Master's Thesis

Heavy Quark Production at the LHC and its Impact on Parton Distribution Functions

Jan Wissmann

September 20, 2023

First Examiner: Dr. Tomáš Ježo
Second Examiner: Prof. Dr. Michael Klasen

Contents

1 Introduction

Parton Distribution Functions (PDFs) are an important ingredient in collider physics. Since only the partonic cross sections are calculable from first principles in Quantum Chromodynamics (QCD) perturbation theory, calculating any hadronic cross section, i.e. involving a collision of two hadrons, requires knowledge about the momentum distribution of the partons inside these hadrons. The PDFs give this information, but as they are not calculable from first principles, it is only possible to determine them by data-driven methods, whose measurements come from e.g. the Large Hadron Collider (LHC) or the Hadron-Electron Ring Accelerator (HERA). As the PDFs are universal, i.e. they are the same for different processes involving the same hadrons, different experiments can be combined to form a global analysis. As the theoretical predictions depend on the PDFs, it is possible to vary them and then compare the predictions to experimental data in global QCD analyses, i.e. PDF fits. Notable examples for proton PDF fits include the so-called CT18 [Hou+21], CJ15 [Acc+16] and NNPDF4.0 [Bal+22] fits.

The discovery by the European Muon Collaboration (EMC) of the so-called EMC effect [Aub+83] provided compelling evidence that free protons could not just be combined to form a nucleus, but that nuclear effects play a role. Through these and subsequent findings, the necessity of nuclear Parton Distribution Functions (nPDFs) became clear, i.e. PDFs that describe the momentum distribution of partons inside a nucleus instead of a proton. Similar global analyses are carried out for nPDFs as for proton PDFs, for example the EPPS21 [Esk+22], nNNPDF2.0 [Kha+20] and nCTEQ15 [Kov+16] fits. A shortage of data makes these less precise than the proton ones. In [Duw+22] it could be shown that inclusive quarkonium and open heavy-flavor meson production data from the LHC, in particular D-meson production data, could be used to constrain the gluon PDF down to unprecedently low x values using a data-driven approach, i.e. using a fitted effective scattering matrix element. This approach was validated using predictions from the General Mass – Variable Flavor Number Scheme (GM-VFNS) [Kni+05a; Kni+05b; Kni+12]. Compared to the effective ansatz, GM-VFNS is a true perturbative QCD (pQCD) approach and could be directly compared to the experimental data. The possibility of integrating this into the `nCTEQ++` framework in the future lets the heavy-quark production process stay relevant.

In this work, we first calculate the heavy quark production cross section at leading order (LO) analytically. This development will eventually lead to a theoretical prediction for heavy quark production suitable for fast convolution in PDF fits. The here obtained result is integrated using Monte Carlo methods and compared to predictions from the `hvq` process implemented in the `POWHEG BOX V2` [FNR07; Ali+10]. Additionally, we present development in the context of the `nCTEQ++` PDF fitting framework: Efforts were made in developing an Apptainer [App] image to ease installation on work stations and computing clusters, a more automated PDF plotting framework named `pdfplotter` was presented and progress in parallelizing the `nCTEQ++` framework was made.

2 Perturbative QCD

QCD describes one of the four fundamental forces, namely the strong force. It deals with the interaction between quarks and gluons, and is a non-Abelian gauge theory (i.e. Yang-Mills theory) based on the gauge group SU(3). Quarks transform in its fundamental representation, whose three components are then associated with the three colors red, green and blue, which gives QCD its name. As a part of the Standard Model, it is one of the most successful theories in physics. A distinctive feature of QCD is the fact that the coupling strength decreases with increasing energy, or decreasing distance. This is called asymptotic freedom, and is the reason why QCD high-energy scattering processes can be calculated within the framework of perturbation theory, while e.g. seemingly simple bound states, like the proton, cannot. Here the non-perturbative PDFs arise in the method of factorization, which combines the high energy partonic cross sections with the low-energy, universal PDFs. In the following we will outline the basics of the theory of pQCD.

2.1 The QCD Lagrangian

As mentioned, the gauge group of QCD is SU(3), the special unitary group of degree 3, a Lie group of 3×3 unitary matrices with determinant 1:

$$\mathrm{SU}(3) = \left\{ U \in \mathrm{GL}(3, \mathbb{C}) \mid U^\dagger U = \mathbb{1},\ \det U = 1 \right\} . \tag{2.1}$$

The quark fields transform in the fundamental representation of SU(3), so every quark spinor field ψ_i has three color components and therefore has a color index $i \in \{1, 2, 3\}$. Gluons, i.e. the gluon vector field A^a_μ and its field strength tensor $F^a_{\mu\nu}$, transform in the adjoint representation of SU(3), which means they have an adjoint color index $a \in \{1, \ldots, 8\}$. We also adopt the convention to choose adjoint color indices starting at a, and fundamental color indices starting at i. The basic Lagrangian (density) of QCD reads

$$\mathcal{L} = \bar{\psi}_{q,i} \left(\mathrm{i} \not{\mathrm{D}}_{ij} - m \delta_{ij} \right) \psi_{q,j} - \frac{1}{4} F^a_{\mu\nu} F^{a\,\mu\nu} , \tag{2.2}$$

where m is the quark mass (generated through Yukawa terms in the Standard Model (SM)) and D is the covariant derivative for QCD, here contracted with the Dirac matrices γ^μ in the Feynman slash notation. A summation over all quark flavors q is assumed. We leave out the gauge-fixing and Faddeev-Popov ghost terms for brevity, they are written out e.g. in [Sch14, p. 509]. The covariant derivative is defined as:

$$\mathrm{D}^\mu_{ij} = \delta_{ij}\, \partial^\mu - \mathrm{i} g_s (t^a)_{ij} A^{a\,\mu} . \tag{2.3}$$

Here $g_s = \sqrt{4\pi\alpha_\mathrm{s}}$ is the strong coupling constant, and t^a are the generators of the SU(3) group in the fundamental representation. They are given by the Gell-Mann matrices λ^a:

$$t^a = \frac{1}{2} \lambda^a , \tag{2.4}$$

where

$$\lambda^1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \lambda^2 = \begin{pmatrix} 0 & -\mathrm{i} & 0 \\ \mathrm{i} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \lambda^3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \lambda^4 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix},$$
$$\lambda^5 = \begin{pmatrix} 0 & 0 & -\mathrm{i} \\ 0 & 0 & 0 \\ \mathrm{i} & 0 & 0 \end{pmatrix}, \quad \lambda^6 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \lambda^7 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -\mathrm{i} \\ 0 & \mathrm{i} & 0 \end{pmatrix}, \quad \lambda^8 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix}. \tag{2.5}$$

We will elaborate on the meaning of these in the next chapter. The gluon field strength tensor is defined as

$$F^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g_s f^{abc} A^b_\mu A^c_\nu . \tag{2.6}$$

Due to the extra term in the covariant derivative for a non-Abelian gauge group, the field strength tensor also contains an extra term, which is quadratic in the gluon field. Thus, if eq. (2.6) is plugged into eq. (2.2), one obtains terms cubic and quartic in the gluon field. This gives rise to the self-interaction of gluons, a feature not present in Quantum Electrodynamics (QED). It is the reason why three- and four-gluon vertices exist.

The SU(3) structure constants f^{abc}, occuring in eq. (2.6), are defined by the commutator of the generators:

$$[t^a, t^b] = \mathrm{i} f^{abc} t^c . \tag{2.7}$$

This shows that the structure constants are antisymmetric in their indices. All the non-zero structure constants for the Gell-Mann matrices as the basis of the Lie algebra are given by (see [Ska13, p. 9])

$$\begin{aligned} f^{123} &= 1 , \\ f^{147} = f^{246} = f^{257} = f^{345} &= \frac{1}{2} , \\ f^{156} = f^{367} &= -\frac{1}{2} , \\ f^{458} = f^{678} &= \frac{\sqrt{3}}{2} . \end{aligned} \tag{2.8}$$

They provide the explicit matrix elements of the generators of SU(3) in the adjoint representation.

The normalization of the generators is chosen such that the following is true for the structure constants: (see [Sch14, p. 485])

$$f^{acd} f^{bcd} = N \delta^{ab} \tag{2.9}$$

for SU(N), where $N = 3$ for QCD. This implies the normalization of the generators:

$$\mathrm{Tr}(t^a t^b) = \frac{1}{2} \delta^{ab} . \tag{2.10}$$

The number in front of δ^{ab} in eqs. (2.9) and (2.10) is called the index of the representation

and is thus given for the fundamental (F) and adjoint (A) representation by

$$\begin{aligned} T_{\mathrm{F}} &= \frac{1}{2}\,, \\ T_{\mathrm{A}} &= N = 3\,. \end{aligned} \tag{2.11}$$

Another invariant often showing up in calculations is called the Casimir of a representation:

$$\begin{aligned} f^{acd}f^{bcd} &= C_{\mathrm{A}}\delta^{ab}\,, \\ (t^a t^a)_{ij} &= C_{\mathrm{F}}\delta_{ij}\,. \end{aligned} \tag{2.12}$$

For the adjoint representation, the index and the Casimir are the same. The Casimirs are given by

$$\begin{aligned} C_{\mathrm{A}} &= N = 3\,, \\ C_{\mathrm{F}} &= \frac{N^2-1}{2N} = \frac{4}{3}\,. \end{aligned} \tag{2.13}$$

2.2 Color Charge

Since the quark spinor fields transform in the fundamental representation of SU(3), they have a color index $i \in \{1,2,3\}$, i.e. we can write them as 3-component vectors (for all flavors) (see [Gri87, p. 355]):

$$\psi = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{pmatrix} \quad \text{and} \quad \bar{\psi} = \begin{pmatrix} \bar{\psi}_1 & \bar{\psi}_2 & \bar{\psi}_3 \end{pmatrix}. \tag{2.14}$$

Each of the components then holds a 4-component Dirac spinor. The gluon fields are 8-dimensional vectors in the color components since they transform in the adjoint representation of SU(3). The possible states of a gluon stem from combining 3 colors with 3 anticolors:

$$3 \otimes \bar{3} = 8 \oplus 1\,. \tag{2.15}$$

Remember that we write this down in terms of the dimension of the representation, different from SU(2), where sometimes the spin is used directly to label a representation. The bar stands here for the antifundamental representation, the complex conjugate of the fundamental representation. These are two different things, since the fundamental representation of SU(N) is a complex representation if $N > 2$. From the 9 possible states, the octet of eq. (2.15) seems to be realized in nature, since the singlet would be a free gluon, violating color confinement. It would also couple to other color singlets, so the strong force would be long-range, which is not observed. The octet states are given by

(see [Gri87, p. 280])

$$\begin{aligned}
&\frac{1}{\sqrt{2}}(|r\bar{b}\rangle + |b\bar{r}\rangle)\,, \quad -\frac{\mathrm{i}}{\sqrt{2}}(|r\bar{b}\rangle - |b\bar{r}\rangle)\,, \quad \frac{1}{\sqrt{2}}(|r\bar{r}\rangle - |b\bar{b}\rangle)\,, \quad \frac{1}{\sqrt{2}}(|r\bar{g}\rangle + |g\bar{r}\rangle)\,,\\
&-\frac{\mathrm{i}}{\sqrt{2}}(|r\bar{g}\rangle - |g\bar{r}\rangle)\,, \quad \frac{1}{\sqrt{2}}(|b\bar{g}\rangle + |g\bar{b}\rangle)\,, \quad -\frac{\mathrm{i}}{\sqrt{2}}(|b\bar{g}\rangle - |g\bar{b}\rangle)\,, \quad \frac{1}{\sqrt{6}}(|r\bar{r}\rangle + |b\bar{b}\rangle - 2\,|g\bar{g}\rangle)\,.
\end{aligned} \tag{2.16}$$

By writing the color basis as follows

$$|r\rangle = |\bar{r}\rangle = \begin{pmatrix}1\\0\\0\end{pmatrix}, \qquad |b\rangle = |\bar{b}\rangle = \begin{pmatrix}0\\1\\0\end{pmatrix}, \qquad |g\rangle = |\bar{g}\rangle = \begin{pmatrix}0\\0\\1\end{pmatrix} \tag{2.17}$$

and performing the tensor products for the product states in eq. (2.16), we see that the octet states in eq. (2.16) are just the Gell-Mann matrices in eq. (2.5) up to normalization. The color singlet state is given by

$$\frac{1}{\sqrt{3}}(|r\bar{r}\rangle + |b\bar{b}\rangle + |g\bar{g}\rangle) \tag{2.18}$$

and corresponds to the identity matrix. As mentioned, this state is not observed in nature. We see that the gluons are not just color-anticolor states (such as e.g. $r\bar{g}$), by which we could have expected that there would be 9 gluons. Instead, they are realized as an octet, therefore there are only 8.

2.3 The Strong Coupling Constant

Dimensional regularization introduces the renormalization scale μ_r. The dependence of the coupling constant (called the running coupling) on this scale is given by the Renormalization Group Equation (RGE):

$$\frac{\mathrm{d}\ln\alpha_\mathrm{s}}{\mathrm{d}\mu_\mathrm{r}^2} = \beta(\alpha_\mathrm{s})\,, \tag{2.19}$$

where the QCD β-function is given by

$$\beta(\alpha_\mathrm{s}) = -\alpha_\mathrm{s}^2(b_0 + b_1\alpha_\mathrm{s} + b_2\alpha_\mathrm{s}^2 + \ldots) \tag{2.20}$$

with the 1-loop and 2-loop coefficients [Par+22, sec. 9, p. 2]

$$\begin{aligned}
b_0 &= \frac{11C_\mathrm{A} - 4T_\mathrm{F}n_\mathrm{f}}{12\pi} \overset{N=3}{=} \frac{33-2n_\mathrm{f}}{12\pi}\,,\\
b_1 &= \frac{17C_\mathrm{A}^2 - n_\mathrm{f}T_\mathrm{F}(10C_\mathrm{A} + 6C_\mathrm{F})}{24\pi^2} \overset{N=3}{=} \frac{153-19n_\mathrm{f}}{24\pi^2}\,,\\
&\vdots
\end{aligned} \tag{2.21}$$

Here, n_f is the number of light quark flavors, i.e. flavors with a mass less than μ_r. More coefficients in the modified Minimal Subtraction ($\overline{\mathrm{MS}}$) scheme can be found in [Col11,

p. 59]. The β-function coefficients are known up to b_5 (4 loops) [Par+22, sec. 9, p. 2].

The negative sign in eq. (2.20) shows the asymptotic freedom of QCD, i.e. that the coupling constant decreases with increasing energy, so that QCD is a theory where quarks and gluons are almost (or asymptotically) free at these energies, in contrast to e.g. QED, where the β-function is positive. At low energies, or large distances, the coupling constant increases and the quarks and gluons enter the regime of confinement, which means that they are bound in hadrons and cannot be encountered as free particles. Qualitatively, this effect can be explained by the anti-screening behavior of the gluons: Every color charge is understood to be surrounded by a cloud of virtual particles. Quark-antiquark pairs screen the color charge like photons do in QED, which means they diminish the field of the charge at large distances. But since gluons carry color charge themselves, they augment the field of the charge at large distances, an effect that is called antiscreening. These effects act in opposite ways, so it makes sense that it depends on the number of quark flavors, i.e. n_f in eq. (2.21). For standard QCD with $N = 3$ colors and 6 flavors, the antiscreening effect dominates. This is the case for $n_f \leq 16$.

The solution to eq. (2.19) provides the connection of the strong coupling constant α_s at an arbitrary scale to a reference scale, which is usually chosen to be the Z boson mass M_Z. It reads [Ska13, p. 11]

$$\alpha_s(\mu_r^2) = \frac{\alpha_s(M_Z^2)}{1 + b_0\,\alpha_s(M_Z^2)\ln\left(\frac{\mu_r^2}{M_Z^2}\right) + O(\alpha_s^2)}\,. \tag{2.22}$$

The current world average, determined by the Particle Data Group (PDG) from hadronic τ and heavy quarkonia decays, PDF fits, $e^+\,e^-$ annihilations, etc. is [Par+22, sec. 9, p. 38]

$$\alpha_s(M_Z^2) = 0.1179 \pm 0.0009\,. \tag{2.23}$$

The different measurements contributing to this average can be seen in fig. 1.

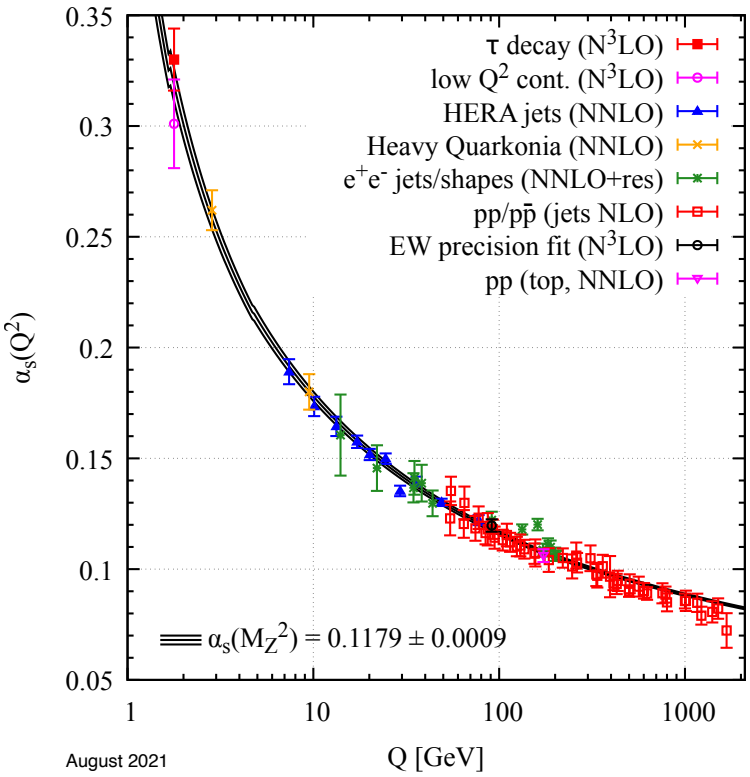


Figure 1: Measurements of α_s contributing to the world average by the PDG, as a function of the renormalization scale, here named Q. Taken from [Par+22, sec. 9, p. 35].

We can also express the result in terms of the constant Λ_{QCD}:

$$\alpha_s(\mu_r^2) = \frac{1}{b_0 \ln\left(\frac{\mu_r^2}{\Lambda_{\text{QCD}}^2}\right)}. \tag{2.24}$$

Λ_{QCD} is of the order of magnitude 200 MeV and describes the energy scale at which the coupling constant diverges, called the Landau pole. As it is a non-perturbative quantity given by perturbative methods, it is not too well defined. But it can give an estimate of the energy scale at which perturbation theory is valid, namely $\mu_r \gg \Lambda_{\text{QCD}}$, which corresponds to $\alpha_s(\mu_r^2) \ll 1$.

2.4 The QCD Feynman Rules

The whole Lagrangian is, now with gauge-fixing and ghost terms, given by (see [Sch14, p. 509])

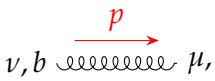
$$\mathcal{L} = \bar{\psi}_{q,i}\left(\mathrm{i}\not{D}_{ij} - m\delta_{ij}\right)\psi_{q,j} - \frac{1}{4}(F^a_{\mu\nu})^2 - \frac{1}{2\xi}(\partial^\mu A^a_\mu)^2 + (\partial^\mu \bar{c}^a)(D^{ac}_\mu c^c) \tag{2.25}$$

The gauge fixing parameter ξ comes from the R_ξ- (covariant) gauge, which gives rise to the ghost and antighost fields c and $\bar{c}$ in Yang-Mills theories. $D_{\mu\, ij}$ is the covariant derivative acting on a field in the fundamental representation, as previously defined in eq. (2.3), and D^{ab}_μ is the covariant derivative acting on a field in the adjoint representation, e.g. the ghost

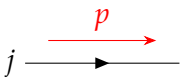
fields:

$$\mathrm{D}^{ac\,\mu} = \delta^{ac}\,\partial^{\mu} + g_s f^{abc}\,A^{b\,\mu}\,. \tag{2.26}$$

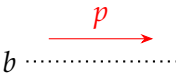
We can now state the Feynman rules for QCD in the R_ξ gauge. We start with the gluon, quark, and ghost propagator:



$$= \mathrm{i}\,\frac{-g^{\mu\nu} + (1-\xi)\frac{p^\mu p^\nu}{p^2}}{p^2 + \mathrm{i}\varepsilon}\,\delta^{ab}\,, \tag{2.27}$$

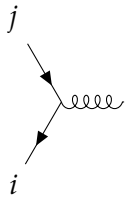


$$= \mathrm{i}\,\frac{\not{p} + m}{p^2 - m^2 + \mathrm{i}\varepsilon}\,\delta_{ij}\,, \tag{2.28}$$

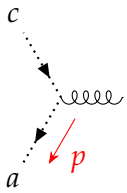


$$= \mathrm{i}\,\frac{1}{p^2 + \mathrm{i}\varepsilon}\,\delta^{ab}\,. \tag{2.29}$$

The quark-gluon and ghost-gluon vertices are

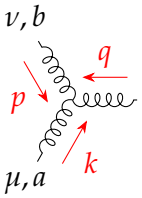


$$= \mathrm{i}g_s\gamma^\mu t^a_{ij}\,, \tag{2.30}$$

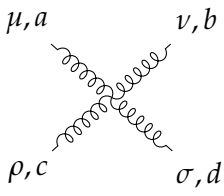


$$= -g_s f^{abc} p^\mu\,. \tag{2.31}$$

And finally, the 3-gluon and 4-gluon self-coupling vertices are



$$\begin{aligned} &= \mathrm{i}g_s f^{abc}(g^{\mu\nu}(k-p)^\rho \\ &\quad + g^{\nu\rho}(p-q)^\mu + g^{\rho\mu}(q-k)^\nu)\,, \end{aligned} \tag{2.32}$$



$$\begin{aligned} &= -\mathrm{i}g_s(f^{abe}f^{cde}(g^{\mu\rho}g^{\nu\sigma} - g^{\mu\sigma}g^{\nu\rho}) \\ &\quad + f^{ace}f^{bde}(g^{\mu\nu}g^{\rho\sigma} - g^{\mu\sigma}g^{\rho\nu}) \\ &\quad + f^{ade}f^{bce}(g^{\mu\sigma}g^{\nu\rho} - g^{\mu\rho}g^{\nu\sigma}))\,. \end{aligned} \tag{2.33}$$

3 Statistics and the Monte Carlo Method

3.1 Introduction to Probability Theory

We cover a brief introduction to form the basis for the following sections. This section is based on [Par+22, sec. 39]. More details are found in [Cow98].

