Approximation of the order α_s^2 Drell-Yan correction terms $\Delta_{ij}^{(2)}$

Felix Haas

Universität Potsdam, Institut für Physik
Tucholskystr. 49, 10117 Berlin

Abstract

In this paper the approximation of the order α_s^2 Drell-Yan correction terms $\Delta_{ij}^{(2)}$ [1] is discussed. As a contribution of this approximation the limits x $\rightarrow$ 0 and x $\rightarrow$ 1 are calculated. In preparation for this, the Mathematica implementation `HPL` [2] of the harmonic polylogarithms is used to check the numerical values (4.16) found in [4].

1 Introduction

1.1 The hadron substructure

The cross section describing deep inelastic scattering of leptons on hadrons can be split into mainly two parts:

$$\sigma \sim L_{\mu\nu} W^{\mu\nu}. \tag{1}$$

Here $L_{\mu\nu}$ is the so called lepton tensor. It describes the leptonic part of the interaction. Equivalently $W^{\mu\nu}$ is the hadron tensor and describes the hadronic part of the interaction. The fundamental structural difference of the two particles is mirrored by their respective tensor. Leptons are point like particles without substructure. Their part of the scattering is described by the point like lepton tensor

$$L_{\mu\nu} = 2[k'_\mu k_\nu + k'_\nu k_\mu + (q^2/2) g_{\mu\nu}]. \tag{2}$$

Where k and k' are the the lepton momenta before and after the scattering. Due to the substructure of the hadron, the hadron tensor takes on the non point like form [1]

$$W_{\mu\nu} = [-g_{\mu\nu} + q_\mu q_\nu / q^2] W_1(q^2, \nu) + [p_\mu - (pq/q^2) q_\mu][p_\nu - (pq/q^2) q_\nu] M^{-2} W_2(q^2, \nu). \tag{3}$$

p is the hadron momentum before the scattering, $q = p' - p$ the momentum difference. M is the hadron mass and $\nu = pq/M$. The functions $W_{1,2}(q^2, \nu)$ are the so called hadron structure functions which describe its substructure i.e. the parton distribution.
In the Bjorken limit

$$Q^2 = -q^2 \rightarrow \infty, \;\; \nu \rightarrow \infty, \;\; x = Q^2/2M\nu \; fixed \tag{4}$$

[1] neglecting the lepton mass

the structure functions show a scaling behavior, i.e. they become functions of only the variable x.

$$MW_1(q^2,\nu) \to F_1(x), \;\; \nu W_2(q^2,\nu) \to F_2(x) \tag{5}$$

In this limit the functions $F_{1,2}(x)$ can always be expressed as a linear combination of the individual parton density functions $N_i(x)$ where the subscript i denotes the kind of parton, i.e. gluon, up, anti-up, strange,..., depending on the scattering process.

1.2 The Drell Yan Process

The Drell-Yan Process describes the annihilation of a quark antiquark pair in a hadron collision into a pair of leptons (electrons, muons or tauons) and a final hadronic state, usually denoted by "X".

$$q + \overline{q} = l + \overline{l} + X \tag{6}$$

It is impossible to observe a DY-process of say only an u (up-quark) and $\overline{u}$ (antiup-quark). We will always, due to confinement, need to use whole hadrons (H and $\overline{H}$) and then sum over all the contributing processes such as up-antiup, down-antidown, up-gluon, etc. What we deal with in a DY-process is thus not only process (6) but rather

