ETP_Bachelor_2019-15

Karlsruher Institut für Technologie

Untersuchung eines automatisierten Arbeitsablaufes zur Erstellung von Interpolationsgittern in perturbativer QCD

Law and order – a workflow for interpolation grids at higher order in perturbative QCD

Alexander Heidelbach

Bachelorthesis

Institut für Experimentelle Teilchenphysik (ETP)

Referent: PD Dr. Klaus Rabbertz
Koreferent: Prof. Dr. Günter Quast

Inhaltsverzeichnis

KAPITEL 1

Einleitung

In ferner Vergangenheit ragte das Verständnis über das Verhalten der eigenen Umgebung meist nur wenig über die vier Elemente Wasser, Erde, Feuer und Luft hinaus. Nach aktuellem Stand sind vier fundamentale Wechselwirkungen für die Existenz aller bekannten Materie verantwortlich. Drei dieser Wechselwirkungen werden im Standardmodell der Teilchenphysik beschrieben. Mithilfe von Experimenten, wie zum Beispiel dem *Large Hadron Collider* (LHC), werden Vorhersagen des Standardmodells mit Messdaten verglichen, um das Modell zu überprüfen. Dank moderner Technologien können Vorhersagen immer genauer getroffen und Messungen sehr präzise durchgeführt werden. In der Forschung tätige Personen arbeiten daran Programme und Strukturen zu entwickeln, die die Nutzung der computerbasierten Technologien effizienter gestalten. Eine Struktur zur Erstellung von Vorhersagen soll in dieser Bachelorarbeit untersucht und verbessert werden.

Ein Aspekt aktueller Forschung beinhaltet die Untersuchung der starken Wechselwirkung. Bei Streuprozessen mit Hadronen im Anfangszustand muss bei perturbativen Berechnungen der Quantenchromodynamik die Struktur des Protons berücksichtigt werden, damit Vorhersagen für den Endzustand getroffen werden können. Die Protonstruktur muss experimentell bestimmt werden. Die Verteilungen der Bestandteile des Protons werden mithilfe von Partonverteilungsfunktionen beschrieben. Deren Anpassung kann zum Beispiel mit den Messdaten des *Compact Muon Solenoid* (CMS) Experiments erfolgen. In dieser Bachelorarbeit werden diejenigen Endzustände der Proton-Proton Kollisionen betrachtet, die ein Z Boson und einen Jet im Endzustand aufweisen. Dieser Endzustand ist durch diejenigen Streuprozesse dominiert, die Gluonen im Anfangszustand besitzen. Die weitere Analyse dieser Ereignisse kann dazu genutzt werden, um die Gluonenverteilung weiter anzupassen.
Des Weiteren können wesentliche Differenzen der gemessenen und vorhergesagter Wirkungsquerschnitte in bestimmten Raumwinkelanteilen der Z+Jet Ereignisse festgestellt werden, wenn nur die Streuprozesse führender Ordnung betrachtet werden. Durch die Hinzunahme weiterer Ordnungen der Störungsrechnung ist es möglich diese Abweichungen teilweise aufzuheben.

Die Berechnungen höherer Ordnungen der Störungsrechnung erfordern eine hohe Rechenkapazität. Analysen, die statistisch signifikante Aussagen treffen, benötigen mehrere hunderttausend CPU Stunden Rechenleistung. Dadurch steht die effiziente Nutzung der zur Verfügung gestellten Computer im Vordergrund bei der Erstellung entsprechender Programmstrukturen. Theoretische Vorhersagen können in Interpolationsgittern festgehalten werden, die in Anpassungen der Protonstruktur iterativ verwendet werden können. Ein automatisierter Arbeitsablauf zur Erstellung der Interpolationsgitter [1] wurde erarbeitet. Mithilfe des Arbeitsablaufes sollen Vorhersagen der Wirkungsquerschnitte der Z+Jet Ereignisse in einem ausgewählten Phasenraum getroffen werden. Durch die Analyse und Verbesserung des Arbeitsablaufes sollen die Wirkungsquerschnitte bis zur dritten Ordnung der Störungsrechnung in guter statistischer Präzision berechnet werden.

Aufbau: Im zweiten Kapitel wird kurz der theoretische Hintergrund zusammengefasst, der benötigt wird zur Diskussionen der analysierten Ereignisse. Das dritte Kapitel erläutert Begrifflichkeiten zu der Betrachtung von Prozessen mit dem Z+Jet Endzustand. Kapitel vier liefert eine Überblick über die verwendete Software und beschreibt den zu untersuchenden Arbeitsablauf. Die erarbeiteten Anpassungen am Arbeitsablauf werden im fünften Kapitel dargestellt. Im letzten inhaltlichen Kapitel wird die Auswertung der erstellten Interpolationsgitter diskutiert.

KAPITEL 2

Theoretischer Hintergrund

Dieser Abschnitt beschreibt in kurzer und zusammengefasster Form den theoretischen Hintergrund, welcher für die physikalischen Diskussionen notwendig ist. Es werden im Folgenden die aufgelisteten Konventionen verwendet.

- **Natürliche Einheiten**
  Die Vakuumlichtgeschwindigkeit und das reduzierte Plancksche Wirkungsquantum werden als Größen der Normierung verwendet durch

$$\hbar = c = 1.$$

- **Einsteinsche Summenkonvention**
  Bei Komponentenrechnungen wird stets über gleiche Indizes summiert und das Summenzeichen wird nicht mitgeführt.

$$\vec{A} \cdot \vec{B} = \sum_{i=1}^{3} A_i B_i \hat{=} A_i B_i$$

  Entsprechend gilt dies für Matrizen und allgemeine Tensoren.

- **Vierer-Vektor Komponenten**
  Auch bei Komponenten eines Vierer-Vektors, und der entsprechenden Tensoren, gilt die Einsteinsche Summenkonvention. Im Folgenden werden zwei Punkte beachtet. Lateinische Buchstaben werden, falls nicht anders angegeben, von den Komponenten 1 bis 3 summiert. Griechische Buchstaben laufen dagegen von 0 bis 3. Desweiteren ist die Stellung der Indizes zu beachten, sodass $x_\mu x_\mu$ keine Summe mehr darstellt.

$$x^\mu = g^{\mu\nu} x_\nu$$

  $g^{\mu\nu}$ stellt die Minkowski-Metrik dar mit

$$g^{\mu\nu} = \begin{pmatrix} 1 & & & \\ & -1 & & \\ & & -1 & \\ & & & -1 \end{pmatrix}.$$

2.1 Standardmodell

Das Standardmodell der Teilchenphysik beschreibt drei der vier elementaren Wechselwirkungen in der heutigen Physik [2]. Unter die durch das Modell charakterisierten Wechselwirkungen fallen die schwache, die elektromagnetische und die starke Kopplung. Ohne Berücksichtigung bleibt die Gravitation. Das in Abbildung 2.1 dargestellte Schaubild stellt die Elementarteilchen dar. Diese Teilchen lassen sich unter Breücksichtigung des Spins in zwei Gruppen einteilen. Bosonen sind jene Teilchen mit einem ganzzahligen Spin und sind für den Austausch der Wechselwirkungen verantwortlich. Fermionen dagegen haben einen halbzahligen Spin und sind die wesentlichen Bestandteile der Materie. Dabei können wieder zwei Gruppen identifiziert werden: die Quarks und die Leptonen. Quarks unterliegen der starken Wechselwirkung und können aufgrund ihrer Farbladung nicht isoliert vorkommen[1]. Leptonen hingegen nehmen nicht an der starken Wechselwirkung teil. An dieser Stelle soll der Vollständigkeit halber erwähnt werden, dass im Standardmodell die Neutrions masselos angenommen werden und somit keine rechtshändigen Neutrinos vorkommen. Die Fermionen werden anhand ihrer Massen in drei Familien unterteilt, wobei nur die Teilchen der ersten Familie stabil sind.
Das Standardmodell stellt mit den Elementarteilchen in Abbildung 2.1 und den drei elementaren Wechselwirkungen eine spontan gebrochene Eichtheorie mit der Eichgruppe [3]

$$SU(3)_C \times SU(2)_L \times U(1)_Y$$

dar. Dabei beschreiben die Indizes jeweils die Art der Kopplung.
Die schwache Wechselwirkung mit der Eichgruppe $SU(2)_L$ beschreibt die Kopplung über den schwachen Isospin. Nur die linkshändigen Fermionen besitzen einen von null verschiedenen Isospin und bilden somit, unter entsprechender Darstellung, ein Schwaches-Isospin-Dublett. Die entsprechenden Eichbosenen/Austauschteilchen sind $W^{\pm}$ und Z^0.
Die elektromagnetische Wechselwirkung wird durch die Quantenelektrodynamik (QED) beschrieben. Dabei werden Wechselwirkungen zwischen Teilchen mit nicht verschwindender Ladung unter Austausch eines Photons betrachtet. Die entsprechende Eichgruppe ist damit die unitäre Gruppe $U(1)_{em}$. Die schwache und die elektromagnetische Wechselwirkung wird unter der Glashow-Weinberg-Salam-Theorie zusammengefasst zu der Vereinigung der Eichgruppen $SU(2)_L \times U(1)_Y$[2]. Diese wird durch den Higgs-Mechanismus

[1] Das Confinement wird in Abschnitt 2.2 genauer erklärt.
[2] Die $U(1)_Y$ Symmetrie ist bezüglich der schwachen Hyperladung.

spontan gebrochen, unter entsprechender Eichtransformation reduziert sich die Eichgruppe auf $U(1)_{em}$. Auf die starke Wechselwirkung wird in Abschnitt 2.2 näher eingegangen.

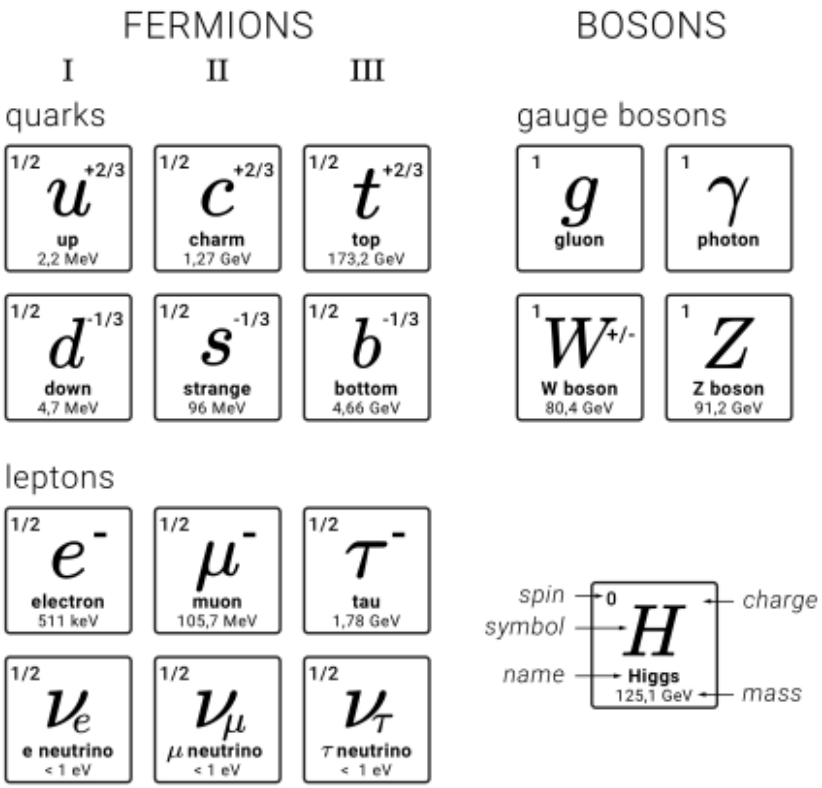


Abbildung 2.1: Schaubild der Elementarteilchen im Standardmodell [1]

2.2 Quantenchromodynamik

Die Quantenchromodynamik (QCD) ist die quantenfeldtheoretische Beschreibung der starken Wechselwirkung. Diese beschreibt die Wechselwirkung der Quarks durch die Gluonen. Die entsprechende Kopplung verläuft über die Farbladung der Elementarteilchen, die unter anderem aufgrund der Notwendigkeit einer antisymmetrischen Wellenfunktion der Baryonen eingeführt wird. Die mathematische Betrachtung erfolgt somit anhand der speziell unitären Gruppe $SU(3)_C$. Die zugehörige Lagrangedichte [4] ergibt sich aus dem eichinvarianten Term des Quarkfeldes und den eichinvarianten Gluonfeld-Tensoren

$$\mathcal{L}_{QCD} = \overline{\psi}\left(i\gamma_\mu D^\mu - m\right)\psi - \frac{1}{4}G^a_{\mu\nu}G^{\mu\nu}_a.$$

Dabei setzt sich das Quarkfeld aus drei Komponenten zusammen, die jeweils eine Farbladung repräsentieren. Die eichinvariante Ableitung besteht aus der kovarianten Ableitung und der Produktsumme aus den Vektorfeldkomponenten und den acht Generatoren T^a der Gruppe $SU(3)_C$.

$$D_\mu = \partial_\mu + igA_\mu = \partial_\mu + igA^a_\mu T_a$$

Hierbei ist g eine dimensionslose Kopplungskonstante. Der Gluonfeld-Tensor ergibt sich bei Berücksichtigung der Eichinvarianz zu

$$G^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu - g f_{abc} A^b_\mu A^c_\nu$$

mit der zugehörigen Strukturkonstanten f^{abc}, welche die Lie-Klammer definiert. Die entsprechenden Wechselwirkungsvertices, die aus der Lagrangedichte resultieren, sind in Abbildung 2.2 dargestellt.

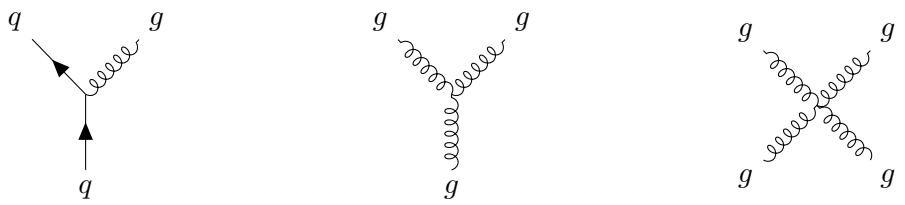


Abbildung 2.2: Darstellung der aus der Lagrangedichte der QCD resultierenden Vertizes: (links) Gluon Abstrahlung durch ein Quark/Aufspaltung des Gluons in Quark-Antiquark-Paar; (mitte & rechts) Selbstkopplung der Gluonen

Ein weiterer, essentieller Faktor der Quantenchromodynamik ist die laufende Kopplungskonstante. Anders als bei der Quantenelektrodynamik, ist die Abhängigkeit von diesem Faktor wesentlich bei der Beschreibung von Wechselwirkungen. Die Kopplungskonstante kann in Abhängigkeit einer frei wählbaren Skala μ_r bestimmt werden. Eine mögliche Wahl dieser Skala ist der Impulsübertrag Q von Prozessen. In erster Ordnung Störungsrechnung ergibt sich für die QCD eine Kopplungskonstante [2] von

$$\alpha_s(Q^2) = \frac{12\pi}{(33 - 2n_p) \ln \frac{Q^2}{\Lambda^2}},$$

wobei n_p die Anzahl der beteiligten Quarktypen ist. Die Anzahl ist ebenfalls abhängig vom Impulsübertrag Q. Λ ist ein rein experimentell zu messender Parameter. In Abbildung 2.3 ist der qualitative Vergleich zwischen den Kopplungskonstanten der starken Wechselwirkung α_s und der elektromagnetischen Wechselwirkung α_{em} dargestellt. Dabei sind beide Grenzwerte im Graph physikalisch motiviert.

In Bereichen von großem Q, entsprechend kleinem Abstand, nimmt die Kopplungskonstante bei der QED zu, da die Abschirmung von einzelnen Ladungen durch die virtuellen Elektron-Positron-Paare im Vakuum der QED abnimmt. Dieser Effekt kann ebenfalls mittels Quark-Antiquark-Paaren bei Farbladung auftreten. Jedoch wird der Effekt überkompensiert durch die Gluonenselbstwechselwirkung. Da die Gluonen auch selbst Farbladung tragen, „verschmieren“ sie die einzelnen Farbladung bei der Erzeugung eines virtuellen Gluonenpaares. Dieser Abfall der Kopplungskonstante wird im Grenzfall auch als asymptotische Freiheit bezeichnet.