Probability measures the degree of randomness of an event. We call a set S the sample space, which is the set of all outcomes, whose subsets $A, B, \ldots$ can be assigned a probability via a real-valued function P. Its definition is given by the three Kolmogorov axioms, simplified here and in [Par+22, p. 1] by leaving out the aspects of measure theory:

$$\begin{aligned} &1.\ \text{For every subset } A \subseteq S,\ P(A) \geq 0, \\ &2.\ \text{for disjoint subsets } A \cap B = \emptyset,\ P(A \cup B) = P(A) + P(B), \\ &3.\ P(S) = 1. \end{aligned} \tag{3.1}$$

A lot of properties follow from these axioms [Cow98, p. 2]:

$$\begin{aligned} P(A^c) = 1 - P(A)\,, \quad P(A \cup A^c) = 1\,, \quad 0 \leq P(A) \leq 1\,, \quad P(\emptyset) = 0\,, \\ A \subseteq B \Rightarrow P(A) \leq P(B)\,, \quad P(A \cup B) = P(A) + P(B) - P(A \cap B)\,. \end{aligned} \tag{3.2}$$

Here, A^c is the complement of A, i.e. $A^c = S \backslash A$. The conditional probability $P(A \mid B)$ ("P of A given B") is defined as

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}\,, \tag{3.3}$$

which is itself a probability function again, and gives the probability of A given that B has already occurred. The definition can be understood by considering the probability of the whole set B, determining the probability of all the events of A that are also a part of B (the subset $A \cap B$), and taking the ratio.

From $P(A \cap B) = P(B \cap A)$ we get Bayes' theorem:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}\,. \tag{3.4}$$

3.2 Parameter Estimation

Monte Carlo methods and fitting models to data are ways of estimating the parameters of a probability density function. We first introduce some general concepts about parameter estimation. This section is based on [Cow98, ch. 5].

We consider a random variable x with a probability density function $f(x)$. A *sample* of size n is a vector

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}\,, \tag{3.5}$$

consisting of n measurements x_i of the random variable x. We assume that the measure-

ments are all independent and identically distributed (i.i.d.), such that the total probability density function of the sample is given by the product of the individual probability density functions:

$$f(\vec{x}) = \prod_{i=1}^{n} f(x_i)\,. \tag{3.6}$$

We now want to infer $f(x)$ from the sample $\vec{x}$, e.g. by fitting a model $f(x;\vec{\theta})$ to the data, where $\vec{\theta}$ are unknown parameters that our model depends on. For this, we introduce the concept of an *estimator* $\hat{\theta}$, which is a function of $\vec{x}$ and estimates some property of a probability density function (e.g. its mean or variance). The true value θ of the estimator $\hat{\theta}$ is written without a hat. The estimator makes an *estimate*, which is the value of the estimator for a given sample $\vec{x}$.

An estimator is a function of the measured values $\vec{x}$, and is, therefore, itself a random variable, with e.g. its own expectation value and variance. The expectation value of $\hat{\theta}$ is given by

$$\mathrm{E}[\hat{\theta}] = \int \hat{\theta}(\vec{x}) f(\vec{x})\, \mathrm{d}^n x\,, \tag{3.7}$$

which can be interpreted as doing the experiment an infinite number of times, where each time a finite number of measurements is carried out, such that the whole sample space is probed. If $\mathrm{E}[\hat{\theta}] = \theta$, $\hat{\theta}$, i.e. the expectation value of $\hat{\theta}$ actually returns the true value θ, is called *unbiased*, otherwise it is called *biased*.

The *sample mean* is defined by

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i\,. \tag{3.8}$$

This is now an estimator for the expectation value of x, which we call $\mu = \mathrm{E}[x]$. The latter can be understood as the true mean of the random variable x, and the former tries to reconstruct this value from a sample. Calculating the mean and the variance of the sample mean $\bar{x}$ gives [Cow98, p. 67]

$$\mathrm{E}[\bar{x}] = \mu\,, \tag{3.9}$$

$$\mathrm{V}[\bar{x}] = \frac{\sigma^2}{n}\,, \tag{3.10}$$

where μ is the true mean and σ^2 is the true variance of the random variable x. From eq. (3.9), we see that $\bar{x}$ is an unbiased estimator for the true mean μ.

For the sample variance, we have two possibilities, depending if we know the true mean μ or have just the estimator $\hat{x}$:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{n}{n-1}(\overline{x^2} - \bar{x}^2)\,, \tag{3.11}$$

$$S^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 = \overline{x^2} - \mu^2\,. \tag{3.12}$$

With these definitions, one can show

$$\mathrm{E}[s^2] = \mathrm{E}[S^2] = \sigma^2\,, \tag{3.13}$$

so that s^2 and S^2 are unbiased estimators. Obtaining unbiased estimators for s^2 and S^2 is the reason for either choosing $\frac{1}{n-1}$ or $\frac{1}{n}$.

3.3 Method of Maximum Likelihood

The method of maximum likelihood (ML) is a way of parameter estimation, i.e. fitting a model to data. This section is based on [Cow98, ch. 6]. We continue the situation of section 3.2, where we have a random variable x and a model for its probability density function $f(x;\vec{\theta})$. $\vec{\theta} = (\theta_1, ..., \theta_m)$ are unknown parameters which our model depends on.

The method of ML now tries to estimate $\vec{\theta}$ from a sample $\vec{x}$ of size n. For this, we define the likelihood function L:

$$L(\vec{\theta}) = \prod_{i=1}^{n} f(x_i;\theta)\,. \tag{3.14}$$

Using this, we can now obtain estimators $\hat{\vec{\theta}} = (\hat{\theta}_1, ..., \hat{\theta}_m)$ for the true parameters $\vec{\theta}$. In contrast to e.g. the sample mean $\bar{x}$, which had an explicit formula given in eq. (3.8), we define the ML estimators $\hat{\vec{\theta}}$ implicitly by maximizing the likelihood function:

$$\left.\frac{\partial L}{\partial \theta_i}\right|_{\vec{\theta}=\hat{\vec{\theta}}} = 0\,, \tag{3.15}$$

where $i \in \{1, ..., m\}$. Thus, the estimates $\hat{\vec{\theta}}$ are where the likelihood function is maximal. We do this since we want the model with the highest probability to be the one that was observed. Since we want this to happen at all of the measured points x_i, we can instead maximize the product of the individual probability density functions evaluated at each point x_i. This exactly describes the likelihood function L of eq. (3.14).

In practice, it is often more convenient to work with the logarithm of the likelihood function, since the product in eq. (3.14) turns into a sum, and exponentials in the probability density functions turn into their arguments.

3.4 χ^2 Least Squares Fitting

Least squares (LS) fitting is another way of parameter estimation, which is directly connected with the method of ML for Gaussian distributed data. The data is denoted by y_i, $i \in \{1, ..., N\}$, which is measured at the points x_i. The distribution of each datum y_i is given by the mean μ_i and the variance σ_i^2:

$$f(y_i;\mu_i,\sigma_i^2) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y_i-\mu_i)^2}{2\sigma_i^2}\right). \tag{3.16}$$

The σ_i are known from the measurement uncertainties, but the μ_i are unknown. Therefore, we introduce a model parametrized by the unknown parameters $\vec{\theta} = (\theta_1, ..., \theta_m)$, so that we write $\mu_i = \mu(x_i;\vec{\theta})$. The log-likelihood function is then given by taking the logarithm of eq. (3.14) together with eq. (3.16) and ignoring additive and multiplicative terms, which do not influence the position of the maximum/minimum of the log-likelihood func-

tion. This is called χ^2:

$$\chi^2(\vec{\theta}) = \sum_{i=1}^{N} \frac{(y_i - \mu(x_i;\vec{\theta}))^2}{\sigma_i^2}\,. \tag{3.17}$$

If the total probability density function is a more general, N-dimensional Gaussian where the parameters are correlated by the covariance matrix V,

$$f(\vec{y};\vec{\mu},V) = \frac{1}{\sqrt{(2\pi)^n \det V}} \exp\Big(-\frac{1}{2}(\vec{y}-\vec{\mu})^{\mathrm{T}} V^{-1} (\vec{y}-\vec{\mu})\Big)\,, \tag{3.18}$$

then the χ^2-function looks like

$$\chi^2(\vec{\theta}) = \sum_{i,j=1}^{N} (y_i - \mu(x_i;\vec{\theta}))\, V_{ij}^{-1}\, (y_j - \mu(x_j;\vec{\theta}))\,. \tag{3.19}$$

To find the LS estimators $\hat{\vec{\theta}}$, the χ^2 function is minimized, which gives the method its name.

It should be noted that the method of LS is defined by eqs. (3.17) and (3.19) also for non-Gaussian distributed data, but only in the Gaussian case is it equivalent to the ML case.

The concrete method used in the nCTEQ15 fits can be found in section 5.

3.5 Monte Carlo Methods

First, we give an overview of the general method. Then, we show how it can be applied to solving integrals and how to reduce their variance. The descriptions are based on [Cow98, ch. 3] and [Wei00, sec. 3].

3.5.1 Introduction

In general, Monte Carlo methods use sequences of random numbers to calculate a desired quantity. As a first step, we need to generate the uniformly distributed random (or in practice, pseudorandom) numbers $r_i \in [0,1]$ according to their probability density function

$$g(r) = \begin{cases} 1 & \text{for } 0 \le r \le 1 \\ 0 & \text{otherwise} \end{cases}\,. \tag{3.20}$$

The field of pseudorandom number generation is vast, with an example being the Mersenne Twister algorithm [MN98]. The MT19937 variant is e.g. available in the C++ standard library as `std::mt19937` and gets its name from having a period length of the Mersenne prime $2^{19937} - 1$. The obtained sequence $r_1, \ldots, r_n$ is then transformed into a sequence $x_1, \ldots, x_n$, distributed according to a probability density function $f(x)$. We call this transformation $x(r)$. If e.g. $f(x)$ is simply a uniform distribution with a different range $x_{\min} \neq 0$ and $x_{\max} \neq 1$, we can just use

$$x_{\text{uniform}}(r) = x_{\min} + r\,(x_{\max} - x_{\min})\,. \tag{3.21}$$

Transformation Method In case $f(x)$ is more complicated than a uniform distribution, but not too much, $x(r)$ may still be obtained analytically. For this, the transformation method is used, which is based on the cumulative distribution function $F(x)$ of $f(x)$. The derivation starts with equating the probabilites for r and x in corresponding intervals:

$$\begin{aligned} P(r \in [r, r+\mathrm{d}r]) &= P(x \in [x, x+\mathrm{d}x]) \\ \Rightarrow \quad g(r)\,\mathrm{d}r &= f(x)\,\mathrm{d}x\,, \end{aligned} \tag{3.22}$$

Next, we can integrate both sides, which yields the definitions of the respective cumulative distribution functions:

$$\begin{aligned} \int_{-\infty}^{r} g(r')\,\mathrm{d}r' &= \int_{-\infty}^{x(r)} f(x')\,\mathrm{d}x' \\ \Rightarrow \quad G(r) &= F(x(r))\,. \end{aligned} \tag{3.23}$$

As the lower integration boundaries are arbitrary, we see that the method is not unique. If we want to express the problem in the cumulative distribution functions, we must use $-\infty$, otherwise the definition does not match. Using the cumulative distribution function of the uniform distribution,

$$G(r) = \begin{cases} 0 & \text{for } r < 0 \\ r & \text{for } 0 \leq r \leq 1 \\ 1 & \text{for } r > 1 \end{cases}, \tag{3.24}$$

i.e. in this case $G(r) = r$, since $r \in [0, 1]$. With this we can invert eq. (3.23) to obtain

$$x(r) = F^{-1}(r)\,. \tag{3.25}$$

This is the general formula for the transformation method.

An example given by [Cow98, p. 42] is the exponential distribution

$$f(x) = \begin{cases} \frac{1}{\xi} e^{-\frac{x}{\xi}} & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}, \tag{3.26}$$

Then, starting at eq. (3.23), we have the following equation

$$r = \int_0^{x(r)} \frac{1}{\xi} e^{-\frac{x}{\xi}}\,\mathrm{d}x'\,, \tag{3.27}$$

which, after evaluating the integral and then solving for $x(r)$, gives

$$x(r) = -\xi \ln(1-r)\,. \tag{3.28}$$

Acceptance-Rejection Method If $f(x)$ is more complicated and the transformation method does not work, we can use the acceptance-rejection method, also called rejection sampling. Again, we want to sample a random variable x with a probability density function $f(x)$, given that we have a uniform pseudorandom number generator. This method then re-

lies on generating two random numbers at each iteration for a one-dimensional $f(x)$, to sample $f(x)$ in a region $x \in [x_{\min}, x_{\max}]$ (cf. [Cow98, sec. 3.3]):

1. Generate a pseudorandom number r_1 and transform it into the interval $x \in [x_{\min}, x_{\max}]$ using eq. (3.21).
2. Generate another pseudorandom number r_2 and transform it into the interval $y \in [0, f_{\max}]$ using $y = r_2 f_{\max}$, where $f_{\max}$ is the maximal value of f in the interval $[x_{\min}, x_{\max}]$.
3. Accept (i.e. return) x if $y < f(x)$, otherwise reject it and resume from step 1.

The accepted values of x then follow the probability density function $f(x)$. If $f(x)$ has sharp peaks, the acceptance-rejection method becomes inefficient since a lot of points will be rejected. This can be circumvented if there is a function $g(x)$, which is greater than $f(x)$ point-wise, and is easy to sample from, either because it is a standard probability density function or because it can be sampled from using the transformation method. The algorithm then becomes [Cow98, p. 43]

1. Generate a random number x from $\frac{g(x)}{\int g(x')\,\mathrm{d}x'}$.
2. Generate a uniformly distributed random number r and transform it into the interval $y \in [0, g(x)]$ using $y = r\,g(x)$.
3. Accept x if $y < f(x)$, otherwise reject it and resume from step 1.

The integral in step 1 acts as a normalization, since $g(x)$ can be an arbitrary function that need not be normalized. The accepted values are then again distributed according to $f(x)$, but only the region below $g(x)$ had to be evaluated, not the whole rectangle up to $f_{\max}$.

3.5.2 Monte Carlo Integration

One of the applications of the Monte Carlo method is integration. We want to calculate the d-dimensional integral

$$I = \int_\Omega f(\vec{x})\,\mathrm{d}^d x\,, \tag{3.29}$$

over a domain Ω.

Since we already know the function $f(\vec{x})$, we could ask ourselves why we can't just use other known methods of integration, such as the trapezoidal rule or Simpson's rule. The problem is that the convergence of these methods depends on the dimension, e.g. $1/N^{2/d}$ for the trapezoidal rule, and $1/N^{4/d}$ for Simpson's rule, whereas the Monte Carlo method has a convergence of $1/\sqrt{N}$, independent of the dimension [Pap20, tab. 3]. N denotes the number of points, i.e. the number of function evaluations we need. Therefore, the Monte Carlo method is used for high-dimensional integrals.

The Monte Carlo estimator for the integral I using N samples is given by

$$I_N = \frac{\mathrm{V}(\Omega)}{N} \sum_{i=1}^{N} f(\vec{x}_i) =: \frac{1}{N} \sum_{i=1}^{N} W_i\,, \tag{3.30}$$

where the W_i are called the Monte Carlo (MC) weights. The x_i are uniformly distributed in Ω, i.e. calculated by eq. (3.21) from the N generated $r_i \in [0,1]$. In the limit of large numbers $N \to \infty$, the estimator I_N converges to the true value I. We introduce an estimator for the variance of the function f:

$$V_N = \frac{1}{N}\sum_{i=1}^{N}(W_i - I_N)^2 = \frac{1}{N}\Big(\sum_{i=1}^{N} W_i^2\Big) - I_N^2 \,. \tag{3.31}$$

Since for Monte Carlo simulations, the N is usually large, the difference between the unbiased and biased estimator is negligible, so we use the biased estimator for the variance.

The variance of the estimator I_N is then given by the variance of the sample mean $\tilde{V}_N = \frac{V_N}{N}$, i.e. we now have estimators for the integral itself and also its standard error:

$$I \approx I_N \pm \sqrt{\tilde{V}_N} \,. \tag{3.32}$$

It should be noted that we do not have to deal with the integration boundaries, as we can transform the integral in such a way that we integrate over the unit hypercube $[0,1]^d$. In one dimension this transformation looks like

$$\int_{x^{\min}}^{x^{\max}} f(x)\,\mathrm{d}x = \int_0^1 [x^{\max} - x^{\min}] f(x(r))\,\mathrm{d}r \,, \tag{3.33}$$

where $x(r)$ is the uniform transformation introduced in eq. (3.21). In d dimensions, the expression gets longer if we write out the dependencies of the integration limits explicitly:

$$\begin{aligned}\int_\Omega f(\vec{x})\,\mathrm{d}^d x &= \int_{x_1^{\min}}^{x_1^{\max}} \int_{x_2^{\min}(x_1)}^{x_2^{\max}(x_1)} \cdots \int_{x_d^{\min}(x_1,\ldots,x_{d-1})}^{x_d^{\max}(x_1,\ldots,x_{d-1})} f(\vec{x})\,\mathrm{d}^d x \\ &= \int_0^1 \int_0^1 \cdots \int_0^1 \Big[x_1^{\max} - x_1^{\min}\Big]\Big[x_2^{\max}(x_1(r_1)) - x_2^{\max}(x_1(r_1))\Big] \\ &\qquad \cdots \Big[x_d^{\max}(x_1(r_1),\ldots,x_{d-1}(r_{d-1})) - x_d^{\max}(x_1(r_1),\ldots,x_{d-1}(r_{d-1}))\Big] \\ &\qquad \cdot f(\vec{x}(\vec{r}))\,\mathrm{d}^d r \,,\end{aligned} \tag{3.34}$$

where $x_i(r_i)$ is again the uniform transformation. This makes the integration easier, as then the volume factor $\mathrm{V}([0,1]^d)$ in eq. (3.30) occurring in the next equation is 1, and does not have to be separately calculated.

Stratified Sampling Stratified Sampling is a way to reduce the variance of the Monte Carlo estimator. The integration space Ω is divided into M different subspaces (strata) Ω_j, $j \in \{1,\ldots,M\}$:

$$\Omega = \bigcup_{j=1}^{M} \Omega_j \,. \tag{3.35}$$

The integral I can then be written as

$$I = \int_\Omega f(\vec{x})\,\mathrm{d}^d x = \sum_{j=1}^{M} \int_{\Omega_j} f(\vec{x})\,\mathrm{d}^d x \,. \tag{3.36}$$

In practice, this is done by partitioning the hypercube $[0,1]^d$ into M sub-hypercubes, to use with eq. (3.34). We define the MC estimator of the integral in each stratum Ω_j as

$$I_{N_j}^{(j)} = \frac{\mathrm{V}(\Omega_j)}{N_j} \sum_{i=1}^{N_j} f(\vec{x}_i^{(j)}) =: \frac{1}{N_j} \sum_{i=1}^{N_j} W_i^{(j)} , \tag{3.37}$$

analogously to eq. (3.30). Here, $\vec{x}_i^{(j)}$ are the N_j generated points in stratum Ω_j, and $V(\Omega_j)$ gives the stratum's volume. The total estimator is then given by adding up the integrals, as in eq. (3.36):

$$I_N = \sum_{j=1}^{M} I_{N_j}^{(j)} , \tag{3.38}$$

where $\vec{x}_i^{(j)}$ are the N_j generated points in each stratum Ω_j, and $V(\Omega_j)$ gives the stratum's volume.

Then, in each Ω_j, we calculate the variances $V_{N_j}^{(j)}$ analogously to eq. (3.31):

$$V_{N_j}^{(j)} = \frac{1}{N_j} \sum_{i=1}^{N_j} \left(W_i^{(j)} - I_{N_j}^{(j)} \right)^2 = \frac{1}{N_j} \left(\sum_{i=1}^{N_j} \left(W_i^{(j)} \right)^2 \right) - \left(I_{N_j}^{(j)} \right)^2 . \tag{3.39}$$

We now want to obtain the variance $\tilde{V}_N = \mathrm{Var}[I_N]$ of the total estimator I_N again. Since the samples in the Ω_j are independent, the total variance reduces to a sum:

$$\tilde{V}_N = \mathrm{Var}[I_N] = \sum_{j=1}^{M} \mathrm{Var}[I_N^{(j)}] = \sum_{j=1}^{M} \frac{V_{N_j}^{(j)}}{N_j} , \tag{3.40}$$

and the total integral with uncertainty is again given by eq. (3.32).

The naive MC, of course, falls out as a special case for $j = 1$. To ensure that this method actually reduces the variance, we need to choose the strata Ω_j and the number of points N_j accordingly. We see from eq. (3.39) that the variance in each stratum gets smaller the more homogeneous f is in this region, so we want to choose the strata such that f is as constant as possible. The optimal N_j can be obtained from minimizing eq. (3.40) with respect to N_j, together with the assumptions that $V_{N_j}^{(j)}$ is constant and that $\sum_{j=1}^{M} N_j = N$, e.g. with the method of Lagrange multipliers:

$$N_j = N \frac{\sqrt{V_{N_j}^{(j)}}}{\sum_{k=1}^{M} \sqrt{V_{N_k}^{(k)}}} . \tag{3.41}$$

This shows that taking $N_j \sim \sqrt{V_{N_j}^{(j)}}$ is a good choice.

Importance Sampling Importance Sampling is another variance reduction method. We introduce a function $g(\vec{x})$, which we use to rewrite the integral I as

$$I = \int_\Omega f(\vec{x})\,\mathrm{d}^d x = \int_\Omega \frac{f(\vec{x})}{g(\vec{x})}\, g(\vec{x})\,\mathrm{d}^d x\,. \tag{3.42}$$

If we require

$$g(x) \geq 0\,, \quad \int_\Omega g(\vec{x})\,\mathrm{d}^d x = 1\,, \tag{3.43}$$

then $g(\vec{x})$ is a probability density function. We can now use the Monte Carlo estimator eq. (3.30) in the same way as before:

$$I_N = \frac{\mathrm{V}(\Omega)}{N} \sum_{i=1}^{N} \frac{f(\vec{x}_i)}{g(\vec{x}_i)} =: \frac{1}{N} \sum_{i=1}^{N} W_i\,. \tag{3.44}$$

where now we have to sample the $\vec{x}_i$ from the distribution $g(x)$ because of the remaining factor $g(\vec{x})$ in eq. (3.42), e.g. with one of the methods described in the preceding section.

The variance of the estimator I_N is now again given by eq. (3.31), but with the new weights W_i defined in eq. (3.44). Therefore we obtain the integral again by eq. (3.32).

The reason to use this method is the choice of $g(\vec{x})$: Regardless of this, the integral I_N should always approximate the true value I, but the variance in eq. (3.31) can be changed for different choices of $g(\vec{x})$ because the weights W_i depend on it. If we choose $g(\vec{x}) = V(\Omega) f(\vec{x})/I$, it follows that $W_i = I$, such that the variance would tend to 0 if $N \to \infty$. This is of course impossible since I is the value we are looking for, but it gives us an idea: Choosing $g(\vec{x})$ such that it is close to being proportional to $f(\vec{x})$ reduces the variances and such the number of evaluations N needed to obtain better accuracy.