$$H + \overline{H} = l + \overline{l} + X. \tag{7}$$

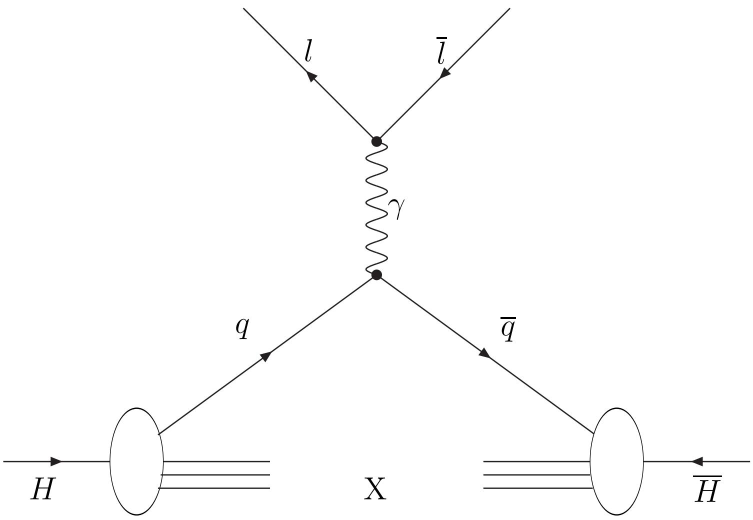

Figure 1 DY-process

For the Drell-Yan Process the hadronic structure function may be written as

$$W(Q^2,\tau) = \sum_{i,j} \int_0^1 dx_1 \int_0^1 dx_2 \int_0^1 dx \, \delta(\tau - x x_1 x_2) \, D_{ij}(x_1, x_2, M^2) \, \Delta_{ij}(x, Q^2, M^2). \tag{8}$$

$\tau = Q^2/S$ where $S^{1/2}$ and $(Q^2)^{1/2}$ is the c.m. energy of the hadrons and the invariant mass of the lepton pair. As already mentioned, we need to sum over the different possible annihilation processes of different parton types. $D_{ij}(x_1, x_2, M^2) \sim N_i(x_1) \cdot N_j(x_2)$ denotes the combination of these different parton distributions. $\Delta_{ij}(x, Q^2, M^2)$ is the Drell-Yan correction term and is nothing but the cross section of the contributing parton annihilations. In the discussion of the DY-process R and M are the renormalization scale and the mass factorization [2]. Like a usual cross section $\Delta_{ij}(x, Q^2, M^2)$ is expand in powers of the running coupling constant $\alpha_s(R^2)$

$$\Delta_{ij}(x, Q^2, M^2) = \sum_{n=0}^{\infty} \alpha_s^n(R^2)\, \Delta_{ij}^{(n)}(x, Q^2, M^2, R^2). \tag{9}$$

1.3 The mathematical structure of the correction terms

The different orders of perturbation theory generate functions of different types. In zero order of $\alpha_s(R^2)$, the feynman rules give us only $\Delta_{q\overline{q}}^{(0)} = \delta(1-x)$. Note that here the only contributing process is the quark-antiquark annihilation, so just the elementary subprocess of the process pictured in Figure 1. Already in first order however one finds divergencies in both the low and high (i.e. 0 and 1) limits in Bjorken-x-space. Redrawing the parton annihilation part of Figure 1 for higher orders gives for example rise to the following process of soft (i.e. low momentum) gluon emission.

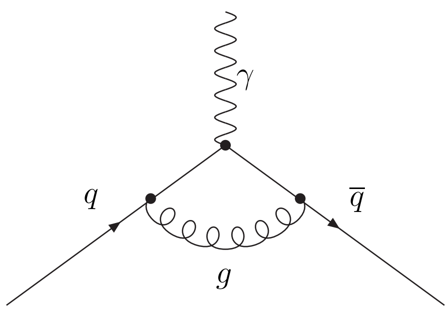

Figure 2 soft gluon emission

This process of emission and reabsorbtion of a virtual soft gluon is a second order α_s process. This can be seen by simply counting the vertices. The vertex of the coupling to the vector boson (here the photon) gives the electromagnetic coupling constant. Whereas the two gluon coupling vertices each give the strong coupling constant, i.e. together we get g_s^2. Squaring the matrix element and taking $\alpha_s \sim g_s^2$ into account, we obtain the predicted α_s^2 dependence.
Doing the explicit calculation for the soft gluon emission does not give the same divergent structure as $\Delta_{q\overline{q}}^{(0)}$ (namely $\delta(1-x)$) but rather $\Delta_{q\overline{q}}^{(2)}$(soft gluon)$\sim ln(1-x)/(1-x)$. Note that a lot of other correction processes contribute to $\Delta_{q\overline{q}}^{(2)}$, thus its divergent term turns out to be a lot more complicated then only $ln(1-x)/(1-x)$.
One should also keep in mind, that $\Delta_{q\overline{q}}^{(2)}$ is not the only kind of second order correction contributing to $W(Q^2, \tau)$. In (8) one has to sum over all possible parton combinations, i.e. not only over quark-antiquark, but also over (anti)quark-(anti)quark, (anti)quark-gluon and gluon-gluon.
Taking all this into account one gets the following structure for the limits of $\Delta_{ij}^{(2)}$

[2] In my calculations R is set equal to M which cancels first order contributions to the second order of $\Delta_{ij}(x, Q^2, M^2)$ due to its proportionality to $log(R^2/M^2)$.