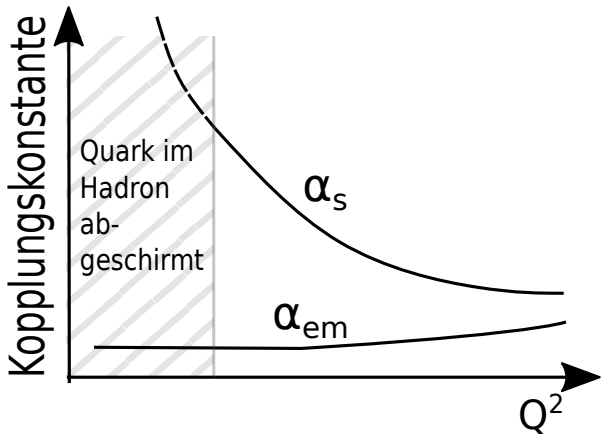


Abbildung 2.3: Qualitative Darstellung[3]des Verlaufs der Kopplungskonstanten für die QED und die QCD in Abhängigkeit des Impulsübertrages Q^2

Bei kleinen Impulsüberträgen (großen Abständen) sind bei elektromagentischen Wechselwirkungen die einzelnen Ladungen durch die virtuellen Paare abgeschirmt. Bei Quarks kommt jedoch noch ein weiterer Faktor hinzu: der Farbeinschluss (*engl.: color confinement*). Aufgrund experimenteller Beobachtungen ist der Farbeinschluss im Potential der Quarkfelder berücksichtigt. Am Beispiel eines Quark-Antiquark-Paares wird dies durch eine anziehende Wirkung beider Partner erklärt. Bei geringem Abstand ist das Potential schwach[4], jedoch steigt es mit wachsendem Abstand an, bis es sich als günstiger erweist, ein weiteres Quark-Antiquark-Paar zu erzeugen. Dieses Verhalten ist schematisch in Abbildung 2.4 dargestellt.

[3] Adaptiert von [2]
[4] In gebundenen Zuständen wird häufig ein Quark als näherungsweise frei betrachtet.

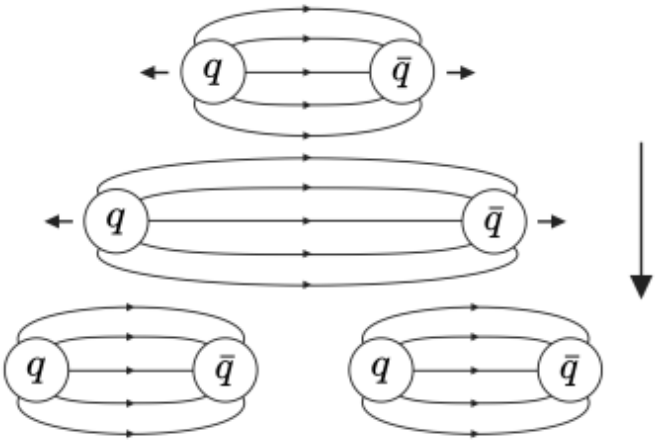


Abbildung 2.4: Schematische Darstellung des Eichfeldverhältnisses bei der Separation eines Quark-Antiquark-Paares [1]

2.3 Partonverteilungsfunktion

Von eminenter Bedeutung im Zusammenhang mit Streuexperimenten ist der Wirkungsquerschnitt σ. Bei Hadron-Hadron-Kollisionen treten Schwierigkeiten bei der Berechnung dieser Größe auf, die sich nicht mehr aus der perturbativen QCD vorhersagen lassen. Die Ursache hierfür ist der Aufbau der Hadronen. Sie bestehen neben den Valenzquarks, die die Quantenzahlen bestimmen, aus Gluonen und Quark-Antiquark-Paaren, die als Seequarks bezeichnet werden. Diese Bausteine werden zusammengefasst als Partonen definiert. Somit bestehen Hadron-Hadron-Kollisionen aus mehreren Subprozessen, die aus Parton-Parton-Kollisionen zusammengesetzt sind. Mithilfe des Faktorisierungstheorems [5] lässt sich der Wirkungsquerschnitt σ einer Parton-Kollision aufstellen. Hierbei werden die skalenabhängigen Partonverteilungsfunktionen $f_i(x, \mu)$ (*engl.: parton distribution function (PDF)*) der beteiligten Partonen von dem Wirkungsquerschnitt der harten Kollisionen $\hat{\sigma}_{ij}$, die sich perturbativ bestimmen lassen, getrennt. Somit kann der Wirkungsquerschnitt durch Gleichung (2.1) [1] angegeben werden, indem über die Impulsbruchteile x_i entsprechenden integriert und über die beteiligten Partonen summiert wird.

$$\mathrm{d}\sigma = \sum_{i,j} \int \mathrm{d}x_1 \mathrm{d}x_2 f_i(x_1, \mu_f) f_j(x_2, \mu_f) \times \mathrm{d}\hat{\sigma}_{ij}(x_1, x_2, \mu_r, \mu_f, \alpha_s(\mu_r)) \tag{2.1}$$

Die Impulsbruchteile x_i sind hierbei die Partonimpulse, die aus dem Protonimpuls resultieren und durch die Bjorken-Skalierung beschrieben werden.
Die PDFs $f_i(x, \mu)$ sind Wahrscheinlichkeitsverteilungen für das Auftreten eines Partons i in Abhängigkeit der Skala μ und dem Impulsbruchteil x bei Proton-Proton-Kollisionen. Diese Verteilungen lassen sich bislang nur experimentell durch verschiedene Streuprozesse

bestimmen. HERA [6] hat beispielsweise tiefinelastische $e^{\pm} - p$-Streuquerschnitte zur Bestimmung dieser Verteilungen verwendet. In Abbildung (2.5) sind Beispiele für diese Verteilungen zu zwei verschieden gewählten Skalen gezeigt.

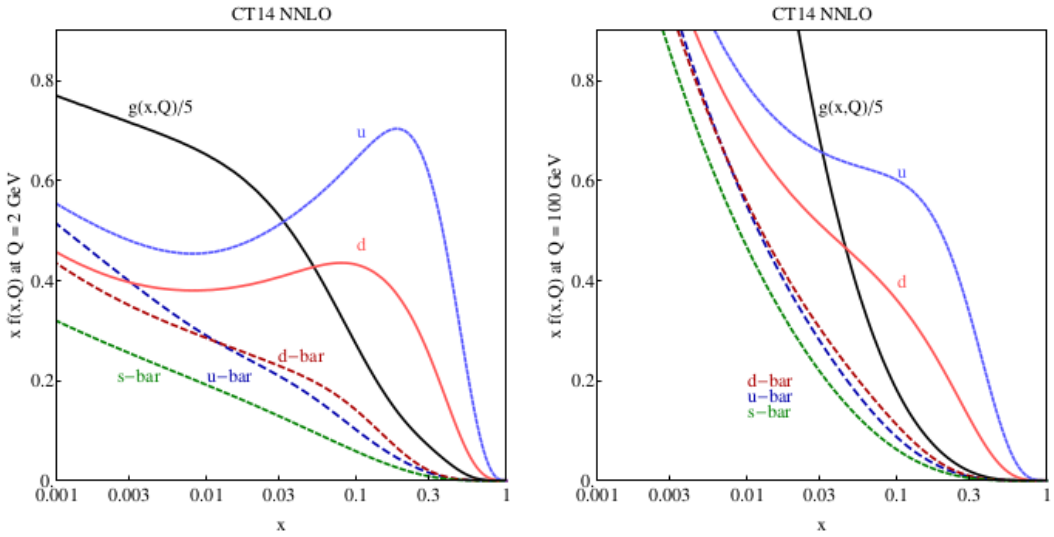


Abbildung 2.5: Partonverteilungen der Gruppe CTEQ [7] für $u, \bar{u}, d, \bar{d}, s = \bar{s}$ und g bei $Q = 2\,\text{GeV}$ und $Q = 100\,\text{GeV}$

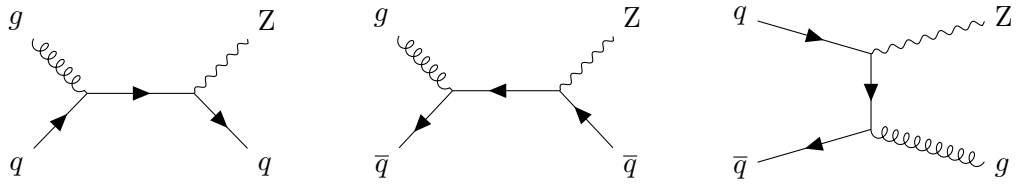
KAPITEL 3

Variablen bei der Z+Jet Analyse

Dieses Kapitel beschreibt in sehr kurzer Form den prinzipiellen Aufbau der Z+Jet Analyse. Das Z Boson ist eines der drei Austauschteilchen der schwachen Wechselwirkung. Es ist elektrisch neutral, besitzt einen Spin von eins und hat eine Masse von 91,187 GeV [8]. Aufgrund der hohen Masse hat das Z Boson nur eine sehr kurze Lebensdauer. Es kann prinzipiell in alle Fermion-Antifermion-Paare zerfallen. Bei der Auswertung wird jedoch der $Z \to \mu\mu$ Kanal betrachtet. Trotz höherer Wahrscheinlichkeit für andere Zerfallskanäle sind Myonen aufgrund der hohen Präzision des CMS Detektors geeigneter für die Auswertung.

3.1 Z+Jet erzeugende Ereignisse

Die hohe Präzision des CMS Detektors im Bezug auf Myonen ist, wie bereits erwähnt, bei der vorliegenden Analyse ausschlaggebend für die Wahl des $Z \to \mu\mu$ Kanals. Es werden die in Abbildung 3.1 dargestellten Ereignisse betrachtet, wobei diese nur für die führende Ordnung (*engl.: leading order (LO)*) der Störungsreihe aufgestellt sind. Hierbei wird deutlich, dass insbesondere jene Ereignisse von Interesse sind, die zu einem Z+Jet Ereignis im Endzustand führen. Die im Endzustand vorkommenden Partonen produzieren jeweils einen hadronischen Schauer, der sich ebenfalls im Detektor nachweisen lässt.



(a) Quark-Gluon **(b)** Antiquark-Gluon **(c)** Quark-Antiquark

Abbildung 3.1: Darstellung der möglichen Subprozesse bei Proton-Proton-Kollisionen für Z+Jet Erzeugung in erser Ordnung Störungsrechnung

Bei Betrachtung der Anfangszustände wird deutlich, warum gute Kenntnisse über QCD Prozesse, α_s Kopplung und vor allem die PDFs notwendig sind. Es handelt sich hierbei

um Prozesse, die unter starker Wechselwirkung bei Proton-Proton-Kollisionen auftreten. Wie anhand der Abbildung 2.5 überlegt werden kann, existieren verschiedene Wahrscheinlichkeiten für die drei LO Prozesse. Der Quark-Gluon Subprozess tritt häufiger auf, als der Antiquark-Gluon Subprozess, der wiederum deutlich häufig ist, als der Quark-Antiquark Prozess. Die genaue Analyse dieser Prozesse erlaubt eine wesentlich präzisere Bestimmung der Gluon PDF, die entscheidend für diese und andere Auswertungen ist. Korrekturen werden bei der Analyse hauptsächlich in der führenden Ordnung der QCD berechnet, da diese den größten Einfluss auf die Prozesse haben. Dabei kommt es zu Korrekturen zu den LO Subprozessen durch das Hinzufügen realer Abstrahlung von Partonen oder virtueller Gluonenschleifen. Desweiteren werden zusätzliche Parton-Parton-Kombinationen im Anfangszustand ermöglicht. Somit kommen die Subprozesse Gluon-Gluon, Quark-Quark und Antiquark-Antiquark hinzu.

3.2 Observablen

Die Analyse der eben beschriebenen Ereignisse erfordert eine geschickte Auswahl der verwendeten Observablen. Zur Beschreibung bieten sich der Impuls, die Ausrichtung und die Energie des vermuteten Z+Jet Ereignisses an. Die räumliche Definition des Laborsystems ist in Abbildung 3.2 dargestellt. Eine häufig verwendete Variable der Beschreibung ist der Transversalimpuls eines Teilchens

$$\vec{p_T} = \begin{pmatrix} p_x \\ p_y \end{pmatrix} \quad \Rightarrow \quad p_T = \sqrt{p_x^2 + p_y^2}.$$

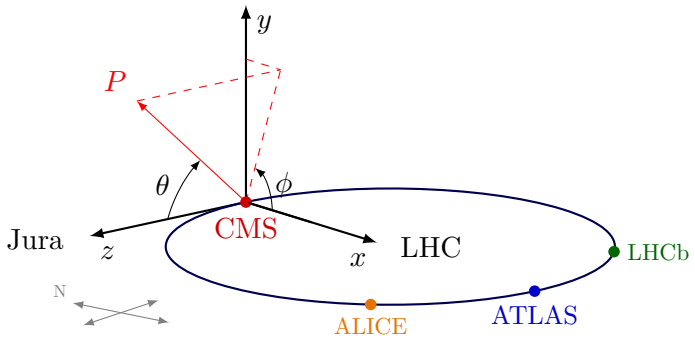


Abbildung 3.2: Schematische Darstellung der Polarkoordinaten-Definition im Laborsystem des CMS Detektors[9]

Desweiteren bieten sich Rapiditäten häufig zur Beschreibung der Kinematik eines Teilchens an. Die Rapidität eines Teilchens ist durch

$$y = \frac{1}{2} \ln \left(\frac{E + p_z}{E - p_z} \right)$$

definiert. Dabei entspricht E der Gesamtenergie und p_z dem Impuls in z-Richtung. Unter einem Lorentzboost entlang der z-Achse ist die Differenz zweier Rapiditäten erhalten. Dies wird später bei der Definition der Ereignistopologie ausgenutzt. In der relativistischen Näherung $E \approx p$ kann die Rapidität durch die Pseudorapidität in Gleichung (3.1) ersetzt werden. Der entscheidende Vorteil hierbei ist, dass diese Größe nur vom Polarwinkel θ abhängig ist.

$$y = \frac{1}{2} \ln \left(\frac{E + p_z}{E - p_z} \right) \overset{E \approx p}{\approx} \frac{1}{2} \ln \left(\frac{p(1 + \cos\theta)}{p(1 - \cos\theta)} \right) = -\ln \tan \frac{\theta}{2} \equiv \eta \tag{3.1}$$

Die Rapiditäten werden in Abschnitt 3.3 zur Klassifizierung der räumlichen Ausrichtung der Z+Jet Ereignisse verwendet. Die Energie kann durch zwei Observablen besonders gut beschrieben werden: p_T^Z und ϕ_η^*. Der Transversalimpuls des Z Boson Kandidaten p_T^Z wird dem des Jets vorgezogen, da die Zerfallsmyonen sehr gut mit dem CMS Detektor rekonstruiert werden können. Ebenfalls eine gute Wahl ist die in Gleichung (3.2) dargestellte Observable ϕ_η^*. Diese Variable ist lorentzinvariant für Boosts entlang der z-Achse, da sie nur von den Differenzen der Azimuthalwinkel $\phi^{\mu^\pm}$ und den Pseudorapiditäten $\eta^{\mu^\pm}$ des Myon-Antimyon Paares abhängt. Es kann gezeigt werden, dass beide Größen p_T^Z, ϕ_η^* stark korreliert sind [10].

$$\begin{aligned} \phi_\eta^* &= \tan \left(\frac{\pi - \left(\phi^{\mu^+} - \phi^{\mu^-} \right)}{2} \right) \sin\theta^* \\ \cos\theta^* &= \tanh \left(\frac{\eta^{\mu^+} - \eta^{\mu^-}}{2} \right) \end{aligned} \tag{3.2}$$

3.3 Binning

Die Ausrichtung der zu untersuchenden Ereignisse lässt sich anhand der Rapiditäten aufteilen. Dazu werden Kombinationen der einzelnen Rapiditäten y^Z, y^{Jet} verwendet. Die Differenz der Rapidität y^*, die durch

$$y^* = \frac{1}{2} \left| y^Z - y^{\text{Jet}} \right|$$

definiert ist, bietet sich an, da sie den Streuwinkel im Schwerpunktsystem darstellt. Die Summe beider Rapiditäten y_b beschreibt den Boost ins Schwerpunktsystem.

$$y_b = \frac{1}{2} \left| y^Z + y^{\text{Jet}} \right|$$

y_b enthält somit Informationen über die einzelnen Impulsbruchteile der am Prozess beteiligten Partonen. Unter Beachtung einer sinnvollen Wahl für die Grenzen der Observablen [10] kann die Ausrichtung eines Z+Jet Ereignisses durch entsprechende Anordnung

graphisch in Abbildung 3.3 dargestellt werden. Die Besetzung der Bins nimmt vom Zentralbereich aus nach Außen ab, wobei der Bereich mit hohem y_b durch die Quark-Gluon-Events dominiert ist [10].

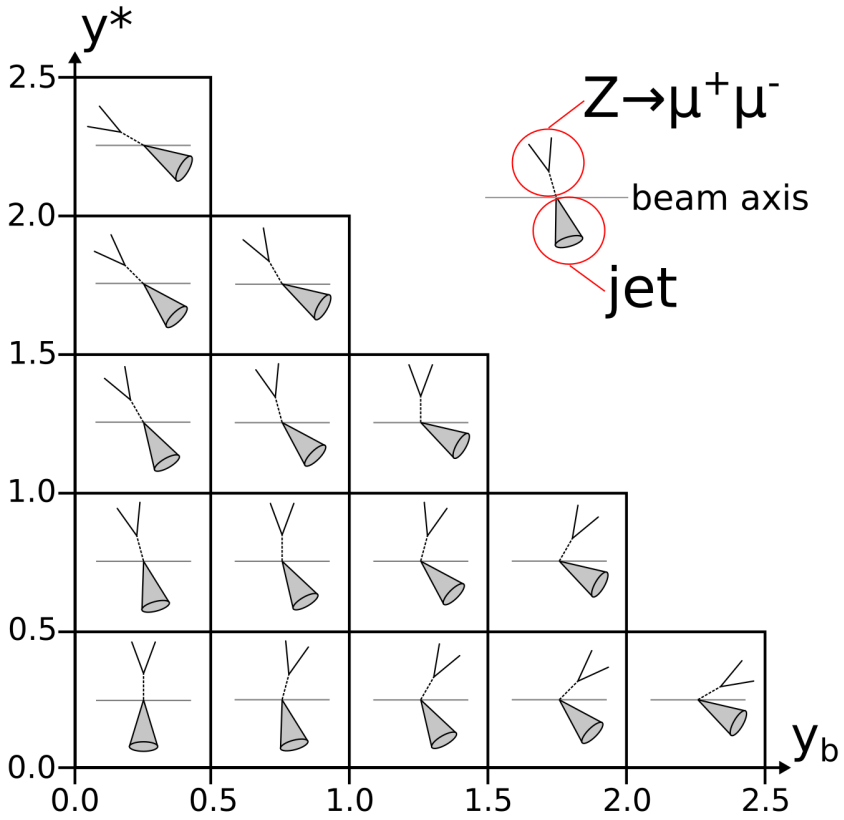


Abbildung 3.3: Geometrische Aufstellung der Z+Jet Topologie anhand der Rapiditäten y^* und y_b[10]

In den Phasenräumen $\left(p_T^Z, y_b, y^*\right)$ bzw. $\left(\phi_\eta^*, y_b, y^*\right)$ kann der dreifachdifferentielle Wirkungsquerschnitt aus den Daten rekonstruiert werden, der es ermöglicht, präzise Aussagen über die PDFs zu machen.