VEGAS algorithm The VEGAS algorithm [Pet78] is an adaptive MC algorithm, i.e. it solves the problem of having to know information about the integrand f beforehand. The goal is to get the optimal choice of $g(x)$ for importance sampling, as discussed in the last paragraph:

$$g_{\mathrm{opt}}(x) = \frac{|f(x)|}{\int_0^1 |f(x')|\,\mathrm{d}^d x'}\,. \tag{3.45}$$

The absolute value is taken so that $g_{\mathrm{opt}}(x) \geq 0$. There is no volume factor in this equation since the VEGAS algorithm only works for the unit hypercube $[0,1]^d$, which can be achieved from the aforementioned rescaling of the integrand. VEGAS uses the subsequent evaluations of the integrand to construct a grid, first of uniform size, but adapting non-uniform step size during subsequent evaluations. The grid undergoes some smoothing after each iteration to circumvent large discontinuities. In more than one dimension, there is one grid for each dimension, which means $g_{\mathrm{opt}}(\vec{x})$ is assumed to factorize:

$$g(\vec{x}) = g_1(x_1)\, g_2(x_2) \ldots g_d(x_d)\,. \tag{3.46}$$

As the algorithm is iterative, a cumulative estimation of the integral from separate iterations, counted by j, can be constructed [Lep21, eq. (30)] (notation adapted to the one

used in this work):

$$I_N = \frac{\sum_j I_N^{(j)} / V_N^{(j)}}{\sum_j 1/V_N^{(j)}} \quad \text{and} \quad V_N = \left(\sum_j \frac{1}{V_N^{(j)}} \right)^{-1/2} . \tag{3.47}$$

The per-iteration quantities are calculated according to the preceding paragraph. Having multiple iterations of the sampling allows the calculation of a χ^2 function to make sure the iterations are consistent with each other:

$$\chi^2 = \sum_j \frac{\left(I_N^{(j)} - I_N\right)^2}{V_N^{(j)}} . \tag{3.48}$$

It is expected that χ^2 is of the order of the number of iterations.

Additionally, stratified sampling is present in the VEGAS algorithm. In the classical one, the integration hypercube is divided into equally sized sub-hypercubes. The sampling is then carried out according to the method described in this section. In a newer version of the VEGAS algorithm, the stratified sampling algorithm is also replaced by an adaptive version, which tries to get to the optimal number of points in each stratum, proportional to the standard error in each stratum, as described above.

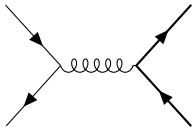
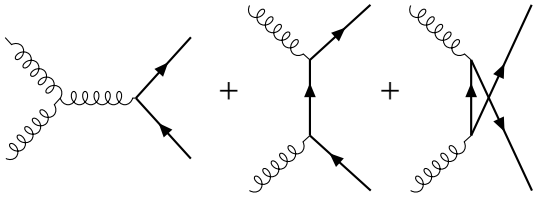
The VEGAS algorithm is heavily used in High Energy Physics (HEP). It is e.g. implemented in the GNU Scientific Library.

4 Heavy Quarks

4.1 Heavy Quark Production Calculation

We turn to the calculation of the cross section for heavy-quark production in proton-proton collisions at LO in QCD, i.e. the process $\mathrm{p\,p} \to \mathrm{Q\,\overline{Q}}$. At the parton level, there are two contributing LO channels, namely $\mathrm{q\,\bar{q}} \to \mathrm{Q\,\overline{Q}}$ and $\mathrm{g\,g} \to \mathrm{Q\,\overline{Q}}$, where q denotes a light (massless) quark and Q a heavy quark. The Feynman diagrams are shown in table 1.

Table 1: LO processes of heavy-quark production at the parton level.

Process	Diagram(s)
$\mathrm{q\,\bar{q}} \to \mathrm{Q\,\overline{Q}}$	
$\mathrm{g\,g} \to \mathrm{Q\,\overline{Q}}$	

4.1.1 Kinematics and the Partonic Cross Section

There are two reference frames of interest: The hadron and parton center-of-momentum system (CMS).

Hadron CMS In the hadron CMS, the beams (e.g. protons) collide with opposite 3-momenta $\vec{P}_{1,2}$, i.e.

$$\vec{P}_1 = -\vec{P}_2 \quad \text{or} \quad \vec{P}_1 + \vec{P}_2 = 0\,. \tag{4.1}$$

We use the first hadronic Mandelstam variable s to parameterize the momenta of the incoming hadrons. It is defined as

$$\begin{aligned} s := (P_1 + P_2)^2 = (E_1 + E_2)^2 - (\underbrace{\vec{P}_1 + \vec{P}_2}_{=0})^2 &\overset{(4.1)}{=} (E_1 + E_2)^2 \\ \Rightarrow \quad \sqrt{s} = E_1 + E_2 &=: E_{\mathrm{CMS}}\,. \end{aligned} \tag{4.2}$$

The incoming protons can be approximated as massless, since, e.g. for an LHC energy of 7 TeV per proton, its rest energy makes up less than 1 ‰. Because the hadron CMS is symmetrical, each proton gets the same energy $E_1 = E_2 = \frac{\sqrt{s}}{2}$ and, from the massless condition, opposite 3-momenta $\vec{P}_{1,2} = \pm\frac{\sqrt{s}}{2}\vec{e}_z$ (where we chose the z axis parallel to the

beam axis). Thus, the momenta of the incoming hadrons in the hadron CMS look like

$$P_1 = \begin{pmatrix} \frac{\sqrt{s}}{2} \\ 0 \\ 0 \\ \frac{\sqrt{s}}{2} \end{pmatrix} \quad \text{and} \quad P_2 = \begin{pmatrix} \frac{\sqrt{s}}{2} \\ 0 \\ 0 \\ -\frac{\sqrt{s}}{2} \end{pmatrix}. \tag{4.3}$$

As the partons are assumed to be essentially free, the momenta of the partons can be different from the hadron momenta. For parton momenta in the partonic CMS we use a lowercase p, where $p_{1,2}$ are the incoming and $p_{3,4}$ the outgoing momenta. We will assume the incoming partons to be massless and their 3-momenta to be collinear to the proton 3-momenta with momentum fractions $x_{1,2}$:

$$\vec{p}_1 = x_1 \vec{P}_1 \quad \text{and} \quad \vec{p}_2 = x_2 \vec{P}_2 \,. \tag{4.4}$$

Due to all the momenta belonging to massless particles, we get from the energy-momentum relation that the energy component is also multiplied by the momentum fraction:

$$(p_i^0)^2 = \vec{p}_i^{\,2} \overset{(4.4)}{=} x_i^2 \, \vec{P}_i^{\,2} = x_i^2 (P_i^0)^2 \,, \tag{4.5}$$

where $i \in \{1, 2\}$ counts the incoming particles. For particles with masses, this is not true. In total, the momenta of the incoming partons in the hadron CMS are

$$p_1 = x_1 P_1 \overset{(4.3)}{=} x_1 \begin{pmatrix} \frac{\sqrt{s}}{2} \\ 0 \\ 0 \\ \frac{\sqrt{s}}{2} \end{pmatrix} \quad \text{and} \quad p_2 = x_2 P_2 \overset{(4.3)}{=} x_2 \begin{pmatrix} \frac{\sqrt{s}}{2} \\ 0 \\ 0 \\ -\frac{\sqrt{s}}{2} \end{pmatrix}. \tag{4.6}$$

The outgoing partons with momenta p_3 and p_4 can scatter in arbitrary directions. We parameterize them by the transverse momentum p_T, which is the component of the momentum orthogonal to the beam axis, and the rapidity y, which describes the boost of the outgoing particles along the beam direction. A general momentum p can be parameterized as follows

$$p = \begin{pmatrix} E_\mathrm{T} \cosh y \\ p_\mathrm{T} \cos \phi \\ p_\mathrm{T} \sin \phi \\ E_\mathrm{T} \sinh y \end{pmatrix} \tag{4.7}$$

The transverse energy E_T is defined as

$$E_\mathrm{T} = \sqrt{m^2 + p_\mathrm{T}^2} \,, \tag{4.8}$$

where m is the mass of the particle with momentum p, i.e. the heavy quark mass in this work. In case of $m = 0$, E_T is equal to the transverse momentum p_T. The angle ϕ describes the angle of the transverse momentum in the x-y plane. As we can still choose the rotation

of the coordinate system around the z-axis, we choose p_T along the x-direction, i.e. $\phi = 0$. Since we assume collinearity between the parton and hadron momenta, the parton CMS is only boosted in the z-direction relative to the hadron CMS, leading to the outgoing partons having the same transverse momentum, but different rapidities. We name the latter $y_{3,4}$ to be consistent with the momentum indices of the momenta. The momenta of the outgoing partons in the hadron CMS are then

$$p_3 = \begin{pmatrix} E_T \cosh y_3 \\ p_T \\ 0 \\ E_T \sinh y_3 \end{pmatrix} \quad \text{and} \quad p_4 = \begin{pmatrix} E_T \cosh y_4 \\ -p_T \\ 0 \\ E_T \sinh y_4 \end{pmatrix} . \tag{4.9}$$

By using 4-momentum conservation, we can relate the momentum fractions $x_{1,2}$ to the rapidities $y_{3,4}$:

$$\begin{aligned}
& p_1 + p_2 = p_3 + p_4 \\
\overset{(4.6),\,(4.9)}{\Rightarrow} \quad & \frac{\sqrt{s}}{2} \begin{pmatrix} x_1 + x_2 \\ 0 \\ 0 \\ x_1 - x_2 \end{pmatrix} = E_T \begin{pmatrix} \cosh y_3 + \cosh y_4 \\ 0 \\ 0 \\ \sinh y_3 + \sinh y_4 \end{pmatrix} \qquad \begin{matrix} \text{(I)} \\ \\ \\ \text{(II)} \end{matrix} \\
\overset{\text{(I)}+\text{(II)}}{\Rightarrow} \quad & x_1 = \frac{E_T}{\sqrt{s}} (\cosh y_3 + \cosh y_4 + \sinh y_3 + \sinh y_4) \\
& \quad = \frac{E_T}{\sqrt{s}} (e^{y_3} + e^{y_4}) \\
\overset{\text{(I)}-\text{(II)}}{\Rightarrow} \quad & x_2 = \frac{E_T}{\sqrt{s}} (\cosh y_3 + \cosh y_4 - \sinh y_3 - \sinh y_4) \\
& \quad = \frac{E_T}{\sqrt{s}} (\cosh(-y_3) + \cosh(-y_4) + \sinh(-y_3) + \sinh(-y_4)) \\
& \quad = \frac{E_T}{\sqrt{s}} (e^{-y_3} + e^{-y_4})
\end{aligned} \tag{4.10}$$

So, in total, we get

$$x_1 = \frac{E_T}{\sqrt{s}} (e^{y_3} + e^{y_4}) \quad \text{and} \quad x_2 = \frac{E_T}{\sqrt{s}} (e^{-y_3} + e^{-y_4}) . \tag{4.11}$$

This will also occur naturally when transforming the integration variables in the cross section calculation.

Parton CMS Analogous to the hadron CMS, in the parton CMS the incoming quarks or gluons collide with opposite 3-momenta $\vec{k}_{1,2}$, i.e.

$$\vec{k}_1 = -\vec{k}_2 \quad \Leftrightarrow \quad \vec{k}_1 + \vec{k}_2 = 0 . \tag{4.12}$$

This is the frame in which the actual calculation takes place. In the end, we want to express our result in Lorentz-invariant quantities. We can do that by using the Mandelstam vari-

ables (summarized in appendix A.1), where we also use the convention to equip partonic variables with a hat:

$$\begin{aligned} \hat{s} &= (k_1 + k_2)^2 = (k_3 + k_4)^2 \,, \\ \hat{t} &= (k_1 - k_3)^2 = (k_2 - k_4)^2 \,, \\ \hat{u} &= (k_1 - k_4)^2 = (k_2 - k_3)^2 \,. \end{aligned} \tag{4.13}$$

The second equality in each line stems from 4-momentum conservation. In analogy to eq. (4.3), the momenta of the incoming partons in the parton CMS are

$$k_1 = \begin{pmatrix} \frac{\sqrt{\hat{s}}}{2} \\ 0 \\ 0 \\ \frac{\sqrt{\hat{s}}}{2} \end{pmatrix} \quad \text{and} \quad k_2 = \begin{pmatrix} \frac{\sqrt{\hat{s}}}{2} \\ 0 \\ 0 \\ -\frac{\sqrt{\hat{s}}}{2} \end{pmatrix} . \tag{4.14}$$

Since the scalar product of two 4-vectors is Lorentz-invariant, we can get a relation between $\sqrt{s}$ and $\sqrt{\hat{s}}$:

$$k_1 \cdot k_2 = p_1 \cdot p_2 \overset{(4.6),\,(4.14)}{\Rightarrow} \hat{s} = x_1\, x_2\, s \tag{4.15}$$

The parton level differential cross section $\mathrm{d}\hat{\sigma}_i$ for a channel $i \in \{\mathrm{q}\,\bar{\mathrm{q}} \to \mathrm{Q}\,\overline{\mathrm{Q}},\ \mathrm{g}\,\mathrm{g} \to \mathrm{Q}\,\overline{\mathrm{Q}}\}$ can be parameterized as follows (see [Sch14]):

$$\mathrm{d}\hat{\sigma}_i = \frac{1}{F} \overline{|\mathcal{M}_i|}^2 \mathrm{dPS}_2 \,, \tag{4.16}$$

where F is the flux factor, $\mathcal{M}_i$ is the matrix element for the process i, averaged over initial and summed over final spins and colors, and dPS_2 is the Lorentz-invariant phase space volume element for the two-body final state. In the following, we want to calculate this differential cross section for the two processes in the parton CMS.

Before the actual calculation, we need the following for any particle with energy E, 3-momentum $\vec{k}$ and velocity $\vec{v}$:

$$\begin{gathered} \vec{k} = \frac{m\vec{v}}{\sqrt{1-\vec{v}^2}} \quad \Rightarrow \quad \vec{k}^2 \left(1 - \vec{v}^2\right) = m^2 \vec{v}^2 \quad \Rightarrow \quad \vec{v}^2 = \frac{\vec{k}^2}{m^2 + \vec{k}^2} \\ \overset{\vec{v} \parallel \vec{k}}{\Rightarrow} \quad \vec{v} = \frac{\vec{k}}{E} \end{gathered} \tag{4.17}$$

and also the relation between the Lorentz invariant 3- and 4-dimensional integration measures over a momentum k of a particle with mass m:

$$\begin{aligned} \mathrm{d}^4k\ \delta(k^2 - m^2)\,\theta(k^0) &= \mathrm{d}^4k\ \delta\left((k^0)^2 - \vec{k}^2 - m^2\right)\theta(k^0) \\ &= \frac{\mathrm{d}^4k}{2\sqrt{\vec{k}^2 + m^2}} \left(\delta\left(k^0 - \sqrt{\vec{k}^2 + m^2}\right) + \delta\left(k^0 + \sqrt{\vec{k}^2 + m^2}\right)\right)\theta(k^0) = \frac{\mathrm{d}^3k}{2E} \,, \end{aligned} \tag{4.18}$$

where in the last equality the particle is assigned a fixed energy $E = \sqrt{\vec{k}^2 + m^2}$.

We will start the actual calculation by considering the flux factor F of eq. (4.16), which

is defined by

$$\begin{aligned} F &= 4E_1E_2\left|\vec{v}_1-\vec{v}_2\right| \overset{(4.17)}{=} 4\left|E_2\vec{k}_1-E_1\vec{k}_2\right| \\ &\overset{(4.12)}{=} 4\left|\vec{k}_1\right|\left|E_1+E_2\right| \overset{(4.14)}{=} 2\hat{s}\,. \end{aligned} \tag{4.19}$$

The calculation of dPS_2 is a bit longer. Since dPS_2 is Lorentz-invariant, we choose the hadron CMS for performing the integration. We again start from its definition:

$$\begin{aligned} \mathrm{dPS}_2 &= \frac{\mathrm{d}^3p_3}{(2\pi)^3 2E_3}\frac{\mathrm{d}^3p_4}{(2\pi)^3 2E_4}(2\pi)^4\,\delta^{(4)}(p_1+p_2-p_3-p_4) \\ &\overset{(4.18)}{=} \frac{1}{8\pi^2}\frac{\mathrm{d}^3p_3}{E_3}\,\mathrm{d}^4p_4\,\delta\left(p_4^2-m^2\right)\theta(p_4^0)\,\delta^{(4)}(p_1+p_2-p_3-p_4) \\ &= \frac{1}{8\pi^2}\frac{\mathrm{d}^3p_3}{E_3}\,\delta\Big((p_3-p_1-p_2)^2-m^2\Big) \\ &= \frac{1}{8\pi^2}\frac{\mathrm{d}^3p_3}{E_3}\,\delta(2p_1\cdot p_2-2\,(p_1+p_2)\cdot p_3) \\ &\overset{(4.6),\,(4.9)}{=} \frac{1}{8\pi^2}\frac{\mathrm{d}^3p_3}{E_3}\,\delta\left(x_1x_2s-\sqrt{s}\left((x_1+x_2)\,p_3^0-(x_1-x_2)\,p_3^3\right)\right)\,. \end{aligned} \tag{4.20}$$

This intermediate result will be simplified later in the full factorization integral.

At last, only the matrix elements $|\mathcal{M}|^2_{\mathrm{g\,g}\to\mathrm{Q\,\overline{Q}}}$ and $|\mathcal{M}|^2_{\mathrm{q\,\bar{q}}\to\mathrm{Q\,\overline{Q}}}$ are missing. We will calculate $|\mathcal{M}|^2_{\mathrm{q\,\bar{q}}\to\mathrm{Q\,\overline{Q}}}$ explicitly. As mentioned before, we define the parton momenta as is displayed in fig. 2

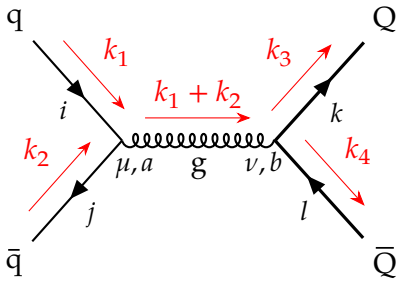


Figure 2: Definition of the parton momenta needed for the calculation of the matrix elements. The $\mathrm{g\,g}\to\mathrm{Q\,\overline{Q}}$ channel is defined analogously.

Using the Feynman rules from section 2.4, we can write down the matrix element as follows:

$$\begin{aligned} \mathrm{i}\mathcal{M}_{\mathrm{q\,\bar{q}}\to\mathrm{Q\,\overline{Q}}} &= \bar{v}(k_2)\left(\mathrm{i}g_s\gamma^\mu t^a_{ji}\right)u(k_1)\;\bar{u}(k_3)\left(\mathrm{i}g_s\gamma^\nu t^b_{kl}\right)v(k_4)\;\frac{-\mathrm{i}g_{\mu\nu}}{(k_1+k_2)^2}\,\delta_{ab} \\ &= \mathrm{i}g_s^2\,t^a_{ji}\,t^a_{kl}\,\frac{g_{\mu\nu}}{\hat{s}}\,[\bar{v}(k_2)\gamma^\mu u(k_1)]\,[\bar{u}(k_3)\gamma^\nu v(k_4)] \end{aligned} \tag{4.21}$$

where u, $\bar{u}$, v and $\bar{v}$ are the momentum-space spinors associated with the incoming and outcoming particles. We left out the gauge parameter ξ in the gluon propagator from the start, since it drops out when squaring the matrix element anyway. Next, we write down

the adjoint of eq. (4.21):

$$-\mathrm{i}\mathcal{M}^{\dagger}_{\mathrm{q}\bar{\mathrm{q}}\to\mathrm{Q}\bar{\mathrm{Q}}} = -\mathrm{i}g_\mathrm{s}^2\, t^a_{ij}\, t^a_{lk}\, \frac{g_{\mu\nu}}{\hat{s}}\, [\bar{u}(k_1)\gamma^\mu v(k_2)]\, [\bar{v}(k_4)\gamma^\nu v(k_3)] \tag{4.22}$$

Remember that the SU(3) generators are hermitian, and therefore $(t^a_{ji})^* = t^a_{ij}$. Multiplying eqs. (4.21) and (4.22) gives

$$\begin{aligned} |\mathcal{M}|^2_{\mathrm{q}\bar{\mathrm{q}}\to\mathrm{Q}\bar{\mathrm{Q}}} = \frac{g_\mathrm{s}^4}{\hat{s}^2}\, t^a_{ji}\, t^a_{kl}\, t^b_{ij}\, t^b_{lk}\, [\bar{v}(k_2)\gamma^\mu u(k_1)] \\ \cdot\, [\bar{u}(k_3)\gamma_\mu v(k_4)]\, [\bar{u}(k_1)\gamma^\rho v(k_2)]\, [\bar{v}(k_4)\gamma_\rho v(k_3)] \end{aligned} \tag{4.23}$$

Next, we average the spins s_1, s_2 over the initial states and sum over the spins s_3, s_4 of the final states. We understand each momentum space spinor to have an implicit spin index, matched to its momentum, so that we can use the following relations:

$$\sum_s u^{(s)}(k)\bar{u}^{(s)}(k) = \not{k} + m\,, \tag{4.24}$$

$$\sum_s v^{(s)}(k)\bar{v}^{(s)}(k) = \not{k} - m\,. \tag{4.25}$$

This leads to the following traces in the matrix element:

$$\frac{1}{4}\sum_{\substack{s_1,s_2,\\ s_3,s_4}} |\mathcal{M}|^2_{\mathrm{q}\bar{\mathrm{q}}\to\mathrm{Q}\bar{\mathrm{Q}}} = \frac{1}{4}\frac{g_\mathrm{s}^4}{\hat{s}^2}\, t^a_{ji}\, t^a_{kl}\, t^b_{ij}\, t^b_{lk}\, \mathrm{Tr}[\not{k}_1\gamma^\rho \not{k}_2\gamma^\mu]\, \mathrm{Tr}[(\not{k}_3 + m)\gamma_\mu(\not{k}_4 - m)\gamma_\rho] \tag{4.26}$$

The first trace does not have masses, since k_1 and k_2 belong to the incoming partons, which are massless. We use the following trace theorems [Sch14, app. A.4]

$$\mathrm{Tr}[\gamma^\mu\gamma^\nu] = 4g^{\mu\nu}\,, \tag{4.27}$$

$$\mathrm{Tr}[\gamma^\mu\gamma^\nu\gamma^\rho\gamma^\sigma] = 4(g^{\mu\nu}g^{\rho\sigma} - g^{\mu\rho}g^{\nu\sigma} + g^{\mu\sigma}g^{\nu\rho}) \tag{4.28}$$

to evaluate the second, massive trace:

$$\begin{aligned} \mathrm{Tr}[(\not{k}_3 + m)\gamma_\mu(\not{k}_4 - m)\gamma_\rho] &= k_3^\nu k_4^\sigma\, \mathrm{Tr}[\gamma_\nu\gamma_\mu\gamma_\sigma\gamma_\rho] + m^2\, \mathrm{Tr}[\gamma_\mu\gamma_\rho] \\ &\overset{(4.27),\,(4.28)}{=} 4[k_{3\rho}\,k_{4\mu} + k_{3\mu}\,k_{4\rho} - g_{\mu\rho}\,(p_3\cdot p_4 + m^2)] \end{aligned} \tag{4.29}$$

The first trace in eq. (4.26) is the massless case of eq. (4.29):

$$\mathrm{Tr}[\not{k}_1\gamma^\rho\not{k}_2\gamma^\mu] = 4[k_1^\mu\,k_2^\rho + k_1^\rho\,k_2^\mu - g^{\mu\rho}\,(k_1\cdot k_2)]\,. \tag{4.30}$$

Next, we do the color average and sum:

$$\frac{1}{N^2}\sum_{i,j,k,l} t^a_{ji}\, t^a_{kl}\, t^b_{ij}\, t^b_{lk} \overset{(2.10)}{=} \frac{1}{4N^2}\,\delta^{ab}\delta^{ab} = \frac{N^2-1}{4N^2} \tag{4.31}$$

From now on, we denote the averaging and summing over spin and colors by a bar:

$$\begin{aligned}
\overline{|\mathcal{M}|}^2_{\mathrm{q}\bar{\mathrm{q}}\to\mathrm{Q}\bar{\mathrm{Q}}} &:= \frac{1}{4N^2}\sum_{\substack{i,j,k,l\\ s_1,s_2,s_3,s_4}} |\mathcal{M}|^2_{\mathrm{q}\bar{\mathrm{q}}\to\mathrm{Q}\bar{\mathrm{Q}}} \\
&= \frac{N^2-1}{16\,N^2}\,\frac{g_\mathrm{s}^4}{\hat{s}^2}\,16\,[k_1^\mu\,k_2^\rho + k_1^\rho\,k_2^\mu - g^{\mu\rho}\,(k_1\cdot k_2)] \\
&\qquad\qquad \cdot [k_{3\rho}\,k_{4\mu} + k_{3\mu}\,k_{4\rho} - g_{\mu\rho}\,(k_3\cdot k_4 + m^2)]\,. \\
&= 2\,\frac{N^2-1}{N^2}\,\frac{g_\mathrm{s}^4}{\hat{s}^2}[(k_1\cdot k_3)(k_2\cdot k_4) + (k_1\cdot k_4)(k_2\cdot k_3) + m^2(k_1\cdot k_2)] \qquad (4.32)
\end{aligned}$$

We can rewrite the scalar products in the Mandelstam variables using eq. (A.2):

$$\begin{aligned}
\overline{|\mathcal{M}|}^2_{\mathrm{q}\bar{\mathrm{q}}\to\mathrm{Q}\bar{\mathrm{Q}}} &= \frac{N^2-1}{2\,N^2}\,\frac{g_\mathrm{s}^4}{\hat{s}^2}\,[(m^2-\hat{t})^2 + (m^2-\hat{u})^2 + 2\,m^2\,\hat{s}] \\
&= \frac{N^2-1}{2\,N^2}\,\frac{g_\mathrm{s}^4}{\hat{s}^2}\,[\hat{t}^2 + \hat{u}^2 + 2\,m^2\,\hat{s} - 2\,m^2(\hat{t}+\hat{u}) + 2\,m^4] \\
&\overset{(A.3)}{=} \frac{N^2-1}{2\,N^2}\,\frac{g_\mathrm{s}^4}{\hat{s}^2}\,[\hat{t}^2 + \hat{u}^2 + 4\,m^2\,\hat{s} - 2\,m^4] \\
&\overset{N=3}{=} \frac{4}{9}\,\frac{g_\mathrm{s}^4}{\hat{s}^2}\,[\hat{t}^2 + \hat{u}^2 + 4\,m^2\,\hat{s} - 2\,m^4]\,. \qquad (4.33)
\end{aligned}$$

One can easily see that, in the massless limit, the result is the same as in [ESW96, p. 249]. The full result eq. (4.33) was checked against a calculation with the Mathematica package FeynCalc [SMO20] and numerically against [ESW96, p. 349].