$$\Delta^{(2)}_{ij,x\to 1} = A\frac{1}{1-x} + B\,\delta(1-x) + \sum_{n=1}^{3} C_n ln^n(1-x) + \sum_{n=1}^{3} D_n \frac{ln^n(1-x)}{1-x} + O(1) \qquad (10)$$

$$\Delta^{(2)}_{ij,x\to 0} = E\frac{1}{x} + \sum_{n=1}^{3} F_n ln^n(x) + \sum_{n=1}^{2} G_n \frac{ln^n(x)}{x} + O(1). \qquad (11)$$

The coefficients $A, B, C_n, D_n, E, F_n, G_n$ generally are functions of x. In the limits (10) and (11) however they become constant ($A(x)_{x\to 1} = A$).
It is the goal of this paper to put the DY correction terms in the form of (10) and (11) and to find the numerical values for the coefficients. For a full approximation, the $O(1)$ term has to be fitted, so that the full expression is preferably close to the correction term. This is discussed in section 3.

2 Harmonic Polylogarithms and HPL

2.1 Harmonic Polylogarithms

The logarithms in (10) and (11), as well as the well behaved polylogarithms, who do not contribute to $\Delta^{(2)}_{ij}$ in the upper limits can be absorbt in a new class of functions called the harmonic polylogarithms (HPLs), first introduced in [3]. The HPLs of weight one are defined as

$$\begin{aligned} H(0;x) &= \ln(x) \\ H(1;x) &= \int_0^x \frac{1}{1-t}dt = -\ln(1-x) \\ H(-1;x) &= \int_0^x \frac{1}{1+t}dt = \ln(1+x). \end{aligned} \qquad (12)$$

HPLs of higher weight are defined recursively through integration of the following functions

$$f_0(x) = \frac{1}{x}, \quad f_1(x) = \frac{1}{1-x}, \quad f_{-1}(x) = \frac{1}{1+x}. \qquad (13)$$

The general recursion formula for HPLs of higher order is

$$H(a, a_{1,\ldots,k}; x) = \int_0^x f_a(t) H(a_{1,\ldots,k}; t)\, dt. \qquad (14)$$

Which gives for example

$$H(-1,1;x) = Li_2(\frac{1+x}{2}) - ln(2)ln(1+x) - Li_2(\frac{1}{2}). \qquad (15)$$

Where $Li_n(x) = \sum_{k=1}^{\infty} \frac{x^k}{k^n}$ are Nielsen‘s polylogarithms. They are non divergent functions (that is why they do not appear in (10) or (11)) with a zero at the origin.
The HPLs fulfill the very useful product algebra

$$H(\vec{p};x)H(\vec{q};x) = \sum_{\vec{r}\in\vec{p}\uplus\vec{q}} H(\vec{r};x), \tag{16}$$

where $\vec{p}$ and $\vec{q}$ are index vectors. The sum is understood to be over all permutations of indices that do not change the original internal order of the respective index vector. For example if we take $\vec{p} = (a,b)$ and $\vec{q} = (c,d)$ we get

$$\begin{aligned} H(a,b;x)H(c,d;x) &= H(a,b,c,d;x) + H(a,c,b,d;x) \\ &\quad + H(a,c,d,b;x) + H(c,a,b,d;x) \\ &\quad + H(c,a,d,b;x) + H(c,d,a,b;x). \end{aligned} \tag{17}$$

a can interchange freely with c and d, so can b. a however always has to be listed before b. c always has to be listed before d.

2.2 Extraction of divergencies with `HPL`

`HPL` [2] is a Mathematica implementation of the harmonic polylogarithms. Next to other operations on HPLs, it also contains the function `HPLLogExtract` which extracts the logarithmic divergencies. These divergencies can be made more explicit by `HPLConvertToKnownFunctions` which changes the HPLs back to regular logarithms and polylogarithms.
If we want to bring $\Delta_{ij}^{(2)}$ into the form of (10) or (11) respectively, we need to run this procedure in order to be able to take the respective limit of the coefficient functions $A(x), B(x), ...$ $(A(x)_{x\to 1} = A)$.
The extraction of the logarithmic divergencies via `HPLLogExtract` makes use of the HPL

product algebra (16).
For $x \to 0$ the divergencies occur only due to the $ln^n(x)$ terms, made explicit in (11). All $ln^n(x)$ divergencies are due to n 0's at the right end of the index vector. Using (16) one can write

$$\begin{aligned} &H(a_{1,\ldots,k};x)H(0;x) = \\ &\qquad H(a_{1,\ldots,k},0;x) + H(a_{1,\ldots,k-1},0,a_k;x) + \ldots + H(0,a_{1,\ldots,k};x). \end{aligned} \tag{18}$$