KAPITEL 4

Verwendete Software

Dieses Kapitel beschäftigt sich mit der Software, die für diese Arbeit verwendet wird. Im Vordergrund stehen dabei die Programmstrukturen, welche für die Produktion der Interpolationsgitter ausschlaggebend sind, bzw. diejenigen, mit denen der Nutzer den Arbeitsprozess von der Erstellung der Interpolationsgitter bis zur graphischen Überprüfung der Zwischenergebnisse überwacht.
Die angesprochenen Interpolationsgitter sind das Ergebnis eines Verfahrens zur Bestimmung des Wirkungsquerschnittes bei den in Abschnitt 2.3 angesprochenen Parton-Parton-Kollisionen. Die Korrekturen höherer Ordnungen (*Next-to-Leading-Order (NLO)* oder *Next-to-Next-to-Leading-Order (NNLO)*) für die Z+Jet Ereignisse sind im Programm NNLOJET [11] implementiert. Die Bestimmung der Wirkungsquerschnitte ist zeitaufwändig und kann mehrere hunderttausend CPU Stunden in Anspruch nehmen. Besonders aufwendig wird es bei Verwendung verschiedener PDFs. Die Projekte FASTNLO [12] und APPLGRID [13] verwenden besonders effiziente Methoden für wiederholte Berechnungen der Wirkungsquerschnitte unter Beachtung verschiedener PDFs. Dabei wird der Integrationsbereich des Wirkungsquerschnittes durch verschiedene Stützstellen in den Impulsbruchteilen und den Energieskalen beschrieben. Die zeitintensive Auswertung an den Stützstellen für den gesamten Phasenraum der harten Proton-Proton-Wirkungsquerschnitte muss somit nur einmal durchgeführt werden. Durch die Interpolation der Ergebnisse an den Stützstellen kann für die folgenden Rechnungen der tatsächliche Wirkungsquerschnitt als Summe über α_s, den PDFs und den Interpolationsgitterkoeffizienten aufgefasst werden.

Die Erstellung der Interpolationsgitter ist eine sehr zeitaufwändige und vielseitige Aufgabe. Zur Vereinfachung dieses Prozesses wurde ein Arbeitsablauf erstellt [1], der es ermöglicht die Arbeitsschritte aufzuteilen, zu strukturieren und gegebenenfalls zu kontrollieren. Ein automatisierter Arbeitsablauf ermöglicht es dem Nutzer, effizient und fehlerfrei zu arbeiten. Das hierzu verwendete Programmgerüst wird im Folgenden aufgestellt und erläutert.

4.1 Luigi

Luigi [14] ist ein von Spotify entwickeltes, softwaretechnisches Werkzeug zum Verwalten komplexer Pipelines bei Batch-Jobs. Eine Pipeline bezeichnet dabei mehrere Arbeitsprozesse, die möglichst parallel durchgeführt werden können. Als Batch Jobs werden Prozesse bezeichnet, die ohne weitere Eingabe im System anhand ihrer Abhängigkeiten durchgeführt werden. Das Python Paket Luigi bietet einen Rahmen für umfangreiche Datenanalyseprozesse mit komplexen internen Abhängigkeiten. Bestimmte Arbeitsschritte (im Folgenden als *engl.: tasks* bezeichnet) können definiert werden, die durch ihre Abhängigkeiten untereinander und ihre Ausgabedateien bestimmt sind. Luigi unterstützt den Nutzer dabei, die Arbeitsschritte in einer strukturierten Weise entsprechend ihrer Abhängigkeiten abzuarbeiten. Wird beispielsweise ein im Programmverlauf später auftretender Task vom Nutzer angefragt, so überprüft Luigi die entsprechenden Abhängigkeiten, welche in Form eines Abhängigkeitsgraphen alle vorher benötigten Tasks und ihre entsprechenden Abhängigkeiten vordefiniert sind. Der sogenannte Luigi *Scheduler* übernimmt dann die Aufgabe, die Tasks in geordneter Reihenfolge abarbeiten zu lassen. Somit werden überflüssige Arbeitsschritte vermieden, was einen besonders großen Einfluss auf die Rechenzeit und die Effizienz hat. Außerdem können dadurch Arbeitsabläufe vollzogen werden, die keine lineare Struktur aufweisen. Ein Beispiel für den Aufbau des Abhängigkeitsbaumes, den der Scheduler für die Bearbeitung übergeben bekommt, ist in Abbildung 4.1 gegeben. Jeder Knoten dieses Diagramms zeigt einen einzelnen Task.

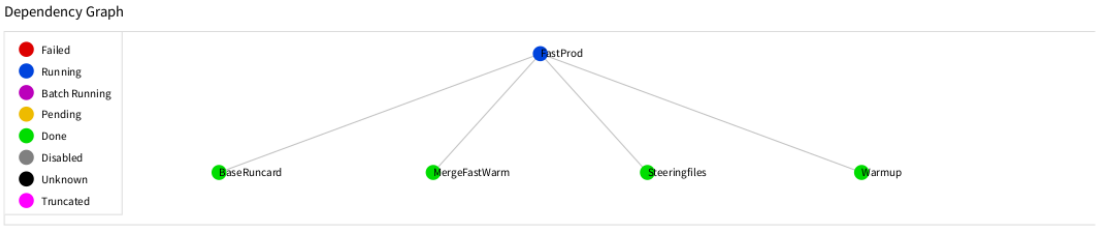


Abbildung 4.1: Die graphische Darstellung zeigt einen aufgebauten Luigi Schedulers. Die grünen Knoten sind bereits fertig gestellte Tasks während der blaue Knoten einen laufenden Prozess symbolisiert.

Listing 4.1 zeigt den prinzipiellen Aufbau einer Luigi Tasks Instanz, welcher grundlegend aus drei Methoden aufgebaut ist. Die Abhängigkeiten werden über die Methode *requires* definiert. Es müssen alle Abhängigkeiten erfüllt sein, bevor der Task durchgeführt werden kann. Die Methode *output* definiert die Speicher- und Benennungsstruktur der Outputdateien, die für folgende Tasks unter anderem für die Überprüfung der Abhängigkeit dienen. Schlussendlich kann in der *run* Methode der eigentliche Nutzprozess (*engl.: payload*) des Tasks definiert werden. In Analogie zu den drei wesentlichen Methoden eines Luigi Tasks können durch den Nutzer entsprechend weitere Methoden hinzugefügt werden. Zusätzlich kann ein Luigi Task globale Parameter übernehmen, die taskübergreifend durch den Nutzer definiert sind.
Luigi ist eine Hilfe zur Erstellung eines Entwurfsmusters und kann prinzipiell für verschiedene Software angewandt werden. Die Abhängigkeiten werden überprüft, indem

festgestellt wird, ob der entsprechende Output der benötigten Tasks existiert. Der logische Inhalt dieser Dateien wird dabei nicht überprüft. Somit können auch Output Dateien desselben Formats übernommen werden, falls diese bereits an anderer Stelle erzeugt wurden und im weiteren Verlauf für den Nutzer kompatibel sind.

```
import luigi
from TaskN import TaskN
from TaskM import TaskM

class TaskName(luigi.Task):
    '''
      Variablen Definition
    '''
    Parameter = luigi.parameter()

    def requires(self):
      '''
        Definitionsbereich aller Abhaengigkeiten
      '''
      return {
        'taskn': TaskN(),
        'taskm': TaskM()
      }

    def output(self):
      '''
        Ort und Name des Outputfiles
      '''
      return luigi.LocalTarget('OutputName.txt')

    def run(self):
      '''
        Definition der Run-Methode
      '''
      [...]
```

Listing 4.1: Darstellung des prinzipiellen Aufbaus eines Luigi Tasks mit den drei Methoden: *requires*, *output* und *run*

4.2 Law

Law (*luigi analysis workflow*) [15] ist ein auf Luigi basierendes Analyse Workflow Mangement Tool. Es hilft dabei, komplexere, groß-skalierte Anreihungen von Tasks zu erstellen, sogenannte *workflows*. Ein Workflow besteht damit aus mehreren Unterarbeitsprozessen (*engl.: branches*), die zu einem Task gehören. Ein essentieller Vorteil ist die Möglichkeit, einen Task mit verschiedenen Parametern parallel bearbeiten zu können. Somit ist ein Workflow erst dann vollständig, wenn der Output aller Branches vorhanden ist.

Die Task Verwaltung wird durch die zuvor angesprochene Abstraktion des Workflows erreicht. Des Weiteren bietet Law Mechanismen zur Speicherung der Output Dateien auf GridKa [16] und das Verschicken der Workflows an Batch Systeme. Das verwendete GridKa ist ein Großspeichersystem, welches mittel Griddateiprotokollen wie XRootD [17] und GridFTP [18] von überall erreichbar ist. Außerdem stellt GridKa selbst Rechenresourcen zur Verfügung. Die einzelnen Branches werden an HTCondor [19] submittiert, welches die anstehenden Arbeitsprozesse auf die angebundenen Ressourcen verteilt, sodass die Auslastung des Gesamtsystems optimiert wird.
Ein weiterer Vorteil von Law ist die in Abschnitt 4.3 beschriebene Umgebung, die dem Nutzer gegebenenfalls eine interaktive Steuerung und Überwachung des gesamten Arbeitsablaufes ermöglicht.

4.3 Aufstellung der Arbeitsumgebung

Das Git Verzeichnis [20] beinhaltet das Analysepaket zur Erstellung von Interpolationsgittern. Dort befinden sich als Submodule bereits Luigi, Law und Six (Python 2 und 3 Kompatibilität) sowie die Analysetasks und die entsprechenden Konfigurationsdateien. Die wesentlichen Bestandteile der Arbeitsumgebung sind die Konfiguration der Umgebungsvariablen, der physikalischen Analyse und Luigi. Im Folgenden wird auf die wichtigsten Aspekte eingegangen.

4.3.1 Umgebungsvariablen

Zunächst müssen die richtigen Umgebungsvariablen gesetzt werden. Dazu sind Dateien im Stil von `setup_law.sh` vorhanden. Diese setzen die wichtigsten Pfadvariablen. Dazu gehören unter anderem die Softwarepakete Luigi und Law aber auch das bereits angesprochene NNLOJet und fastNLO zur Erstellung und Auswertung der Interpolationsgitter. Ebenso muss GFAL2, die Bibliothek zur Kommunikation mit unter anderem dem Speichersystem bei GridKa, richtig einbezogen werden, da diese systemabhängig ist. Ein Beispiel für das Setzen der Umgebungsvariablen findet sich im Anhang A.1.
Mit dem korrekten Setzen der Pfadvariablen können bereits durch das Terminal die verschiedenen Law Tasks und Workflows angesprochen werden. Diese werden durch das Argument *run* aufgerufen.

```
$ law run TaskName
```

Nach dem entsprechenden Tasknamen können weitere Argumente folgen. Diese bieten interaktiven Zugriff auf Luigi Parameter[1], darunter zählen auch zahlreiche HTCondor Einstellungen, und weitere Optionen zur Verarbeitung. Die häufigst gebrauchten Optionen sind hierbei *print-status* und *remove-ouput*. Die weiteren Argumente werden in Form von *option flags* aufgerufen. Die folgenden Kommandos zeigen den Nutzen von *print-status*. Dabei gibt die erste Zahl die Tiefe der Abhängigkeit eines Tasks an, die zweite Zahl die Tiefe der einzelnen Strukturen.

[1] Diese werden in Abschnitt 4.3.3 erklärt.

```
$ law run TaskName --print-status 0,1
$ law run TaskName --print-status -1
```

Listing 4.2: Darstellung der Nutzung der *print-status* Option für Law Workflows

Die Ausgabe der Optionen im Listing 4.2 ist im Anhang A.2 dargestellt.
Ein weiteres relevantes Argument ist das Entfernen der Ausgabedateien. Dies kann mit der Option Flag *remove-output* durchgeführt werden. Die Tiefenangabe der Tasks funktioniert genauso wie oben nur ohne die Angabe für die Tiefe der einzelnen Strukturen. Sollten Zwischenergebnisse notwendigerweise gespeichert werden, so können die Verzeichnisse oder entsprechenden Dateien einfach umbenannt werden, so dass Law diese nicht mehr überprüft.

4.3.2 Analyse Konfiguration

Das Analysepaket law-analysis erfordert für die Analyse speziell angefertigte Anweisungen für NNLOjet und fastNLO. Diese liegen in Form von der *runcard* für NNLOjet und in Form von *steering files* für fastNLO vor. Die Runcard enthält unter anderem Informationen über die betrachteten physikalischen Prozesse in der jeweiligen Produktion, Massen und Breiten der Elementarteilchen und entsprechende Skalen, die Auswahlkriterien für Ereignisse sowie das Binning der betrachteten Observablen. Eine vollständige Runcard für die Produktion der ϕ^*_η-Tabellen befindet sich im Anhang A.3. Das Binning wurde entsprechend der Auswahlkriterien [10], dargestellt in der Tabelle 4.1 und der Abbildung 4.2, durchgeführt. Die Steering Files können anhand von vorhandenen Beispielen erstellt werden und müssen mit der Runcard kompatibel sein.

Tabelle 4.1: Auflistung der Bingrenzen zur Analyse der ϕ^*_η Observablen [10]

Observable		Bingrenzen
y		0,0; 0,5; 1,0; 2,0; 2,5
y_b		0,0; 0,5; 1,0; 2.0; 2.5
ϕ^*_η	Zentral (C)	0,4; 0,5; 0,6; 0,7; 0,8; 0,9; 1; 1,2; 1,5; 2; 3; 4; 5; 7; 10; 15; 20; 30; 50
	Grenzen (E)	0,4; 0,5; 0,6; 0,7; 0,8; 0,9; 1, 1,2; 1,5; 2; 3; 5; 10; 50
	Extrem (X)	0,4; 0,6; 0,8; 1; 5; 50

4.3.3 Luigi Konfiguration

Den Kern für die Nutzerkontrolle der Analyse stellt die Konfiguration der Luigi Tasks dar. Jeder Task besitzt eine Vielzahl an Parametern, die bei Bedarf individuell eingestellt werden können. Dies erlaubt dem Nutzer die einzelnen Analyseschritte zu überwachen und gegebenenfalls anzupassen.
Die Kontrolle der Teilschritte erfolgt über die Manipulation der `luigi.cfg` Datei oder

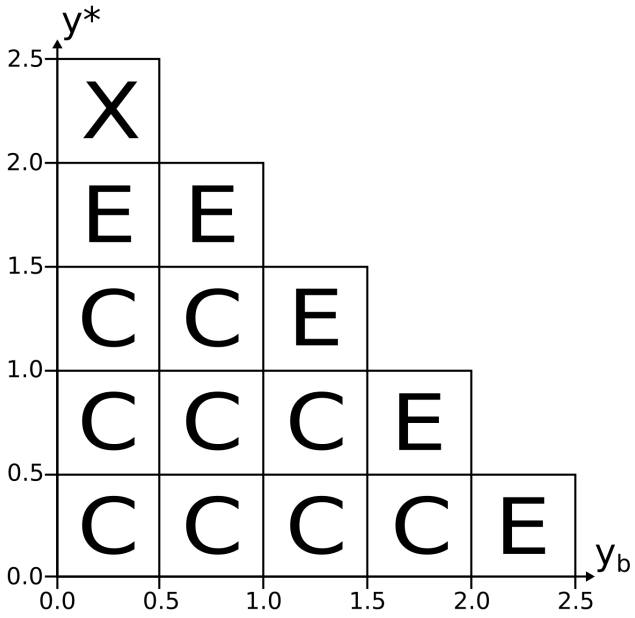


Abbildung 4.2: Sortierte Darstellung der Bingrenzen anhand der Observablen y_b und y^*, wie in Tabelle 4.1 definiert

in der Kommandozeile. Diese enthält die Definitionen aller Luigi Parameter. Die Konfiguration ist in zwei Abschnitte eingeteilt: den `[DEFAULT]` Bereich und den einzelnen `[TaskName]` Bereichen. Im `[DEFAULT]` Abschnitt werden die grundlegenden Parameter gesetzt, die für die Tasks als Standard eingestellt werden. Darunter fallen globale Parameter wie die NNLOjet Kanäle, die Bins der Observablen und einige zusätzliche Umgebungsvariablen aber auch die einzelnen HTCondor Konfigurationen. In den einzelnen `[TaskName]` Bereichen können dann Luigi Parameter umdefiniert werden. Ein Beispiel für die `luigi.cfg` Datei ist im Anhang A.4 dargestellt.

