0.023 < P_y < 0.003$ GeV/c (blue lines).

Figure 8: Histograms of reconstructed x and y momenta.

The P_y distribution had an average, initial value of zero and is narrow compared to the other components. The P_y distribution does not have a tail. The mean value of the fit to the central peak is 0.0 ± 0.063 GeV/c so we selected a test value of -0.01 GeV/c which is the same as the previous test value.

We have added the vertex position of the test track to the unit test. The distributions of the components of the track vertex are shown in Figures 9 and 10. In Figure 9 the central region of the v_Z distribution is Gaussian, but the high and low vertex tails are not. We fitted the central region to a Gaussian and obtained a value of 0.169 ± 0.485 cm. The reconstruction of the previous event is shown by the green point which is a full eight standard deviations away from the mean of the V_z distribution. We selected a test value of 0.40 cm.

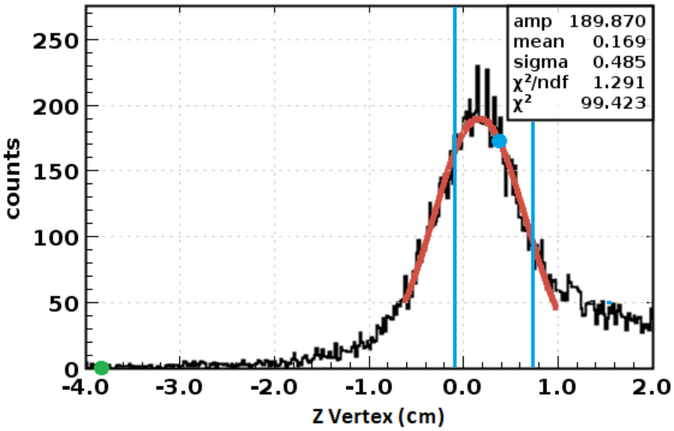
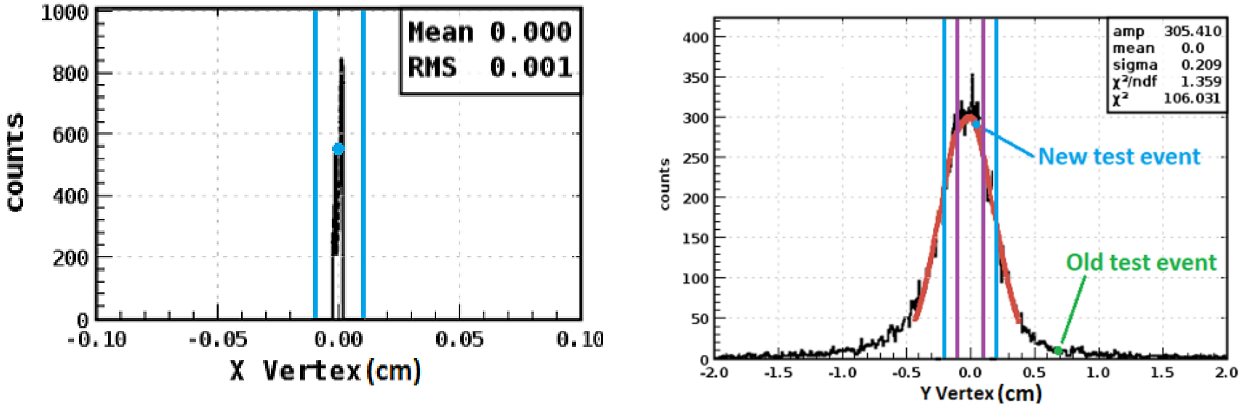


Figure 9: Simulated data V_z histogram with new test event at 0.40 ± 0.485 cm indicated with blue dot and vertical lines. The previous test event is more than 8 sigma from the mean at -3.84 cm indicated with green dot.



(a) Reconstructed V_x histogram spike at zero, new and previous test event both reconstructed at zero. New test limits at $0.993 < V_x < 1.001$ cm (blue lines)

(b) Reconstructed V_y histogram with previous test event at $V_y = 0.69$ cm (green dot), new test event at 0.0 ± 0.2 cm (blue dot and lines). New test event 0.5σ selection criteria indicated with purple vertical lines.

Figure 10: Histograms of reconstructed x and y vertices.

The x component of the vertex in Figure 10a is extremely narrow because the events all had small P_y components (the azimuthal angle range was $\phi = 0 \pm 5°$) so the track vertex is never far from zero ($V_x = 0.000 \pm 0.001$). We set $V_x = 0.0$ cm. The y component of the vertex in Figure 10b has a Gaussian distribution with a width in between the values for the x and z components. We set $V_y = 0.0$ cm. A summary of the central values for the momentum and vertex components is shown in Table 2.

