

Version 3 of RunDec and CRunDec

Florian Herren and Matthias Steinhauser

*Institut für Theoretische Teilchenphysik
Karlsruhe Institute of Technology (KIT)
76128 Karlsruhe, Germany*

Abstract

We present new versions of the packages `RunDec` and `CRunDec` which can be used for the running and decoupling of the strong coupling constant and quark masses. Furthermore several conversion formulae for heavy quark masses are implemented. The new versions include five-loop corrections of the QCD beta function and four-loop decoupling effects. Furthermore, various relations between the heavy quark mass defined in the $\overline{\text{MS}}$ and other short-distance schemes are implemented to next-to-next-to-next-to-leading order. We discuss in detail the improvements and provide several examples which show how `RunDec` and `CRunDec` can be used in frequently occurring situations.

PACS numbers: 12.38.-t 12.38.Bx 14.65.-q

Program summary

Title of program: RunDec, CRunDec

Available from: <https://www.ttp.kit.edu/preprints/2017/ttp17-011/>

Computer for which the program is designed and others on which it is operable: Any computer where **Mathematica** or a C++ compiler is available.

Operating system or monitor under which the program has been tested: Linux

No. of bytes in distributed program including test data etc.: 400 000

Distribution format: source code

Keywords: Quantum Chromodynamics, running coupling constant, running quark mass, on-shell mass, $\overline{\text{MS}}$ mass, threshold masses, decoupling of heavy particles

Nature of physical problem: The value for the coupling constant of Quantum Chromodynamics, $\alpha_s^{(n_f)}(\mu)$, depends on the considered energy scale, μ , and the number of active quark flavours, n_f . The same applies to light quark masses, $m_q^{(n_f)}(\mu)$, if they are, e.g., evaluated in the $\overline{\text{MS}}$ scheme. In the programs **RunDec** and **CRunDec** all relevant formulae are collected and various procedures are provided which allow for a convenient evaluation of $\alpha_s^{(n_f)}(\mu)$ and $m_q^{(n_f)}(\mu)$ using the state-of-the-art correction terms. Furthermore, the programs contain several conversion formulae which allow to transform the $\overline{\text{MS}}$ value of a heavy quark into other short-distance values or the on-shell definition.

Method of solution: **CRunDec** is implemented in C++. For the solution of the differential equations an adaptive Runge-Kutta procedure has been implemented. The **Mathematica** version **RunDec** uses function provided by **Mathematica** to solve the differential equations.

Restrictions on the complexity of the problem: It could be that for an unphysical choice of the input parameters the results are nonsensical.

Typical running time: In general the run time for the individual operations is below a millisecond. In some cases it can increase to the order of a second.

1. Introduction

The fundamental parameters of QCD are the quark masses (m_q) and the strong coupling constant (α_s) which are usually defined in the $\overline{\text{MS}}$ scheme. As a consequence their numerical values depend on the renormalization scale μ , which in general is of the same order as the energy scale of the considered process, and the number of active quark flavours n_f . In practice, α_s and m_q are determined for certain values of μ and n_f . Afterwards they enter predictions which require other choices. It is thus crucial to have at hand precise relations for m_q and α_s evaluated at different renormalization scales and different number of active flavours.

For the heavy quark masses there are two widely used renormalization schemes, the on-shell and the $\overline{\text{MS}}$ scheme. Depending upon the considered physical process, there are further short-distance definitions which have advantageous properties and are thus often used to parametrize perturbative predictions. It is important to have available precise relations between the various schemes in order not to lose precision due to the transformations.

The **Mathematica** program `RunDec` [1] and its C++ counterpart `CRunDec` [2] achieve these requirements: they implement the highest available perturbative QCD corrections both for the running and decoupling of α_s and m_q , and for the conversion of the heavy quark masses among different schemes. In this paper we describe versions 3 of `RunDec` and `CRunDec` which contain several improvements. They are described in Section 2. Section 3 contains several practical examples. In this paper we refrain from presenting explicit formulae which are implemented in the routines. They can either be found in Ref. [1], in the original papers where the higher order corrections have been computed, or can easily be extracted from the source codes of `RunDec` and `CRunDec`.