4.4 Die Arbeitsschritte

Die Produktion und Analyse eines Datensatzes besteht aus 17 Schritten [1], die in drei Bereiche aufgeteilt werden können. Der erste Bereich beinhaltet die Produktion der Interpolationsgitter mithilfe von NNLOjet. Hier wird für eine ausreichende statistische Genauigkeit die meiste Rechenzeit benötigt. Aus diesem Grund werden die tatsächlichen Berechnungen an HTCondor submittiert. Der zweite Bereich ist zuständig für das Zusammenfügen der erzeugten Interpolationsgitter. Die Arbeitsschritte innerhalb dieses Bereiches sind vom Prinzip her komplexer, erfordern jedoch weniger Rechenzeit und werden deshalb lokal durchgeführt. Der letzte Bereich stellt die Ergebnisse graphisch dar. Entsprechenden Skripte zur graphischen Darstellung sind in fastNLO eingebun-

den und können deshalb ebenfalls lokal ausgeführt werden. Abbildung 4.3 stellt die 17 implementierten Arbeitsschritte mit ihren Abhängigkeiten graphisch dar. Die folgenden Unterabschnitte fassen die einzelnen Arbeitsschritte zusammen.
Die Berechnungen der Wirkungsquerschnitte für die höheren Ordnungen der Störungsreihe orientieren sich dabei an der NNLOjet Aufteilung. Die Subprozesse werden in sieben Kanäle aufgeteilt. Der erste Kanal (LO) beinhaltet die Berechnungen der führenden Ordnung der Störungsrechnung. Die entsprechenden Feynman Diagramme sind bereits in Abbildung 3.1 dargestellt. Zu NLO Berechnungen sind zwei Kanäle definiert. Ein Kanal beinhaltet eine virtuelle Schleife in den Feynman Diagrammen (V), während der andere Kanal ein drittes Auslaufteilchen, welches in den Jet übergeht, berücksichtigt (R). Bei Berechnungen für die zweite Ordnung der Stöhrungsreihe sind die Kanäle entsprechend aufgeteilt für zwei zusätzliche auslaufende Teilchen (RRa und RRb), zwei Schleifen (VV) und die Mischung aus beiden (RV).

4.4.1 Produktion der Interpolationsgitter

Die Produktion der Interpolationsgitter ist, wie bereits angeführt, der zeitaufwändigste Schritt. Ein Großteil der Arbeitschritte wird an HTCondor submittiert und auf dem dCache bei GridKa gespeichert.
BaseRuncard und *Steeringfiles* sind lokale Luigi Tasks, welche die Konfiguration in richtiger Form auf das GridKa Verzeichnis übertragen. *Warmup* führt eine Vegas Integration des zur Verfügung stehenden Phasenraums durch. Dabei werden für jeden NNLOjet Kanal Verteilungsfunktionen für die Bereiche bestimmt, welche besonders relevant für die Untersuchung sind. Wie bereits zu Beginn des Kapitels angesprochen, kann der gleiche Phasenraum für die Berechnung verschiedener Wirkungsquerschnitte genutzt werden. Dadurch können gegebenenfalls bereits vorhandene Datensätze von anderen Analysen genutzt werden, um diesen Schritt zu überspringen. Dabei müssen die *Warmup* Archive nur korrekt benannt auf dem GridKa Verzeichnis hinterlegt werden. *FastWarm* bestimmt dann den benötigten Phasenraum und gibt die Grenzen für jedes Bin zwecks Optimierung an fastNLO weiter. Dieser Task wird genauso wie der *Warmup* Task an HTCondor submittiert. Im Anschluss fügt *MergeFastWarm* die Ergebnisse von *FastWarm* zusammen und kann aufgrund dessen lokal durchgeführt werden. Der letzte Schritt ist die Produktion der Interpolationsgitter. Dieser Schritt benötigt mit Abstand die meiste Zeit, besonders da hier die Anzahl der Ereignisse festgelegt wird. Für eine aussagekräftige statistische Genauigkeit benötigt der *FastProd* Task mehrere hunderttausend CPU-Stunden für den Fall von NNLO Berechnungen.

4.4.2 Zusammenfügen der Interpolationsgitter

Die durch *FastProd* erzeugten Interpolationsgitter werden durch drei verschiedene Prozesse zusammengefügt. Am Ende dieser Prozesse stehen fertige, richtig gewichtete und sortierte Interpolationsgitter für jeden NNLOjet Kanal und jede Obervable zur Verfügung. Zunächst werden jedoch die Ergebnisse von *FastProd* durch *CopyTables* auf einen lokalen Speicher verlagert. Im Anschluss beschäftigen sich die Schritte *Combine*,

MergeFastProd und *MergeFinal* mit dem richtigen Zusammensetzen der Interpolationsgitter. Während dieser Schritte wird ersichtlich, ob die Produktion der Interpolationsgitter überall erfolgreich durchgeführt wurde, da leere oder korrumpierte Output Dateien von *FastProd* zu Fehlern führen. Die Schritte *FnloCppread* und *FnloCppreadFinal* führen eine erste Auswertung der zusammengesetzten Ergebnisse durch.

4.4.3 Graphische Darstellung

Der letzte Abschnitt der Arbeitsprozesse beinhaltet die graphische Auswertung der Interpolationsgitter. Diese dienen zur Kontrolle der entstandenen Daten. *SingleScalecheck* nimmt aus den *FastProd* Datensätzen für jeden NNLOJET Kanal und jedes Observablenpaar den ersten Wert eines Satzes an Daten und vergleicht die Ergebnisse der Interpolation mit denen von NNLOJET. Dabei wird graphisch das Verhältnis und die Asymmetrie dargestellt. *Approxtest* stellt einen ähnlichen Vergleich auf, nur mit statistischen Größen für alle Interpolationsgitter. Die graphische Darstellung von *Absolute* beinhaltet die Darstellung des Wirkungsquerschnittes zu jedem zusammengefügten Interpolationsgitter und vergleicht diese mit den Wirkungsquerschnitten von NNLOJET. *AbsoluteAll* produziert eine ähnliche Ausgabe mit den Beiträgen aller Kanäle. Beispiele für diese Darstellungen werden in Kapitel 6 aufgeführt und genauer beschrieben.

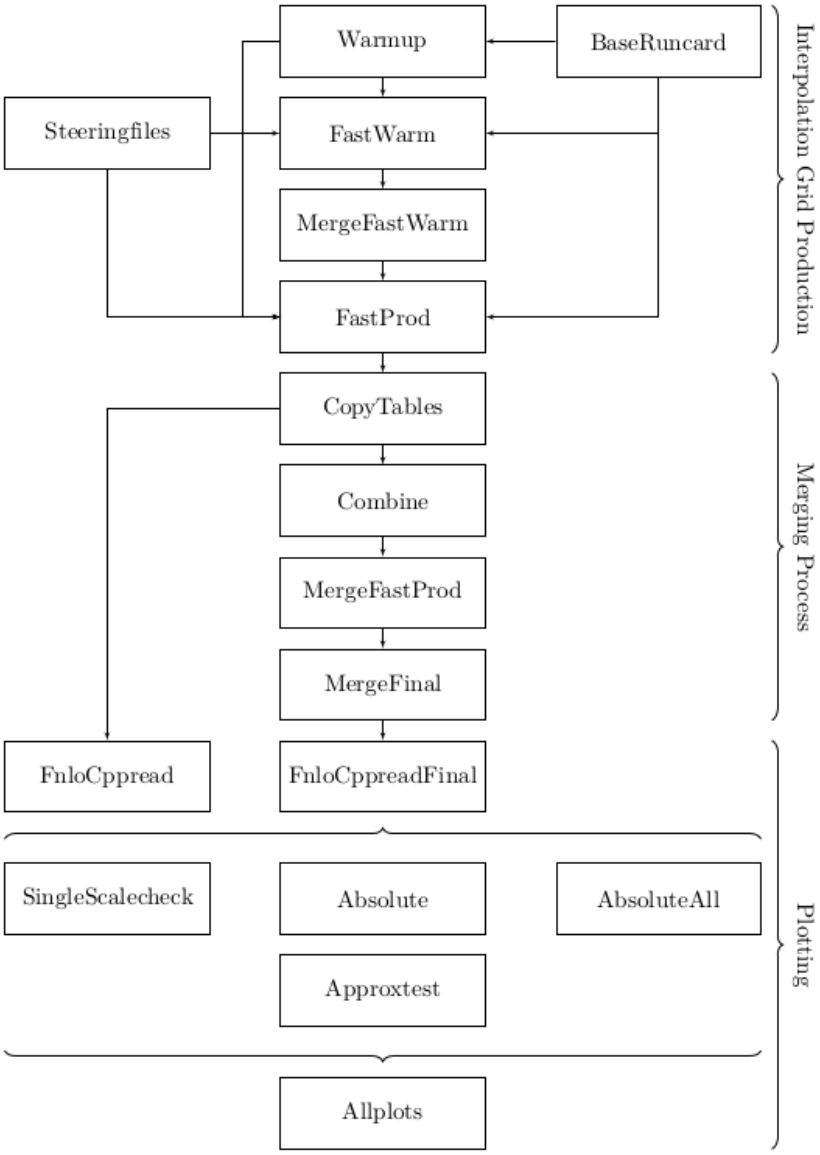


Abbildung 4.3: Schematische Aufstellung aller Arbeitsprozesse und ihrer Abhängigkeiten zur Produktion und Analyse der Interpolationsgitter [1]

KAPITEL 5

Anpassung des Arbeitsprozesses

Der erarbeitete Arbeitsablauf [1] zur Produktion der Interpolationsgitter ermöglicht es dem Nutzer die komplexen Rechenprozesse in einer strukturierten Weise zu kontrollieren. Dieses Kapitel beschreibt die genaue Untersuchung des Arbeitsablaufes, die dadurch entstandenen Verbesserungen und gibt Konfigurationsempfehlungen für eine effizientere Bearbeitung. Die Kernaspekte der Analyse der automatisierten Interpolationsgitterproduktion sind der zu Beginn durchgeführte Umstieg zu einer neueren LAW-Version und die Anpassung der Module des Software Rahmens und der LUIGI Tasks. Bei der Erstellung der Interpolationsgitter zu der ϕ_η^* Observable haben sich Konfigurationen hervorgehoben, die bei großen Datenmengen zu Schwierigkeiten führen können. Durch geschicktes Setzen der Umgebungsvariablen kann die Produktion auch für größere Datenmengen effizienter gestaltet werden. Zunächst soll der Ablauf einer Interpolationsgitterproduktion beschrieben werden.
Der Nutzer legt zu Beginn die Umgebungsvariablen fest. Diese wurden bereits in Abschnitt 4.3.1 diskutiert und ein Beispiel für die `setup_law.sh` Datei befindet sich im Anhang A.1. Im Anschluss kann mit den angepassten Runcards, Steeringfiles und WLCG, GSI Pfaden in der `luigi.cfg` die Produktion beginnen. Für eine statistisch signifikante Produktion der NNLO Interpolationsgitter müssen mehrere hunderttausende CPU Stunden geleistet werden. Somit ist es von Vorteil den prinzipiellen Aufbau mit wenigen Ereignissen zu testen. Dazu sollte die gesamte Pipeline durchlaufen werden. Die Ergebnisse müssen im Anschluss evaluiert werden und gegebenenfalls muss die Konfiguration angepasst werden. Anschließend kann die Produktion der Interpolationsgitter mit erhöhten Ereigniszahlen pro parallelisiertem Job erneut durchgeführt werden. Die Ausgaben der Zusammenfügung der Interpolationstabellen und die Abbildungen aus der graphischen Auswertung müssen vor einem erneuten Durchlauf gelöscht oder umbenannt werden, damit LAW die Workflows nicht als „beendet“ erkennt. Die Wiederholung der Arbeitsschritte kann mit diesem Schema durchgeführt werden, bis die gewünschte statistische Signifikanz erreicht ist.
Für die sich wiederholenden Schritte zur Produktion der Interpolationsgitter sollten die folgenden Aspekte berücksichtigt werden, um Rechenzeit zu sparen.

- Solange der, in der Runcard definierte, (x, μ^2) Phasenraum nicht verändert wird, kann, wie in Abschnitt 4.4.1 bereits erwähnt, die Vegas Integration durch *Warmup*

für verschiedene Observablen übernommen werden.

- Bei Änderungen des Binnings muss jeder Schritt ab inklusive *FastWarm* vollständig wiederholt werden.
- Sollten keine Änderungen im Binning bestehen, sondern nur die Anzahl der Ereignisse erhöht werden, dann kann der *FastProd* Task ohne das Entfernen der Ausgabe erneut gestartet werden. Es müssen nur die Submittierungs- und Statusdateien aus dem Verzeichnis entfernt werden, damit der Task neu submittiert wird.

5.1 Law-Version

Ein wesentlicher Bestandteil des *law-analysis* Verzeichnisses ist der *analysis* Unterordner. An dieser Stelle wird das von Law aufgestellte Gerüst um Luigi an den Nutzer und die Analyse angepasst. In diesem Verzeichnis wird das Grundgerüst von Law an die Analyse angepasst und die in Abschnitt 4.4 vorgestellten Workflows und Tasks definiert. Das Grundgerüst ist in der Datei *framework.py* definiert. Dort werden unter anderem die Law Tasks bzw. Workflow Typen und der *HTCondorJobManager* angepasst. Es werden zwei Arten an Law Workflows verwendet: *LocalWorkflow* und *HTCondorWorkflow*. Beim ersteren Workflow werden die einzelnen Branches auf dem gleichen Computer, von dem aus die Bearbeitung initialisiert wurde, gerechnet. Entsprechend wird beim anderen jeder einzelne Branch individuell über HTCondor auf dem Batch System bearbeitet.

In der Framework Datei werden zum Anpassen der eben angesprochenen Bestandteile ursprüngliche Klassen und Methoden von Law überschrieben. Die Analyse des Arbeitsablaufes und die dadurch entstandenen Verbesserungen resultieren aus dem Umstieg von der ursprünglich verwendeten Law-Version v0.0.21 auf die Version v0.0.30. Die durch die Aktualisierung eingeführten Änderungen an einigen Komponenten von Law, ziehen Änderungen am Analyse Framework nach sich. Im Wesentlichen sind bei bei dem Umstieg zwei Kollisionen aufgetreten, die den Arbeitsprozess verhindert haben. Beide sind durch den HTCondorJobManager begründet.
Der HTCondorJobManager ist eine Law Klasse, die hauptsächlich für die Kommunikation mit HTCondor verantwortlich ist und ist damit ein wesentlicher Bestandteil der HTCondorWorkflows. Der Jobmanager ist die Instanz, die die von Law aufgestellten Aufgaben submittiert, abfragt, abbricht und bereinigt.
Eine Änderung führt dazu, dass die Prozesse nicht an HTCondor submittiert werden. Dies wird durch die Umdefinition der Methode *htcondor_create_job_manager* hervorgerufen, die für die Einbindung des HTCondorJobManagers verantwortlich ist. Der Jobmanager wird bei jeder Initialisierung eines HTCondorWorkflows aufgerufen.

```
class HTCondorWorkflow(law.contrib.htcondor.HTCondorWorkflow):

    [...]

    def htcondor_create_job_manager(self):
        return HTCondorJobManager()
```

Dies hat bei der vorherigen Version ausgereicht. Durch zahlreiche Änderungen in der Definition des LAW HTCondorJobManagers hat die Methode *htcondor_create_job_manager* zusätzliche optionale Parameter erhalten. Die Methode wird somit durch

```
def htcondor_create_job_manager(self, **kwargs):
```

definiert. Die ursprüngliche Überschreibung der Methode in `framework.py` hat somit die Methode überladen, statt sie aufzunehmen. Eine entsprechende Anpassung der Methode in der Framework Datei behebt die Kollision.