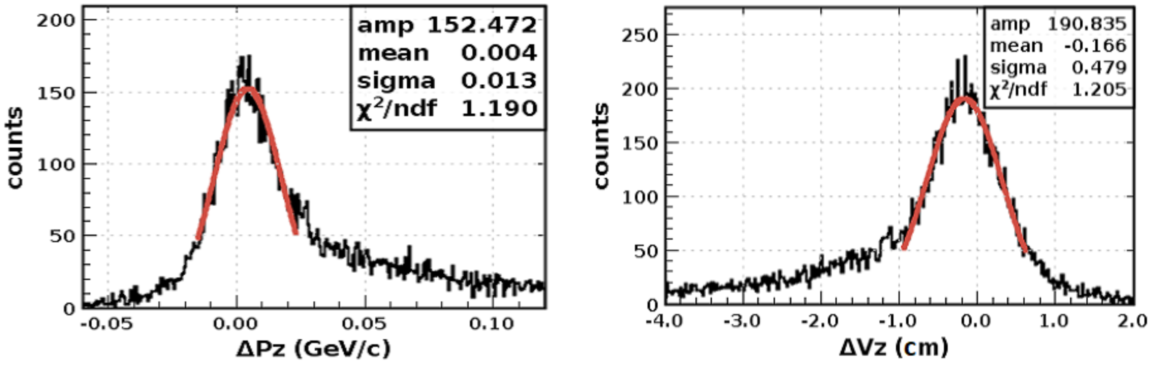
Property	New Test Average	New Test Limits
P_x (GeV/c)	0.997	0.997 ± 0.278
P_y (GeV/c)	-0.013	-0.010 ± 0.013
P_z (GeV/c)	2.04	2.04 ± 0.013
x vertex (cm)	0.00	0.000 ± 0.2
y vertex (cm)	-0.024	-0.00 ± 0.21
z vertex (cm)	0.40	0.40 ± 0.48

Table 2: New DC unit test particle properties and test particle reconstruction tolerance limits.

3.4 DC Unit Test Tolerance Limits

In order to update the DC unit test with new tolerance limits that match the new reconstruction version, the reconstruction resolution was studied. The resolution for each momentum and vertex component was determined by finding the difference between the reconstructed values and the generated ones. This difference is a measure of reconstruction error and is defined as $\Delta P = P_{gen} - P_{recon}$ and $\Delta V = V_{gen} - V_{recon}$ for the momentum and vertex respectively. The new unit test resolution would be used to determine the range around the selected central value taken from the results shown in Figures 7-10 and Table 2.

Histograms showing the frequency distribution of each ΔP and ΔV component are plotted in Figures 11-13. The energy loss tails observed in each figure are reversed when compared to the those in Figures 7-8. This difference is a result of defining these values as generated minus reconstructed so a loss in reconstructed momentum creates a positive difference. Figure 11a is a histogram of ΔP_z with a peak centred close to zero. The peak of the distribution has been fitted with a Gaussian function ($\chi^2_{reduced} = 1.19$) with $\sigma = 0.013$ GeV/c. The new unit test limits for P_z were originally selected at 2.04 ± 0.013 GeV/c, with the one sigma resolution (0.013 GeV/c) used as the tolerance limits around the central value where the test event is expected to be reconstructed at 2.04 GeV/c. At the discretion of the software working group the limits were later widened to avoid over-sensitivity to small changes in the geometry and the resulting false positives. The final range was 2.04 ± 0.36 GeV/c. A similar situation occurred for the P_x and ΔV_x ranges. The result of the fit to ΔP_x was $\sigma = 0.004$ GeV/c which we adjusted to $\sigma = 0.278$ GeV/c. The result of the fit to ΔV_x was $\sigma = 0.001$ cm which we adjusted to $\sigma = 0.20$ cm. The remaining ranges for the momentum, vertex position, and reconstruction crosses (see below) followed the procedure described here. See Section 3.3 and Table 2 for final momentum and vertex results that were implemented in the code.

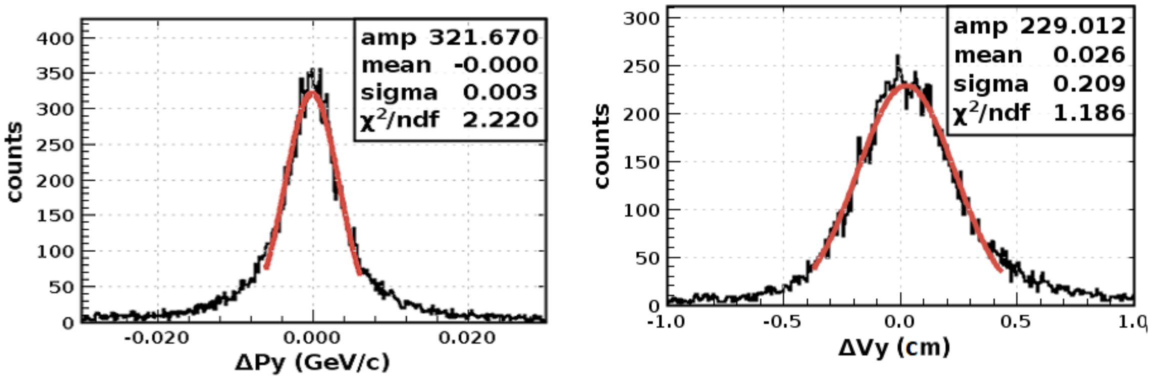


(a) Simulated data ΔP_z, distributed about mean of 0.004 GeV/c with $\sigma = 0.013$ GeV/c