The use of the **Mathematica** version `RunDec` is straightforward. After loading `RunDec.m` all **Mathematica**-modules are available and can immediately be used. `CRunDec` is written in C++ which offers several possibilities to access the implemented functions. In the following we present two skeleton files which exemplify the usage. They can easily be adapted to the problem at hand. On the one hand it is possible to work with pointers to an object of the type `CRunDec` and access the member functions correspondingly:

```
#include <iostream>
#include "CRunDec.h"
using namespace std;
int main(){
    CRunDec * <pointer> = new CRunDec();
    double <result> = <pointer> -> <function>(<parameters>);
    return(0);
}
```

Alternatively, also the following realization is possible:

```
#include <iostream>
#include "CRunDec.h"
using namespace std;
int main(){
```

```

CRunDec <object>;
double <result> = <object>.<function>(<parameters>);
return(0);
}

```

Several explicit examples are discussed in Section 3. They are illustrated with the help of `Mathematica` code; the corresponding C++ code can be found in the ancillary files to this paper. Section 4 contains a brief summary.

2. New and updated routines in `RunDec` and `CRunDec`

All routines connected to the running (of α_s and m_q) and to the decoupling of heavy quarks have been updated. In all of them it is now possible to use five-loop running accompanied by four-loop decoupling. We have also updated the routines establishing the relations between α_s and the QCD scale parameter Λ which allow for a more precise extraction of Λ . The corresponding analytic expressions have been obtained in the following papers: The four-loop decoupling constants for α_s is available from Refs. [3, 4] and the one for the light quark masses has been computed in [5]. The five-loop anomalous dimension of the quark masses has been computed by two groups [6, 7, 8] and the five-loop beta function is available from [9, 10, 11].

As a second major update we have implemented the four-loop relation between the heavy quark masses defined in the $\overline{\text{MS}}$ and on-shell scheme [12, 13]. In this context a couple of new routines have been introduced which allow the transformation between the $\overline{\text{MS}}$ or on-shell and so-called threshold masses. In particular, we have implemented the potential subtracted (PS) [14], the 1S [15, 16, 17] and the renormalon subtracted (RS) [18] masses. In Section 3 we show how the new routines can be used to obtain the $\overline{\text{MS}}$ mass from an experimentally determined threshold mass including the uncertainty on the strong coupling constant.

We refrain from providing analytic results for the new perturbative corrections which have been implemented in `RunDec` and `CRunDec`. One can either find them in the original papers or easily extract them from the `Mathematica` source code. For example, the decoupling constant which establishes the transition from α_s defined in the n_f -flavour theory to α_s in the $(n_f - 1)$ -flavour theory using the $\overline{\text{MS}}$ definition of the heavy quark mass, can be found up to four loops with the help of

```
RunDec'Modules'as6to5ms /. setdec
```

The inverted relations and the ones where different renormalization schemes for the heavy quark mass have been chosen are contained in the variables `as6to5si`, `as6to5os`, `as5to6ms`, `as5to6si` and `as5to6os`. Analytic results up to three loops are given in Eqs. (20) to (25) of Ref. [1]. The coefficients of the QCD β function and the mass anomalous dimension can be extracted from the variables `setbeta` and `setgamma`, respectively.

In the following subsections we provide merely the name of the function in case it has only been extended to higher loop order. For a detailed description of the arguments we refer

to Refs. [1] and [2]. For the new functions and for the functions where additional arguments have been introduced¹ we display the complete function header.

2.1. Running and decoupling of α_s and m_q

The following functions related to the running of α_s and m_q have been updated to five-loop accuracy:

LamImpl, LamExpl, AlphasLam,
 AlphasExact, mMS2mMS, mMS2mSI, AsmMSrunexact,
 mMS2mRGI, mMS2mRGImod.

Correspondingly we implemented the four-loop decoupling relations into the following routines:

DecAsUpOS, DecAsDownOS, DecAsUpMS, DecAsDownMS, DecAsUpSI, DecAsDownSI,
 DecMqUpOS, DecMqDownOS, DecMqUpMS, DecMqDownMS, DecMqUpSI, DecMqDownSI,
 DecLambdaUp, DecLambdaDown.

As a consequence also the functions A1L2A1H and A1H2A1L, which perform a combined running and decoupling can be used at five-loop order.