Eine weitere Änderung betrifft die Klassenmethode *map_status* des HTCondorJobManagers. Diese ist die naive Schnittstelle der Kommunikation zwischen LAW und HTCONDOR. An dieser Stelle wird der Status übersetzt, der von HTCONDOR übergeben wird. Der Nutzer kann den Zustand eines laufenden HTCondorWorkflows durch die Ausgabe im Terminal einsehen.

```
16:13:37: all: 3800, pending: 1954 (+0), running: 1123 (+0), finished:
    723 (+0), retry: 0 (+0), failed: 0 (+0)
```

Das obige Beispiel zeigt die möglichen Zustände an, die ein LAW Workflow Branch annehmen kann. Bevor ein Branch als fehlgeschlagen angezeigt wird, wird dieser resubmittiert und somit im *retry* Bereich angezeigt. Dies wird so lange durchgeführt, bis ein Branch eine vordefinierte Anzahl Resubmittierungen durchlaufen hat. Der aktuelle Zustand wird durch eine vorgegebene Kodierung von HTCONDOR als sogenannte *status_flag* ausgegeben. Diese werden durch die Methode *map_status* ausgelesen und weitergegeben.

```
@classmethod
def map_status(cls, status_flag):
    if status_flag in ("U", "I","S"):
        return cls.PENDING
    elif status_flag in ("R",">","<"):
        return cls.RUNNING
    elif status_flag in ("C"):
        return cls.FINISHED
    elif status_flag in ("H", "E"):
        return cls.FAILED
    else:
        return cls.FAILED
```

Der Ausnahmefall in der *map_status* Methode sorgt dafür, dass jeder nicht definierte Zustand eines Prozesses als fehlgeschlagen interpretiert wird. Bei der Änderung der LAW Version verändert sich auch die Kodierung der zu interpretierenden Zustände eines Prozesses. Dies ist durch den Umstieg von einer string-artigen zu einer integer-artigen Kodierung [21] von HTCONDOR hervorgerufen. Eine entsprechende Anpassung in der Framework Datei behebt somit das Problem, dass durch die neuen Zustände der Kodierung alle Branches unabhängig ihres tatsächlichen Status als fehlgeschlagen markiert werden.

```
def map_status(cls, status_flag):
    if status_flag in ("1","0","U", "I", "S"):
        return cls.PENDING
```

```
    elif status_flag in ("2","R","<",">"):
        return cls.RUNNING
    elif status_flag in ("4","C"):
        return cls.FINISHED
    elif status_flag in ("5","H", "E"):
        return cls.FAILED
    else:
        return cls.FAILED
```

Eine zusätzliche Änderung, die den Arbeitsprozess nicht unterbricht, aber für den Nutzer zu Beginn bei der Einstellung einer neuen Arbeitsumgebung wichtig ist, muss bei der neuen Law Version beachtet werden. Bei dem Submittieren eines Workflows wird ein Verzeichnis erstellt, welches die Arbeitsumgebung für jeden Branch enthält. Darüber hinaus ist in diesem Verzeichnis auch die entsprechende Ausgabe eines jeden Branches hinterlegt, welche beim Debuggen besonders hilfreich ist. Eine genaue Beschreibung dieses Verzeichnisses wird in Abschnitt 5.2 erfolgen. Bei dem Umstieg der Law Version wird die neue Variable *job_file_dir_cleanup* eingeführt, die es ermöglicht, das erstellte Verzeichnis automatisch löschen zu lassen. Das globale Setzen des Standardwertes dieser Variable erfolgt bei Law durch ihre *config.py* Datei. Die Werte der Law Konfigurationsvariablen können in dem *law-analysis* Verzeichnis durch die *law.cfg* Datei angepasst werden. Diese ist ähnlich wie *luigi.cfg* aufgebaut. Im Bereich `[job]` kann die Variable *job_file_dir_cleanup* durch das Setzen des entsprechenden booleschen Wertes manuell verändert werden.

```
[job]
[...]
job_file_dir_cleanup: False
```

Da standardmäßig die Variable auf *true* gesetzt ist, wird empfohlen diese zu Beginn auf *false* zu setzen, damit, wie angesprochen, die Ausgabedateien eines jeden Branches überprüft werden können.

5.2 Benutzerfreundlichkeit

Dieser Abschnitt behandelt die Untersuchung des Analysepakets unabhängig der jeweiligen Law Version. Dabei stehen die Empfehlungen im Vordergrund, welche dem Nutzer eine effiziente Arbeitsweise bei der Erstellung der Interpolationsgitter ermöglichen.
Wie bereits im bisherigen Verlauf erläutert, manipuliert der Nutzer die Analyse hauptsächlich mit der Konfiguration der vorhandenen Luigi Parameter bei der Erstellung eines Interpolationsgitterdatensatzes. Damit die entsprechenden Workflows diese und andere Einstellungen nutzen können, werden die Einstellung beim Submittieren mit übergeben. Dazu wird ein komprimiertes Verzeichnis *analysis.tar.gz* erstellt, welches den aktuellen Status des Analyse-Verzeichnisses enthält. Umgesetzt wird die Erstellung durch die Methode *htcondor_job_config* in der HTCondorWorkflow Klasse des Frameworks. Die Methode erlaubt ebenfalls, individuelle Nutzervariablen der Konfiguration zu übergeben.

```
def htcondor_job_config(self, config, job_num, branches):
    [...]
```

```
    prevdir = os.getcwd()
    os.system('cd $ANALYSIS_PATH')
    if not os.path.isfile('analysis.tar.gz'):
        os.system('tar --exclude=luigi/.git -czvf analysis.tar.gz
    analysis luigi.cfg law.cfg luigi law six')
    os.chdir(prevdir)

    config.input_files.append(law.util.rel_path(__file__, '../analysis.
    tar.gz'))
    return config
```

Diese Einbindung erfordert jedoch, dass das komprimierte Verzeichnis *analysis.tar.gz* bei jeder Änderung manuell gelöscht wird. Der Nutzer muss bei der Wiederholung des Arbeitsablaufes häufig die *luigi.cfg* Datei bearbeiten und somit ist das Löschen des Verzeichnisses eine zu automatisierende Tätigkeit. Die daraus resultierende Änderung übernimmt diese Aufgabe und erstellt das Verzeichnis bei jedem Submittiervorgang neu.

```
def htcondor_job_config(self, config, job_num, branches):
    [...]
    prevdir = os.getcwd()
    os.system('cd $ANALYSIS_PATH')
    if os.path.isfile('analysis.tar.gz'):
        os.remove('analysis.tar.gz')
    os.system('tar --exclude=luigi/.git -czf analysis.tar.gz analysis
    luigi.cfg law.cfg luigi law six')
    os.chdir(prevdir)

    config.input_files.append(law.util.rel_path(__file__, '../analysis.
    tar.gz'))
    return config
```

Eine weitere Änderung der *framework.py* Datei im *analysis* Verzeichnis beinhaltet die standardmäßige Nutzung eines lokalen Schedulers. Luigi stellt den Scheduler so zur Verfügung, dass mehrere Nutzer direkt an dem selben Task arbeiten können und damit der Scheduler zusätzliche Arbeitsschritte auf verschiedene Nutzer aufteilt. Bearbeitet jedoch ein Nutzer einen Task oder Workflow alleine, so ist die Nutzung eines zentralen Schedulers im Allgemeinen nicht notwendig. Damit der Nutzer in dem vorliegenden Analysepaket selbst die Wahl über die Nutzung des lokalen oder zentralen Schedulers hat, wird eine zusätzliche Methode in das lokale Framework eingebaut. Diese ist bereits durch Law vollständig umgesetzt und kann bei dem Aufruf eines Workflows durch die zusätzliche Option *local-scheduler* manuell aufgerufen werden. Die entsprechende Methode *htcondor_use_local_scheduler* ist Teil der HTCondorWorkflow Klasse und wird im personalisierten Framework hinzugefügt.

```
class HTCondorWorkflow(law.contrib.htcondor.HTCondorWorkflow):
  [...]
  def htcondor_use_local_scheduler(self):
      return True
```

Dies hat den Vorteil, dass wenn der zentrale Scheduler nicht gebraucht wird, keine externe Rechenmaschine bereitgestellt werden muss, die vom lokalen System aus erreichbar ist.

Wie bereits unter anderem in der Einleitung des Kapitels erwähnt, besteht der Arbeitsablauf zu einem großen Anteil aus der Wiederholung der einzelnen Workflows. Für eine statistisch signifikante Aussage werden viele Ereignisse benötigte, die wiederum eine große Datenmenge bedeuten. Die entsprechenden Interpolationsgitter werden auf Großspeichersystemen des GridKa hinterlegt und sind damit kein direktes Anliegen des Nutzers. Lokal werden dennoch viele Dateien gespeichert, die den zur Verfügung stehenden lokalen Speicherplatz auslasten. Im Wesentlichen werden die ausgegebenen Dateien lokal in drei Verzeichnissen gespeichert, welche ausgelagert werden können.

Beim Submittieren von Worklflows an HTCondor wird das bereits in Abschnitt 5.1 angesprochene Archiv mit Informationen zu der Submittierung erstellt. Dieses enthält für jeden Branch die benötigten Analysevariablen und die entsprechenden Ausgaben. Die Analysevariablen setzen sich zusammen aus einer branchspezifischen Version des *analysis.tar.gz* Archivs, der Definition der Pfadvariablen und der zusätzlichen Luigi Parameter. Die Ausgabe wird in einer Textdatei festgehalten. Diese beinhaltet neben formalen Informationen die Law Ausgabe, welche für das Debuggen benötigt wird. Bei einer großen Anzahl an Branches für HTCondor Workflows übersteigt ein einzelnes dieser Verzeichnisse schnell eine Speichergröße von 10 GB. Die Pfadvariable dafür wird in der *law.cfg* definiert.

```
[job]
[...]
job_file_dir = $ANALYSIS_PATH/data/jobs
```

Die *ANALYSIS_PATH* Variable wird in der *setup_law.sh* Datei definiert. Da der Pfad zusätzlich für die Ausgaben von *Combine* und *FnloCppread* verwendet wird, welche nicht stark speicherplatzverbrauchend sind, wird für den Pfad zu den Logging Verzeichnissen eine neue Variable *JOB_FILE_DIR* in der *setup_law.sh* Datei verwendet. Dies soll die besonders speicherplatzlastigen Verzeichnisse entkoppeln und die Möglichkeit bieten diese an geeigneten Orten zu hinterlegen.
Zusätzlich werden zwei Pfade in der Datei *luigi.cfg* gesetzt. Diese beinhalten die Variablen zu den Speicherorten für die zusammengefügten Interpolationsgitter und die graphischen Ausgaben. Die *merge_dir* Variable setzt den Speicherort für alle Law Workflows von *CopyTables* bis zu *FnloCppreadFinal*. Jedes der dort vorliegenden Verzeichnisse beinhaltet bei entsprechender, gewünschter statistischer Signifikanz mehrere hunderte Gigabyte, sodass der Gesamtordner für eine Analyse schnell mehr als 1 TB benötigt. Das Verzeichnis für die Abbildungen wird durch die *plots_dir* Variable festgelegt. Da bei jedem Workflow für jeden NNLOjet Kanal und jedes Bin Abbildungen erstellt werden, kann auch dieses Verzeichnis viel Speicherplatz benötigen und sollte falls nötig entsprechenden verlagert werden.

KAPITEL 6

Auswertung der ϕ^*_η Interpolationsgitter

Dieses Kapitel behandelt die Auswertung der erzeugten Interpolationsgitter zu der ϕ^*_η Observable. Dabei werden die graphischen Kontrollauswertungen, welche bereits in Abschnitt 4.4.3 kurz erwähnt wurden, betrachtet und insbesondere dessen Veränderung bei Erhöhung der produzierten Interpolationsdaten. Zunächst soll die Interpolationsqualität der fastNLO Methode betrachtet werden. Dazu werden die Interpolation der Daten direkt mit der ursprünglichen Berechnung von NNLOjet in Relation gesetzt. Daraufhin werden die interpolierten, theoretischen Vorhersagen dazu verwendet, die Wirkungsquerschnitte und die Korrekturen in höheren Ordnungen darzustellen.

6.1 Beschreibung der graphischen Abbildungen

SingleScalecheck dient dazu, die entstandenen Interpolationsgitter durch *FastProd* auf direktem Wege mit den Ergebnissen von NNLOjet zu vergleichen. Die *FastProd* Daten werden durch *CopyTables* und *FnloCppread* in das richtige Format übertragen. Anschließend wird aus jedem NNLOjet Kanal, jedem Rapiditätspaar und jeder Skala das Ergebnis des ersten Datensatzes herausgenommen und zueinander in Relation gesetzt. Dazu wird der Quotient und die Asymmetrie zwischen den Ergebnissen der fastNLO[1] Interpolation und der Berechnung durch NNLOjet gebildet. Die entsprechenden Vorschriften zum Erstellen dieser Vergleiche sind in Abbildung 6.1 zusammen mit den Ergebnissen eines Datensatzes dargestellt. Beide Verhältnisse sollten nicht um mehr als 1 ‰ von der Eins abweichen. Bei Wirkungsquerschnitten nahe der Null kann es von Vorteil sein, die Asymmetrie zu betrachten, da diese durch die Summe im Nenner ein weniger verzerrtes Ergebniss liefert. Die *SingleScalecheck* Ergebnisse können dazu genutzt werden, zu Beginn der Analyse systematische Fehler direkt zu erkennen, indem die Ausgaben der führenden Ordnung der Störungsrechnung betrachtet werden.

Approxtest benötigt viele Einzelergebnisse für die Wirkungsquerschnitte. Dabei werden alle *FastProd* Datensätze zu einem Kanal, einem (y_b, y^*) Paar und einer Skala in Relation

[1] Die Bezeichnung APPLfast in den Abbildungen entstehen aus der Kooperation zwischen fastNLO und APPLgrid

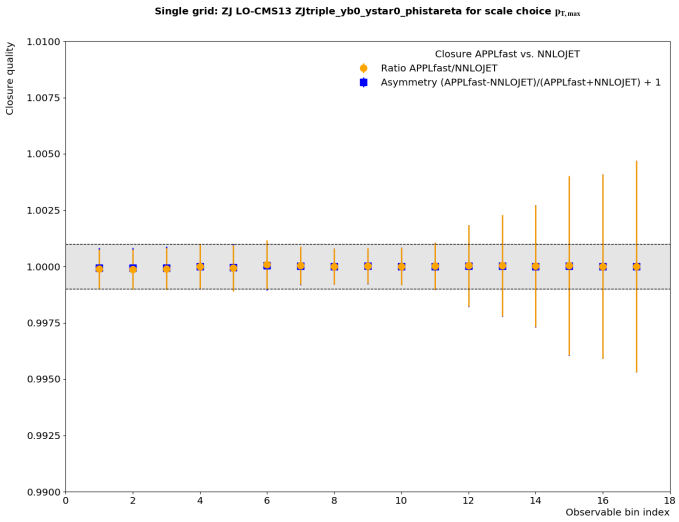


Abbildung 6.1: Diese Beispielabbildung zeigt eine graphische Ausgabe des *SingleScalecheck* Workflows. Dabei werden die ersten Werte eines interpolierten Datensatzes durch fastNLO mit den tatsächlichen Werten von NNLOjet in Relation gesetzt. Es werden der Quotient (gelb) und die Asymmetrie (blau) beider Größen dargestellt. Die dargestellten Fehlerbalken zeigen die statistische Unsicherheit, die durch NNLOjet angegeben wird. Diese ist für den Vergleich jedoch nicht relevant, da identische Ereignisse ausgewertet werden.

gesetzt wie bei *SingleScalecheck*. Statt einzelner Datenpunkt werden die drei statistischen Größen Mittelwert, gewichteter[2] Mittelwert und der Median dargestellt. Hierbei ist besonders der Median von Interesse, da dieser Schätzer unempfindlich gegenüber statistischen Fluktuationen ist.

Absolute stellt die differentiellen Wirkungsquerschnitte, welche aus den Interpolationsgittern resultieren, graphisch dar. Dabei wird erneut für jedes Rapiditätspaar und jede Skala der dazugehörige Wirkungsquerschnitt ermittelt und mit dem Ergebnis von NNLOjet verglichen. Abbildung 6.3 zeigt die berechneten Wirkungsquerschnitte in führender Ordnung der Störungsrechnung. Entsprechend kann dies unter Berücksichtigung der höheren Ordnungen dargestellt werden. Für die NLO und NNLO Prozesse kann der *Absolute* Workflow einzelne Kanäle auftrennen. Im unteren Abschnitt der Abbildung wird der Quotient zwischen Interpolation und NNLOjet Vorhersage dargestellt.

AbsoluteAll stellt die einzelnen Wirkungsquerschnitte für die drei Ordnungen der Störungsrechnung gegenüber. Dabei ist der Einfluss der jeweiligen Ordnung meist deutlich anhand der Verschiebung der Wirkungsquerschnitte zu erkennen. Zusätzlich wird in dem unteren Teil der Abbildung der sogenannte K-Faktor dargestellt. Dieser gibt das Ver-

[2]Die entsprechenden Gewichte werden durch den *Combine* Worklfow berechnet.

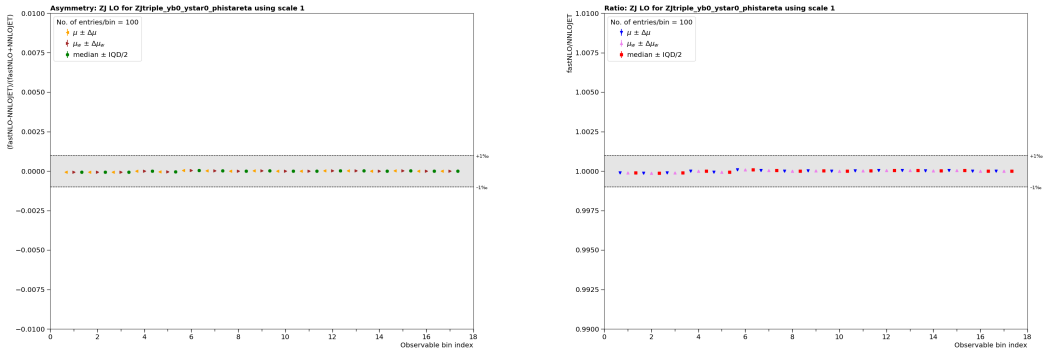


Abbildung 6.2: Der Vergleich durch APPROXTEST zwischen den interpolierten Wirkungsquerschnitten und den durch NNLOJET ist in diesen Abbildungen dargestellt. Für je einen Kanal, ein Rapiditätenpaar und eine Skala eines vollständigen Datensatzes wird der Mittelwert, der gewichtete Mittelwert und der Median für den Quotienten (rechts) und die Asymmetrie (links) aufgezeigt.

hältnis der Beiträge der jeweiligen Ordnung gegenüber der vorherigen an. So stehen in Abbildung 6.4 die jeweiligen K Faktoren für

$$\frac{\sigma_{\mathrm{NLO}}}{\sigma_{\mathrm{LO}}} \quad \text{bzw.} \quad \frac{\sigma_{\mathrm{NNLO}}}{\sigma_{\mathrm{NLO}}}.$$

Hierbei ist der Beitrag bei Beachtung der Prozesse höherer Ordnung größer, wenn dabei mehr Subprozesse berücksichtigt werden. Die Veränderung der Beiträge kann unter der Berücksichtigung der Ereignis-Topologie dazu genutzt werden, die PDFs einzelner Komponenten besser zu bestimmen. So kann beispielsweise der hohe Gluonanteil innerhalb des Protons für erhöhte Korrekturen höherer Ordnung verantwortlich sein.