The only function that exhibits a divergency at 0 is $H(a_{1,\ldots,k},0;x)$. Thus solving (18) for $H(a_{1,\ldots,k},0;x)$ and recalling that $H(0;x) = ln(x)$ makes the $ln(x)$ stand explicit. For functions with more then one trailing 0 one has to use this method recursively.
The logarithmic divergencies $ln^n(1-x)$ for $x \to 1$ only occur in HPLs with leading 1's. The equivalent of (18) reads here

$$\begin{aligned} &H(1;x)H(a_{1,\ldots,k};x) = \\ &\qquad H(1,a_{1,\ldots,k};x) + H(a_1,1,a_{2,\ldots,k};x) + \ldots + H(a_{1,\ldots,k},1;x). \end{aligned} \tag{19}$$

Now we solve for $H(1,a_{1,\ldots,k};x)$ and recall $H(1;x) = -ln(1-x)$. This gives us the the explicit divergency.

2.3 Probing the `HPL` package

To test the `HPL` package, I reproduced the numerical values of the expansion coefficients of the three loop splitting function $P^{(2)}(x)$ for the $x \to 0$ case (4.16) in [4].Generally the divergent structure turns out similar to (10) and (11) however for $x \to 0$ lacks the soft gluon emission term discussed in 3. In the lower limit the $ln^n(x)$ expansion differs also, as it contains terms up to $n = 4$. The $ln^n(x)$ expansion with x dependent coefficients is obtained by the commands

$$\begin{aligned}
&\texttt{\$HPLAutoConvertToKnownFunctions} = \texttt{True};\\
&\texttt{ReplaceAll}[\texttt{P}^{(2)}(\texttt{x}), \texttt{DiracDelta}[1 - \texttt{x}]- > 0];\\
&\texttt{HPLLogExtract}[\%];\\
&\texttt{Collect}[\%, \texttt{Log}[\texttt{x}]].
\end{aligned} \tag{20}$$

The x dependent coefficients are taken independently and the limit $x \to 0$ is taken with a simple $\texttt{ReplaceAll}[\texttt{D}_\texttt{i}(\texttt{x}), \texttt{x}- > 0]$ command. This procedure gives the following numerical values [3] in powers of the the flavor number n_f

$$\begin{aligned}
&D_0^+ \simeq 1.580247\\
&D_1^+ \simeq 29.629630 - 2.370370 n_f\\
&D_2^+ \simeq 295.041787 - 32.197531 n_f + 0.592593 n_f^2\\
&D_3^+ \simeq 1261.11350 - 152.596678 n_f + 4.345679 n_f^2.
\end{aligned} \tag{21}$$

These values correspond to the values (4.16) in [4] up to their given accuracy. The `HPLLogExtract` and `HPLConvertToKnownFunctions` commands of the `HPL` package seem to work with sufficient precision.

3 Approximation of the Drell Yan correction terms

3.1 General procedure

The Approximation of the DY-correction terms follows two separate steps. First, as already discussed, one is to find the numerical values for the coefficients of the divergent terms listed in (22) and (23). This is a calculation only valid for the limits under consideration. Second, one seeks a general approximation of the DY-correction terms. This is done by subtracting the divergent terms (they have already been treaded in their respective limit) and running a numerical approximation routine for the remaining part, the rest term $O(1)$. Hereby certain expressions that did not contribute to the coefficient values have to be considered. The approximation of the whole correction term is then obtained by simply adding (22) and (23) to the numerical approximation. For convenience (10) and (11) are reproduced here as (22) and (23)

$$\Delta^{(2)}_{ij,x\to 1} = A\frac{1}{1-x} + B\,\delta(1-x) + \sum_{n=1}^{3} C_n ln^n(1-x) + \sum_{n=1}^{3} D_n \frac{ln^n(1-x)}{1-x} + O(1) \tag{22}$$

$$\Delta^{(2)}_{ij,x\to 0} = E\frac{1}{x} + \sum_{n=1}^{3} F_n ln^n(x) + \sum_{n=1}^{2} G_n \frac{ln^n(x)}{x} + O(1). \tag{23}$$

[3]The notation of the coefficients follows [4] and differs from the one used in (11).