(b) Simulated data ΔV_z, distributed about of zero mean of -0.166 with $\sigma = 0.479$ cm

Figure 11: Reconstruction resolution analysis z component momentum and vertex histograms.

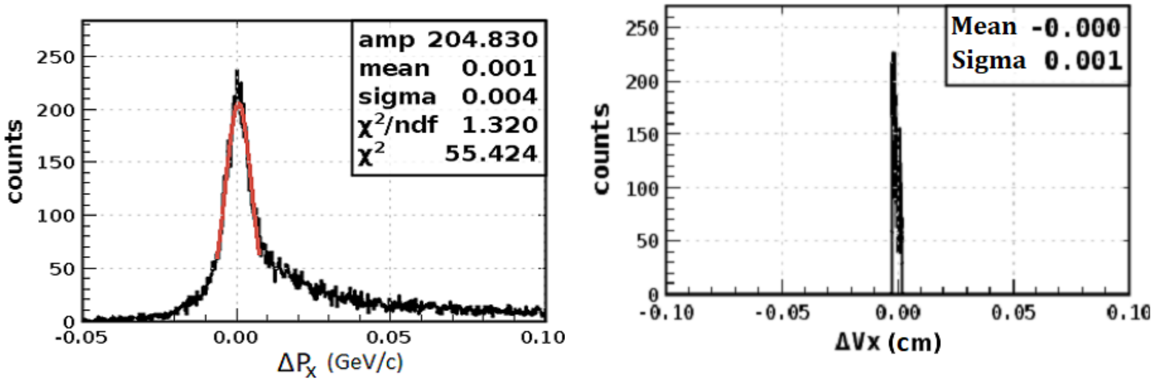
It is still possible that a test particle can be reconstructed outside the tolerance limits by random chance rather than a coding change. The software group made the choice of a one sigma tolerance by balancing the concerns that too tight a tolerance would produce an unacceptably high rate of false positives and that too loose a cut would increase the chance of software issues remaining undetected. The same analysis of the reconstruction resolution was applied to all other momentum and vertex components leading to the selection of the new unit test limits listed in Table 2 except for the cases mentioned above. The results for the reconstruction resolution for the momentum and vertex components are shown in Figures 11 - 13.



(a) Simulated data ΔP_y, distributed about mean of zero with $\sigma = 0.003$ GeV/c

(b) Simulated data ΔV_y, distributed about mean of 0.026 cm with $\sigma = 0.209$ *cm*

Figure 12: Reconstruction resolution analysis y component momentum and vertex histograms.



(a) Simulated data ΔP_x, distributed about mean of 0.01 GeV/c with $\sigma = 0.004$ GeV/c

(b) Simulated data ΔV_x, distributed about mean of zero with $\sigma = 0.001$ *cm*

Figure 13: Reconstruction resolution analysis x component momentum and vertex histograms.

3.5 DC Unit Test Reconstruction Crosses

The DC reconstruction relies upon the placement of reconstruction crosses to ensure the reconstructed trajectory is accurate as it passes between DC superlayers. The proper placement of these crosses is an important indicator that a new version of DC reconstruction software is performing as intended. The DC unit test was therefore expanded to check that reconstruction crosses are being accurately positioned.

3.5.1 DC Reconstruction Cross Position Analysis

Crosses are positioned at a fixed midpoint along the z axis (in the sector coordinate system) between the three pairs of DC superlayers and vary in the $x - y$ plane depending on the reconstructed HBT trajectory. The position of the crosses for the three DC regions was extracted from the reconstructed GEMC simulation and used to produce the x and y cross position histograms in Figures 14-16. The electrons were generated as before with a momentum of $P = 2.5 \pm 0.5$ GeV/c with angles $\theta = 25^\circ \pm 10^\circ$ and $\phi = 0^\circ \pm 5^\circ$. As charged particles travel through each DC region, they lose momentum when they ionize a DC gas molecule and multiply scatter off detector material. The multiple scattering spreads out the distribution of trajectories of the charged particles. This effect is seen with each successive DC region the electrons pass through. The energy loss and multiple scattering broadens the the x and y position distributions in Figures 14a, 15a, and 16a. The range of the distribution of the cross positions in x corresponds to a range in θ and goes from $-40\ cm \ \rightarrow\ +40\ cm$ in Region 1 to $-130\ cm \ \rightarrow\ +80\ cm$ in Region 3. The range of the distribution of the cross positions in y corresponds to a range in the azimuthal angle ϕ and changes very little in the same DCs going from $-12\ cm \ \rightarrow\ +12\ cm$ in Region 1 to $-24\ cm \ \rightarrow\ +24\ cm$ in Region 3. The results are listed in Table 3.