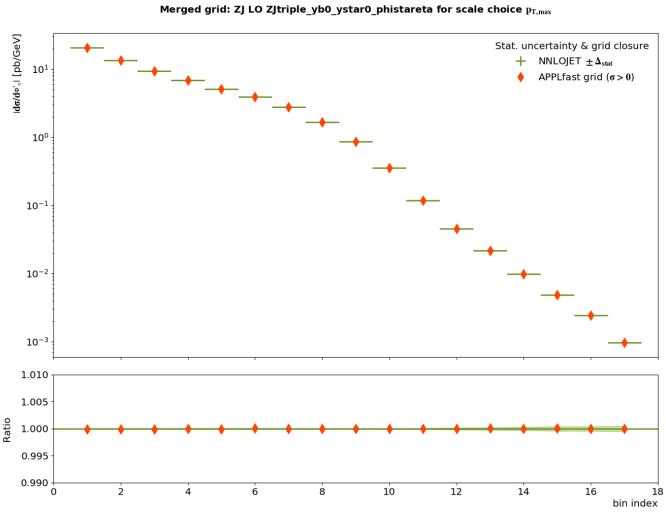


Abbildung 6.3: Die graphische Ausgabe durch *Absolute* zeigt den direkten Vergleich zwischen den berechneten Wirkungsquerschnitten aus den interpolierten Daten (rot) und dem Ergebniss durch NNLOJET (grün) und zusätzlich den entsprechenden Quotienten mit dem angegebenen statistischen Fehler (hellgrünes Band) von NNLOJET.

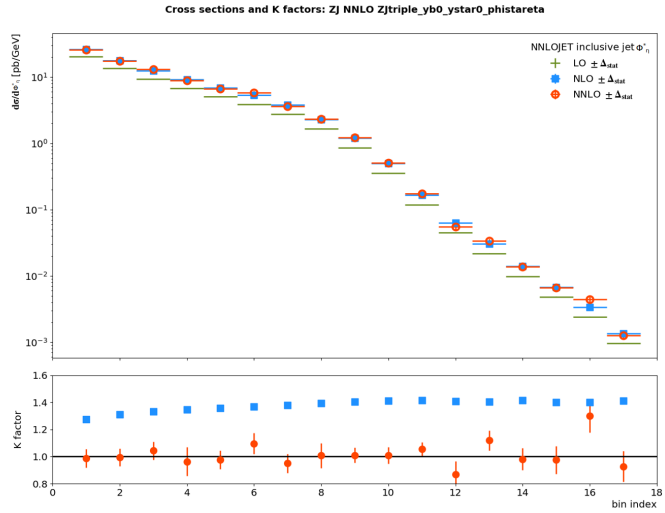


Abbildung 6.4: Die differentiellen Wirkungsquerschnitte unter Berücksichtigung der jeweiligen Ordnung der Störungsrechnung werden zusammengefasst durch *AbsoluteAll* dargestellt. Die Beiträge der NLO (blau) und NNLO (rot) Subprozesse verschieben den Wirkungsquerschnitt der Berechnungen führender Ordnung (grün). Die K Faktoren im unteren Bildteil beschreiben die jeweilige Veränderung des Wirkungsquerschnittes für NLO gegenüber LO (blau) beziehungsweise NNLO gegenüber NLO (rot) Rechnungen.

6.2 Verbesserung der statistischen Signifikanz

Die verwendeten Auswahlkriterien zur Untersuchung des dreifachdifferentiellen Phasenraums, welche in die Runcard einfließen, sind bereits in Abschnitt 4.3.2 dargestellt. Im Wesentlichen müssen bei der Wiederholung des Arbeitsablaufes bei unverändertem Binning die Anzahl der simulierten Ereignisse erhöht werden. Dies wird bei der Erstellung der Interpolationsgitter durch die Anzahl der *FastProd* Ereignisse und Jobs festgelegt. Es werden für jeden NNLOJET Kanal die Anzahl der *fastprod_events* für jeden der *fastprod_jobs* erstellt. Die zur Auswertung der ϕ_η^* Interpolationsgitter verwendeten Einstellungen sind in Tabelle 6.1 festgehalten.

Tabelle 6.1: Auflistung der eingestellten Ereignis- und Jobanzahlen beim *FastProd* Workflow zur Erzeugung der Interpolationsgitter in Abhängigkeit der NNLOJET Kanäle

#		LO	R	V	RRa	RRb	RV	VV
1	Events	$250 \cdot 10^3$	$45 \cdot 10^3$	$22 \cdot 10^3$	800	700	250	800
	Jobs	3	3	3	3	3	3	3
2	Events	$250 \cdot 10^6$	$45 \cdot 10^6$	$22 \cdot 10^6$	$800 \cdot 10^3$	$700 \cdot 10^3$	$250 \cdot 10^3$	$800 \cdot 10^3$
	Jobs	100	100	100	10	10	10	10
3	Events	$250 \cdot 10^6$	$45 \cdot 10^6$	$22 \cdot 10^6$	$800 \cdot 10^3$	$700 \cdot 10^3$	$250 \cdot 10^3$	$800 \cdot 10^3$
	Jobs	100	100	100	100	100	100	100

Im Folgenden werden zur Analyse jeweils drei Bereiche des Rapiditäts $(y_b; y^*)$-Phasenraums betrachtet, da die Ergebnisse eines benachbarten Bereiches sich nicht signifikant unterscheiden. Die zentralen Bins werden durch $(0{,}0 \leq y_b \leq 0{,}5;\ 0{,}0 \leq y^* \leq 0{,}5)$, die Grenzbins durch $(2{,}0 \leq y_b \leq 2{,}5;\ 0{,}0 \leq y^* \leq 0{,}5)$ und das Extremalbin durch $(0{,}0 \leq y_b \leq 0{,}5;\ 2{,}0 \leq y^* \leq 2{,}5)$ beschrieben. Das Grenzbin mit hohem y_b enthält Ereignisse mit unterschiedlichen Impulsbruchteilen der kollidierenden Partonen und dem daraus resultierenden stark geboosteten Massenschwerpunkt, während das Extremalbin mit hohem y^* die Streuung zweier Partonen mit ähnlichen Impulsbruchteilen und großem Streuwinkel im Schwerpunktsystem repräsentiert.

Als erstes soll angemerkt werden, dass die Interpolationsqualität nahezu konstant über alle drei Durchläufe ist. In den Abbildungen, die durch *SingleScalecheck* und *Approxtest* entstehen, kann nur selten eine Abweichung in den Quotienten oder der Asymmetrie beobachtet werden. Abbildung 6.5 zeigt deutlich, dass zwar häufiger Fluktuationen bei der Auswertung aller Mittelwerte beziehungsweise gewichteter Mittelwerte auftreten, jedoch der Median, der einzelne Fluktuationen besser kompensiert, im angestrebten 1 ‰ Intervall liegt. Das häufigere Auftreten von Fluktuationen ist im Bereich der Wirkungsquerschnitte in dritter Ordnung Störungsrechnung (NNLO) dadurch erklärt, dass die Wirkungsquerschnitte häufiger um die Null verteilt sind und dadurch kleine Unterschiede in den Wirkungsquerschnitten zu größeren Abweichungen der Relationen führen.
Der einzige wesentliche Unterschied der *SingleScalecheck* und *Approxtest* Abbildungen bei dem Vergleich zwischen unterschiedlichen Ereigniszahlen besteht in der Anzahl der

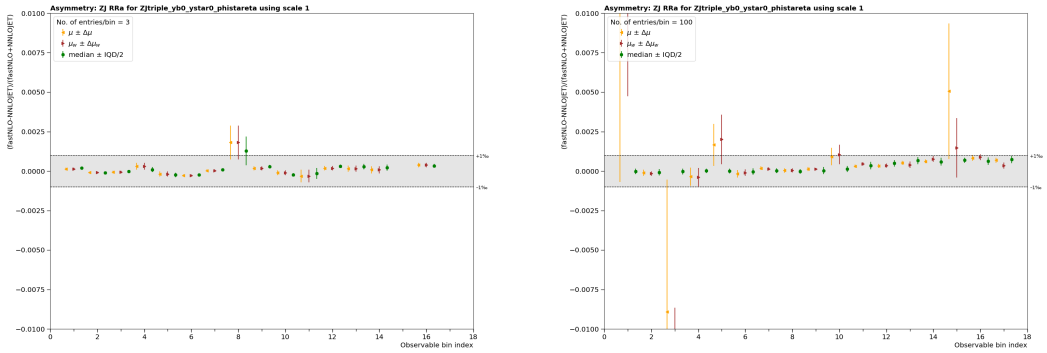


Abbildung 6.5: Dieser Vergleich zeigt die Asymmetrie bei drei Jobs (je 800 Ereignisse)(links) und 100 Jobs (je $800 \cdot 10^3$ Ereignisse)(rechts), die durch *Approxtest* berechnet wird. Die höhere Anzahl an Fluktuationen bei den (gewichteten) Mittelwerten wird durch die Größenordnung der Wirkungsquerschnitte im RRa Kanal hervorgerufen. Der Median ist jedoch in einem Bereich, der für eine weitere Analyse der Datensätze spricht.

beitragenden, simulierten Ereignisse. Im ersten Testdurchlauf werden wenige bis gar keine Ereignisse in den Bins des VV Kanals registriert. Erst im zweiten und dritten Durchlauf können Bins in den äußeren Rapiditätsbereichen beobachtet werden.
Der Unterschied der drei Durchläufe kann besonders gut an den *Absolute* Abbildungen erkannt werden. Insbesondere im Bereich der NNLO Berechnungen sind wesentliche Unterschiede der Wirkungsquerschnitte zu sehen. Abbildung 6.6 zeigt die NNLO Beiträge zum Wirkungsquerschnitt im Extremalbin. Es ist deutlich zu erkennen, dass sich physikalisch sinnvolle Verläufe erst mit steigender Ereigniszahl herauskristallisieren. Negative Wirkungsquerschnitte sind unter der Betrachtung von reinen NLO und NNLO Beiträgen nicht ungewöhnlich, sehr wohl aber sich häufig wiederholende Vorzeichenwechsel innerhalb eines Rapiditätsbereiches. Insgesamt kann ebenfalls beobachtet werden, dass sich der Quotient aus FASTNLO und NNLOJET mit höherer Ereigniszahl deutlich stabilisiert. Besonders gut kann der Einfluss höherer Ereigniszahlen für die Erstellung der Interpolationsgitter in dem finalen Zusammenschluss aller Ergebnisse durch *AbsoluteAll* betrachtet werden. Die starken Fluktuationen und häufigen, unphysikalischen Vorzeichenwechsel der Wirkungsquerschnitte verschwinden bei Betrachtung der Gesamtwirkungsquerschnitte in Abbildung 6.7. Unter der Voraussetzung, dass die Interpolationsqualität für eine Aussage ausreicht, können hier signifikante Verläufe betrachtet und diskutiert werden. Im betrachteten Zentralbin kann beobachtet werden, dass die Korrekturen durch die NLO Berechnungen den wesentlichen Beitrag liefern. In Bereichen mit großem ϕ^*_η fluktuiert jedoch noch die NNLO Korrektur stark. Ein weiterer Durchlauf mit mehr Ereignissen sollte zeigen, ob es sich um tatsächliche Abweichungen handelt oder ob es nur fehlende Präzision aufgrund weniger Ereignisse in dem Bereich ist.
In den äußeren Rapiditätsbereichen kann ebenfalls festgestellt werden, dass mehr Ereignisse für eine vollständige Analyse der NNLO Korrekturen notwendig sind. Der K-Faktor fluktuiert in den Randbereichen noch stark, sollte aber im Vergleich zum Zentralbereich

ebenfalls einen annähernd konstanten Verlauf annehmen. Unter Berücksichtigung dieses Aspektes kann jedoch, wie in Abbildung 6.8 dargestellt, im Bereich mit hohem y_b ein ähnlicher Verlauf wie im Zentralbereich festgestellt werden. Dies kann durch den dominanten Quark-Gluon Subprozess in führender Ordnung der Störungsrechnung begründet werden. Die Streuprozesse mit unterschiedlichen Impulsbruchteilen der Partonen tragen zu den Wirkungsquerschnitten in diesem Bereich wesentlich bei. Im Rapiditätsbereich mit hohem y^* kann jedoch, wie Abbildung 6.9 zeigt, ein wesentlicher Unterschied erkannt werden. Die NLO Korrekturen liefern einen großen Beitrag zu dem totalen Wirkungsquerschnitt. Subprozesse wie Gluon-Gluon-/Quark-Quark-Kollisionen tragen wesentlich zu diesem Rapiditätsbereich bei, treten jedoch erst in der Störungsrechnung in höherer Ordnung auf. Der Vergleich mit den experimentellen Daten [10] stützt diese Beobachtung. Die Wirkungsquerschnitte unter der Berücksichtigung der NLO Korrekturen stellen eine neue Grundlage für Präzisionsanalysen dar.

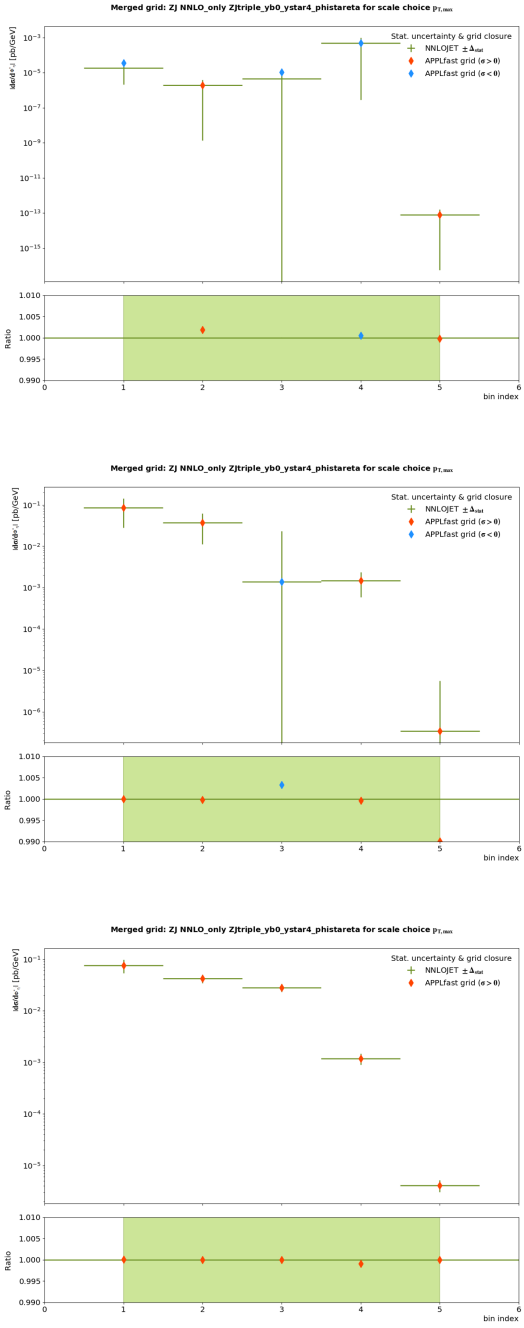


Abbildung 6.6: Anhand dieser drei Abbildungen soll zeigt werden, wie sich der fluktuierende NNLO Beitrag zum Wirkungsquerschnitt mit der steigenden Anzahl an Ereignissen stabilisiert. Im ersten Durchlauf (oben) ist zu sehen, dass die Wirkungsquerschnitte und der Quotient zwischen Interpolation und NNLOJET Berechnung stark fluktuieren. Die Wirkungsquerschnitte führen zwei Vorzeichenwechsel durch. Im zweiten Durchlauf (mitte) verschwinden die Vorzeichenwechsel des differentiellen Wirkungsquerschnittes (blau entspricht negativ; rot positiv) beinahe vollständig und im dritten Durchlauf (unten) sind keine Vorzeichenwechsel zu sehen und der Quotient ist verträglich mit der Eins.