Their structure naturally depends on the process under consideration. As discussed in 1.3, in a perturbation series that does not consider the soft gluon emission for example, one would not encounter the D_n terms. One has to carefully go through all expressions and make sure to list all divergent terms. Hereby combinations of divergent terms must be treaded separately, otherwise one of the divergencies would remain in the coefficient term of the other, thus yielding infinity in the limit. That is why in (22) and (23) the term $\sum_{n=1}^{3} D_n \frac{ln^n(1-x)}{1-x}$ appears separately.
The Mathematica algorithm (Appendix A) of course also relies on the correct structure of (22) and (23). If an algorithm for example only considered $ln^n(1-x)$ terms up to n=2, it would not only not calculate C_3, but also yield a wrong result for C_2 as it would also incorporate terms normally belonging to the coefficient C_3. That is the reason why, even though the DY-correction terms of order α_s^2 only contain terms up to n=3, the algorithm was written to calculate terms up to n=4.

Terms such as $ln^n(x) \cdot ln^m(1-x)$ will neither contribute to the low nor to the high limit. For $x \to 0$ $log^n(x)$ is divergent and thus $ln^m(1-x)$ would normally contribute to its coefficient F_n, but as $x \to 0$ $ln^m(1-x)$ vanishes and the whole term does not contribute. For $x \in \,]0,1[$ generally it does of course contribute. Thus one has to carefully extract the following terms individually and approximate them just like the rest term $O(1)$. For $x \to 0$ generally any divergent term (i.e. $\frac{1}{x}$, $ln^n(x)$, and $\frac{ln^n(x)}{x}$), multiplied by one of the following expressions, is to be approximated

$$ln^n(1-x),\;\; ln^n(x+1)\;\; x^n,\;\; Li_n(x),\;\; Li_n(-x),\;\; S_{n,m}(x). \tag{24}$$

For $x \to 1$ any divergent term (i.e. $\frac{1}{1-x}$, $\delta(1-x)$, $ln^n(1-x)$ and $\frac{ln^n(1-x)}{1-x}$), multiplied by one of the following expressions, is to be approximated

$$ln^n(x),\;\; Li_n(1-x),\;\; Li_n(\frac{1-x}{1+x}),\;\; S_{n,m}(1-x). \tag{25}$$

Where

$$S_{n,m}(x) = \frac{(-1)^{n+m-1}}{(n-1)!m!} \int_0^1 \frac{dz}{z} \log^{n-1}(z) \log^m(1-zx) \tag{26}$$

are the harmonic sums [5] who also exhibit a zero at their origin.
For the purpose of extracting the above terms (24) and (25) that do not contribute to the numerical value of the coefficients in (22) and (23), another Mathematica algorithm was written. Due to its length and structural similarity to the algorithm in Appendix A (with the significant difference that of course no limit is taken), it is however not displayed in this paper.

3.2 Results for the limits

The algorithm to be found in Appendix A was used to derive the coefficients A, B, C_n, D_n for $x \to 1$ and E, F_n, G_n for $x \to 0$. Following the distinction being made in the Appendix B of [1], the following contributions to Δ_{ij} are treaded separately

(i) quark-antiquark (non-singlet)
(ii) (anti)quark-gluon
(iii) quark-antiquark (singlet) and non-identical quark-quark

(iv) identical quark-quark and
(v) gluon-gluon.

In the following results the color factors C_A and C_F, appearing in the original correction terms, have been replaced by their respective values, namely $C_A = 3,\ C_F = 4/3$.

(i) the quark-antiquark (non-singlet (NS)) contribution $\Delta^{(2),NS}_{q\bar{q}}$

$$\begin{aligned} &A = -12.33289 n_f + 962.93399 \\ &B = 13.75044 n_f - 68.3263 \\ &C_1 = 572.421002 + 23.7037037 n_f,\ C_2 = 234.666667 - 14.2222222 n_f,\ C_3 = -227.555556 \\ &D_1 = -23.7037037 n_f - 563.532113,\ D_2 = 14.2222222 n_f - 234.666667,\ D_3 = 227.555556 \\ &F_1 = -146.138022 + 2133333 n_f,\ F_2 = -28 + 8 n_f,\ F_2 = -15.11111 \\ &E = G_n = 0 \end{aligned} \tag{27}$$