2.2. Quark mass relations

The functions implementing the transitions between the heavy quark masses defined in the on-shell and $\overline{\text{MS}}$ scheme have been extended to four loops. Since the non-logarithmic four-loop term is not known analytically but has an uncertainty of the order of 0.2% [13] we have introduced an additional argument for the following functions where this can be taken into account

```
mOS2mMS[mOS_,mq_,asmu_,mu_,nnf_,loops_,fdelm_]
mOS2mMS[mOS_,mq_,asmu_,mu_,nnf_,loops_]
mOS2mSI[mOS_,mq_,asM_,nnf_,loops_,fdelm_]
mOS2mSI[mOS_,mq_,asM_,nnf_,loops_]
mMS2mOS[mMS_,mq_,asmu_,mu_,nnf_,loops_,fdelm_]
mMS2mOS[mMS_,mq_,asmu_,mu_,nnf_,loops_]

```

Here `fdelm` multiplies the non-logarithmic four-loop coefficient of order α_s^4 . The default value is 1. A 0.2% variation is obtained with `fdelm=0.998`. The function prototype of the corresponding routines in `CRunDec` read

```
double mOS2mMS(double mOS, std::pair<double,double>* mq,
               double asmu, double mu, int nf, int nloops, double fdelm);
double mOS2mMS(double mOS, std::pair<double,double>* mq,
               double asmu, double mu, int nf, int nloops);
double mOS2mSI(double mOS, std::pair<double,double>* mq,
               double asM, int nf, int nloops, double fdelm);
double mOS2mSI(double mOS, std::pair<double,double>* mq,

```

¹Note that the functions have been overloaded such that it is still possible to call them with the old argument list. The only exception is the argument handling the light quark masses in the relation between the $\overline{\text{MS}}$ and on-shell masses as implemented in `CRunDec`, see Subsection 2.2.

```

        double asM,int nf, int nloops);
double mM2mOS(double mMS, std::pair<double,double>* mq,
        double asmu, double mu, int nf, int nloops, double fdelm);
double mM2mOS(double mMS, std::pair<double,double>* mq,
        double asmu, double mu, int nf, int nloops);

```

The arguments of the following functions are unchanged:
mOS2mMSrun, mOS2mMSit, mM2mOSrun.

A further improvement in these functions is related to the light quark mass effects. Up to now they have only been implemented up to two-loop order [19]. We have extended the implementation to three loops using the results of Ref. [20]. The light quark masses, m_q , are renormalized in the $\overline{\text{MS}}$ scheme at their own renormalization scale μ_q . In the previous versions of RunDec the argument mq was a list containing the light quark masses. For example, finite charm and bottom quark masses in the top quark mass relation are taken into account via {4.163,0.986}. After including the three-loop corrections the scale μ_q has to be specified and the argument mq has been extended to {{4.163,4.163},{0.986,3.0}} to specify $m_b(m_b)$ and $m_c(3 \text{ GeV})$. Note that it is still possible to provide a non-nested list. In that case the quark masses are interpreted as $m_q(m_q)$.

In the C++ version modifications in the parameter passing were necessary, which are incompatible with older versions in case non-zero light quark masses are used. In version 3 an array of pairs of double variables has to be passed instead of a double array with four elements. The first element of each pair is the mass, the second is the scale. Note that the array must either contain four elements or can be a null pointer.