The gluon-gluon channel was calculated from the Feynman diagrams using FeynCalc. In conclusion, the matrix elements for the two channels are

$$\begin{aligned}
\overline{|\mathcal{M}|}^2_{\mathrm{g}\,\mathrm{g}\to\mathrm{Q}\bar{\mathrm{Q}}} &= g_\mathrm{s}^4\,\frac{\left(9m^4 - 9m^2\,\hat{s} + 4\,\hat{s}^2 - 9\,\hat{t}\,\hat{u}\right)\left(m^2\left(\hat{s}^3 + 4\,\hat{s}\,\hat{t}\,\hat{u}\right) - 3m^4\,\hat{s}^2 - 2m^8 + \hat{t}\,\hat{u}\left(\hat{t}^2+\hat{u}^2\right)\right)}{24\,\hat{s}^2\,(t-m^2)^2\,(u-m^2)^2}\,, \\
\overline{|\mathcal{M}|}^2_{\mathrm{q}\bar{\mathrm{q}}\to\mathrm{Q}\bar{\mathrm{Q}}} &= \frac{4}{9}\,\frac{g_\mathrm{s}^4}{\hat{s}^2}\,(\hat{t}^2 + \hat{u}^2 + 4\,m^2\,\hat{s} - 2\,m^4)\,. \qquad (4.34)
\end{aligned}$$

Here, $g_\mathrm{s} = \sqrt{4\pi\alpha_\mathrm{s}}$ is the strong coupling constant. The matrix elements were numerically checked against [ESW96, p. 349].

4.1.2 Hadronic Cross Section

Since protons are composite objects, the cross-section for the process $\mathrm{p\,p} \to \mathrm{Q\,\overline{Q}}$ is given by the following factorization integral:

$$\mathrm{d}\sigma = \sum_{i,j}\int_0^1 \mathrm{d}x_1\mathrm{d}x_2\,f_i(x_1,\mu_\mathrm{f})\,f_j(x_2,\mu_\mathrm{f})\,\mathrm{d}\hat{\sigma}_{ij\to\mathrm{Q}\bar{\mathrm{Q}}} \qquad (4.35)$$

where i,j are possible initial states of light (i.e. assumed massless) partons, e.g. $(i,j) \in \{(\mathrm{g},\mathrm{g}),(\mathrm{u},\bar{\mathrm{u}}),(\bar{\mathrm{u}},\mathrm{u}),\ldots\}$. f_i is a PDF for the parton i inside a proton. μ_f is the factorization scale, and a dependence on the renormalization scale μ_r hides in $\mathrm{d}\hat{\sigma}$ through α_s. Our goal is to substitute the integration variables by experimentally more accessible ones, namely

the transverse momentum and rapidity of an outgoing heavy quark. We start to do this by transforming $\mathrm{d}x_1 \mathrm{d}^3 p_3$:

$$\mathrm{d}x_1\, \mathrm{d}^3 p_3 = \left| \det\left(\frac{\partial(x_1, p_{3x}, p_{3y}, p_{3z})}{\partial(y_3, y_4, p_\mathrm{T}, \phi)} \right) \right| \mathrm{d}y_3\, \mathrm{d}y_4\, \mathrm{d}p_\mathrm{T}\, \mathrm{d}\phi\,. \tag{4.36}$$

For the calculation of the determinant we use eqs. (4.7) and (4.11) and the following relation:

$$\frac{\partial E_\mathrm{T}}{\partial p_\mathrm{T}} = \frac{\partial}{\partial p_\mathrm{T}} \sqrt{m^2 + p_\mathrm{T}^2} = \frac{1}{2} \left(m^2 + p_\mathrm{T}^2\right)^{-\frac{1}{2}} \cdot 2p_\mathrm{T} = \frac{p_\mathrm{T}}{E_\mathrm{T}}\,. \tag{4.37}$$

With this we obtain

$$\begin{aligned} \det\left(\frac{\partial(x_1, p_{3x}, p_{3y}, p_{3z})}{\partial(y_3, y_4, p_\mathrm{T}, \phi)} \right) &= \det \begin{pmatrix} \frac{E_\mathrm{T}}{\sqrt{s}} e^{y_3} & \frac{E_\mathrm{T}}{\sqrt{s}} e^{y_4} & \frac{p_\mathrm{T}}{E_\mathrm{T}\sqrt{s}} \left(e^{y_3} + e^{y_4}\right) & 0 \\ 0 & 0 & \cos\phi & -p_\mathrm{T} \sin\phi \\ 0 & 0 & \sin\phi & p_\mathrm{T} \cos\phi \\ E_\mathrm{T} \cosh y_3 & 0 & \frac{p_\mathrm{T}}{E_\mathrm{T}} \sinh y_3 & 0 \end{pmatrix} \\ &= -\frac{E_\mathrm{T}^2 p_\mathrm{T}}{\sqrt{s}} \cosh y_3\, e^{y_4}\,, \end{aligned} \tag{4.38}$$

so, in total, we get

$$\mathrm{d}x_1 \mathrm{d}^3 p_3 = \frac{E_\mathrm{T}^2 p_\mathrm{T}}{\sqrt{s}} \cosh y_3\, e^{y_4}\, \mathrm{d}y_3\, \mathrm{d}y_4\, \mathrm{d}p_\mathrm{T}\, \mathrm{d}\phi\,. \tag{4.39}$$

We also have to change variables in the δ-function argument in eq. (4.20):

$$\begin{aligned} &\delta\left(x_1 x_2 s - \sqrt{s}\left((x_1 + x_2)\, p_3^0 - (x_1 - x_2)\, p_3^3\right)\right) \\ &= \delta\Big(x_1 x_2 s - \sqrt{s}\, E_\mathrm{T}\, x_1 \big(\underbrace{\cosh y_3 - \sinh y_3}_{= e^{-y_3}}\big) - \sqrt{s}\, E_\mathrm{T}\, x_2 \big(\underbrace{\cosh y_3 + \sinh y_3}_{e^{y_3}}\big)\Big) \\ &= \delta\left(x_2 \left(x_1\, s - \sqrt{s}\, E_\mathrm{T}\, e^{y_3}\right) - \sqrt{s}\, E_\mathrm{T}\, x_1 e^{-y_3}\right) \\ &= \delta\left(x_2\, \sqrt{s}\, E_\mathrm{T}\, e^{y_4} - E_\mathrm{T}^2 \left(1 + e^{y_4 - y_3}\right)\right) \\ &= \frac{1}{\sqrt{s}\, E_\mathrm{T}\, e^{y_4}}\, \delta\left(x_2 - \frac{E_\mathrm{T}}{\sqrt{s}} \left(e^{-y_3} + e^{-y_4}\right)\right)\,, \end{aligned} \tag{4.40}$$

where we changed $\vec{p}_3$ in the first equality and x_1 in the third. Since, as before, we plugged the variable changes into momentum conservation, the result eq. (4.11) pops out again.

Plugging this into eq. (4.20) we get (using $E_3 = E_\mathrm{T} \cosh y_3$)

$$\begin{aligned} \mathrm{d}x_1\, \mathrm{dPS}_2 &= \frac{1}{8\pi^2 \hat{s}}\, p_\mathrm{T}\, \delta\left(x_2 - \frac{E_\mathrm{T}}{\sqrt{s}} \left(e^{-y_3} + e^{-y_4}\right)\right) \mathrm{d}y_3\, \mathrm{d}y_4\, \mathrm{d}p_\mathrm{T}\, \mathrm{d}\phi \\ &= \frac{1}{4\pi \hat{s}}\, p_\mathrm{T}\, \delta\left(x_2 - \frac{E_\mathrm{T}}{\sqrt{s}} \left(e^{-y_3} + e^{-y_4}\right)\right) \mathrm{d}y_3\, \mathrm{d}y_4\, \mathrm{d}p_\mathrm{T}\,, \end{aligned} \tag{4.41}$$

where the second equality comes from integrating out ϕ. This is possible since the integrand is independent of ϕ. We can now gather everything and plug it into the factorization

formula eq. (4.35):

$$\frac{\mathrm{d}^3\sigma}{\mathrm{d}p_\mathrm{T}\,\mathrm{d}y_3\,\mathrm{d}y_4} = \frac{p_\mathrm{T}}{8\pi\hat{s}^2}\sum_{i,j} x_1 f_i(x_1,\mu_\mathrm{f})\; x_2 f_j(x_2,\mu_\mathrm{f})\,\overline{|\mathcal{M}|}^2_{ij\to\mathrm{Q}\overline{\mathrm{Q}}}\,. \tag{4.42}$$

Because of the x_1-substitution according to eq. (4.11) and the δ-function in eq. (4.40), x_1 and x_2 in eq. (4.42) are just functions of p_T, y_3 and y_4 and are both given by eq. (4.11). This result can be verified by eq. (10.54) in [ESW96, p. 349].

To get actual differential distributions, we now have to integrate out two of the three variables. First, we will look at the p_T-distribution, i.e. $\mathrm{d}\sigma/\mathrm{d}p_\mathrm{T}$, thus we need the integration limits for y_3 and y_4. We start with the limits for y_4: Plugging eq. (4.11) into eq. (4.51) gives

$$0 \le \frac{E_\mathrm{T}}{\sqrt{s}}\left(e^{y_3}+e^{y_4}\right) \le 1 \quad \text{and} \quad 0 \le \frac{E_\mathrm{T}}{\sqrt{s}}\left(e^{-y_3}+e^{-y_4}\right) \le 1 \tag{4.43}$$

The "$0 \le \ldots$" sides do not give new information, since all occurring quantities are either square roots or exponentials, and therefore positive. But if we solve for y_4 on the "$\ldots \le 1$" sides of both inequalities, we obtain the integration limits for y_4:

$$\underbrace{-\ln\left(\frac{\sqrt{s}}{E_\mathrm{T}} - e^{-y_3}\right)}_{=:\,y_4^{\min}(p_\mathrm{T},y_3)} \le y_4 \le \underbrace{\ln\left(\frac{\sqrt{s}}{E_\mathrm{T}} - e^{y_3}\right)}_{=:\,y_4^{\max}(p_\mathrm{T},y_3)}\,. \tag{4.44}$$

In the next section we will see that the integration region for y_3 is bounded by the intersections of $y_4^{\min}(p_\mathrm{T},y_3)$ and $y_4^{\max}(p_\mathrm{T},y_3)$. Equating them and solving for y_3 gives

$$\begin{gathered} y_4^{\min}(p_\mathrm{T},y_3) = y_4^{\max}(p_\mathrm{T},y_3) \\ \overset{(4.44)}{\Rightarrow} \quad \underbrace{-\operatorname{arcosh}\left(\frac{\sqrt{s}}{2E_\mathrm{T}}\right)}_{=:\,y_3^{\min}(p_\mathrm{T})} \le y_3 \le \underbrace{\operatorname{arcosh}\left(\frac{\sqrt{s}}{2E_\mathrm{T}}\right)}_{=:\,y_3^{\max}(p_\mathrm{T})}\,. \end{gathered} \tag{4.45}$$

Thus, the p_T-distribution is given by

$$\frac{\mathrm{d}\sigma}{\mathrm{d}p_\mathrm{T}} = \sum_{i,j}\int_{y_3^{\min}(p_\mathrm{T})}^{y_3^{\max}(p_\mathrm{T})}\mathrm{d}y_3\int_{y_4^{\min}(p_\mathrm{T},y_3)}^{y_4^{\max}(p_\mathrm{T},y_3)}\mathrm{d}y_4\;\frac{p_\mathrm{T}}{8\pi\hat{s}^2}\,x_1 f_i(x_1,\mu_\mathrm{f})\;x_2 f_j(x_2,\mu_\mathrm{f})\,\overline{|\mathcal{M}|}^2_{ij\to\mathrm{Q}\overline{\mathrm{Q}}}\,. \tag{4.46}$$

For the y_3-distribution, i.e. $\mathrm{d}\sigma/\mathrm{d}y_3$, we can recycle the integration limits for y_4 from eq. (4.44). The integration limits for p_T are given, again, by the intersections:

$$\begin{gathered} y_4^{\min}(p_\mathrm{T},y_3) = y_4^{\max}(p_\mathrm{T},y_3) \\ \overset{(4.44)}{\Rightarrow} \quad \underbrace{-\sqrt{\frac{s}{4\cosh^2(y_3)} - m^2}}_{=:\,p_\mathrm{T}^{\min}(y_3)} \le p_\mathrm{T} \le \underbrace{\sqrt{\frac{s}{4\cosh^2(y_3)} - m^2}}_{=:\,p_\mathrm{T}^{\max}(y_3)}\,. \end{gathered} \tag{4.47}$$

And again the y_3-distribution reads

$$\frac{\mathrm{d}\sigma}{\mathrm{d}y_3} = \sum_{i,j} \int_{p_\mathrm{T}^\mathrm{min}(y_3)}^{p_\mathrm{T}^\mathrm{max}(y_3)} \mathrm{d}p_\mathrm{T} \int_{y_4^\mathrm{min}(p_\mathrm{T},y_3)}^{y_4^\mathrm{max}(p_\mathrm{T},y_3)} \mathrm{d}y_4 \, \frac{p_\mathrm{T}}{8\pi\hat{s}^2} \, x_1 f_i(x_1,\mu_\mathrm{f}) \, x_2 f_j(x_2,\mu_\mathrm{f}) \, \overline{|\mathcal{M}|}^2_{ij\to\mathrm{Q}\overline{\mathrm{Q}}} \,. \tag{4.48}$$

For the total cross section we integrate over all three variables. The integration limits for y_3 and y_4 are given by eq. (4.45) and eq. (4.44), respectively. The lower limit for the p_T-integration is $0 \leq p_\mathrm{T}$. We can get the upper from energy conservation:

$$\begin{aligned} & p_3^0 + p_4^0 = p_1^0 + p_2^0 \\ \overset{(4.6),\,(4.9)}{\Rightarrow} \quad & E_\mathrm{T}(\underbrace{\cosh y_3 + \cosh y_4}_{\geq 2}) = \frac{\sqrt{s}}{2}(\underbrace{x_1 + x_2}_{\leq 2}) \\ \Rightarrow \quad & 2E_\mathrm{T} \leq \sqrt{s} \\ \overset{(4.8)}{\Rightarrow} \quad & p_\mathrm{T} \leq \underbrace{\sqrt{\frac{s}{4} - m^2}}_{=:\, p_\mathrm{T}^\mathrm{max}} \,. \end{aligned} \tag{4.49}$$

This amounts to the following total cross section:

$$\begin{aligned} \sigma = \sum_{i,j} \int_0^{p_\mathrm{T}^\mathrm{max}} \mathrm{d}p_\mathrm{T} \int_{y_3^\mathrm{min}(p_\mathrm{T})}^{y_3^\mathrm{max}(p_\mathrm{T})} \mathrm{d}y_3 \int_{y_4^\mathrm{min}(p_\mathrm{T},y_3)}^{y_4^\mathrm{max}(p_\mathrm{T},y_3)} \mathrm{d}y_4 \\ \cdot \, \frac{p_\mathrm{T}}{8\pi\hat{s}^2} \, x_1 f_i(x_1,\mu_\mathrm{f}) \, x_2 f_j(x_2,\mu_\mathrm{f}) \, \overline{|\mathcal{M}|}^2_{ij\to\mathrm{Q}\overline{\mathrm{Q}}} \,. \end{aligned} \tag{4.50}$$

4.2 Numerical Results

At first, the integration region is validated. We compare the analytic expressions for the limits corresponding to p_T-distribution, eqs. (4.44) and (4.45), with the kinematically allowed region in fig. 3. The integration limits used in the y_3-distribution, eqs. (4.44) and (4.47), compared to the kinematically allowed region are shown in fig. 4.

More precisely, the points in figs. 3 and 4 are obtained by sampling 80^2 points from $[-3,3]^2$ and $[-5,5] \times [0, 500\,\mathrm{GeV}]$ for the p_T-distribution limits and the y_3-distribution limits, respectively. This is done by varying the mass of the heavy quark, i.e. in fig. 3 $m \in \{1\,\mathrm{GeV}, 100\,\mathrm{GeV}, 200\,\mathrm{GeV}\}$, and for each of these looking at 3 different p_T or y_3 values in the allowed regions. When we talk about the kinematically allowed regions, we mean transforming the generated points according to eq. (4.11) and only keeping the ones that satisfy

$$0 \leq x_1 \leq 1 \quad \text{and} \quad \frac{4m^2}{s\,x_1} \leq x_2 \leq 1 \,. \tag{4.51}$$

From figs. 3 and 4 it is apparent that the analytical integration regions correspond to the sampled regions, i.e. the integration limits are correct.

The numerically calculated p_T-distributions are shown in figs. 5 to 7 for a produced $\mathrm{c\bar{c}}$-, $\mathrm{b\bar{b}}$- and $\mathrm{t\bar{t}}$-pair, respectively. They were implemented in a C++ program from eq. (4.46), together with the integral transformation to the unit hypercube described in eq. (3.34). The `LHAPDF` [Buc+15] library was used with the `CT18NLO` set to get the PDF values.

The integration was done using the VEGAS function of the CUBA [Hah05] library, a Monte Carlo integration library written in C. The results were checked against the `hvq` [FNR07] process of the `POWHEG BOX V2` [Ali+10] program. p_T-distributions (and also y-distributions) were obtained from adapting a custom existing interface connecting the `POWHEG hvq` process to `Rivet` [Bie+20]. We implemented a custom `Rivet` analysis, binning the `POWHEG` events in p_T and y of the produced heavy quark (i.e. not the quantities of the antiquark). In this process, the interface was also extended to obtain fixed-order channel decompositions, which is not shown here since the distributions summed over the channels show good agreement. It should be noted, that instead of calculating the integral eq. (4.46) for one p_T-value at a time, the comparison to `POWHEG` facilitates also having to sample p_T over the bin intervals used in the `Rivet` analysis and then averaging over them afterward. `hvq` makes the scale choice of $\mu_r = \mu_f = E_T$, so this convention was also adopted in our implementation.

In addition to the absolute values of $\mathrm{d}\sigma/\mathrm{d}p_T$, the ratio to the `POWHEG` results are shown. It is apparent, that the two curves are compatible with each other within the uncertainties. To show this further, a two-sample z-test,

$$z = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\sigma_1^2 + \sigma_2^2}}, \tag{4.52}$$

is also plotted. It shows the difference of two means $\bar{X}_{1,2}$ with the standard errors $\sigma_{1,2}$ in units of standard deviations. The p_T-distributions show that our implementation is mostly in a 1σ-interval around `POWHEG`, with some values in a 2σ-interval. This can be taken as agreement between the distributions.

The shape of the `POWHEG` standard errors is noticeable, with the errors getting bigger for bigger p_T. This behavior could not be reproduced by our implementation, although the ratios seem to spread more for higher p_T. Running `POWHEG` with higher statistics could be tried to investigate this.

The y_3-distributions in figs. 8 to 10 are generated using the same method as the p_T-distributions. The y_3-distributions show the same level of agreement, but the `POWHEG` errors are much more uniformly distributed and expectedly lower for higher values of $\mathrm{d}\sigma/\mathrm{d}y_3$, since the `POWHEG` performs importance sampling in the integration region, leading to more samples and thus lower errors in the peak region.

At last, channel contributions are compared for the p_T- and y_3-distributions in figs. 11 and 12. It is apparent that the gg-channel dominates for lower p_T and then eventually the $q\bar{q}$ starts to dominate. For the y_3-distributions, the gg channels dominates in the plotted region, except for $t\bar{t}$-production. This may be connected to the y_3-region being bigger than is actually shown, and the $q\bar{q}$ increasing there. For $t\bar{t}$, the shown region is almost the whole kinematically allowed region.