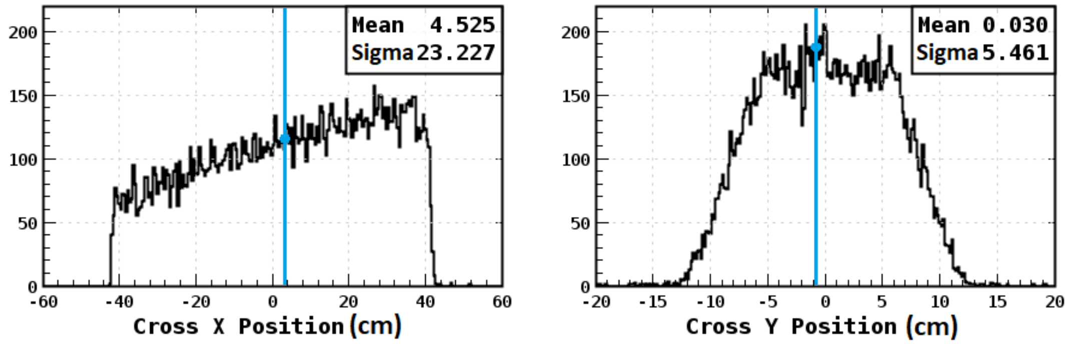
	Region 1	Region 2	Region 2
x range (cm)	$-40/40$	$-72/64$	$-130/80$
y range (cm)	$-12/12$	$-20/18$	$-24/24$
x average (cm)	4.53	3.12	-12.3
y average (cm)	0.03	0.01	0.02

Table 3: Ranges and averages for reconstruction crosses.

The standard CLAS12 configuration has the torus magnet polarity set so negative particles bend inward towards the beamline. Here, the generated electrons are centered on $\phi = 0^\circ$, *i.e.* along the x axis, so as they bend in the torus field the distribution shifts to smaller x. The effect on the y distribution is small. These features are observed in Figures 14-16. The mean x cross position decreases from 4.53 cm in Region 1 to 3.12 cm and -12.312 cm in Regions 2 and 3 respectively as they travel along this trajectory. The electrons were generated with little momentum in the y direction so the y position of the crossing points stays roughly the same, going from $y = 0.03 \pm 5.46\ cm$ in Region 1 to $y = 0.01 \pm 8.21\ cm$ and $y = 0.023 \pm 10.41\ cm$ in Regions 2 and 3 respectively. The new test event's reconstruction cross was found to be close to the mean of each distribution with $\langle x \rangle = 4.07\ cm$ and $\langle y \rangle = -1.95\ cm$ for Region 1, $\langle x \rangle = 4.07\ cm$ and $\langle y \rangle = -1.27\ cm$ for Region 2 and $\langle x \rangle = -11.0\ cm$ $\langle y \rangle = 2.85\ cm$ for Region 3. In order to put tolerance limits on these cross positions the resolution of each cross position were studied.

3.5.2 DC Reconstruction Cross Position Resolution

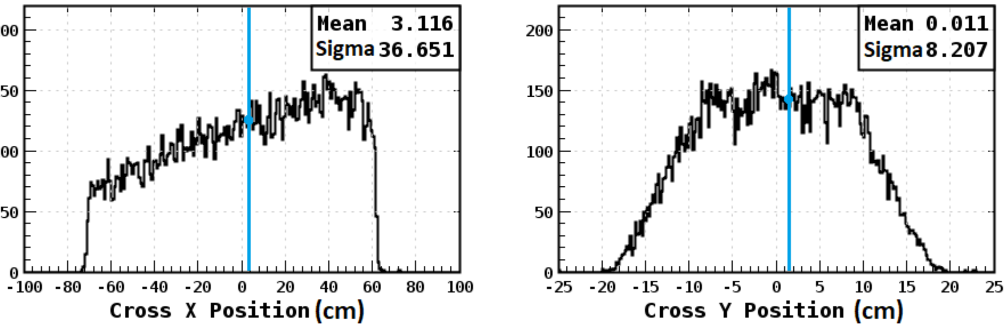
The starting point and initial momentum of each generated particle is exactly known from the simulation input parameters. Using these starting points and by taking into account the geometry of the detector and toroidal field, the expected crossing points were calculated. The difference between the generated positions and the reconstructed ones were extracted and plotted in Figures 17, 18, and 19 where $\Delta X = x_{gen} - x_{recon}$



(a) Simulated data cross x position in Region 1. New test event reconstruction has cross at x = 4.07 cm (blue line).