(ii) the (anti)quark-gluon contribution $\Delta^{(2)}_{qg}$

$$\begin{aligned} &C_1 = -95.76359 n_f,\ C_2 = -8 n_f,\ C_3 = 28.55556 n_f \\ &E = 32.76339 n_f \\ &F_1 = 146.70544 n_f,\ F_2 = -3.16667 n_f,\ F_3 = 5.22222 n_f \\ &A = B = D_n = G_n = 0 \end{aligned} \tag{28}$$

(iii) the quark-antiquark (singlet) and non-identical quark-quark contribution $\Delta^{(2),S}_{q\bar{q}} + \Delta^{(2)}_{q_i q_j}$

$$\begin{aligned} &E = 241.23232 n_f \\ &F_1 = 131.90248 n_f,\ F_2 = -20 n_f,\ F_3 = 24 n_f \\ &A = B = C_n = D_n = G_n = 0 \end{aligned} \tag{29}$$

(iv) the identical quark-quark contribution $\Delta^{(2)}_{qq}$

$$\begin{aligned} &F_1 = -18.34479,\ F_2 = 8.88889,\ F_3 = 4.14815 \\ &A = B = C_n = D_n = E = G_n = 0 \end{aligned} \tag{30}$$

(v) the gluon-gluon contribution $\Delta^{(2)}_{gg}$

$$\begin{aligned} &F_1 = 4\frac{N^2}{N^2-1} - 9.84053,\ F_2 = 1.33333\frac{N^2}{N^2-1} - 6,\ F_3 = -2 \\ &A = B = C_n = D_n = E = G_n = 0 \end{aligned} \tag{31}$$

Where n_f is the flavor number and N the color number.
Here again, one sees how the different physical processes are mirrored in the mathematical structure of their coefficients. All the hard limit coefficients are only present in (i) whereas they vanish completely in (iii)-(v).
The G_n coefficients always vanish, so it was not necessary to add them to (23). The terms however are present in (iii), it is only that their coefficients vanish due to the factors (24) discussed in the last section.

3.3 The approximation routine

The numerical values for the coefficients found in the last section can now be used to obtain an overall approximation of the second order DY correction terms.
The second algorithm that was written (not shown in the appendix), extracts the terms (24) and (25). These terms, together with the rest term $O(1)$ are added to the divergent terms in their limit ((27)-(31)). The rest term and the divergent terms with vanishing coefficients have now to be fitted to the original function, vanishing at the endpoints 0 and 1, to ensure the right limit behavior.
Let us clarify the procedure, following the example (iv), the identical quark-quark contribution $\Delta_{qq}^{(2)}$. For shorter notation we set $L_1 := ln(1-x),\ L_0 := ln(x)$. For (iv) one finds terms proportional to the following divergent terms with vanishing coefficients:

$$L_1 L_0,\ L_1 L_0^2,\ x L_0,\ x^2 L_0,\ x^2 L_0^2,\ x^2 L_0^3. \tag{32}$$

In an approximation routine, one would now have to tread the following expression

$$\begin{aligned}\Delta_{qq}^{(2)} \simeq\ & -18.34479 L_1 + 8.88889 L_1^2 + 4.14815 L_1^3 \\ & + a_1 L_1 L_0 + a_2 L_1 L_0^2 + a_3 x L_0 + a_4 x^2 L_0 + a_5 x^2 L_0^2 \\ & + a_6 x^2 L_0^3 + a_7 x^3 + a_8 x^2 + a_9 x + a_{10}.\end{aligned} \tag{33}$$

Where the a_i are real valued fitting parameters and the first line of (33) is just the limit (30). The same procedure needs to be applied to all five DY correction terms, always inserting the limits and fitting the parameters of the ln terms that vanish in the respective limit.

4 Summary

The approximation of the Drell Yan correction terms was split into three steps. First one has to extract all divergent expressions both for x → 0 and x → 1. For this, the algorithm in appendix A was used and the results were presented in section (3.2). Another algorithm was written to perform the next step of extracting the terms (24) and (25) that have a vanishing coefficient in the respective limit. The last step of fitting the extracted finite terms as well as the rest term to match the original function was discussed in section (3.3) for that purpose fitting parameters a_i were introduced. However they still remain to be calculated.

Acknowledgements

I would like to thank my supervisor, Sven-Olaf Moch, for his help and advise during my stay at Desy Zeuthen. Also I would like to thank Johannes Blümlein for encouraging me to apply for the Desy summer school and for his teaching at the University of Potsdam. Furthermore I would like to thank Karlheinz Hiller for all the work that he puts into the summer school each year, and finally thanks to all the summer students who spent their last 8 weeks here with me at Desy.