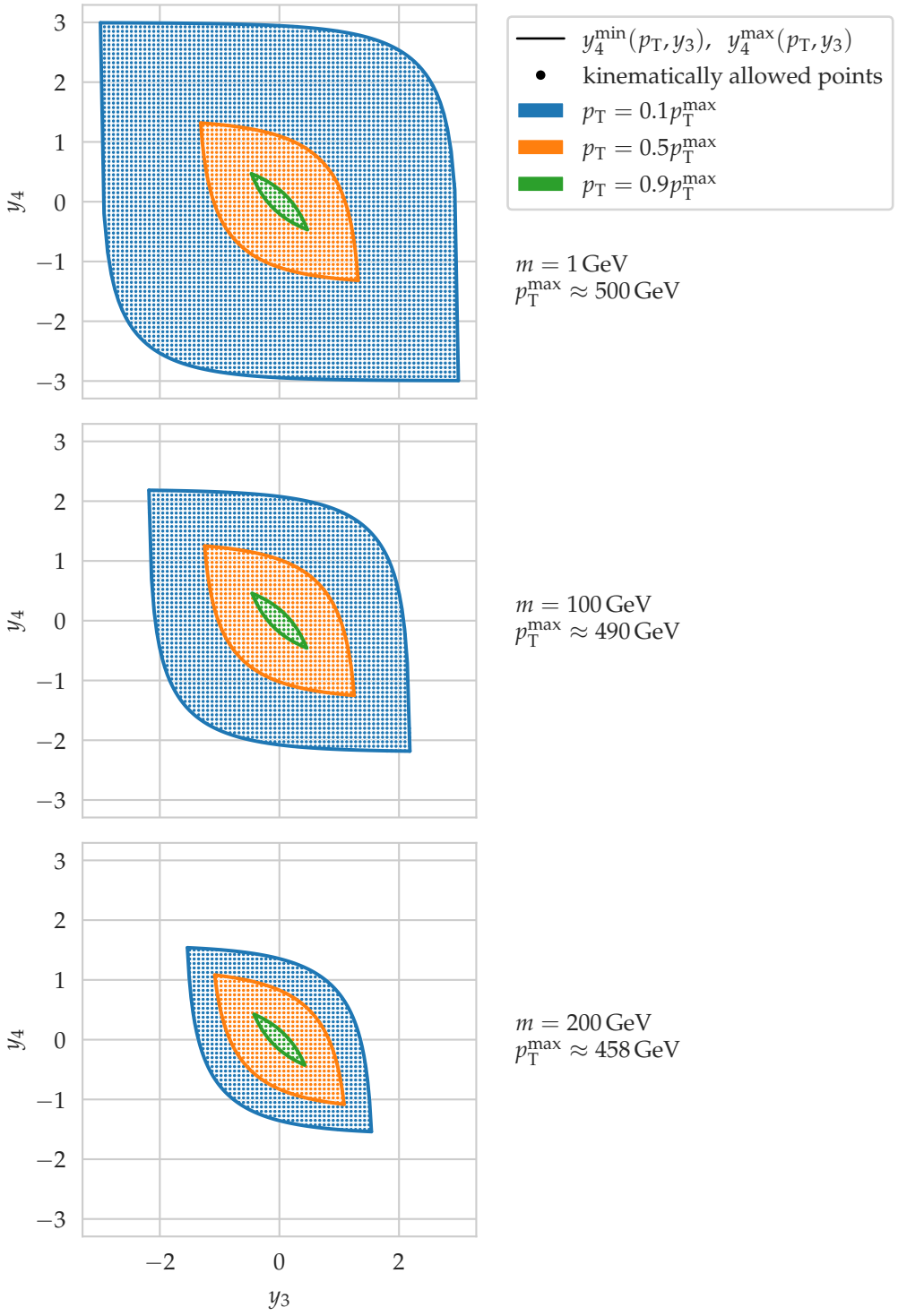


Figure 3: The integration region for integrating over y_3 and y_4, e.g. for $\mathrm{d}\sigma/\mathrm{d}p_T$ as in eq. (4.46). 80^2 points were sampled from $[-3,3]^2$, and only kinematically allowed ones were kept. The line plots indicate the integration limits calculated in eqs. (4.44) and (4.45). We set $\sqrt{s} = 1$ TeV, different values look qualitatively similar.

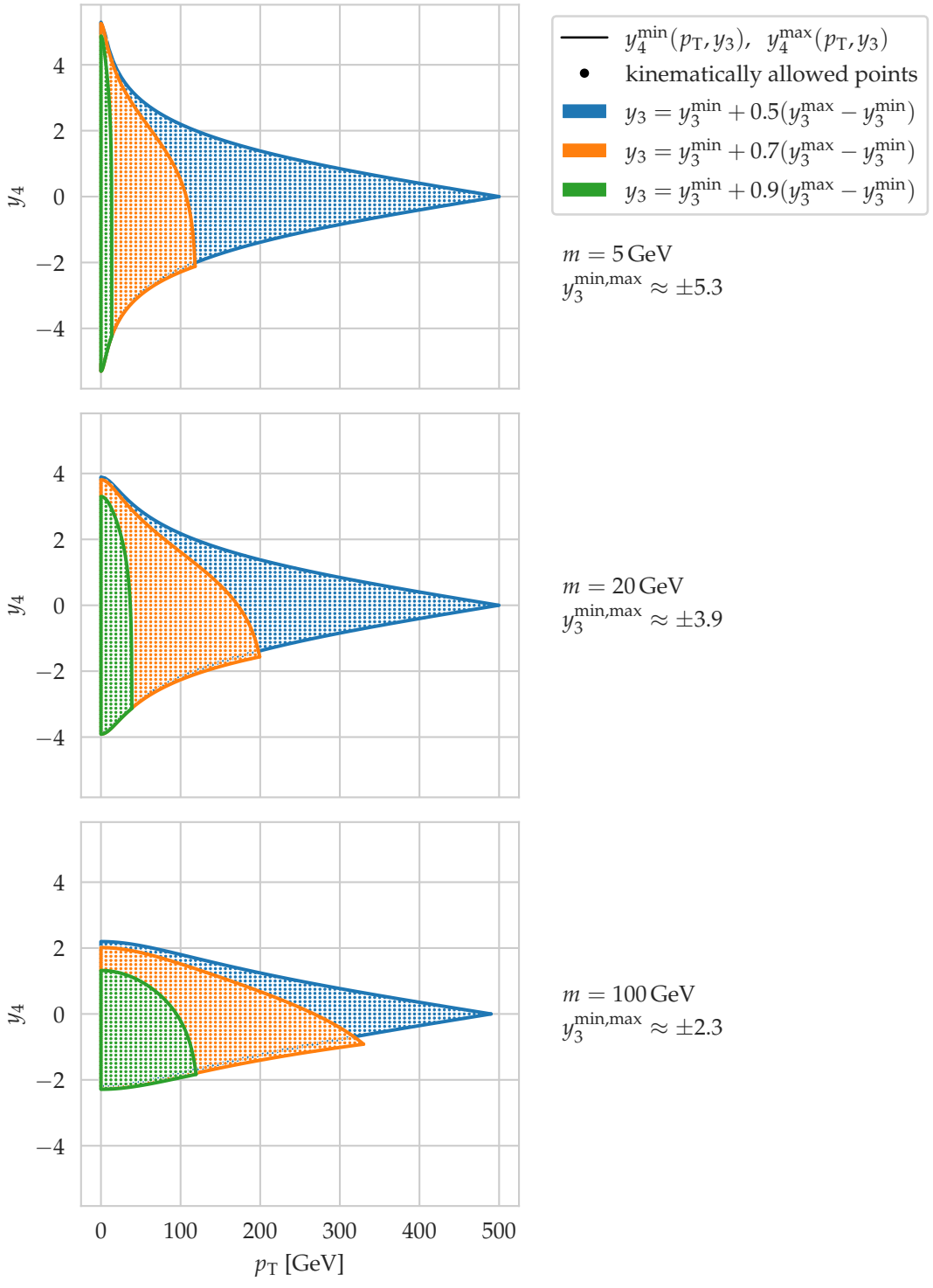


Figure 4: The integration region for integrating over p_T and y_4, e.g. for $d\sigma/dy_3$ as in eq. (4.48). As in fig. 3, 80^2 points were sampled from $[-5, 5] \times [0, 500\,\text{GeV}]$, and again only kinematically allowed ones were kept. The line plots indicate the integration limits calculated in eqs. (4.44) and (4.47). Again, we set $\sqrt{s} = 1\,\text{TeV}$, different values look qualitatively similar.

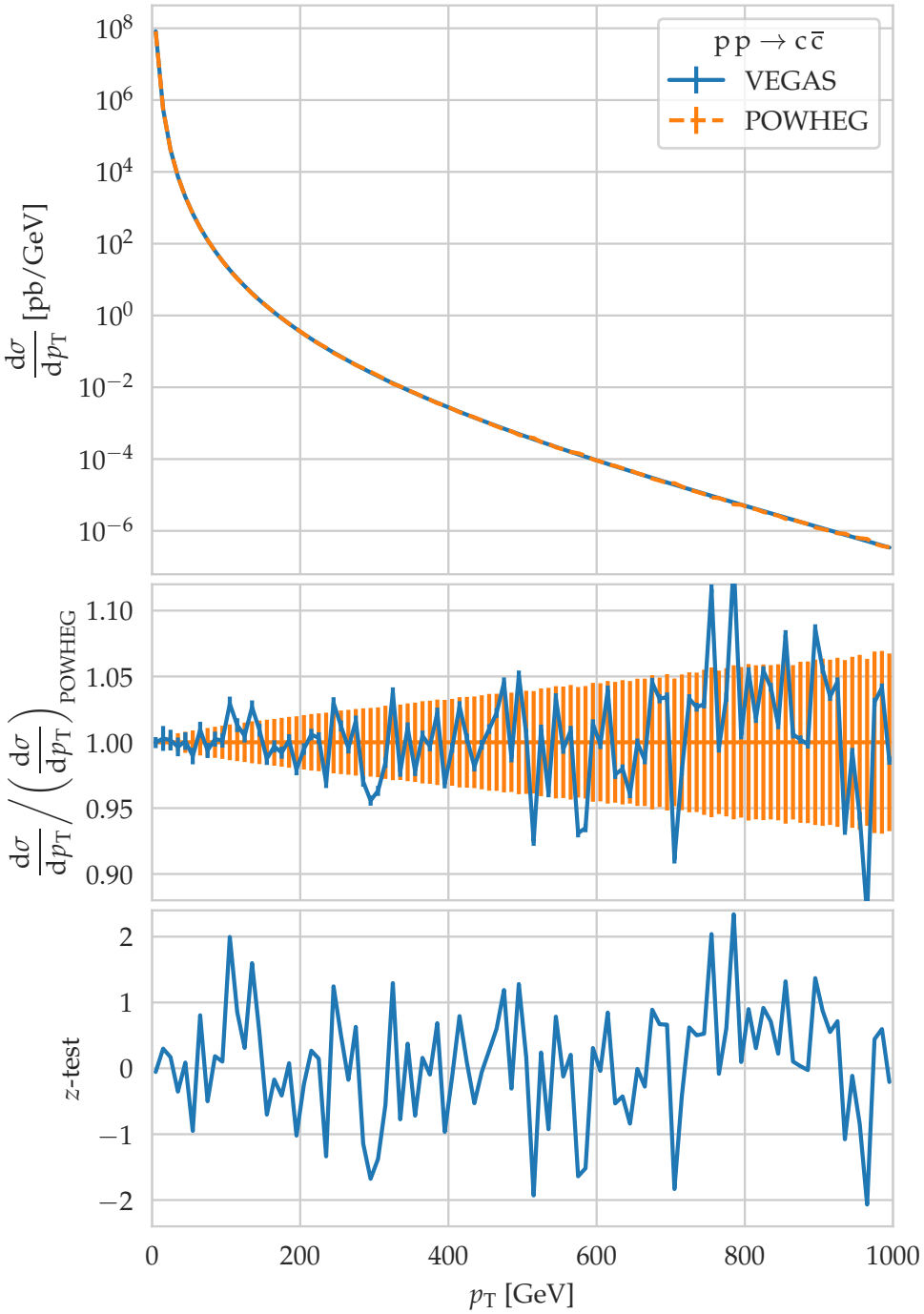


Figure 5: Comparison of the p_T-distribution of $c\bar{c}$-production between our calculation and the `hvq` process of `POWHEG`. We set $\sqrt{s} = 5\,\text{TeV}$ and $m_c = 1.27\,\text{GeV}$.

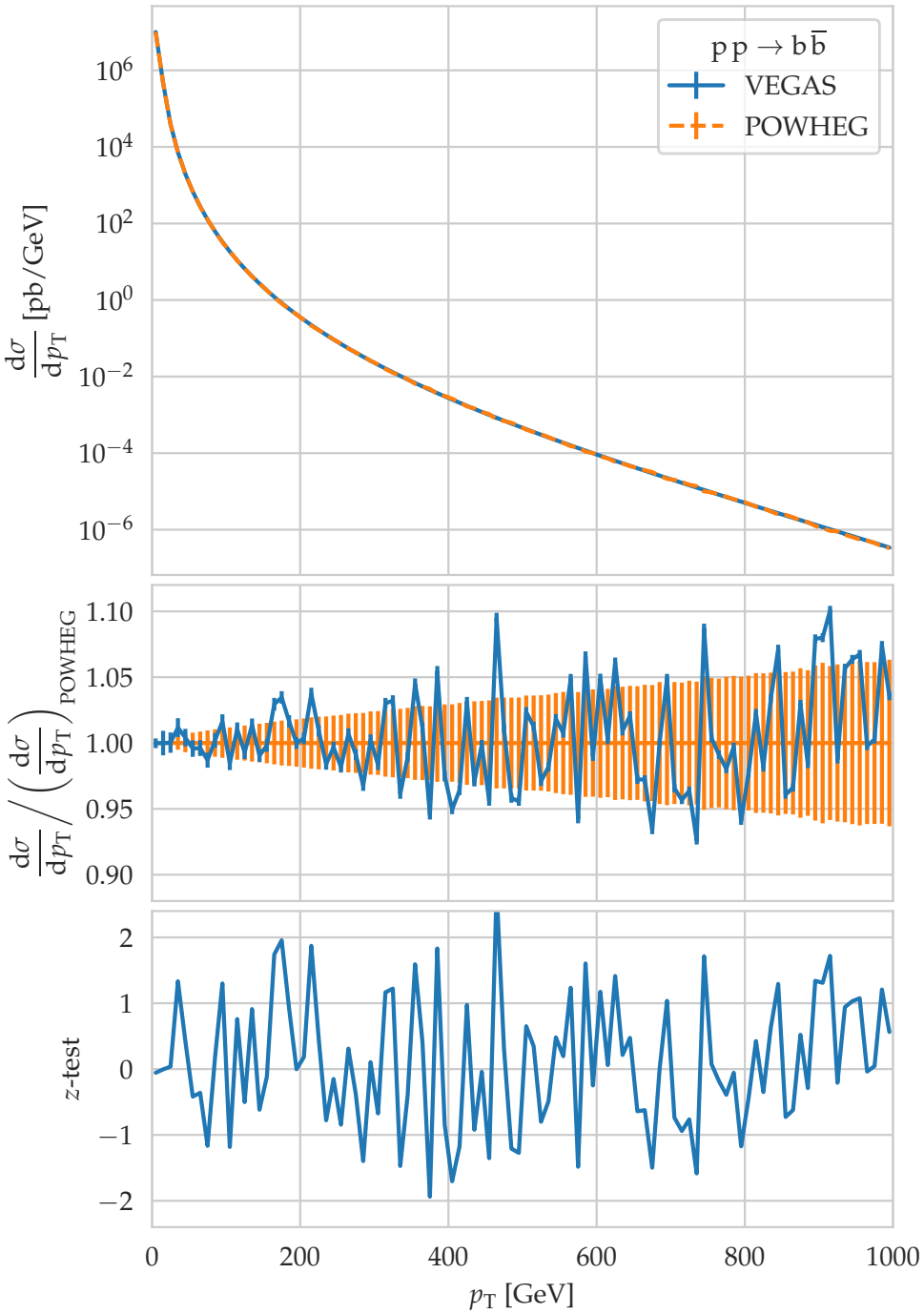


Figure 6: Comparison of the p_T-distribution of $b\bar{b}$-production between our calculation and the `hvq` process of `POWHEG`. We set $\sqrt{s} = 5\,\text{TeV}$ and $m_b = 4.18\,\text{GeV}$.

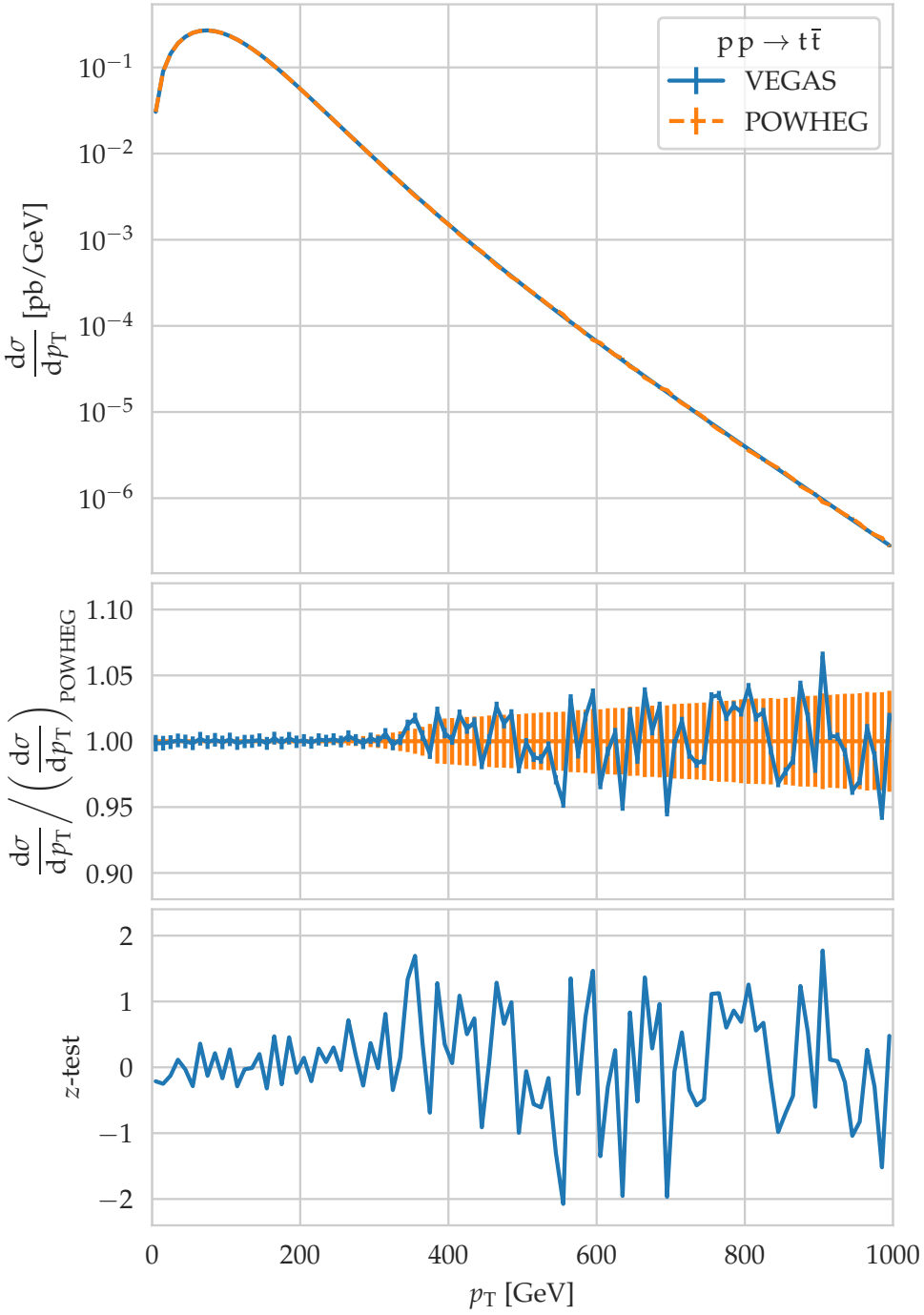


Figure 7: Comparison of the p_T-distribution of $\mathrm{t\bar{t}}$-production between our calculation and the `hvq` process of `POWHEG`. We set $\sqrt{s} = 5\,\mathrm{TeV}$ and $m_\mathrm{t} = 172.69\,\mathrm{GeV}$.

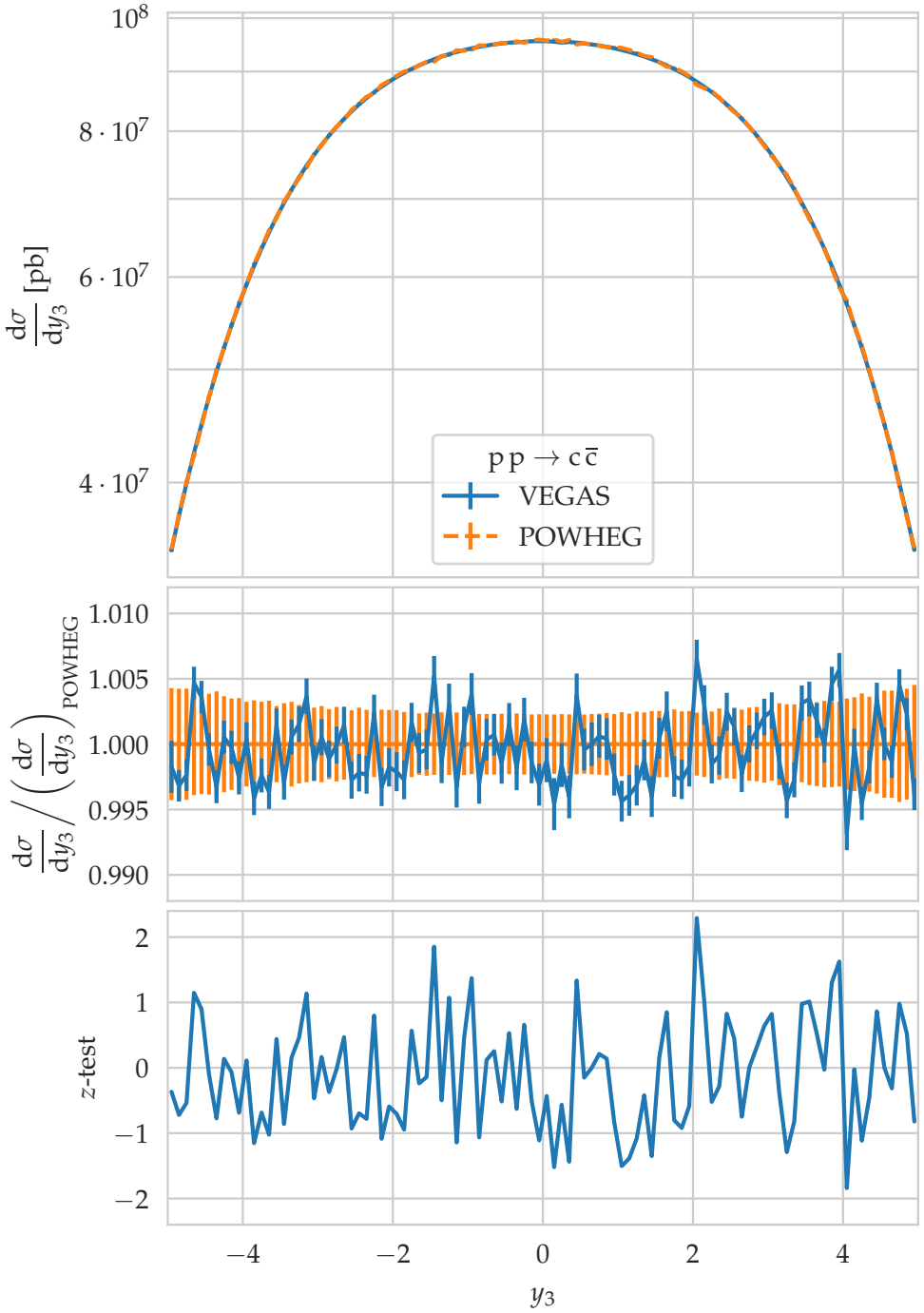


Figure 8: Comparison of the y-distribution of $c\bar{c}$-production between our calculation and the `hvq` process of `POWHEG`. We set $\sqrt{s} = 5\,\text{TeV}$ and $m_c = 1.27\,\text{GeV}$.