The four-loop relation between the $\overline{\text{MS}}$ and on-shell quark mass can be used to derive the relations to the so-called threshold masses to the corresponding accuracy (see Ref. [13] for more details). In particular, we have implemented the following relations to the potential subtracted mass (PS), to the 1S mass (1S), to the renormalon subtracted mass (RS), and to a modified version of the latter (RSp):

```

mOS2mPS[mOS_,mq_,asmu_,mu_,muf_,nl_,loops_]
mMS2mPS[mMS_,mq_,asmu_,mu_,muf_,nl_,loops_]
mMS2mPS[mMS_,mq_,asmu_,mu_,muf_,nl_,loops_,fdelm_]
mPS2mMS[mPS_,mq_,asnlmu_,mu_,muf_,nl_,loops_]
mPS2mMS[mPS_,mq_,asnlmu_,mu_,muf_,nl_,loops_,fdelm_]
mPS2mSI[mPS_,mq_,asfct_,muf_,nl_,loops_]
mPS2mSI[mPS_,mq_,asfct_,muf_,nl_,loops_,fdelm_]

mOS2m1S[mOS_,mq_,asmu_,mu_,nl_,loops_]
mMS2m1S[mMS_,mq_,asmu_,mu_,nl_,loops_]
mMS2m1S[mMS_,mq_,asmu_,mu_,nl_,loops_,fdelm_]
m1S2mMS[m1S_,mq_,asnlmu_,mu_,nl_,loops_]
m1S2mMS[m1S_,mq_,asnlmu_,mu_,nl_,loops_,fdelm_]
m1S2mSI[m1S_,mq_,asfct_,nl_,loops_]
m1S2mSI[m1S_,mq_,asfct_,nl_,loops_,fdelm_]

mOS2mRS[mOS_,mq_,asmu_,mu_,nuf_,nl_,loops_]
mMS2mRS[mMS_,mq_,asmu_,mu_,nuf_,nl_,loops_]
mMS2mRS[mMS_,mq_,asmu_,mu_,nuf_,nl_,loops_,fdelm_]

```

```

mRS2mMS [mRS_,mq_,asnlmu_,mu_,nuf_,nl_,loops_]
mRS2mMS [mRS_,mq_,asnlmu_,mu_,nuf_,nl_,loops_,fdelm_]
mRS2mSI [mRS_,mq_,asfct_,nuf_,nl_,loops_]
mRS2mSI [mRS_,mq_,asfct_,nuf_,nl_,loops_,fdelm_]

```

```

mOS2mRSp [mOS_,mq_,asmu_,mu_,nuf_,nl_,loops_]
mMS2mRSp [mMS_,mq_,asmu_,mu_,nuf_,nl_,loops_]
mMS2mRSp [mMS_,mq_,asmu_,mu_,nuf_,nl_,loops_,fdelm_]
mRSp2mMS [mRS_,mq_,asnlmu_,mu_,nuf_,nl_,loops_]
mRSp2mMS [mRS_,mq_,asnlmu_,mu_,nuf_,nl_,loops_,fdelm_]
mRSp2mSI [mRS_,mq_,asfct_,nuf_,nl_,loops_]
mRSp2mSI [mRS_,mq_,asfct_,nuf_,nl_,loops_,fdelm_]

```

In the C++ version the function prototypes are given by

```

double mOS2mPS(double mOS, std::pair<double,double>* mq, double asmu,
               double mu, double muf, int nl, int nloops);
double mMS2mPS(double mMS, std::pair<double,double>* mq, double asmu,
               double mu, double muf, int nl, int nloops);
double mMS2mPS(double mMS, std::pair<double,double>* mq, double asmu,
               double mu, double muf, int nl, int nloops, double fdelm);
double mPS2mMS(double mPS, std::pair<double,double>* mq, double asmu,
               double mu, double muf, int nl, int nloops);
double mPS2mMS(double mPS, std::pair<double,double>* mq, double asmu,
               double mu, double muf, int nl, int nloops, double fdelm);
double mPS2mSI(double mPS, std::pair<double,double>* mq, double (*as)(double),
               double muf, int nl, int nloops);
double mPS2mSI(double mPS, std::pair<double,double>* mq, double (*as)(double),
               double muf, int nl, int nloops, double fdelm);

double mOS2m1S(double mOS, std::pair<double,double>* mq, double asmu,
               double mu, int nl, int nloops);
double mMS2m1S(double mMS, std::pair<double,double>* mq, double asmu,
               double mu, int nl, int nloops);
double mMS2m1S(double mMS, std::pair<double,double>* mq, double asmu,
               double mu, int nl, int nloops, double fdelm);
double m1S2mMS(double m1S, std::pair<double,double>* mq, double asmu,
               double mu, int nl, int nloops);
double m1S2mMS(double m1S, std::pair<double,double>* mq, double asmu,
               double mu, int nl, int nloops, double fdelm);
double m1S2mSI(double m1S, std::pair<double,double>* mq,
               double (*as)(double), int nl, int nloops);
double m1S2mSI(double m1S, std::pair<double,double>* mq,
               double (*as)(double), int nl, int nloops, double fdelm);

double mOS2mRS(double mOS, std::pair<double,double>* mq, double asmu,
               double mu, double nuf, int nl, int nloops);
double mMS2mRS(double mMS, std::pair<double,double>* mq, double asmu,
               double mu, double nuf, int nl, int nloops);
double mMS2mRS(double mMS, std::pair<double,double>* mq, double asmu,
               double mu, double nuf, int nl, int nloops, double fdelm);
double mRS2mMS(double mRS, std::pair<double,double>* mq, double asmu,
               double mu, double nuf, int nl, int nloops);