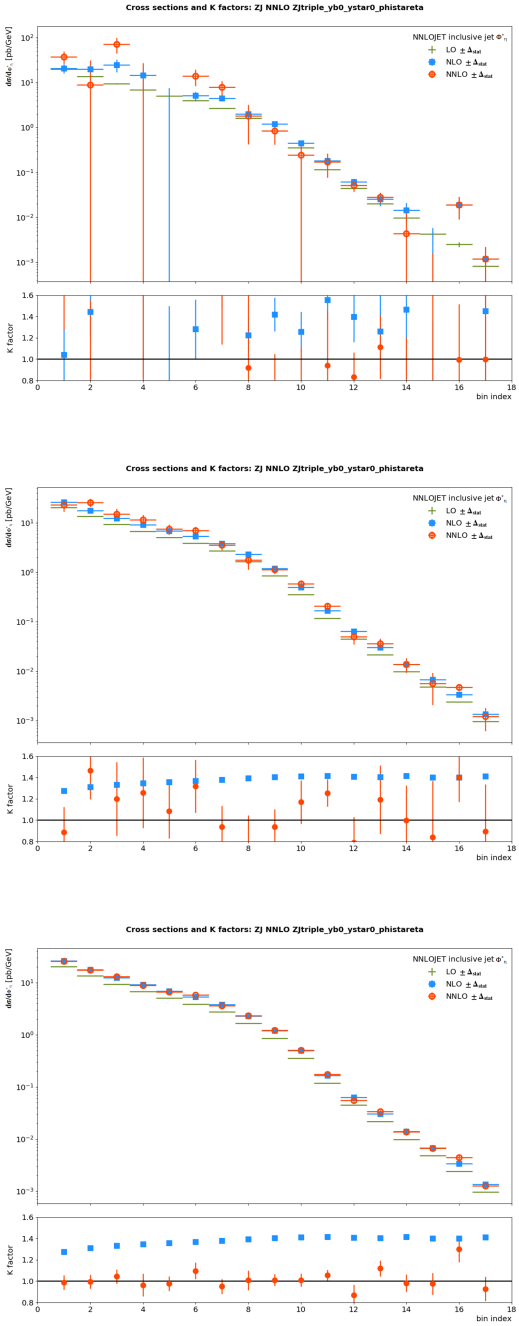


Abbildung 6.7: Der Vergleich dieser drei Abbildungen zeigt die differentiellen Wirkungsquerschnitte unter der Berücksichtigung aller Teilergebnisse. Signifikante Verbesserungen für die physikalische Interpretation können mit mehr Ereignissen während der Interpolation im *FastProd* Schritt erreicht werden. Der erste Durchlauf (oben) ist für eine Analyse der NNLO Korrekturen nicht ausreichend, was schon in den fehlenden Ereignissen bei NNLO Prozessen festgestellt werden kann. Die Analyse der NLO Korrekturen ist bereits im zweiten Durchlauf (mitte) ausreichend gut vorbereitet. Im dritten Durchlauf (unten) ist zu erkennen, dass sich auch die Fluktuationen der NNLO Korrekturen im Zentralbereich stabilisieren. Die äußeren Rapiditätsbereiche erfordern jedoch eine Erhöhung der Ereigniszahl.

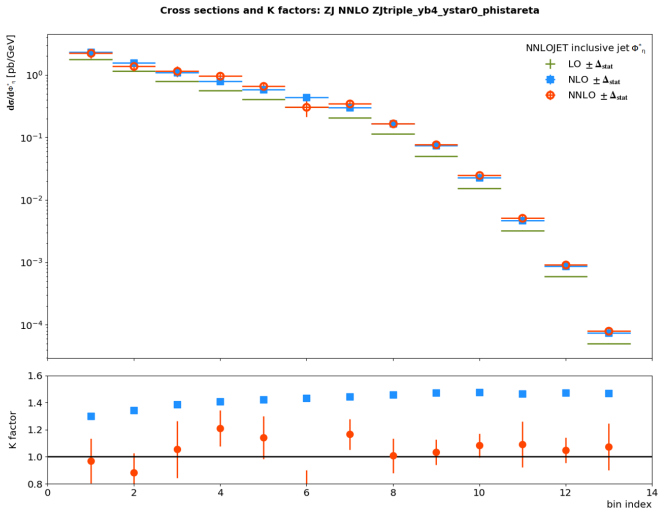


Abbildung 6.8: Bei der Betrachtung der differentiellen Wirkungsquerschnitte im Randbereich mit hohem y_b kann beobachtet werden, dass der entsprechende Verlauf keine großen Fluktuationen aufweist. Damit ist das Grenzbin ähnlich gut wie das Zentralbin beschrieben.

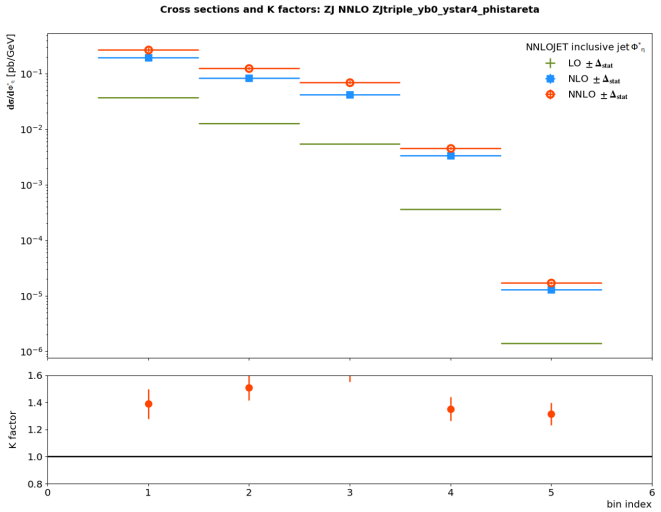


Abbildung 6.9: Die Darstellung der differentiellen Wirkungsquerschnitte unter Berücksichtigung der NLO und NNLO Korrektur zeigen, dass im Rapiditätsbereich mit hohem y^* die NLO Korrekturen wesentlich zum Wirkungsquerschnitt beitragen. Ein K-Faktor in der Größenordnung zehn ist ein starker Indikator dafür, dass die NLO Korrekturen wesentlich für die Präzisionsanalyse ist.

KAPITEL 7

Zusammenfassung

Die Analyse des automatisierten Arbeitsablaufes und der Umstieg auf eine neue Law Version bot die Möglichkeit einer genauen Auseinandersetzung mit Workflow Management Systemen. Darüber hinaus stellten sich dadurch Aspekte bei dem Umgang mit dem Arbeitsablauf heraus, die festgehalten werden können, um zukünftigen Nutzern einen schnelleren Einstieg zu ermöglichen. Der Wechsel zu einer neuen Law Version brachte in erster Linie Verbesserungen in der Law internen Struktur mit sich. Unter anderem werden für den Nutzer mehr Möglichkeiten bereitgestellt, die Produktion zu beeinflussen. Nach einer geeigneten Konfiguration mit wenigen Ereignissen kann ein effizienter Ablauf der Law Workflows mit vielen Ereignissen erfolgen, der idealerweise wenig Eingriff durch den Nutzer erfordert. Dabei ist dem Nutzer selbst überlassen, ob die Produktion mit einem lokalen oder zentralen Luigi Scheduler durchgeführt wird. Außerdem besteht die Möglichkeit einer effizienteren Speicherplatznutzung durch die Kontrolle über die Speicherung der Log-Dateien.

Die Berechnung der Wirkungsquerschnitte bis zur dritten Ordnung der Störungsrechnung für die ϕ_η^* Observable erbrachte eine deutliche Steigerung der systematischen Präzision, erfordert jedoch eine hohe Ereigniszahl für eine hinreichende statistische Präzision. Mit nur wenigen Ereignissen sind willkürliche Fluktuationen deutlich sichtbar. Die statistische Unsicherheit sinkt mit steigender Ereigniszahl, wodurch die Interpolationsgitter für weitere Analysen verwendet werden können. In Rapiditätsbereichen mit hohem y^* und niedrigem y_b ist bei den graphischen Ausgaben der *AbsolutAll* Auswertung zu erkennen, dass die führende Ordnung unzureichende Vorhersagen liefert und erst Berechnungen in nächstführender Ordnung die Messungen grundlegend beschreiben. Für Präzisionsvergleiche sind die hier vorgestellten Vorhersagen in nächst-zu-nächstführender Ordnung daher unerlässlich.

Bei den Berechnungen mit hohen Ereigniszahlen kann gelegentlich der Fall auftreten, dass ungültige Ergebnisse geliefert werden. Diese werden dann beispielsweise als *Not a Number* abgespeichert und können nicht weiter verwendet werden. Dies kann vor dem Zusammenfügen der Interpolationsgitter überprüft werden. Dadurch empfiehlt sich die Erstellung eines zusätzlichen Arbeitsschrittes nach CopyTables, der überprüft, ob ein-

zelne Berechnungen verwendet werden können. Zudem kann dem Nutzer die Aufgabe der Konfiguration seiner Arbeitsumgebung erleichtert werden. Technische Voraussetzungen wie die Verfügbarkeit korrekter PYTHON Bibliotheken können durch LAW Worklflow spezifische virtuelle Umgebungen erfüllt werden. Damit können auch Voraussetzungsdifferenzen einfacher behandelt werden. LAW stellt für solche Zwecke die sogenannte *sandbox* Methode zur Verfügung. Diese ist jedoch noch nicht vollständig ausgearbeitet und soll in Zukunft erweitert werden.

Anhang

A.1 Beispiel für das Setzen von Umgebungsvariablen

```
#!/usr/bin/env bash

action() {

    # law + luigi env variables
    local base="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"
    export LAW_HOME="$base/.law"
    export LAW_CONFIG_FILE="$base/law.cfg"
    export LUIGI_CONFIG_PATH="$base/luigi.cfg"
    export ANALYSIS_PATH="$base"
    export ANALYSIS_DATA_PATH="$ANALYSIS_PATH/data"

    # create directory for law logfiles
    export JOB_FILE_DIR="/portal/ekp$HOSTNAME/home/$USER/law-analysis/
    ZJtriple/data"
    if [ ! -d $JOB_FILE_DIR ]; then
        printf "Creating directory for law logfiles:\n$JOB_FILE_DIR\n"
        mkdir -p "$JOB_FILE_DIR"
    fi

    # luigi + law
    export PATH="$base/law/bin:$base/luigi/bin:$PATH"
    export PYTHONPATH="$base/law:$base/luigi:$base/six:$base:$PYTHONPATH"

    # fastNLO tools + plotting
    export PATH="/home/aheidelbach/git/fastNLO/tools:/home/aheidelbach/
    git/fastNLO/tools/plotting:$PATH"

    # gfal2
    # SLC6
    #    source /cvmfs/grid.cern.ch/emi3ui-latest/etc/profile.d/setup-ui-
    example.sh
    # C7
    source /cvmfs/grid.cern.ch/centos7-ui-v03/etc/profile.d/setup-c7-ui-
    example.sh
```

```
    # NNLOJET + APPLfast
    source /cvmfs/etp.kit.edu/fnlo/fnlosrc_source.sh

    # NNLOJET combine script
    export PATH="/cvmfs/etp.kit.edu/fnlo/src/NNLOJET_rev5088/driver/bin:
    $PATH"

    source "$( law completion )"
}
action
```

A.2 Darstellung der Ausgabe für die *print-status* Option bei Law Workflows

```
  $ law run TaskName --print-status 0,1

print task status with max_depth 0 and target_depth 1
  > check status of TaskName(branch=-1, start_branch=-1, end_branch=-1, {
    "LuigiParameter =..."})
|   - check WLCGFileTarget(path=/pathto/TaskName/htcondor_submission_0ToN
    .json, optional)
|     -> status
|   - check WLCGFileTarget(path=/pathto/TaskName/htcondor_status_0ToN.
    json, optional)
|     -> status
|   - check TargetCollection(len=N, threshold=N)
|     -> existent (M/N)
|        0: existent (WLCGFileTarget(path=/pathto/TaskName/Outputfile_a))
|        1: existent (WLCGFileTarget(path=/pathto/TaskName/Outputfile_b))
|        2: existent (WLCGFileTarget(path=/pathto/TaskName/Outputfile_c))
         [...]

  $ law run TaskName --print-status -1

print task status with max_depth -1 and target_depth 0
> check status of TaskName(branch=-1, start_branch=-1, end_branch=-1, {"
    LuigiParameter =..."})
|   - check WLCGFileTarget(path=/pathto/TaskName/htcondor_submission_0ToN
    .json, optional)
|     -> status
|   - check WLCGFileTarget(path=/pathto/TaskName/htcondor_status_0ToN.
    json, optional)
|     -> status
|   - check TargetCollection(len=N, threshold=N)
|     -> existent (M/N)
|
|   > check status of RequieredTask1({"LuigiParameter =..."})
|   |   - check SiblingFileCollection(len=A, threshold=A, dir=/pathto/
    RequieredTask1)
|   |     -> existent (B/A)
|
|   > check status of Warmup(branch=-1, start_branch=-1, end_branch=-1,{"
    LuigiParameter =..."})
```

```
|   |   - check WLCGFileTarget(path=/pathto/RequieredTask2/
    htcondor_submission_0ToL.json, optional)
|   |     -> status
|   |   - check WLCGFileTarget(path=/pathto/RequieredTask2/
    htcondor_status_0ToL.json, optional)
|   |     -> status
|   |   - check TargetCollection(len=L, threshold=K)
|   |     -> existent (S/L)
```

A.3 Beispiel für die Runcard der ϕ_η^*-Tabellen

```
2.4952d0              ! Width of the Z Boson
80.385d0              ! Mass of the W Boson
2.085d0               ! Width of the W Boson
173.21d0              ! Mass of the Top Quark
1.41d0                ! Width of the Top Quark
4.18d0                ! Mass of the Bottom Quark
0d0                   ! Width of the Bottom Quark
1.275d0               ! Mass of the Charm Quark
0d0                   ! Width of the Charm Quark
1.777d0               ! Mass of the Tau lepton
0d0                   ! Width of the Tau lepton

@UNIT_PHASE@

SELECTORS
      !----- jet veto
      select jets_abs_eta max = 2.4
      select jets_pt min = 20

      !----- for Z production we have two leptons (l^[+-] = l[pm])
      select abs_ylp max = 2.4
      select abs_ylm max = 2.4
      select ptlp min = 25
      select ptlm min = 25

      !----- V = {lp, lm}
      select mll min = 71.1876 max = 111.1876

      !----- for inclusive ptz
      select njets min = 1
END_SELECTORS

HISTOGRAMS

      phi_star > ZJtriple_yb0_ystar0_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
     0.9, 1.0, 1.2, 1.5, 2, 3, 4, 5, 7, 10, 15, 20, 30, 50] grid =
    ZJtriple_yb0_ystar0_phistareta.fast
      HISTOGRAM_SELECTORS
              select yboost_Zj max = 0.5
              select ystar_Zj max = 0.5
      END_HISTOGRAM_SELECTORS
```

```
  phi_star > ZJtriple_yb0_ystar1_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 4, 5, 7, 10, 15, 20, 30, 50] grid =
ZJtriple_yb0_ystar1_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj max = 0.5
          select ystar_Zj min = 0.5 max = 1.0
  END_HISTOGRAM_SELECTORS

  phi_star > ZJtriple_yb0_ystar2_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 4, 5, 7, 10, 15, 20, 30, 50] grid =
ZJtriple_yb0_ystar2_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj max = 0.5
          select ystar_Zj min = 1.0 max = 1.5
  END_HISTOGRAM_SELECTORS

  phi_star > ZJtriple_yb0_ystar3_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 5, 10, 50] grid =
ZJtriple_yb0_ystar3_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj max = 0.5
          select ystar_Zj min = 1.5 max = 2.0
  END_HISTOGRAM_SELECTORS

  phi_star > ZJtriple_yb0_ystar4_phistareta [0.4, 0.6, 0.8, 1.0, 5]
grid = ZJtriple_yb0_ystar4_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj max = 0.5
          select ystar_Zj min = 2.0 max = 2.5
  END_HISTOGRAM_SELECTORS

  phi_star > ZJtriple_yb1_ystar0_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 4, 5, 7, 10, 15, 20, 30, 50] grid =
ZJtriple_yb1_ystar0_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj min = 0.5 max = 1.0
          select ystar_Zj max = 0.5
  END_HISTOGRAM_SELECTORS

  phi_star > ZJtriple_yb1_ystar1_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 4, 5, 7, 10, 15, 20, 30, 50] grid =
ZJtriple_yb1_ystar1_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj min = 0.5 max = 1.0
          select ystar_Zj min = 0.5 max = 1.0
  END_HISTOGRAM_SELECTORS

  phi_star > ZJtriple_yb1_ystar2_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 4, 5, 7, 10, 15, 20, 30, 50] grid =
ZJtriple_yb1_ystar2_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj min = 0.5 max = 1.0
          select ystar_Zj min = 1.0 max = 1.5
  END_HISTOGRAM_SELECTORS
```

```
  phi_star > ZJtriple_yb1_ystar3_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 5, 10, 50] grid =
ZJtriple_yb1_ystar3_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj min = 0.5 max = 1.0
          select ystar_Zj min = 1.5 max = 2.0
  END_HISTOGRAM_SELECTORS

  phi_star > ZJtriple_yb2_ystar0_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 4, 5, 7, 10, 15, 20, 30, 50] grid =
ZJtriple_yb2_ystar0_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj min = 1.0 max = 1.5
          select ystar_Zj max = 0.5
  END_HISTOGRAM_SELECTORS

  phi_star > ZJtriple_yb2_ystar1_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 4, 5, 7, 10, 15, 20, 30, 50] grid =
ZJtriple_yb2_ystar1_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj min = 1.0 max = 1.5
          select ystar_Zj min = 0.5 max = 1.0
  END_HISTOGRAM_SELECTORS

  phi_star > ZJtriple_yb2_ystar2_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 5, 10, 50] grid =
ZJtriple_yb2_ystar2_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj min = 1.0 max = 1.5
          select ystar_Zj min = 1.0 max = 1.5
  END_HISTOGRAM_SELECTORS

  phi_star > ZJtriple_yb3_ystar0_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 4, 5, 7, 10, 15, 20, 30, 50] grid =
ZJtriple_yb3_ystar0_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj min = 1.5 max = 2.0
          select ystar_Zj max = 0.5
  END_HISTOGRAM_SELECTORS

  phi_star > ZJtriple_yb3_ystar1_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 5, 10, 50] grid =
ZJtriple_yb3_ystar1_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj min = 1.5 max = 2.0
          select ystar_Zj min = 0.5 max = 1.0
  END_HISTOGRAM_SELECTORS

  phi_star > ZJtriple_yb4_ystar0_phistareta [0.4, 0.5, 0.6, 0.7, 0.8,
 0.9, 1.0, 1.2, 1.5, 2, 3, 5, 10, 50] grid =
ZJtriple_yb4_ystar0_phistareta.fast
  HISTOGRAM_SELECTORS
          select yboost_Zj min = 2.0 max = 2.5
```

```
              select ystar_Zj max = 0.5
      END_HISTOGRAM_SELECTORS

END_HISTOGRAMS

SCALES
        muf = z_ht mur = etz
        muf = 2.718281828 mur = 2.718281828
        muf = 4.48168907 mur = 4.48168907
        muf = 4.48168907 mur = 2.718281828
        muf = 2.718281828 mur = 4.48168907
        muf = 12.18249396 mur = 2.718281828
        muf = 2.718281828 mur = 12.18249396
END_SCALES

!####### SYNTAX ######################
!##
!## REWEIGHT [{val_factor} * ] [{observable-name} | {val}] [ ** {
    val_power}]
!##
!#####################################

REWEIGHT ht_part**2

CHANNELS
    @CHANNEL@
END_CHANNELS

SETUP
END_SETUP
```