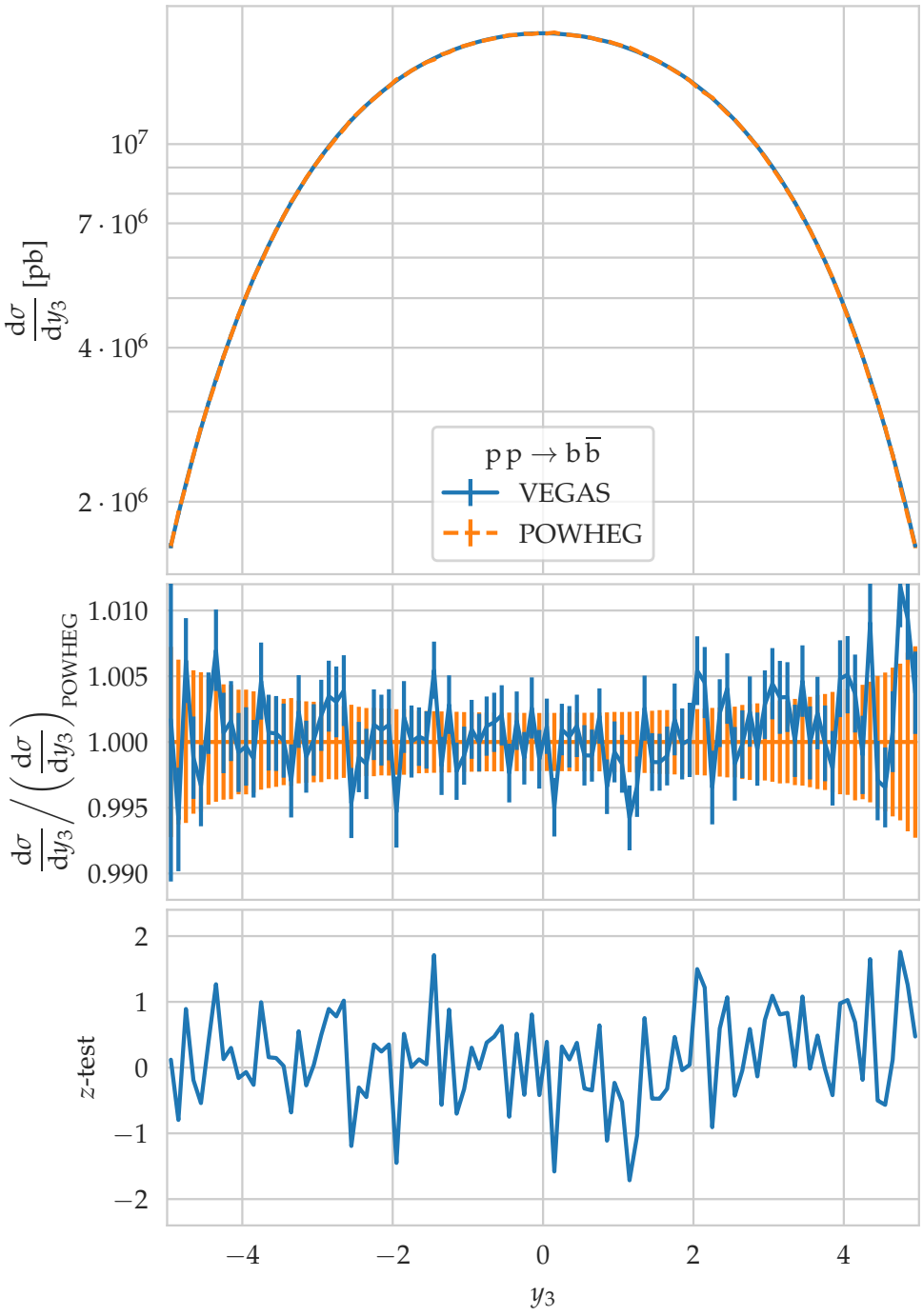


Figure 9: Comparison of the y-distribution of $b\bar{b}$-production between our calculation and the `hvq` process of `POWHEG`. We set $\sqrt{s} = 5\,\text{TeV}$ and $m_b = 4.18\,\text{GeV}$.

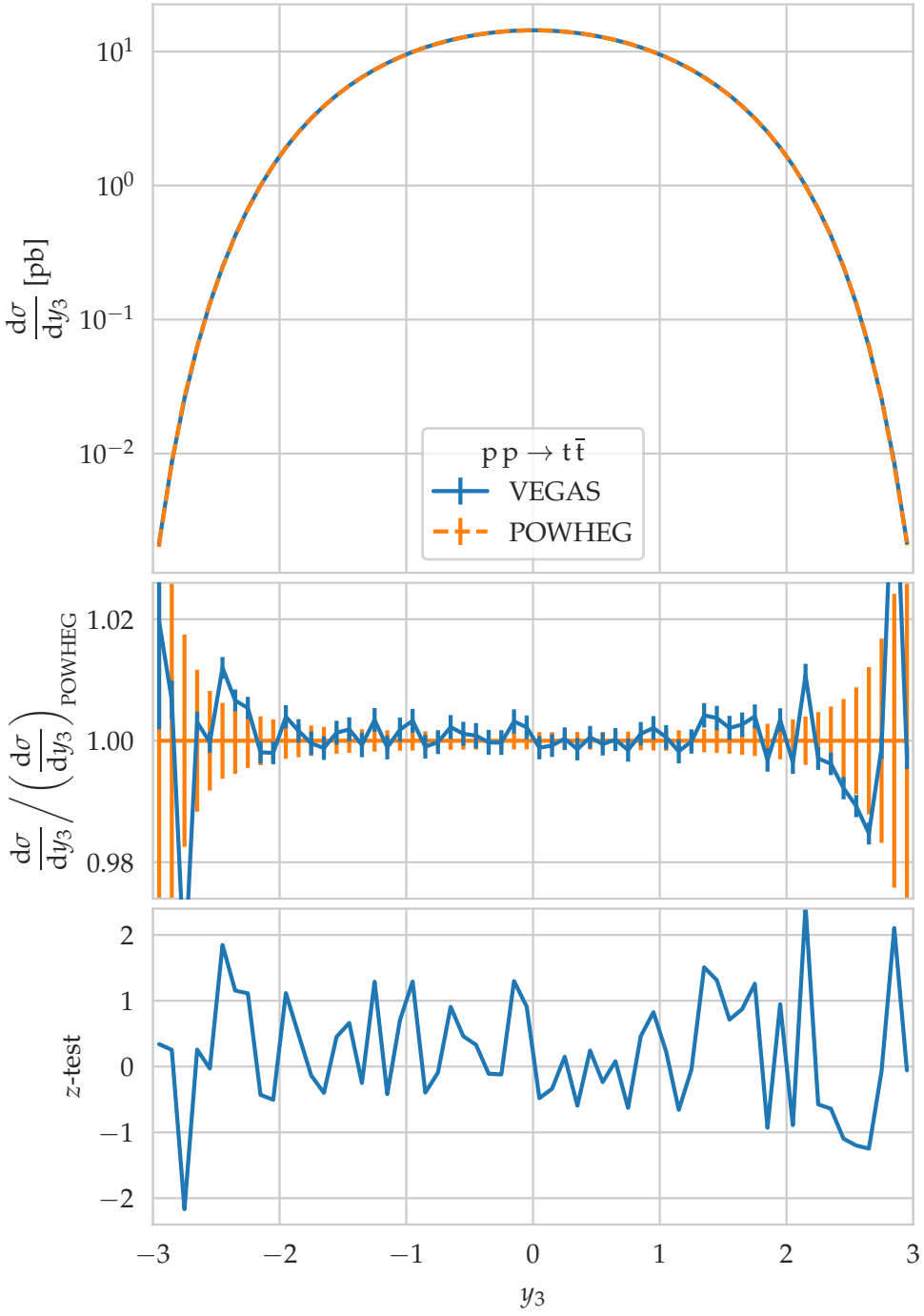
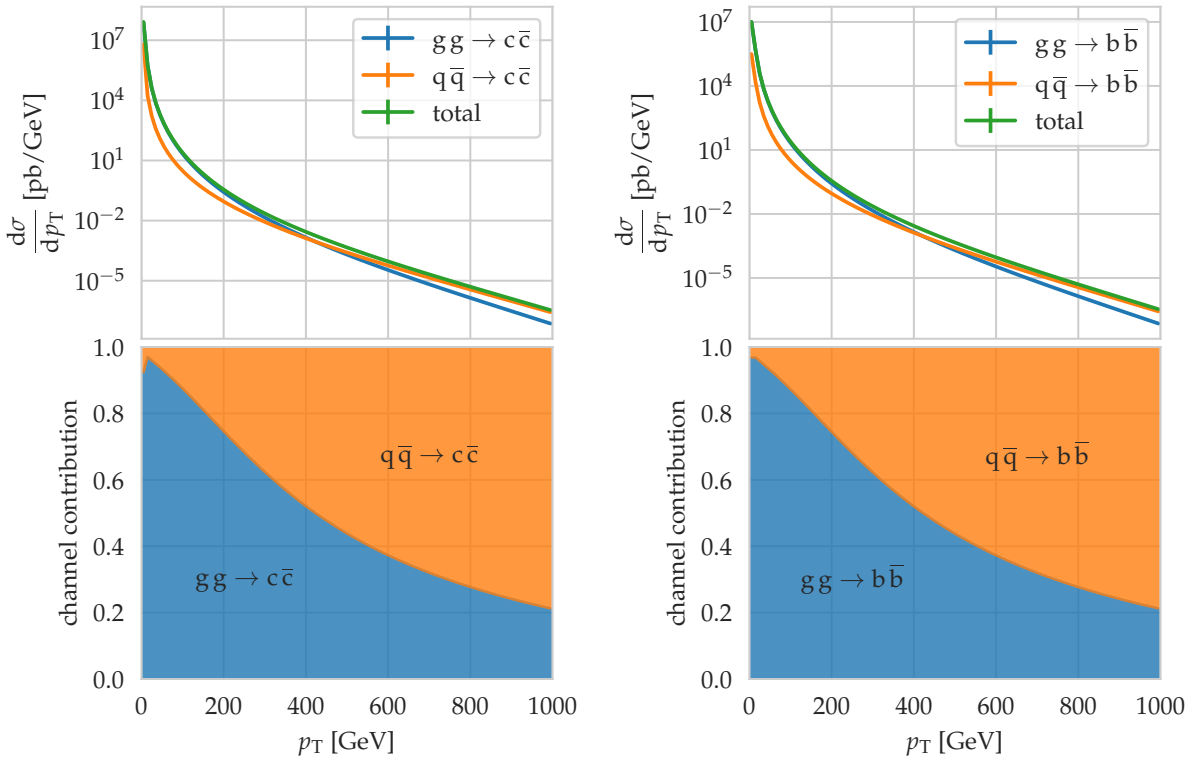
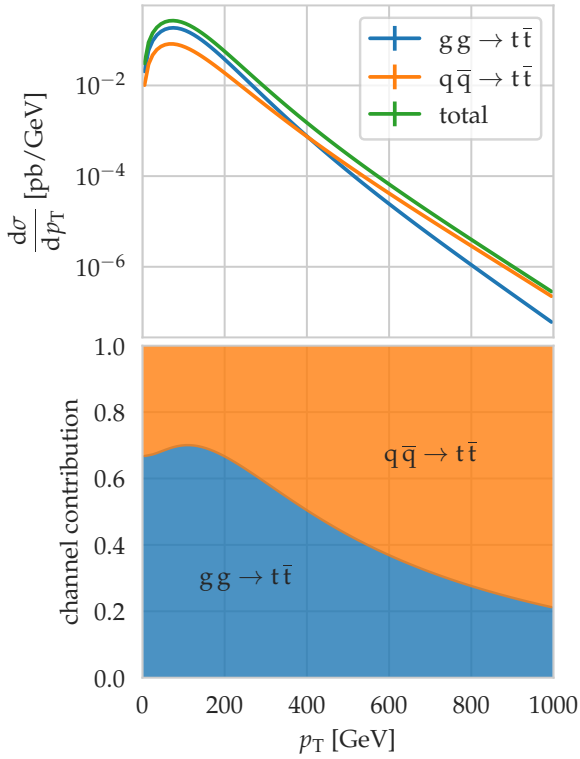


Figure 10: Comparison of the y-distribution of $t\bar{t}$-production between our calculation and the `hvq` process of `POWHEG`. We set $\sqrt{s} = 5\,\text{TeV}$ and $m_t = 172.69\,\text{GeV}$.



(a) (b)



(c)

Figure 11: Channel decomposition for the p_T-distributions at $\sqrt{s} = 5\,\text{TeV}$ for (a) $c\bar{c}$, (b) $b\bar{b}$ and (c) $t\bar{t}$-production.

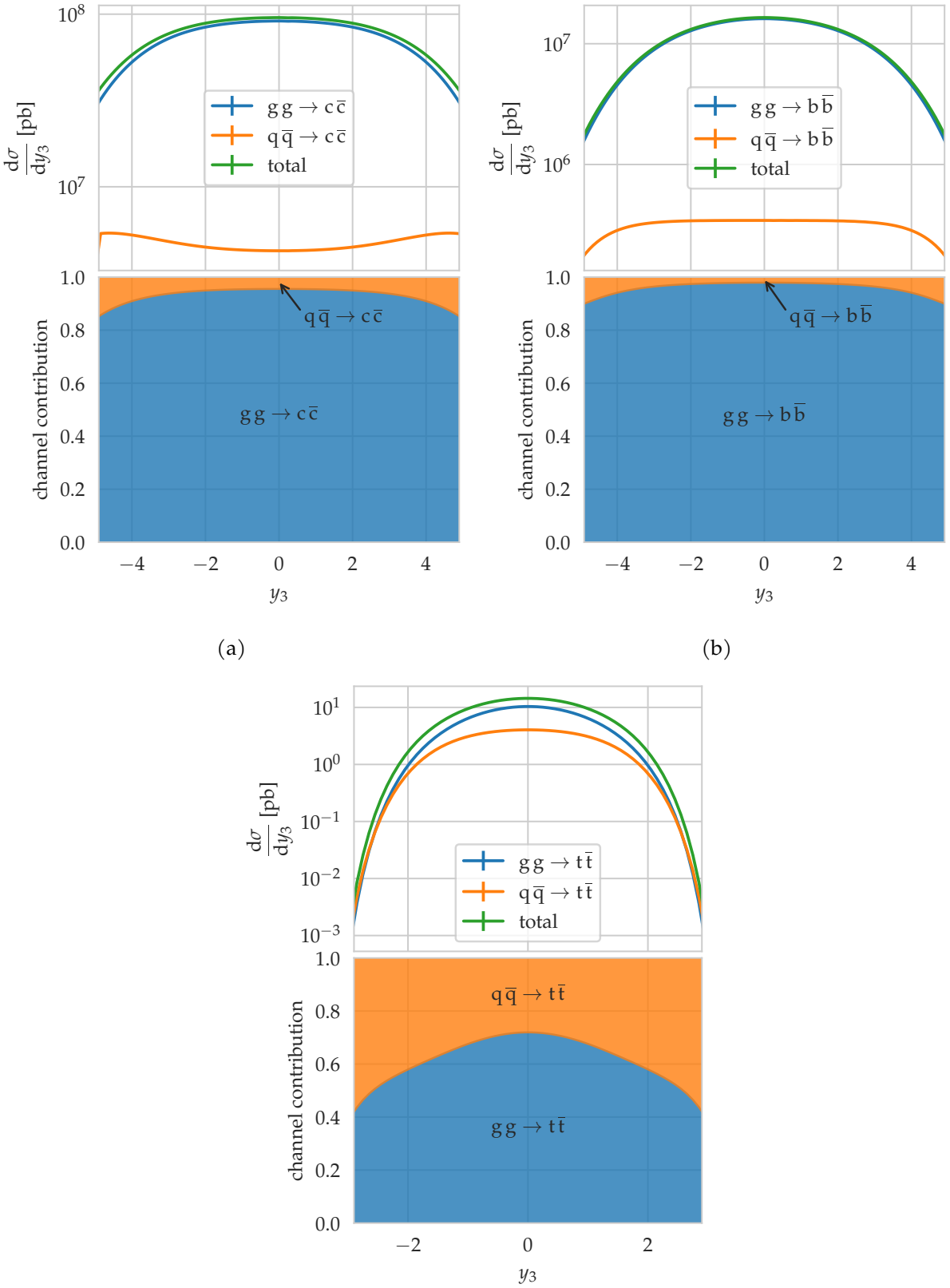


Figure 12: Channel decomposition for the y_3-distributions at $\sqrt{s} = 5\,\text{TeV}$ for (a) $c\bar{c}$, (b) $b\bar{b}$ and (c) $t\bar{t}$-production.

5 PDF Fitting

5.1 The nCTEQ15 PDF Fits

Hard scattering processes involving nuclei cannot use proton PDFs for theoretical predictions. One of the nuclear effects is called the EMC effect [Aub+83], named after the European Muon Collaboration, where it was found that the ratio of iron to deuterium Deep Inelastic Scattering (DIS) structure functions $F_2^{\mathrm{Fe}}/F_2^{\mathrm{D}}$ differed from the theoretical predictions at the time. These understood nuclei to be a collection of slowly moving nucleons weakly bound to each other, so the Deuterium corrections were transferred to the high A case. This means that the PDFs for nuclei are not simply given by the proton PDFs and have to be fitted separately. The nCTEQ collaboration then does just that. The fitting procedure is laid out in this section, following the nCTEQ15 paper [Kov+16], which is a global fit of the data, meaning that it includes data from a wide range of experiments.

5.1.1 Parametrization

The parametrizations are given at the input scale $Q_0 = 1.3\,\mathrm{GeV}$ as functions of the momentum fraction x, after the corresponding CTEQ parametrizations:

$$\begin{aligned} x f_i^{\mathrm{p}/A}(x,Q_0) &= c_0\, x^{c_1}\,(1-x)^{c_2}\, e^{c_3 x}\,(1+c_4 x)^{c_5}\,, \\ \left(\frac{\bar{\mathrm{d}}}{\bar{\mathrm{u}}}\right)^{\mathrm{p}/A}(x,Q_0) &= c_0\, x^{c_1}\,(1-x)^{c_2} + (1+c_3 x)\,(1-x)^{c_4}\,. \end{aligned} \tag{5.1}$$

Here, $f_i^{\mathrm{p}/A}$ is the PDF of flavor or observable $i \in \{\mathrm{u_v} = \mathrm{u}-\bar{\mathrm{u}}, \mathrm{d_v} = \mathrm{d}-\bar{\mathrm{d}}, \mathrm{g}, \bar{\mathrm{u}}+\bar{\mathrm{d}}, \mathrm{s}+\bar{\mathrm{s}}, \mathrm{s}-\bar{\mathrm{s}}\}$ in a proton, bound in a nucleus of mass number A. The notation $f_{\mathrm{u}}(x,Q) = \mathrm{u}(x,Q)$ is used interchangeably, and in the same way for the other observables.

The CTEQ parametrizations of eq. (5.1) are then modified to include the A-dependence in the fit parameters c_k, where $k \in \{1,\ldots,5\}$:

$$c_k \to c_k(A) = c_{k,0} + c_{k,1}\,(1 - A^{-c_{k,2}})\,. \tag{5.2}$$

$c_k(A=1)$ recovers the parametrization of a free proton ("proton baseline"), which is put into the fit, and for nCTEQ15 is given by [Owe+07].

Not all of the c_0 parameters are free, but are instead constrained by the sum rules for each value of A:

$$\int_0^1 \mathrm{u_v}^{\mathrm{p}/A}(x,Q_0)\,\mathrm{d}x = 2\,, \tag{5.3}$$

$$\int_0^1 \mathrm{d_v}^{\mathrm{p}/A}(x,Q_0)\,\mathrm{d}x = 1\,, \tag{5.4}$$

$$\int_0^1 \sum_i x f_i^{\mathrm{p}/A}(x,Q_0)\,\mathrm{d}x = 1\,, \tag{5.5}$$

where eqs. (5.3) and (5.4) are the number sum rules for the valence quarks, and eq. (5.5) is the momentum sum rule, where $i \in \{\mathrm{u_v}, \mathrm{d_v}\}$.

The gluon momentum fraction is parameterized as

$$\int_0^1 x\, \mathrm{g}^{\mathrm{p}/A}(x, Q_0)\, \mathrm{d}x = M_\mathrm{g} \exp\Big(c^\mathrm{g}_{0,0} + c^\mathrm{g}_{0,1}(1 - A^{-c^\mathrm{g}_{0,2}})\Big)\,, \tag{5.6}$$

where M_g and $c^\mathrm{g}_{0,0}$ are again parameters of the proton fit.

Due to insufficient data at the time, the strange quark parametrization is different from the ones in eq. (5.1), and given by $\bar{\mathrm{u}}$ and $\bar{\mathrm{d}}$ instead:

$$\mathrm{s}^{\mathrm{p}/A}(x, Q_0) = \bar{\mathrm{s}}^{\mathrm{p}/A}(x, Q_0) = \frac{\kappa(A)}{2}(\bar{\mathrm{u}} + \bar{\mathrm{d}})^{\mathrm{p}/A}(x, Q_0)\,, \tag{5.7}$$

with $\kappa(A)$ given by

$$\kappa(A) = c^{\mathrm{s}+\bar{\mathrm{s}}}_{0,0} + c^{\mathrm{s}+\bar{\mathrm{s}}}_{0,1}(1 - A^{-c^{\mathrm{s}+\bar{\mathrm{s}}}_{0,2}})\,. \tag{5.8}$$

The momentum sum rule of $\mathrm{s} + \bar{\mathrm{s}}$ is then

$$\int_0^1 x(\mathrm{s} + \bar{\mathrm{s}})^{\mathrm{p}/A}(x, Q_0) = \frac{\kappa}{2 + \kappa}\Big(1 - \int_0^1 \sum_i x f_i^{\mathrm{p}/A}\, \mathrm{d}x\Big)\Big(c^{\mathrm{s}+\bar{\mathrm{s}}}_{0,0} + c^{\mathrm{s}+\bar{\mathrm{s}}}_{0,1}(1 - A^{-c^{\mathrm{s}+\bar{\mathrm{s}}}_{0,2}})\Big)\,, \tag{5.9}$$

where $i \in \{\mathrm{u_v}, \mathrm{d_v}, \mathrm{g}\}$.

To get the full nuclear PDFs for a nucleus with mass number A and charge number Z, its bound proton and neutron PDFs are averaged, where they are weighted with the number of protons Z and neutrons $A - Z$:

$$f_i^{(Z,A)}(x, Q) = \frac{Z}{A} f_i^{\mathrm{p}/A}(x, Q) + \frac{A - Z}{A} f_i^{\mathrm{n}/A}(x, Q)\,. \tag{5.10}$$

To obtain $f_i^{\mathrm{n}/A}$ from $f_i^{\mathrm{p}/A}$, isospin symmetry is assumed, which means that one has to perform the isospin symmetry transformation of exchanging up- and down-quarks, as well as their antiquarks.

16 of the parameters are actually fitted (see [Kov+16, tab. V]), since the number of data points is not sufficient for opening more parameters. The rest of the points are kept at fixed values.