References

[1] R. Hamberg, W. L. van Neerven and T. Matsuura, Nucl. Phys. B **359**, 343 (1991) [Erratum-ibid. B **644**, 403 (2002)].

[2] D. Maitre, arXiv:hep-ph/0507152.

[3] E. Remiddi and J. A. M. Vermaseren, Int. J. Mod. Phys. A **15**, 725 (2000) [arXiv:hep-ph/9905237].

[4] S. Moch, J. A. M. Vermaseren and A. Vogt, Nucl. Phys. B **688**, 101 (2004) [arXiv:hep-ph/0403192].

[5] J. Blumlein and S. Kurth, Phys. Rev. D **60**, 014018 (1999) [arXiv:hep-ph/9810241].

[6] I. J. R. Aitchison and A. J. G. Hey, "Gauge theories in particle physics: A practical introduction. Vol. 1: From relativistic quantum mechanics to QED,"

A Mathematica algorithm

```
QM[t_] := ReplaceAll[t, Log[Q^2/M^2]-> 0];
Ca[t_] := ReplaceAll[t, ca-> 3];
Cf[t_] := ReplaceAll[t, cf-> 4/3];
(******************Here the algorithm for the coefficients A and Dn****)
Collect[F[x], 1/(1 - x)];
ReplaceAll[%, 1/(1 - x)-> 0];
%% - %;
ReplaceAll[%, 1/(1 - x)-> 1];
QM[%];
Ca[%];
Cf[%];
ReplaceAll[%, Log[1 - x]-> 0];
ReplaceAll[%, x-> 1];
SequenceForm["[The 1/(1 - x) coefficient A = ", N[%, 9], "]"]
ReplaceAll[%%%%, Log[1 - x]^2-> 0];
ReplaceAll[%, Log[1 - x]^3-> 0];
% - Out[-5];
ReplaceAll[%, Log[1 - x]-> 1];
ReplaceAll[%, x-> 1];
SequenceForm["[The Log[1 - x]/(1 - x) coefficient D1 = ", N[%, 9], "]"]
ReplaceAll[Out[-10], Log[1 - x]^2-> 0];
Out[-11] - %;
```

```
ReplaceAll[%, Log[1 − x]− > 1];
ReplaceAll[%, x− > 1];
SequenceForm[“[The Log[1 − x]^2/(1 − x) coefficient D2 = “, N[%, 9], “]“]
ReplaceAll[Out[−15], Log[1 − x]^3− > 0];
Out[−16] − %;
ReplaceAll[%, Log[1 − x]− > 1];
ReplaceAll[%, x− > 1];
SequenceForm[“[The Log[1 − x]^3/(1 − x) coefficient D3 = “, N[%, 9], “]“]
ReplaceAll[Out[−20], Log[1 − x]^4− > 0];
Out[−21] − %;
ReplaceAll[%, Log[1 − x]− > 1];
ReplaceAll[%, x− > 1];
SequenceForm[“[The Log[1 − x]^4/(1 − x) coefficient D4 = “, N[%, 9], “]“]
(∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗Here the algorithm for the coefficient B ∗ ∗ ∗ ∗ ∗∗)
Collect[F[x], DiracDelta[1 − x]];
ReplaceAll[%, DiracDelta[1 − x]− > 0];
%% − %;
ReplaceAll[%, DiracDelta[1 − x]− > 1];
QM[%];
Ca[%];
Cf[%];
ReplaceAll[%, x− > 1];
SequenceForm[“[The DiracDelta coefficient B = “, N[%, 9], “]“]
(∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ Here the algorithm for the coefficients Cn ∗ ∗∗)
Collect[F[x], 1/(1 − x)];
ReplaceAll[%, 1/(1 − x)− > 0];
ReplaceAll[%, DiracDelta[1 − x]− > 0];
ReplaceAll[%, Log[1 − x]− > 0];
%% − %;
QM[%];
Ca[%];
Cf[%];
Collect[%, Log[1 − x]];
ReplaceAll[%, Log[1 − x]^2− > 0];
ReplaceAll[%, Log[1 − x]^3− > 0];
ReplaceAll[%, Log[1 − x]^4− > 0];
Collect[%, Log[1 − x]];