(b) Simulated data cross y position in Region 1. New test event reconstruction has cross at y = -1.95 cm (blue line).

Figure 14: Region 1 cross position histograms from simulated data reconstructions.



(a) Simulated data cross x position in Region 2. New test event reconstruction has cross at x = 4.02 cm (blue line).

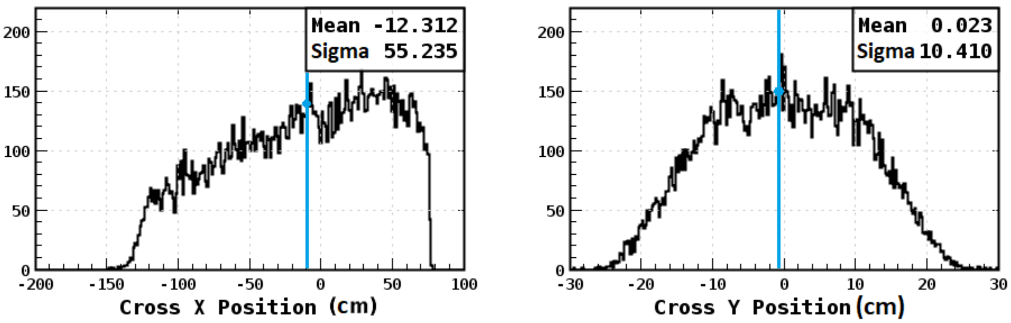
(b) Simulated data cross y position in Region 2. New test event reconstruction has cross at y = -1.27 cm (blue line).

Figure 15: Region 2 cross position histograms from simulated data reconstructions.

and $\Delta Y = y_{gen} - y_{recon}$. As with the previous reconstruction resolution analysis in Section 3.4, this difference is defined as generated minus reconstructed. The energy loss tails in Figures 11, 12, and 13 therefore move to the positive side of the cross-position-difference histograms as a more negative cross position gives a positive difference between the reconstructed and the generated.

The software working group decided the reconstruction cross tolerance limits would be one standard deviation above and below where the new unit test particle's reconstruction crosses are currently positioned. As with the unit test limit selection in Section 3.4, this decision was reached by balancing the rate of possible false positives that could arise due to random chance with the possibility that a software failure may go undetected.

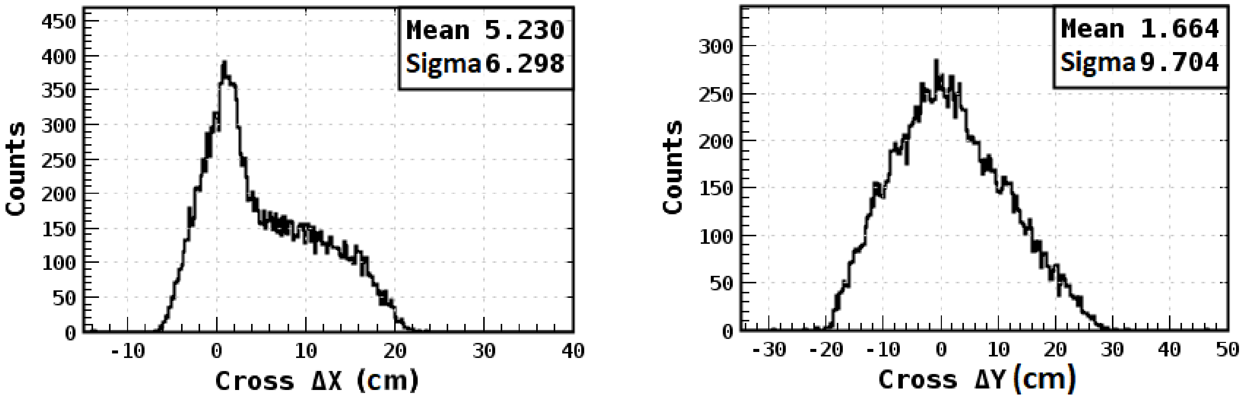
The standard deviations of the x position of the reconstruction crosses for Regions 1, 2 and 3 were 6.3 cm, 9.99 cm and 14.01 cm. The central values were assigned using the result in Table 2 and Section 3.4. The unit test reconstruction cross x position tolerance limits were therefore put at 4.07 ± 6.30 cm



(a) Simulated data cross x position in Region 3. New test event reconstruction has cross at x = -11.0 cm (blue line).

(b) Simulated data cross y position in Region 3. New test event reconstruction has cross at y = 2.85 cm (blue line).

Figure 16: Region 3 cross position histograms from simulated data reconstructions.



(a) Simulated data Region 1 ΔX cross position. Mean of 5.23 cm with $\sigma = 6.298\ cm$.