5.2 χ^2 for PDF Fitting

The global χ^2 function for the nCTEQ15 fit is a weighted sum of the individual experiments:

$$\chi^2(\vec{x}) = \sum_n w_n \chi^2_n(\vec{x})\,, \tag{5.11}$$

where the sum goes over the experiments, $\vec{x}$ is the vector of fit parameters and w_n is the weight of the experiment n. The default weight would be 1, excluded data sets get the weight 0, and emphasis on different experiments is possible with values between 0 and 1. The naive ansatz for the χ^2 functions for each experiment would be (cf. eq. (3.17))

$$\chi^2_n(\vec{x}) = \sum_{i=1}^{N_n} \frac{(D_i - T_i(\vec{x}))^2}{\alpha_i^2}\,, \tag{5.12}$$

with N_n the number of data points of experiment n, D_i the i'th data point of experiment n, T_i its respective theory prediction and $\alpha_i^2 = \sigma_i^2 + u_i^2$ the statistical uncertainties and the uncorrelated systematic uncertainties of D_i, respectively, added in quadrature. We denote the theory prediction with a dependence on the fit parameters $\vec{x}$ because these depend on the PDF model.

Since eq. (5.12) neglects the correlated systematic uncertainties of the experiments, we turn to the ansatz shown in eq. (3.19):

$$\chi_n^2(\vec{x}) = \sum_{i,j=1}^{N_n} (D_i - T_i(\vec{x}))\, V_{ij}^{-1}\, (D_j - T_j(\vec{x}))\,, \tag{5.13}$$

with the covariance matrix V:

$$V_{ij} = \alpha_i^2\, \delta_{ij} + \sum_{k=1}^{K} \beta_{ki}\beta_{kj}\,. \tag{5.14}$$

We denote the correlated systematic uncertainty of a data point $i \in \{1, ..., N_n\}$ due to a source $k \in \{1, ..., K\}$ for this uncertainty by β_{ki}. Using the Woodbury matrix identity, we can formulate the inverse of the covariance matrix, and thus the χ^2 function, as

$$\chi_n^2(\vec{x}) = \sum_{i=1}^{N_n} \frac{(D_i - T_i(\vec{x}))^2}{\alpha_i^2} - \sum_{k,k'=1}^{K} B_k(\vec{x})\, A_{kk'}^{-1}\, B_{k'}(\vec{x})\,, \tag{5.15}$$

where we use the abbreviations

$$A_{kk'} = \delta_{kk'} + \sum_{i=1}^{N_n} \frac{\beta_{ki}\,\beta_{k'i}}{\alpha_i^2} \qquad \text{and} \qquad B_k(\vec{x}) = \sum_{i=1}^{N_n} \frac{\beta_{ki}\,(D_i - T_i(\vec{x}))}{\alpha_i^2}\,. \tag{5.16}$$

This simplifies the calculation of the χ^2 function, since now not the $N_n \times N_n$ covariance matrix has to be inverted, but the $K \times K$ matrix A. The global χ^2 function eq. (5.11) together with eq. (5.15) is then minimized to estimate the PDF parameters $\vec{x}$.

5.3 Hessian Method

The Hessian method is a way to estimate the uncertainties of the resulting central values of the PDFs. Instead of using confidence intervals, as we would do for a 1D random variable, we use confidence regions. If the χ^2 function is a quadratic function of the parameters, the confidence regions turn out to be ellipsoids. Their shape can be conveniently described by the two principal axes, which we can then use to define the PDF uncertainties. This section is based on the description in the nCTEQ15 paper [Kov+16] and on [Pum+01].

The Hessian method starts with expanding the χ^2 function around its minimum $\chi_0^2 = \chi^2(\vec{x}_0)$ to second order. This is a reasonable assumption for the nCTEQ15 fit, as seen in

[Kov+16, figs. 26-27]. The expansion is given by

$$\begin{aligned}\chi^2(\vec{x}) &\approx \chi_0^2 + (\vec{x} - \vec{x}_0)^{\mathrm{T}} H (\vec{x} - \vec{x}_0) \\ &=: \chi_0^2 + \vec{y}^{\mathrm{T}} H \vec{y} \\ &=: \chi_0^2 + \Delta\chi^2(\vec{x}) \,,\end{aligned} \tag{5.17}$$

where H is the $d \times d$-dimensional Hessian matrix occurring through the Taylor expansion, with d the size of $\vec{x}$, i.e. the number of fit parameters. It is given by

$$H_{ij} = \left.\frac{\partial^2 \chi^2}{\partial y_i \, \partial y_j}\right|_{\vec{x}=\vec{x}_0} . \tag{5.18}$$

We define two abbreviations in eq. (5.17): $\vec{y}$ as the distance from the minimum of the χ^2 function in parameter space, and $\Delta\chi^2(\vec{x})$ as the lines of constant χ^2 in an ellipsoid, defined by $\Delta\chi^2(\vec{x}) = \text{const}$. This comes about since $\vec{y}^{\mathrm{T}} H \vec{y} = \text{const}$. is precisely the defining equation for a d-dimensional ellipsoid, if H is real, symmetric and positive semi-definite. The ellipsoid has the nice property that the eigenvectors $\vec{v}_k$ of H give its principal axes, and $\sqrt{\lambda_k}^{-1}$ of its corresponding eigenvalues λ_k are the length of the semi-axes.

The eigenvalue equation looks as follows:

$$H \vec{v}_k = \lambda_k \vec{v}_k \,. \tag{5.19}$$

Since H is real and symmetric, its eigenvectors are mutually orthogonal and its eigenvalues are real; and since H is positive semi-definite, the eigenvalues are non-negative, i.e. $\lambda_k \geq 0$. As only the eigenvector direction is uniquely determined, we normalize them, such that they are orthonormal:

$$\vec{v}_k \cdot \vec{v}_l = \delta_{kl} \,. \tag{5.20}$$

We now want to transform the ellipsoid into a d-sphere, which is achieved by expressing the coordinates in terms of a rescaled eigenvector basis, defined by

$$\tilde{\vec{v}}_k = \frac{1}{\sqrt{\lambda_k}} \vec{v}_k \,. \tag{5.21}$$

These are the vectors along the principle axes of the ellipsoid $\Delta\chi^2(\vec{y}) = 1$. We adopt the convention for the change-of-basis matrix $\widetilde{D}$ analogous to the one in [Kov+16, eq. (2.23)]:

$$\widetilde{D} = \begin{pmatrix} | & & | \\ \tilde{\vec{v}}_1 & \cdots & \tilde{\vec{v}}_d \\ | & & | \end{pmatrix} . \tag{5.22}$$

Since $\widetilde{D}$ contains the new basis expressed as coordinate vectors in terms of the old basis, applying it transforms coordinate vectors from the new basis into the old basis. Therefore,

we use the inverse $\widetilde{D}^{-1}$

$$\widetilde{D}^{-1} = \begin{pmatrix} - & \lambda_1 \vec{\tilde{v}}_1^{\,\mathrm{T}} & - \\ & \vdots & \\ - & \lambda_d \vec{\tilde{v}}_d^{\,\mathrm{T}} & - \end{pmatrix}, \tag{5.23}$$

to transform the old coordinates $\vec{y}$ into the new coordinates $\vec{\tilde{z}}$ (again, using the notation of [Kov+16]):

$$\vec{\tilde{z}} = \widetilde{D}^{-1}\, \vec{y}\,. \tag{5.24}$$

In these coordinates, $\Delta\chi^2(\vec{\tilde{z}})$ reduces then to a simple form, namely the radius of the d-sphere for $\Delta\chi^2(\vec{\tilde{z}}) = \text{const.}$. For this we first show that $\widetilde{D}^{\mathrm{T}} H \widetilde{D} = \mathbb{1}$:

$$(\widetilde{D}^T H \widetilde{D})_{ij} \overset{(5.22)}{=} \sum_{k,l=1}^{d} \tilde{v}_{ik} H_{kl} \tilde{v}_{jl} \overset{(5.19)}{=} \lambda_j \sum_{k=1}^{d} \tilde{v}_{ik} \tilde{v}_{jk} \overset{(5.21)}{=} \sqrt{\frac{\lambda_j}{\lambda_i}}\, \vec{v}_i \cdot \vec{v}_j = \delta_{ij}\,. \tag{5.25}$$

where $\widetilde{D}_{ij} = (\vec{\tilde{v}}_j)_i = \tilde{v}_{ji}$ according to eq. (5.22). The $\Delta\chi^2$ function then reads

$$\Delta\chi^2(\vec{\tilde{z}}) = \vec{y}^{\mathrm{T}} H \vec{y} = \vec{\tilde{z}}^{\mathrm{T}} \widetilde{D}^T H \widetilde{D}\, \vec{\tilde{z}} \overset{(5.25)}{=} \vec{\tilde{z}}^{\,2}\,. \tag{5.26}$$

We then define a tolerance T (just called $\Delta\chi^2$ in [Kov+16]), acting as the radius for the d-sphere that will be translated into the PDF uncertainty bands, also called "error PDFs". In this way, we define d parameter variation vectors for each of the parameters $\tilde{z}_k$:

$$\Delta\vec{\tilde{z}}_k = T\, \vec{e}_k\,, \tag{5.27}$$

where $\vec{e}_k$ is the k'th unit vector. These can then be translated back into the original coordinates $\vec{x}$, with $\Delta\vec{x}^{(k)}$ the parameter variation in the k'th eigendirection:

$$\Delta\vec{x}^{(k)} = \Delta\vec{y}^{(k)} = \widetilde{D}\, \Delta\vec{\tilde{z}}_k = \frac{T}{\sqrt{\lambda_k}} \vec{v}_k\,. \tag{5.28}$$

From these we define the eigenvector PDFs $f_k^{\pm}$:

$$f_k^{\pm} = f(\vec{x}_0 \pm \Delta\vec{x}^{(k)})\,, \tag{5.29}$$

which there are $2d$ of, since there are d parameters. These are then distributed together with the central value PDF via tools like `LHAPDF`. The actual uncertainties on an observable O, e.g. a structure-function, can be obtained by calculating the observable for each of the $2d$ eigenvector PDFs and then using the master formula

$$\Delta O = \frac{1}{2} \sqrt{\sum_{k=1}^{d} \left(O(f_k^+) - O(f_k^-) \right)^2}\,, \tag{5.30}$$

which is used by the nCTEQ collaboration for the Hessian method. It should be noted that different choices exist for the definition of the eigenvector PDFs, e.g. using Lagrange multipliers, Monte Carlo replicas, or recently, Markov Chain Monte Carlo methods.

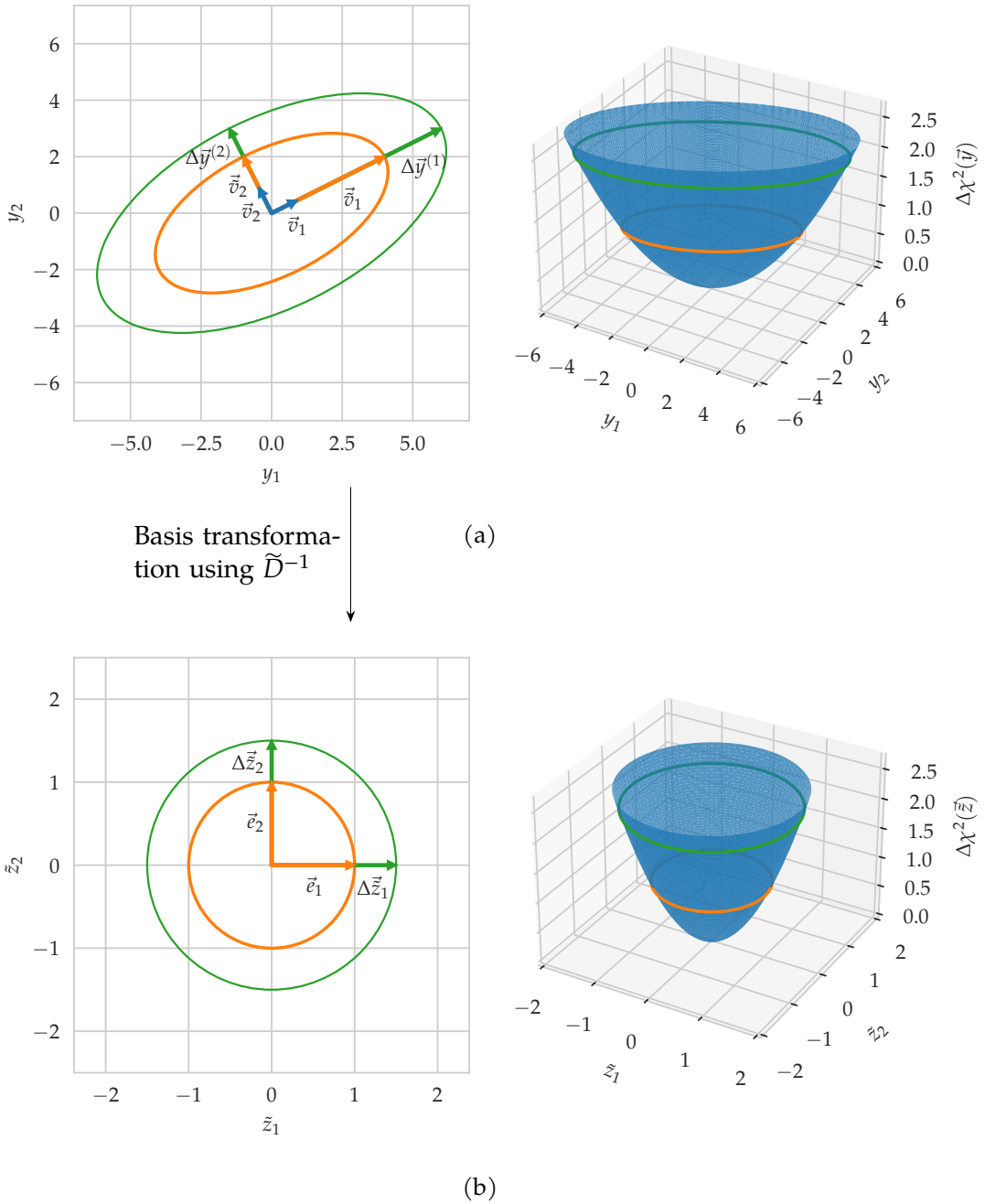


Figure 13: Visualisation of the basis change from $\vec{y}$ to $\vec{\tilde{z}}$, here for two parameters y_1 and y_2. In (a) we show the χ^2 function and its contours before the transformation, and in (b) we show them afterward. The noted symbols are the normalized eigenvectors of the Hessian matrix $\vec{v}_1$ and $\vec{v}_2$, the rescaled ones $\vec{\tilde{v}}_1$ and $\vec{\tilde{v}}_2$, the standard basis $\vec{e}_1$ and $\vec{e}_2$, onto which $\vec{\tilde{v}}_1$ and $\vec{\tilde{v}}_2$ are mapped, and the parameter variations $\Delta\vec{y}^{(1)}$ and $\Delta\vec{y}^{(2)}$, as well as, after the transformation, $\Delta\vec{\tilde{z}}_1$ and $\Delta\vec{\tilde{z}}_2$. The variations are shown for an exemplary $T = 1.5$.

6 Developments in the Context of nCTEQ++

In this section, we present the code developments in the context of the nCTEQ++ framework that were done during this thesis. As PDF fitting is an inherently computational problem, we need a way to translate the methods presented in section 5 into code. This task is achieved by the nCTEQ++ framework, which is an application written in C++. It takes care of e.g. combining all the different theoretical predictions (supplied by external codes often written in Fortran) with the corresponding data, minimizing the χ^2 function, the error analysis using the Hessian method, parameter scans, output for LHAPDF, etc. The following sections describe containerization of the nCTEQ++ dependencies using Apptainer, the pdfplotter tool for plotting observables depending on PDFs and efforts to parallelize parts of nCTEQ++ using the Message Passing Interface (MPI).

6.1 Apptainer

Apptainer [App] is an application aiding in the creation and usage of so-called containers, which are a way to package software and its dependencies. This means a way of circumventing the necessity to install the dependencies of a program on every system where the program is used. Instead, the dependencies and maybe the program itself are installed once in a container, and can then be distributed in a portable and reproducible way to workstations, computing clusters, etc. Only the Apptainer library itself has to be installed on these systems, which makes it a good fit for High Performance Computing (HPC) systems, where the software environment is usually very restrictive, and new software can only be installed by administrators.

In the case of Apptainer, containers are a single file and are run with a command-line interface (CLI). We will present the basic usage of a container given as an Singularity Image Format (SIF) file here called container.sif. We assume it is used for running an executable called program, i.e. the container contains the dependencies for program, but not program itself, as is the case for nCTEQ++. The presented commands can be found in the Apptainer documentation [App].

```
apptainer exec container.sif program
```

runs program inside the container. Instead of program, any other command can be used. program then has access to all the libraries installed in container.sif.

```
apptainer shell container.sif
```

opens a shell inside a container, i.e. is the interactive version of exec.

```
apptainer run container.sif
```

is used if the container already includes the executable, i.e. program is already installed in container.sif. run then executes an executable inside the container that was specified with a so-called runscript at the stage of building the container.

```
apptainer build container.sif container.def
```

builds a container file named `container.sif` from a definition file named `container.def`. A definition file is basically a recipe on how the container should be built. We present the basic anatomy of a definition file used for building an image for `nCTEQ++` in the following:

```
Bootstrap: docker
From: ubuntu:22.04
Stage: build

%files
...

%environment
...

%post
...
apt-get update && apt-get install -y ...
...

%labels
Author ...
Version 1

%help
An example container.
```

The three lines at the beginning form the *header*, and the lines prepended by the % sign give the *sections* of the definition file. The `Bootstrap` keyword supplies the bootstrap agent, i.e. where the base image that this container is built upon comes from, or if it should be built from scratch. `Bootstrap: docker` will pull the base image from the Docker Hub, whose name is given by the `From` option. In this example and, at the time of writing, the base image for `nCTEQ++` is Ubuntu 22.04.

The first section we use here is `%files`. It copies a file from the host system into the container. This can be used e.g. if you have a library only locally on your machine or if you want to copy the source code of your program into the container. The `%environment` section is used for setting environment variables which are then available at runtime in the container. The actual installation steps happen in `post`: Here you can download and install software, create directories, etc. In this example the Ubuntu package manager is used for this purpose, but libraries can also downloaded from the internet directly, e.g. using `wget`. The `%labels` section is used for supplying metadata about the container, e.g. the author or its version. At last, the `%help` section can be used to give a short description of the container and to provide information on how to run it, which is why it is displayed when running `apptainer run-help container.sif` on a built container.

In the context of this work, I wrote such a recipe for building a container to use with

nCTEQ++. It is now used to run the nCTEQ++ code on workstations and the local HPC cluster called Palma-II.

6.2 pdfplotter

pdfplotter is a tool for plotting PDFs from LHAPDF. It automates the need to calculate the error PDFs for each observable that you want to plot, which before had to be done manually for each new observable. It is written in Python and uses the LHAPDF Python wrapper to interact with LHAPDF, uses sympy [Meu+17] Symbols for the representation of the PDF flavors and stores its underlying data in pandas [McK10] DataFrames. The basic usage of pdfplotter and its inner workings are laid out in the following.

At the time of writing, pdfplotter does not supply actual plotting functions yet. But the core functionality is to be a wrapper around lhapdf.PDFSet. First, you load a PDF set by its name, e.g. CT18NNLO, and some x and $\mu_f = Q$ values you are interested in:

```
import numpy as np
import pdfplotter as pp

x_values = np.logspace(-5, 0, 200)
Q_values = np.array([2, 100])
ct18nnlo = pp.PDFSet("CT18NNLO", x_values, Q_values)
```

In the background, pdfplotter loads the PDF set from LHAPDF and initializes the three DataFrames, which will hold the data, by their indices. We will call them D (data), O (observables) and R (ratios). We will call the function which gets the index values from a DataFrame ind:

$$\text{ind}\, D = \{1, ..., n_{\text{members}}\} \times \texttt{x\_values} \times \texttt{Q\_values}\,, \tag{6.1}$$

$$\text{ind}\, O = \text{ind}\, R = \texttt{pdf\_types} \times \texttt{x\_values} \times \texttt{Q\_values}\,, \tag{6.2}$$

where n_{members} are the number of members in the PDF set (i.e. central and eigenvector PDFs), and pdf_types $= \{\text{uncertainty}^-, \text{central}, \text{uncertainty}^+\}$ are the types of PDFs that we actually want to plot.

As mentioned before, flavors of the PDFs are realized as sympy Symbols. For every flavor there is a variable in pdfplotter, e.g. pp.u, pp.g or pp.dbar. These can then be added, divided, etc. to represent observables that depend on the PDF flavors, e.g.

```
R_s = (pp.s + pp.sbar) / (pp.ubar + pp.dbar)
```

We will call an expression of the flavors or the flavors themselves o. We can now get actual values for o from a PDFSet object by calling

```
ct18nnlo.get_central(o, Q)
ct18nnlo.get_uncertainties(o, Q)
```

get_central returns a numpy array for the central values of the observable o at a given Q, and get_uncertaintes returns a tuple of the lower and upper uncertainties of o at a

given `Q`. The latter use the correct uncertainty prescription specified in the `.info` file of the LHAPDF set. We can then plot these values.

All the heavy stuff happens in the background: Whenever `get_central` or `get_uncertaintes` are called with a new observable, D is filled with the values of the flavors that are involved in the observable `o`. After that, the actual uncertainties of `o` are calculated using the LHAPDF Python interface, and are stored in O together with the central value of the observable. Subsequent calls with the same observable are then just a lookup in O. These automatic calculations are made possible by using the flavor variables instead of using the numbers themselves. Values of the column labels col O are then just the observables that were passed to `get_central` or `get_uncertaintes` at some point, and values of the labels col D are the actual flavors that occurred in one or more observables.

R stores ratios to other `PDFSets`, given by the `ratio_to` optional arguments in `get_central` and `get_uncertaintes`. That means the columns of R, namely col R, are the observables together with the name of the other `PDFSet` given by `ratio_to`.

More information can be found in the `pdfplotter` documentation and examples.

6.3 Multiprocessing with MPI

The more data sets are added to `nCTEQ++`, the longer takes the evaluation of the χ^2 function. To speed up the calculation, we can parallelize parts of the code that sequentially make a lot of calls to the χ^2 function without depending on the order of the calls. First we will look at the basics of parallelization.

We can classify types of parallelization by the usage of hardware memory: First, parallelization using shared memory, where all processes may access the same memory space. This means that e.g. variables can be seen and modified by all processes. An example of this is threading, realized e.g. in C++ by `std::thread`, or libraries like OpenMP. Second, parallelization using distributed memory, with each process having its own memory address space. In this way, processes truly run in parallel independent from one another directly from the start. An example of this is MPI, which handles starting the processes in such a way that they can communicate with each other. This then happens by explicitly sending data to other processes. This is often used in computing clusters, such that the nodes do not need shared memory to communicate with each other, but this can happen via a network. Shared memory models are always limited by the number of cores on a single node.

The shared memory model also requires more care in the implementation: We have to watch out that we do not modify variables by accident, as this can lead to wrong results or crashes. It can also be harder to implement into existing code, since the code is not guaranteed to be thread-safe. Since in this work we are using `nCTEQ++`, i.e. an already existing code base, and because of the availability of computing cluster with `Palma-II`, we use MPI for parallelization here.