A.4 Beispiel für die Konfiguration der `luigi.cfg` Datei

```
[core]

no_lock = True
# set local scheduler
#local_scheduler = True
default-scheduler-host = condorcentral.etp.kit.edu
default-scheduler-port = 8082

[worker]

keep_alive = False
ping_interval = 20
wait_interval = 20
max_reschedules = 0

[DEFAULT]

# name of your analysis
name = ZJtriple_phistareta
```

```
# NNLOJET process
process = ZJ

# NNLOJET channels (append "a", "b" for RR region flag)
channels = LO R V RRa RRb RV VV

# merged grids (make sure it's compatible with your combine.ini config)
final_tables = {
    "NLO": ["LO", "R", "V"],
    "NLO_only": ["R", "V"],
    "NNLO_only": ["RRa", "RRb", "RV", "VV"],
    "NNLO": ["LO", "R", "V", "RRa", "RRb", "RV", "VV"]
  }

# list of all observables (APPLfast grid names for NNLOJET histograms)
observables = [
    "ZJtriple_yb0_ystar0_phistareta",
    "ZJtriple_yb0_ystar1_phistareta",
    "ZJtriple_yb0_ystar2_phistareta",
    "ZJtriple_yb0_ystar3_phistareta",
    "ZJtriple_yb0_ystar4_phistareta",
    "ZJtriple_yb1_ystar0_phistareta",
    "ZJtriple_yb1_ystar1_phistareta",
    "ZJtriple_yb1_ystar2_phistareta",
    "ZJtriple_yb1_ystar3_phistareta",
    "ZJtriple_yb2_ystar0_phistareta",
    "ZJtriple_yb2_ystar1_phistareta",
    "ZJtriple_yb2_ystar2_phistareta",
    "ZJtriple_yb3_ystar0_phistareta",
    "ZJtriple_yb3_ystar1_phistareta",
    "ZJtriple_yb4_ystar0_phistareta"
  ]

# grid storage protocol and path usable from submitting machine and
    worker nodes of cluster
# job in- and output will be stored in $wlcg_path under subdirectory of
    analysis $name
wlcg_path = srm://cmssrm-kit.gridka.de:8443/srm/managerv2?SFN=/pnfs/
    gridka.de/cms/disk-only/store/user/aheidelb/law-analysis
gsi_path  = gsiftp://cmssrm-kit.gridka.de:2811//pnfs/gridka.de/cms/disk-
    only/store/user/aheidelb/law-analysis


# default htcondor job submission configuration (modifiable for each task
    )
htcondor_accounting_group = cms.jet
htcondor_requirements = (TARGET.ProvidesCPU==true)
htcondor_remote_job = True
htcondor_user_proxy = /tmp/x509up_u12265
# time in seconds
htcondor_walltime = 84000
htcondor_request_cpus = 1
# for all cores in total
htcondor_request_memory = 4096
```

```
htcondor_universe = docker
htcondor_docker_image = mschnepf/slc6-condocker

# create log files in htcondor jobs
transfer_logs = True

# set tolerance for workflow success with failed branches
tolerance = 0

# submit only missing htcondor workflow branches (should always be true)
only_missing = True

# bootstrap file to be sourced at beginning of htcondor jobs
bootstrap_file = bootstrap.sh

# local directory for merging of grids inside $merge_dir/$name
merge_dir = /ceph/aheidelbach/law-analysis/ZJtriple/mergedgrids


# fastNLO cppread options for grid/table evaluation
pdf = CT14nnlo
scalecombs = -6
ascode = LHAPDF
norm = no
scale = scale12

# local directory for plots
plots_dir = plots

#
# START of the ACTION
#

[BaseRuncard]
# copied to grid storage into BaseRuncard

# path to base runcard file
source_path = setup/runcards/ZJ.ZJtriple_phistareta.run

[Steeringfiles]
# copied to grid storage into Steeringfiles

# local directory with all steering files
source_path = steeringfiles

[Warmup]
# produced @ grid storage under Warmup

# override some defaults for this task
# htcondor config
htcondor_request_cpus = 24
bootstrap_file = multicore_bootstrap.sh
# for all cores in total
htcondor_request_memory = 35000
```

```
# time in seconds
#htcondor_walltime = 180000

# NNLOJET event count and integration steps for every channel
# MUST be of the same length as channels!
# MUST have exactly ONE space between numbers
# (test setup with low number of iterations)
starting_seed = 0
warmup_events = 10000 10000 10000 1000 1000 1000 1000
warmup_iterations = 10 10 10 10 10 10 10
# time in seconds
htcondor_walltime = 3600

[FastWarm]
# produced @ grid storage under FastWarm

# override tolerance to 5% failed jobs
tolerance = 0.05

# override some defaults for this task
# htcondor config
htcondor_requirements = ((TARGET.CLOUDSITE=="BWFORCLUSTER")||(TARGET.
    ProvidesEkpResources==true))
# If ETP blades are too slow
#htcondor_requirements = (TARGET.CloudSite=="BWFORCLUSTER")

# NNLOJET event count and number of jobs for each channel
fastwarm_events = 200000000 200000000 200000000 100000000 100000000
    100000000 100000000
fastwarm_jobs = 10 10 10 10 10 10 10

# Test setup
#fastwarm_events = 200000 200000 200000 100000 100000 100000 100000
#fastwarm_jobs = 1 1 1 1 1 1 1
starting_seeds = 1000 2000 3000 4000 5000 6000 7000
# time in seconds
htcondor_walltime = 1800

[MergeFastWarm]
# produced @ grid storage under MergeFastWarm

[FastProd]
# produced @ grid storage under FastProd

# override some defaults for this task
# htcondor config
htcondor_requirements = ((TARGET.CLOUDSITE=="BWFORCLUSTER")||(TARGET.
    ProvidesEkpResources==true))
#htcondor_requirements = (TARGET.CLOUDSITE=="BWFORCLUSTER")
htcondor_request_memory = 8000

# NNLOJET event count and number of jobs for each channel
starting_seeds = 10000 20000 30000 40000 50000 60000 70000
```

```
# <24h
#fastprod_events = 250000000 45000000 22000000 800000 700000 250000
    800000
#fastprod_jobs = 100 100 100 100 100 100 100

# <48h NNLO
fastprod_events = 250000000 45000000 22000000 1600000 1300000 550000
    1800000
fastprod_jobs = 100 100 100 1000 1000 1000 500
htcondor_walltime = 170000

# Test setup
#fastprod_events = 250000 45000 22000 800 700 250 800
#fastprod_jobs = 3 3 3 3 3 3 3
#starting_seeds = 10000 20000 30000 40000 50000 60000 70000
# time in seconds
#htcondor_walltime = 1800

[CopyTables]
# copied to local storage under $merge_dir/$name/CHANNEL, see above

[Combine]
# produced @ local storage under $merge_dir/$name/Combined

# path to combine.ini config
combine_ini = combine.ini

# number of cores for NNLOJET combine script
cores = 20

[MergeFastProd]
# produced @ local storage under $merge_dir/$name/Combined/Final

htcondor_requirements = (TARGET.ProvidesEkpResources==true)

# execute workflow as local workflow instead of htcondor workflow (useful
    for merging small amount of grids, to be removed later)
workflow = local

[MergeFinal]
# produced @ local storage under $merge_dir/$name/Combined/Final

[FnloCppread]
# produced @ local storage under $merge_dir/$name/CHANNEL
ignore_incompleteness = True


[FnloCppreadFinal]
# produced @ local storage under $merge_dir/$name/Combined/Final

[SingleScalecheck]
# produced @ local storage under $plots_dir

[Approxtest]
```

```
# produced @ local storage under $plots_dir
fscl = 7

[Absolute]
# produced @ local storage under $plots_dir

[AbsoluteAll]
# produced @ local storage under $plots_dir
```

ANHANG B

Abbildungsverzeichnis

ANHANG C

Tabellenverzeichnis

ANHANG D

Literatur

[1] Miguel Santos Correa. "Automated Production of Interpolation Grids at NNLO for the Triple-Differential Z+Jet Cross Section Measurement at the LHC". Karlsruhe Institute of Technology, Masterarbeit, 2018. MS. Karlsruhe Institute of Technology, 2018.

[2] Bogdan Povh u. a. "Teilchen und Kerne. Eine Einführung in die physikalischen Konzepte". Sixth. Springer-Verlag GmbH, 2004. ISBN: 3540210652.

[3] Lisa Edelhäuser und Alexander Karl Knochel. "Tutorium Quantenfeldtheorie. was Sie schon immer über QFT wissen wollten, aber bisher nicht zu fragen wagten". ger. LEHRBUCH. Berlin ; Heidelberg: Springer Spektrum, 2016, xii, 539 Seiten. ISBN: 978-3-642-37675-7.
DOI: 10.1007/978-3-642-37676-4.

[4] Michael E. Peskin und Daniel V. Schroeder. "An Introduction to quantum field theory". Reading, USA: Addison-Wesley, 1995. ISBN: 9780201503975, 0201503972.

[5] John C. Collins, Davison E. Soper und George F. Sterman. "Factorization of Hard Processes in QCD". *Adv. Ser. Direct. High Energy Phys.* 5 (1989), S. 1–91.
DOI: 10.1142/9789814503266_0001. arXiv: hep-ph/0409313 [hep-ph].

[6] H1 Collaboration. "Combination of measurements of inclusive deep inelastic $e^{\pm}p$ scattering cross sections and QCD analysis of HERA data". *Eur. Phys. J.* C75.12 (2015), S. 580.
DOI: 10.1140/epjc/s10052-015-3710-4. arXiv: 1506.06042 [hep-ex].

[7] Sayipjamal Dulat u. a. "New parton distribution functions from a global analysis of quantum chromodynamics". *Phys. Rev.* D93.3 (2016), S. 033006.
DOI: 10.1103/PhysRevD.93.033006. arXiv: 1506.07443 [hep-ph].

[8] Particle Data Group Collaboration. "Review of Particle Physics". *Phys. Rev. D* 98 (3 Aug. 2018), S. 030001.
DOI: 10.1103/PhysRevD.98.030001.

[9] Izaak Neutelings. "Simple example of 3D axes with spherical coordinates". Letzter Zugriff: 30.08.2019 12:40 Uhr. Juli 2017. URL: https://wiki.physik.uzh.ch/cms/latex:example_spherical_coordinates.

[10] Thomas Joachim Berger. "Jet energy calibration and inclusive triple differential cross section measurements with Z ($\rightarrow \mu\mu$) + jet events at 13 TeV recorded with the CMS detector". PhD thesis. Karlsruhe Institute of Technology, 2019.

[11] A. Gehrmann-De Ridder u. a. "Z+jet production at NNLO". *PoS* LL2016 (2016), S. 056.
DOI: 10.22323/1.260.0056. arXiv: 1607.01749 [hep-ph].

[12] T. Kluge, K. Rabbertz und M. Wobisch. "FastNLO: Fast pQCD calculations for PDF fits". *Deep inelastic scattering. Proceedings, 14th International Workshop, DIS 2006, Tsukuba, Japan, April 20-24, 2006*. 2006, S. 483–486.
DOI: 10.1142/9789812706706_0110. arXiv: hep-ph/0609285 [hep-ph].

[13] Tancredi Carli u. a. "A posteriori inclusion of parton density functions in NLO QCD final-state calculations at hadron colliders: The APPLGRID Project". *Eur. Phys. J.* C66 (2010), S. 503–524.
DOI: 10.1140/epjc/s10052-010-1255-0. arXiv: 0911.2985 [hep-ph].

[14] Spotify. "Luigi". Letzter Zugriff: 09.09.2019 16:57 Uhr. 2012. URL: https://github.com/spotify/luigi.

[15] M. Erdmann u. a. "Design and Execution of make-like, distributed Analyses based on Spotify's Pipelining Package Luigi". *J. Phys. Conf. Ser.* 898.7 (2017), S. 072047.
DOI: 10.1088/1742-6596/898/7/072047. arXiv: 1706.00955 [physics.data-an].

[16] Karlsruhe Institut für Technologie. "Grid Computing Centre Karlsruhe". Letzter Zugriff: 13.09.2019 18:40 Uhr. URL: http://www.gridka.de/cgi-bin/frame.pl?seite=/welcome.html.

[17] Alvise Dorigo u. a. "XROOTD/TXNetFile: A Highly Scalable Architecture for Data Access in the ROOT Environment". *Proceedings of the 4th WSEAS International Conference on Telecommunications and Informatics*. TELE-INFO'05. Letzter Zugriff: 13.09.2019 18:50 Uhr. Prague, Czech Republic: World Scientific, Engineering Academy und Society (WSEAS), 2005. ISBN: 960-8457-11-4.

[18] William Allcock u. a. "The Globus Striped GridFTP Framework and Server". *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*. SC '05. Letzter Zugriff: 13.09.2019 18:55 Uhr. Washington, DC, USA: IEEE Computer Society, 2005. ISBN: 1-59593-061-2.
DOI: 10.1109/SC.2005.72.

[19] Douglas Thain, Todd Tannenbaum und Miron Livny. "Distributed computing in practice: the Condor experience". *Concurrency and Computation: Practice and Experience* 17.2-4 (2005), S. 323–356.
DOI: 10.1002/cpe.938. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.938.

[20] Klaus Rabbertz et. al. "law-analysis". Letzter Zugriff: 10.09.2019 18:00 Uhr. URL: https://gitlab.etp.kit.edu/qcd-public/law-analysis.

[21] HTCondor. "Magic Numbers". Letzter Zugriff: 15.09.2019 17:00 Uhr. URL: https://htcondor-wiki.cs.wisc.edu/index.cgi/wiki?p=MagicNumbers.

Danksagung

In erster Linie möchte ich bei Herrn Professor Doktor Günter Quast und Herrn Doktor Klaus Rabbertz für die Möglichkeit der Auseinandersetzung mit dem Thema der Thesis bedanken. Herrn Doktor Klaus Rabbertz danke ich insbesondere für das Korrekturlesen der Thesis, der daraus resultierenden, konstruktiven Kritik und der Bemühung mich stets weiterzubilden.
Des Weiteren danke ich dem ETP und insbesondere der QCD und Computing Arbeitsgruppe für ihre Unterstützung und zahlreiche Hilfestellungen. Insbesondere möchte ich Herrn Florian von Cube für das Korrekturlesen und all seine Zeit, die er investiert hat, um mir bei Problemen und Fragen zu helfen, danken.
Ein weiterer Dank geht an die Fachschaft für die generelle Unterstützung im Studium und an Frau Jasmin Häberle, Frau Svenja Britting und Frau Maraike Dortmund für die Korrektur der sprachlichen Fehler der Arbeit.

Erklärung der selbständigen Anfertigung der Bachelorthesis

Hiermit erkläre ich, dass ich die Bachelorthesis mit dem Titel

»*Untersuchung eines automatisierten Arbeitsablaufes zur Erstellung von Interpolationsgittern in perturbativer QCD*«

selbständig und unter ausschließlicher Verwendung der angegebenen Hilfsmittel angefertigt habe.

Alexander Heidelbach
Karlsruhe, den 26. November 2019