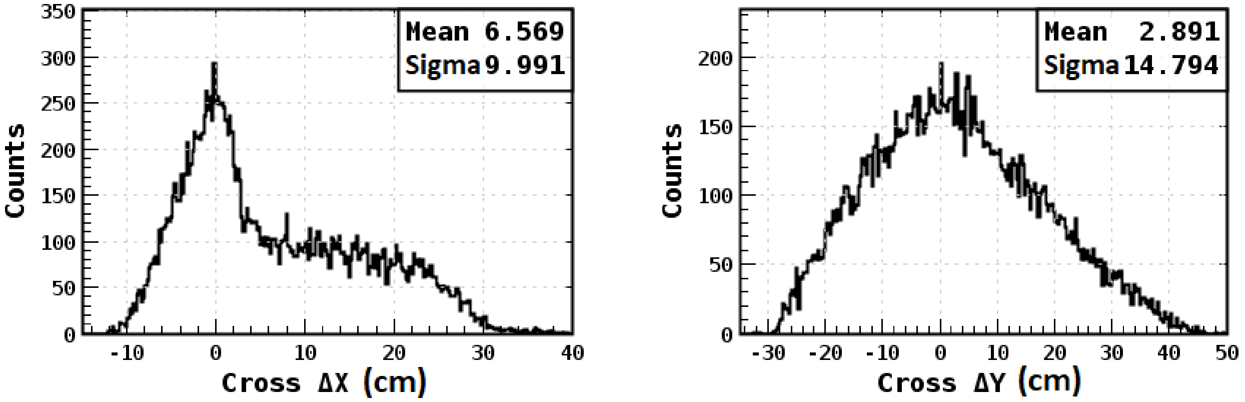
(b) Simulated data Region 1 ΔY cross position. Mean of zero with $\sigma = 2.162\ cm$.

Figure 17: Region 1 cross position resolution histograms.

(Region 1), $4.02 \pm 9.99\ cm$ (Region 2), and $-11 \pm 14.01\ cm$ (Region 3) respectively. Similarly the y position standard deviations were $9.70\ cm$, $14.79\ cm$ and $20.81\ cm$ so the limits were $-1.95 \pm 9.70\ cm$ (Region 1), $-1.27 \pm 14.79\ cm$ (Region 2), and $-2.85 \pm 20.81\ cm$ (Region 3). The new test event cross positions and tolerance limits are summarized in Table 4.

4 Updated DC Unit Test Code

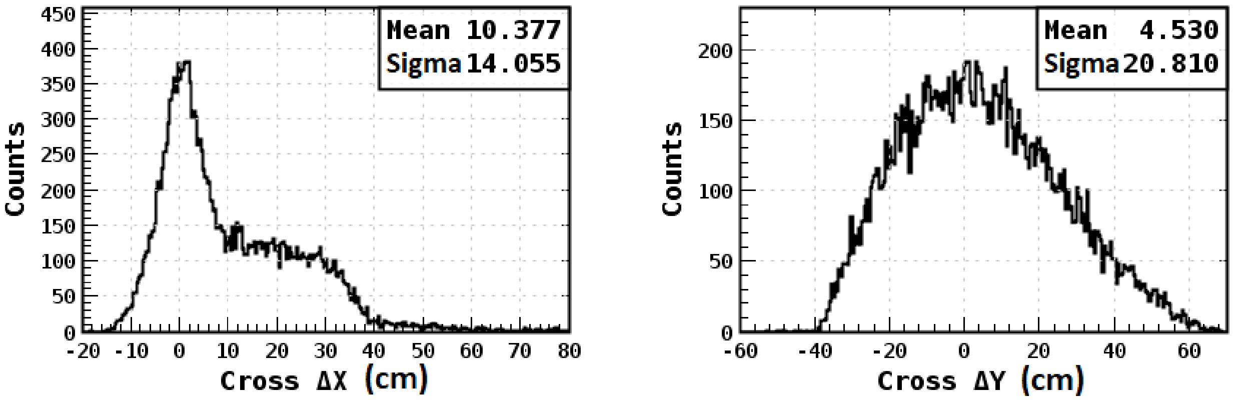
The new DC unit test works similar to the previous one, beginning by reconstructing the properties of the new test event from preprogrammed detector data. The unit test checks the reconstructed momentum, vertex and reconstruction crosses lay within the new tolerance limits and signaling that there has been software failure if they don't. The new unit test was written in JAVA and has been included in the latest versions of



(a) Simulated data Region 2 ΔX cross position. Mean of 6.569 cm with $\sigma = 9.991$ *cm*.

(b) Simulated data Region 2 ΔY cross position. Mean of 2.891 cm with $\sigma = 14.794$ *cm*.

Figure 18: Region 2 cross position resolution histograms.



(a) Simulated data Region 3 ΔX cross position. Mean of 10.377 cm with $\sigma = 14.055$ *cm*.

(b) Simulated data Region 3 ΔY cross position. Mean of 4.53 cm with $\sigma = 20.81$ *cm*.