ReplaceAll[%, Log[1 − x]− > 0];
%% − %;
ReplaceAll[%, Log[1 − x]− > 1];
ReplaceAll[%, x− > 1];
SequenceForm[[, N[%, 9], ]]
Out[−10] − Out[−9];
ReplaceAll[%, Log[1 − x]^2− > 1];
ReplaceAll[%, x− > 1];
```

```
SequenceForm["[The Log[1 − x]^2 coefficient C2 = ", N[%, 9], "]"]
Out[−14] − ReplaceAll[Out[−14], Log[1 − x]^3−> 0];
ReplaceAll[%, Log[1 − x]^3−> 1];
ReplaceAll[%, x−> 1];
SequenceForm["[TheLog[1 − x]^3coefficientC3 = ", N[%, 9], "]"]
Out[−18] − ReplaceAll[Out[−18], Log[1 − x]^4−> 0];
ReplaceAll[%, Log[1 − x]^4−> 1];
ReplaceAll[%, x−> 1];
SequenceForm["[The Log[1 − x]^4 coefficient C4 = ", N[%, 9], "]"]
(*********************Here the algorithm for the coefficient E*****)
Collect[F[x], Log[x]];
ReplaceAll[%, Log[x]−> 0];
Collect[%, 1/x];
ReplaceAll[%, 1/x−> 0];
%% − %;
% * x;
QM[%];
Ca[%];
Cf[%];
ReplaceAll[%, x−> 0];
SequenceForm["[The Log[1 − x]^2/(1 − x) coefficient D2 = ", N[%, 9], "]"]
(*****************Here the algorithm for the coefficients Fn******)
Collect[F[x], 1/x];
ReplaceAll[%, 1/x−> 0];
QM[%];
Ca[%];
Cf[%];
Collect[%, Log[x]];
ReplaceAll[%, Log[x]^2−> 0];
ReplaceAll[%, Log[x]^3−> 0];
ReplaceAll[%, Log[x]^4−> 0];
Collect[%, Log[x]];
ReplaceAll[%, Log[x]−> 0];
%% − %;
ReplaceAll[%, Log[x]−> 1];
ReplaceAll[%, x−> 0];
SequenceForm["[The Log[x] coefficient F1  = ", N[%, 9], "]"];
Out[−10] − Out[−9];
Collect[%, Log[x]];
ReplaceAll[%, Log[x]^2−> 1];
ReplaceAll[%, x−> 0];
SequenceForm["[The Log[x]^2 coefficient F2  = ", N[%, 9], "]"];
Out[−15] − ReplaceAll[Out[−15], Log[x]^3−> 0];
Collect[%, Log[x]];
ReplaceAll[%, Log[x]^3−> 1];
```

```
ReplaceAll[%, x-> 0];
SequenceForm["[The Log[x]^3 coefficient F3 = ", N[%, 9], "]"];
Out[-20] - ReplaceAll[Out[-20], Log[x]^4-> 0];
Collect[%, Log[x]];
ReplaceAll[%, Log[x]^4-> 1];
ReplaceAll[%, x-> 0];
SequenceForm["[The Log[x]^4 coefficient F4 = ", N[%, 9], "]"];
(* * * * * * * * * * * * * * * * * * Here the algorithm for the coefficients Gn * * * * * *)
QM[F[x]];
Cf[%];
Ca[%];
Collect[%, Log[x]];
ReplaceAll[%, Log[x]-> 0];
%% - %;
ReplaceAll[%, Log[x]^3-> 0];
ReplaceAll[%, Log[x]^2-> 0];
ReplaceAll[%, Log[x]-> 1];
Expand[%, x];
ReplaceAll[%, x^-1-> 0];
%% - %;
ReplaceAll[%, x^-1-> 1];
ReplaceAll[%, x-> 0];
SequenceForm["[The Log[x]/x coefficient G1 = ", N[%, 9], "]"]
QM[F[x]];
Cf[%];
Ca[%];
Collect[%, Log[x]];
ReplaceAll[%, Log[x]-> 0];
%% - %;
ReplaceAll[%, Log[x]^2-> 0];
%% - %;
ReplaceAll[%, Log[x]^2-> 1];
Expand[%, x];
ReplaceAll[%, x^-1-> 0];
%% - %;
Expand[%, x];
ReplaceAll[%, x^-1-> 1];
ReplaceAll[%, Log[x]-> 1];
ReplaceAll[%, x-> 0];
SequenceForm["[The Log[x]^2/x coefficient G2 = ", N[%, 9], "]
```