MPI itself is a specification [Mes21], which means it specifies e.g. which functions exist and what they do. The implementation is then done by a library, for example OpenMPI or MPICH. In this work, we use OpenMPI, which has C bindings to the MPI functions

and can thus be called directly from C++. A good reference to the MPI functionality is provided by [Ann22]. The basic structure of a program using MPI is as follows: Suppose we ran our program on the command line prepended by `mpirun -n 4`, where 4 is just an arbitrary number and represents the number of processes we want to start. At the beginning of the program, we need to initialize MPI and in the end, finalize it. Before and afterward, MPI functionality is not available. It is done like this (here) shown in C++:

```
#include <mpi.h>

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);
    // ...
    MPI_Finalize();
}
```

Passing the command-line arguments is not required by the standard, and OpenMPI should actually strip e.g. the `mpirun` command from the arguments. In case we use a different implementation at some point, we can do it even if it is not needed for OpenMPI. After that, we can call MPI functions. Normally, you want to know which process you are in, and how many processes there are in total. This can be achieved by the following two functions (where the position of the code is assumed somewhere between `MPI_Init` and `MPI_Finalize`):

```
int world_size, world_rank;
MPI_Comm_size(MPI_COMM_WORLD, &world_size);
MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
```

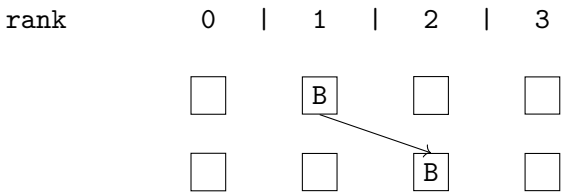
Afterward, `world_size` contains the number of processes present in `MPI_COMM_WORLD`, which is the default `MPI_Communicator`. `world_rank` contains the index of the current process, i.e. `MPI_Comm_rank` returns a different value for each process in a communicator. Usually, the process with rank 0 is called the root process. It is sometimes used as an angle point for the other process, i.e. coordinating the executions. As for the root process, the rank of each process is used to determine what a process should do.

Now to the actual communication: We present a brief overview over MPI functions that are of importance for us and will visualize what they do. Grouped boxes denote different processes where an array of the values in the boxes is present. For simplicity, we will assume that the size of this communicator, which can be the number of processes, is 4. We do not provide the full function signature here, see e.g. [Ann22] for this.

Basic sending and receiving is done by

```
MPI_Send(...);
MPI_Recv(...);
// or
MPI_SendRecv(...);
```

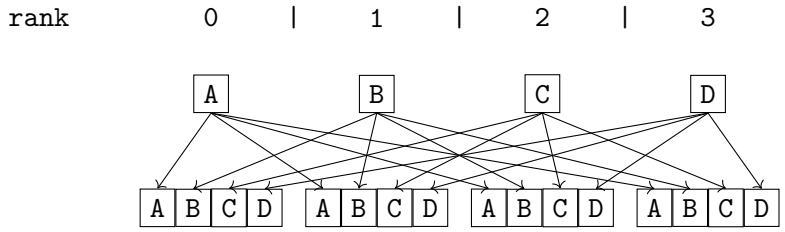
just sends a value to one process, which in turn receives it:



Sending one value to just another process is cumbersome in practice. What we look for is a way for each process to share a value with all other processes. This is done by

```
MPI_Allgather(...);
```

which looks like this:



There is just one more modification: The suffix `v` lets us send variable amounts of data from each process, not just one as displayed in the preceding diagram:

```
MPI_Allgatherv(...);
```

This is the function we will use in the implementation. Each process can calculate some values and send them to all of the other processes. Of course, there are many more functions, these can be found in [Ann22].

We briefly describe the partitioning system for MPI which was implemented during this thesis: The base class for everything is

```
class Partition1D {

// ...

public:
    explicit Partition1D(const int size, const int num_bins);

    int size() const;

    int num_bins() const;

    int bin_size(const int bin_index) const;

    std::vector<int> bin_sizes() const;

    std::vector<int> indices(const int bin_index) const;
```

```
    int start_index(const int bin_index) const;

    int end_index(const int bin_index) const;

    std::vector<int> total_indices() const;

    std::vector<int> displacements() const;

    template <typename T, typename U>
    T Allgatherv(const U& v) const;
};
```

It can partition a one-dimensional sequence of `size()` into `num_bins()` bins. In practice, if we have a `for`-loop like this:

```
std::vector<int> v(size);
for (int i = 0; i < size; i++) {
    v.at(i) = calculate_something(i);
}
```

We can partition this with `Partition1D` like this:

```
int world_size, world_rank;
MPI_Comm_size(MPI_COMM_WORLD, &world_size);
MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

std::vector<int> v(size);
Partition1D partition(world_size, size);
std::vector<int> temp;
for (int i : partition.indices(world_rank)) {
    temp.push_back(calculate_something(i));
}

v = partition.Allgatherv<std::vector<int>>(temp);
```

We see that we do not have to think about the partitioning ourselves anymore. The implementation takes care of this for us. `Partition1D` is now subclassed to different kinds of partitions, like `EquiPartition1D`, which tries to make the bins as equal as possible. This is still under active development and will hopefully help to parallelize the `nCTEQ++` application more in the future. A system like this has already been implemented for the Hessian calculation shown in section 5, since the calculation of the Hessian can be parallelized in the calculation of its components. On the computing cluster Palma-II here in Münster this is then able to speed up the calculation from days to barely under an hour.

7 Conclusion

In this work, we first calculated the heavy quark production cross section at LO analytically. The result was integrated using a `VEGAS` Monte Carlo integrator [Hah05] and compared to predictions from the `hvq` process implemented in the `POWHEG BOX V2` [FNR07; Ali+10]. It could be shown to get agreement of the order of 1σ to 2σ for both the p_T and y distributions. Furthermore, the distributions followed shapes usually expected. In the p_T-distributions, increasing errors with increasing p_T were observed for the `POWHEG` predictions, the origin of which could not yet be fully determined. The channel decompositions for p_T showed higher gg contributions for lower p_T, which switched to higher $q\bar{q}$ contributions with rising p_T, which is also expected.

Additionally, we presented development for the `nCTEQ++`. An Apptainer [App] image was successfully developed, and is used to not have to install the `nCTEQ++` dependencies, which is great for new students since they can start using the code base right away. The same will hopefully happen with the PDF plotting framework `pdfplotter`, which should reduce the size of code just used for reading in the data a lot. Lastly, parallelization efforts were made, reducing the Hessian calculation for the uncertainties from days to an hour. The parallelization framework presented will hopefully mature and then be used more in the future to help in reducing the time needed for the PDF fits, or, alternatively, reduce the uncertainties with equal runtime.

In this work, I was able to understand what goes into producing theory predictions using Monte Carlo integration. Hopefully this can be applied to modifying the GM-VFNS framework and integrate it into the `nCTEQ++` framework to provide fast execution. This would then allow for a more direct comparison to the experimental data in the case of heavy quarks, which would be a great addition to the `nCTEQ++` framework.

A Appendix

A.1 Mandelstam Variables

$$\begin{aligned} s &= (p_1 + p_2)^2 = (p_3 + p_4)^2 \\ t &= (p_1 - p_3)^2 = (p_2 - p_4)^2 \\ u &= (p_1 - p_4)^2 = (p_2 - p_3)^2 \end{aligned} \tag{A.1}$$

$$\begin{aligned} s = (p_1 + p_2)^2 &= m_1^2 + 2\,p_1 \cdot p_2 + m_2^2 \quad &\Rightarrow\ p_1 \cdot p_2 &= \frac{s - m_1^2 - m_2^2}{2} \\ = (p_3 + p_4)^2 &= m_3^2 + 2\,p_3 \cdot p_4 + m_4^2 \quad &\Rightarrow\ p_3 \cdot p_4 &= \frac{s - m_3^2 - m_4^2}{2} \\ t = (p_1 - p_3)^2 &= m_1^2 - 2\,p_1 \cdot p_3 + m_3^2 \quad &\Rightarrow\ p_1 \cdot p_3 &= \frac{m_1^2 + m_3^2 - t}{2} \\ = (p_2 - p_4)^2 &= m_2^2 - 2\,p_2 \cdot p_4 + m_4^2 \quad &\Rightarrow\ p_2 \cdot p_4 &= \frac{m_2^2 + m_4^2 - t}{2} \\ u = (p_1 - p_4)^2 &= m_1^2 - 2\,p_1 \cdot p_4 + m_4^2 \quad &\Rightarrow\ p_1 \cdot p_4 &= \frac{m_1^2 + m_4^2 - u}{2} \\ = (p_2 - p_3)^2 &= m_2^2 - 2\,p_2 \cdot p_3 + m_3^2 \quad &\Rightarrow\ p_2 \cdot p_3 &= \frac{m_2^2 + m_3^2 - u}{2} \end{aligned} \tag{A.2}$$

$$s + t + u = m_1^2 + m_2^2 + m_3^2 + m_4^2 \tag{A.3}$$

Acronyms

$\overline{\text{MS}}$ modified Minimal Subtraction. 6

CLI command-line interface. 47

CMS center-of-momentum system. 20, 21, 22, 23, 24

CTEQ Coordinated Theoretical-Experimental Project on QCD. 41, 56

DIS Deep Inelastic Scattering. 41

EMC European Muon Collaboration. 2, 41

GM-VFNS General Mass – Variable Flavor Number Scheme. 2, 54

HEP High Energy Physics. 19

HERA Hadron-Electron Ring Accelerator. 2

HPC High Performance Computing. 47, 49

i.i.d. independent and identically distributed. 11

LHAPDF Les Houches Accord PDF. 45, 49, 50

LHC Large Hadron Collider. 2

LO leading order. 2, 20, 54

LS least squares. 12, 13

MC Monte Carlo. 16, 17, 18

ML maximum likelihood. 12, 13

MPI Message Passing Interface. 47, 50, 51, 52

nCTEQ nuclear CTEQ. 13, 41, 42, 43, 45

nPDF nuclear Parton Distribution Function. 2

PDF Parton Distribution Function. 2, 3, 7, 26, 29, 41, 42, 43, 45, 47, 49, 54

PDG Particle Data Group. 7, 8

pQCD perturbative QCD. 2, 3

QCD Quantum Chromodynamics. 2, 3, 4, 6, 7, 9, 20

QED Quantum Electrodynamics. 4, 7

RGE Renormalization Group Equation. 6

SIF Singularity Image Format. 47

SM Standard Model. 3

References

[Acc+16] A. Accardi et al. "Constraints on large-x parton distributions from new weak boson production and deep-inelastic scattering data." In: *Physical Review D* 93.11 (June 20, 2016), p. 114017. ISSN: 2470-0010, 2470-0029. DOI: 10.1103/PhysRevD.93.114017. arXiv: 1602.03154[hep-ph,physics:nucl-th]. URL: http://arxiv.org/abs/1602.03154 (visited on 09/20/2023).

[Ali+10] Simone Alioli et al. "A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX." In: *Journal of High Energy Physics* 2010.6 (June 2010), p. 43. ISSN: 1029-8479. DOI: 10.1007/JHEP06(2010)043. arXiv: 1002.2581[hep-ph]. URL: http://arxiv.org/abs/1002.2581 (visited on 09/20/2023).

[Ann22] AnnaDaly. *Microsoft MPI - Message Passing Interface*. Sept. 21, 2022. URL: https://learn.microsoft.com/en-us/message-passing-interface/microsoft-mpi (visited on 09/20/2023).

[App] Apptainer project. *Apptainer*. URL: https://apptainer.org (visited on 09/20/2023).

[Aub+83] J.J. Aubert et al. "The ratio of the nucleon structure functions F2N for iron and deuterium." In: *Physics Letters B* 123.3 (Mar. 1983), pp. 275–278. ISSN: 03702693. DOI: 10.1016/0370-2693(83)90437-9. URL: https://linkinghub.elsevier.com/retrieve/pii/0370269383904379 (visited on 09/03/2023).

[Bal+22] Richard D. Ball et al. "The Path to Proton Structure at One-Percent Accuracy." In: *The European Physical Journal C* 82.5 (May 2022), p. 428. ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-022-10328-7. arXiv: 2109.02653[hep-ex,physics:hep-ph]. URL: http://arxiv.org/abs/2109.02653 (visited on 09/20/2023).

[Bie+20] C. Bierlich et al. "Robust Independent Validation of Experiment and Theory: Rivet version 3." In: *SciPost Physics* 8.2 (Feb. 11, 2020), p. 026. ISSN: 2542-4653. DOI: 10.21468/SciPostPhys.8.2.026. arXiv: 1912.05451[hep-ex,physics:hep-ph]. URL: http://arxiv.org/abs/1912.05451 (visited on 09/20/2023).

[Buc+15] Andy Buckley et al. "LHAPDF6: parton density access in the LHC precision era." In: *The European Physical Journal C* 75.3 (Mar. 2015), p. 132. ISSN: 1434-6044, 1434-6052. DOI: 10.1140/epjc/s10052-015-3318-8. arXiv: 1412.7420[hep-ex,physics:hep-ph]. URL: http://arxiv.org/abs/1412.7420 (visited on 09/20/2023).

[Col11] John Collins. *Foundations of Perturbative QCD*. 1st ed. Cambridge University Press, Apr. 28, 2011. ISBN: 978-0-521-85533-4 978-0-511-97559-2 978-1-107-64525-7. DOI: 10.1017/CBO9780511975592. URL: https://www.cambridge.org/core/product/identifier/9780511975592/type/book (visited on 05/01/2023).

[Cow98] Glen Cowan. *Statistical data analysis*. Oxford science publications. Oxford : New York: Clarendon Press ; Oxford University Press, 1998. 197 pp. ISBN: 978-0-19-850156-5 978-0-19-850155-8.

[Duw+22] P. Duwentäster et al. "Impact of heavy quark and quarkonium data on nuclear gluon PDFs." In: *Physical Review D* 105.11 (June 27, 2022), p. 114043. ISSN: 2470-0010, 2470-0029. DOI: 10.1103/PhysRevD.105.114043. arXiv: 2204.09982[hep-ph]. URL: http://arxiv.org/abs/2204.09982 (visited on 09/20/2023).

[Esk+22] Kari J. Eskola et al. "EPPS21: A global QCD analysis of nuclear PDFs." In: *The European Physical Journal C* 82.5 (May 2022), p. 413. ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-022-10359-0. arXiv: 2112.12462[hep-ph]. URL: http://arxiv.org/abs/2112.12462 (visited on 09/20/2023).

[ESW96] R. K. Ellis, W. J. Stirling, and B. R. Webber. *QCD and Collider Physics*. 1st ed. Cambridge University Press, Oct. 24, 1996. ISBN: 978-0-521-58189-9 978-0-521-54589-1 978-0-511-62878-8. DOI: 10.1017/CBO9780511628788. URL: https://www.cambridge.org/core/product/identifier/9780511628788/type/book (visited on 05/01/2023).

[FNR07] Stefano Frixione, Paolo Nason, and Giovanni Ridolfi. "A Positive-Weight Next-to-Leading-Order Monte Carlo for Heavy Flavour Hadroproduction." In: *Journal of High Energy Physics* 2007.9 (Sept. 28, 2007), pp. 126–126. ISSN: 1029-8479. DOI: 10.1088/1126-6708/2007/09/126. arXiv: 0707.3088[hep-ph]. URL: http://arxiv.org/abs/0707.3088 (visited on 09/20/2023).

[Gri87] David Griffiths. *Introduction to Elementary Particles*. 1st ed. Wiley, Dec. 23, 1987. ISBN: 978-0-471-60386-3 978-3-527-61846-0. DOI: 10.1002/9783527618460. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9783527618460 (visited on 08/31/2023).

[Hah05] T. Hahn. "Cuba - a library for multidimensional numerical integration." In: *Computer Physics Communications* 168.2 (June 2005), pp. 78–95. ISSN: 00104655. DOI: 10.1016/j.cpc.2005.01.010. arXiv: hep-ph/0404043. URL: http://arxiv.org/abs/hep-ph/0404043 (visited on 09/20/2023).

[Hou+21] Tie-Jiun Hou et al. "New CTEQ global analysis of quantum chromodynamics with high-precision data from the LHC." In: *Physical Review D* 103.1 (Jan. 11, 2021), p. 014013. ISSN: 2470-0010, 2470-0029. DOI: 10.1103/PhysRevD.103.014013. arXiv: 1912.10053[hep-ex,physics:hep-ph,physics:nucl-th]. URL: http://arxiv.org/abs/1912.10053 (visited on 09/20/2023).

[Kha+20] Rabah Abdul Khalek et al. "nNNPDF2.0: Quark Flavor Separation in Nuclei from LHC Data." In: *Journal of High Energy Physics* 2020.9 (Sept. 2020), p. 183. ISSN: 1029-8479. DOI: 10.1007/JHEP09(2020)183. arXiv: 2006.14629[hep-ex,physics:hep-ph,physics:nucl-ex,physics:nucl-th]. URL: http://arxiv.org/abs/2006.14629 (visited on 09/20/2023).

[Kni+05a] B. A. Kniehl et al. "Collinear Subtractions in Hadroproduction of Heavy Quarks." In: *The European Physical Journal C* 41.2 (May 2005), pp. 199–212. ISSN: 1434-6044, 1434-6052. DOI: 10.1140/epjc/s2005-02200-7. arXiv: hep-ph/0502194. URL: http://arxiv.org/abs/hep-ph/0502194 (visited on 05/02/2023).

[Kni+05b] B. A. Kniehl et al. "Inclusive D^{*+-} Production in p p-bar Collisions with Massive Charm Quarks." In: *Physical Review D* 71.1 (Jan. 20, 2005), p. 014018. ISSN: 1550-7998, 1550-2368. DOI: 10.1103/PhysRevD.71.014018. arXiv: hep-ph/0410289. URL: http://arxiv.org/abs/hep-ph/0410289 (visited on 05/02/2023).

[Kni+12] B. A. Kniehl et al. "Inclusive Charmed-Meson Production at the CERN LHC." In: *The European Physical Journal C* 72.7 (July 2012), p. 2082. ISSN: 1434-6044, 1434-6052. DOI: 10.1140/epjc/s10052-012-2082-2. arXiv: 1202.0439[hep-ph]. URL: http://arxiv.org/abs/1202.0439 (visited on 05/02/2023).

[Kov+16] K. Kovarik et al. "nCTEQ15 - Global analysis of nuclear parton distributions with uncertainties in the CTEQ framework." In: *Physical Review D* 93.8 (Apr. 28, 2016), p. 085037. ISSN: 2470-0010, 2470-0029. DOI: 10.1103/PhysRevD.93.085037. arXiv: 1509.00792[hep-ph]. URL: http://arxiv.org/abs/1509.00792 (visited on 09/03/2023).

[Lep21] G. Peter Lepage. "Adaptive Multidimensional Integration: VEGAS Enhanced." In: *Journal of Computational Physics* 439 (Aug. 2021), p. 110386. ISSN: 00219991. DOI: 10.1016/j.jcp.2021.110386. arXiv: 2009.05112[hep-ph,physics:physics]. URL: http://arxiv.org/abs/2009.05112 (visited on 09/17/2023).

[McK10] Wes McKinney. "Data Structures for Statistical Computing in Python." In: Python in Science Conference. Austin, Texas, 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a. URL: https://conference.scipy.org/proceedings/scipy2010/mckinney.html (visited on 09/20/2023).

[Mes21] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard Version 4.0*. June 2021. URL: https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf.

[Meu+17] Aaron Meurer et al. "SymPy: symbolic computing in Python." In: *PeerJ Computer Science* 3 (Jan. 2, 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103. URL: https://peerj.com/articles/cs-103 (visited on 09/20/2023).

[MN98] Makoto Matsumoto and Takuji Nishimura. "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator." In: *ACM Transactions on Modeling and Computer Simulation* 8.1 (Jan. 1998), pp. 3–30. ISSN: 1049-3301, 1558-1195. DOI: 10.1145/272991.272995. URL: https://dl.acm.org/doi/10.1145/272991.272995 (visited on 09/15/2023).

[Owe+07] J. F. Owens et al. "The impact of new neutrino DIS and Drell-Yan data on large-x parton distributions." In: *Physical Review D* 75.5 (Mar. 26, 2007), p. 054030. ISSN: 1550-7998, 1550-2368. DOI: 10.1103/PhysRevD.75.054030. arXiv: hep-ph/0702159. URL: http://arxiv.org/abs/hep-ph/0702159 (visited on 09/03/2023).

[Pap20] Andreas Papaefstathiou. "How-to: Write a parton-level Monte Carlo event generator." In: *The European Physical Journal Plus* 135.6 (June 2020), p. 497. ISSN: 2190-5444. DOI: 10.1140/epjp/s13360-020-00499-1. arXiv: 1412.4677[hep-ph]. URL: http://arxiv.org/abs/1412.4677 (visited on 09/15/2023).

[Par+22] Particle Data Group et al. "Review of Particle Physics." In: *Progress of Theoretical and Experimental Physics* 2022.8 (Aug. 8, 2022), p. 083C01. ISSN: 2050-3911. DOI: 10.1093/ptep/ptac097. URL: https://academic.oup.com/ptep/article/doi/10.1093/ptep/ptac097/6651666 (visited on 09/02/2023).

[Pet78] G Peter Lepage. "A new algorithm for adaptive multidimensional integration." In: *Journal of Computational Physics* 27.2 (May 1978), pp. 192–203. ISSN: 00219991. DOI: 10.1016/0021-9991(78)90004-9. URL: https://linkinghub.elsevier.com/retrieve/pii/0021999178900049 (visited on 09/17/2023).

[Pum+01] J. Pumplin et al. "Uncertainties of predictions from parton distribution functions II: the Hessian method." In: *Physical Review D* 65.1 (Dec. 12, 2001), p. 014013. ISSN: 0556-2821, 1089-4918. DOI: 10.1103/PhysRevD.65.014013. arXiv: hep-ph/0101032. URL: http://arxiv.org/abs/hep-ph/0101032 (visited on 09/13/2023).

[Sch14] Matthew Dean Schwartz. *Quantum field theory and the standard model*. New York: Cambridge University Press, 2014. 850 pp. ISBN: 978-1-107-03473-0.

[Ska13] Peter Skands. "Introduction to QCD." In: *Searching for New Physics at Small and Large Scales*. Nov. 2013, pp. 341–420. DOI: 10.1142/9789814525220_0008. arXiv: 1207.2389[hep-ph,physics:hep-th]. URL: http://arxiv.org/abs/1207.2389 (visited on 08/31/2023).

[SMO20] Vladyslav Shtabovenko, Rolf Mertig, and Frederik Orellana. "FeynCalc 9.3: New features and improvements." In: *Computer Physics Communications* 256 (Nov. 2020), p. 107478. ISSN: 00104655. DOI: 10.1016/j.cpc.2020.107478. arXiv: 2001.04407[hep-ph,physics:hep-th]. URL: http://arxiv.org/abs/2001.04407 (visited on 09/20/2023).

[Wei00] Stefan Weinzierl. *Introduction to Monte Carlo methods*. June 23, 2000. arXiv: hep-ph/0006269. URL: http://arxiv.org/abs/hep-ph/0006269 (visited on 09/15/2023).

Declaration of Academic Integrity

I hereby confirm that this thesis is solely my own work and that I have used no sources or aids other than the ones stated. All passages in my thesis for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited.

Münster, October 16, 2023 _____

(Signature of Student)

I agree to have my thesis checked in order to rule out potential similarities with other works and to have my thesis stored in a database for this purpose.

Münster, October 16, 2023 _____

(Signature of Student)