Figure 19: Region 3 cross position resolution histograms.

COATJAVA used by the collaboration since December 2019. See the Appendix for the DC unit test source code.

5 DC Reconstruction Software Validation Conclusion

The CLAS12 collaboration relies upon the COATJAVA software package to reconstruct the momentum, vertex and identity of particles from the raw detector data they collect in their electron scattering experiments with JLab's 12 GeV continuous electron beam. COATJAVA is split into many modules, each with a software unit test that validates new versions of source code before they are integrated into new builds. The unit test associated with the module of code that reconstructs particle momenta and trajectory from detector data

Property	Cross x Position	Cross y Position	Cross x limits	Cross y Limits
Region 1 (cm)	4.07	-1.95	4.1 ± 6.3	-1.95 ± 9.70
Region 2 (cm)	4.02	-1.27	4.0 ± 10.0	-1.27 ± 14.8
Region 3 (cm)	-11	3	-11 ± 14	-3 ± 21

Table 4: Reconstruction cross position test positions and ranges.

collected by CLAS12's DCs was falsely signaling failures in new versions and so had become unusable.

The unit test works by reconstructing a test particle's momentum from preprogrammed detector data and checking that it's results lie within a range of values. The previous test particle data was found to be reconstructed outside these ranges when using COATJAVA version 6.3.1. The reconstruction software had evolved so the limits had become outdated and needed to be replaced.

Simulated data representing the passage of electrons through CLAS12's detector systems was generated using GEMC and reconstructed with COATJAVA. The test particle was reconstructed and compared with the generated data. The value of P_z was well outside the previous tolerance limits. The previous test particle was shown to have a V_z more than eight σ from the mean for particles generated under the same conditions. A new set of preprogrammed data was selected from the simulated data representing a new test particle. This particle was selected so that it would have momentum and vertex x, y, and z components within the range $(0.5 - 1.0)\sigma$ from the mean of the other particles in the simulation.

In order to select new unit test tolerance limits, the reconstruction momentum and vertex resolution was studied. These resolutions were determined by comparing the reconstruction results with the generated ones. The new unit test limits were selected with the width of the resolution around where the new test particle is reconstructed. Some adjustments were made to the tolerance limits to reduce the sensitivity to small changes. The unit test was expanded to examine the test particle's reconstructed vertex as well as its momentum.

An important step in the reconstruction process is the interpolation of reconstruction crosses that characterize the path of particles between DC superlayers. The placement of reconstruction crosses in the reconstruction of the simulated data was compared to how they should have been based on how the particles were generated. From this comparison the cross placement resolution was extracted. The unit test was expanded to check that reconstruction crosses are positioned accurately to within the resolution as an additional indicator of a potential software failure. The updated DC unit test has since been included in the latest development builds of COATJAVA as of December 2019.

6 Bibliography

[1] V.D. Burkert et al, “The CLAS12 Spectrometer at Jefferson Laboratory”, Nucl. Instrum. Methods A 959 (2020) 163419.

[2] V. Ziegler, “CLAS Software Overview”, 23rd Feb 2016, Proceedings of CLAS12 collaboration meeting, Jefferson lab, Newport News, Virginia, USA, URL:https://www.jlab.org/indico/event/125/

[3] V. Ziegler, “CLAS12 Software Document, Hall B 12 GeV Upgrade”, 22nd November 2013, Version 1.4, URL:https://www.jlab.org/Hall-B/clas12-software-nov13.pdf.

[4] J. Dudek et al, “Physics opportunities with the 12 GeV upgrade at Jefferson Lab”, December 2012, European Physical Journal A, DOI:10.1140/epja/i2012-12187-1.

[5] J. Arrington, “The Science and Experimental Equipment for the 12 GeV Upgrade of CEBAF”, 10th January 2005, Jefferson Lab.

[6] D. Kashy, “CLAS12 TORUS Magnet”, March 2017, Jefferson Lab, URL:https://www.jlab.org/Hall-B/clas12-web/specs/torus.pdf.

[7] Nuclear Science Advisory Committee, “A Long Range Plan for Nuclear Science”, 1979, URL: https://www.clasweb.jlab.org/wiki/images/c/ce/science.energy.gov/np/nsac/.

A New DC Reconstruction Unit Test Source Code

```
package org.jlab.service.dc;

import cnuphys.magfield.MagneticFields;
import java.util.Map;
import org.junit.Test;
import static org.junit.Assert.*;

import org.jlab.io.base.DataEvent;
import org.jlab.service.dc.DCHBEngine;
import org.jlab.service.dc.DCTBEngine;

import org.jlab.analysis.physics.TestEvent;
import org.jlab.analysis.math.ClasMath;

import org.jlab.clas.swimtools.MagFieldsEngine;
import org.jlab.jnp.hipo4.data.SchemaFactory;
import org.jlab.utils.CLASResources;
import org.jlab.utils.system.ClasUtilsFile;

/**
 *
 * @author nharrison, marmstr
 */
public class DCReconstructionTest {

  @Test
  public void testDCReconstruction() {
    System.setProperty("CLAS12DIR", "../../");

    String mapDir = CLASResources.getResourcePath("etc")+"/data/magfield";
        try {
            MagneticFields.getInstance().initializeMagneticFields(mapDir,
                    "Symm_torus_r2501_phi16_z251_24Apr2018.dat","Symm_solenoid_r601_phi1_z1201_13June2018.dat");
        }
        catch (Exception e) {
            e.printStackTrace();
        }

    String dir = ClasUtilsFile.getResourceDir("CLAS12DIR", "etc/bankdefs/hipo4");
    SchemaFactory schemaFactory = new SchemaFactory();
    schemaFactory.initFromDirectory(dir);

    DataEvent testEvent = TestEvent.getDCSector1ElectronEvent(schemaFactory);

    MagFieldsEngine enf = new MagFieldsEngine();
    enf.init();
    enf.processDataEvent(testEvent);
    DCHBEngine engineHB = new DCHBEngine();
    engineHB.init();
    engineHB.processDataEvent(testEvent);
    if(testEvent.hasBank("HitBasedTrkg::HBTracks")) {
        testEvent.getBank("HitBasedTrkg::HBTracks").show();
    }

    //Compare HB momentum to expectation
    assertEquals(testEvent.hasBank("HitBasedTrkg::HBTracks"), true);
    assertEquals(testEvent.getBank("HitBasedTrkg::HBTracks").rows(), 1);
    assertEquals(testEvent.getBank("HitBasedTrkg::HBTracks").getByte("q", 0), -1);
    assertEquals(ClasMath.isWithinXPercent(16.0, testEvent.getBank("HitBasedTrkg::HBTracks").getFloat("p0_x", 0), 1.057), true);
    assertEquals(testEvent.getBank("HitBasedTrkg::HBTracks").getFloat("p0_y", 0) > -0.1, true);
    assertEquals(testEvent.getBank("HitBasedTrkg::HBTracks").getFloat("p0_y", 0) < 0.1, true);
    assertEquals(ClasMath.isWithinXPercent(16.0, testEvent.getBank("HitBasedTrkg::HBTracks").getFloat("p0_z", 0), 2.266), true);

    //TB reconstruction
    DCTBEngine engineTB = new DCTBEngine();
    engineTB.init();
    engineTB.processDataEvent(testEvent);
    if(testEvent.hasBank("TimeBasedTrkg::TBTracks")) {
        testEvent.getBank("TimeBasedTrkg::TBTracks").show();
    }


    assertEquals(testEvent.hasBank("TimeBasedTrkg::TBTracks"), true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBTracks").rows(), 1);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBTracks").getByte("q", 0), -1);

    assertEquals(ClasMath.isWithinXPercent(27.9, testEvent.getBank("TimeBasedTrkg::TBTracks").getFloat("p0_x", 0), 0.997), true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBTracks").getFloat("p0_y", 0) > -0.0702, true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBTracks").getFloat("p0_y", 0) < 0.0438, true);
    assertEquals(ClasMath.isWithinXPercent(17.5, testEvent.getBank("TimeBasedTrkg::TBTracks").getFloat("p0_z", 0), 2.04), true);

    assertEquals(testEvent.getBank("TimeBasedTrkg::TBTracks").getFloat("Vtx0_x", 0) < 0.2, true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBTracks").getFloat("Vtx0_x", 0) > -0.2, true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBTracks").getFloat("Vtx0_y", 0) < 0.18, true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBTracks").getFloat("Vtx0_y", 0) > -0.228, true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBTracks").getFloat("Vtx0_z", 0) < 0.885, true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBTracks").getFloat("Vtx0_z", 0) > -0.0753, true);


    //Region 1
    assertEquals(ClasMath.isWithinXPercent(155, testEvent.getBank("TimeBasedTrkg::TBCrosses").getFloat("x", 0), 4.02), true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBCrosses").getFloat("y", 0 ) < 9.25, true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBCrosses").getFloat("y", 0 ) > -11.78, true);

    //Region 2
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBCrosses").getFloat("x", 1 ) < 14.2, true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBCrosses").getFloat("x", 1 ) > -5.8, true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBCrosses").getFloat("y", 1 ) < 13.9, true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBCrosses").getFloat("y", 1 ) > -17.8, true);

    //Region 3
    assertEquals(ClasMath.isWithinXPercent(127, testEvent.getBank("TimeBasedTrkg::TBCrosses").getFloat("x", 2), -11.0), true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBCrosses").getFloat("y", 2 ) < 17.96, true);
    assertEquals(testEvent.getBank("TimeBasedTrkg::TBCrosses").getFloat("y", 2 ) > -23.66, true);


  }

}
```