```

```

double mRS2mMS(double mRS, std::pair<double,double>* mq, double asmu,
               double mu, double nuf, int nl, int nloops, double fdelm);
double mRS2mSI(double mRS, std::pair<double,double>* mq, double (*as)(double),
               double nuf, int nl, int nloops);
double mRS2mSI(double mRS, std::pair<double,double>* mq, double (*as)(double),
               double nuf, int nl, int nloops, double fdelm);

double mOS2mRSp(double mOS, std::pair<double,double>* mq, double asmu,
                double mu, double nuf, int nl, int nloops);
double mM2mRSp(double mMS, std::pair<double,double>* mq, double asmu,
                double mu, double nuf, int nl, int nloops);
double mM2mRSp(double mMS, std::pair<double,double>* mq, double asmu,
                double mu, double nuf, int nl, int nloops, double fdelm);
double mRSp2mMS(double mRS, std::pair<double,double>* mq, double asmu,
                double mu, double nuf, int nl, int nloops);
double mRSp2mMS(double mRS, std::pair<double,double>* mq, double asmu,
                double mu, double nuf, int nl, int nloops, double fdelm);
double mRSp2mSI(double mRS, std::pair<double,double>* mq, double (*as)(double),
                double nuf, int nl, int nloops);
double mRSp2mSI(double mRS, std::pair<double,double>* mq, double (*as)(double),
                double nuf, int nl, int nloops, double fdelm);

```

The meaning of the arguments is in close analogy to the function implementing the $\overline{\text{MS}}$ -on-shell relation, see also the description of the individual modules in the `Mathematica` version. In particular, also the argument `mq` for light quark mass effects is present. However, in the relations involving threshold masses it is not active.

For the relations involving PS and RS masses additional factorization scales are introduced which are denoted by `muf` and `nuf`, respectively.² Note that for technical reasons it is necessary to provide just the function name and not the numerical value of α_s for those functions which convert to the scale-invariant mass (“SI”). This is denoted by `asfact` in the argument list. In the C++ version a pointer to a function taking the renormalization scale μ as argument has to be passed instead.

In `RunDec` there is also the routine `AsRunDec` which extracts from the renormalization scales specified in the input the necessary information about the decoupling steps. Since for some cases this might lead to inconsistencies we recommend not to use this module but use instead `A1L2A1H` or `A1H2A1L`, see also the examples in Section 3. For this reason we do not maintain this routine. This also concerns the versions of `mOS2mMS` and `mMS2mOS` with the headers `mOS2mMS[mOS_,nnf_,loops_]` and `mMS2mOS[mMS_,nnf_,loops_]`.

3. Useful examples

In this section we present various concrete examples which are useful for many everyday situations. We discuss in detail numerical effects and provide the source codes which can easily be adapted to the own programs.

²See the original publications [14] and [18] for more details.

The examples require various input values which we choose as follows [21, 22, 23]

<p>(asMz) $\alpha_s^{(5)}(M_Z) = 0.1181 \pm 0.0011$</p> <p>(Mz) $M_Z = 91.1876 \pm 0.0021$ GeV</p> <p>(muc) $m_c(m_c) = 1.279 \pm 0.013$ GeV</p> <p>(mub) $m_b(m_b) = 4.163 \pm 0.016$ GeV</p> <p>(Mc) $M_c = 1.5$ GeV</p> <p>(Mt) $M_t = 173.21 \pm 0.87$ GeV</p>	<p>(asMtau) $\alpha_s^{(5)}(m_\tau) = 0.332 \pm 0.016$</p> <p>(Mh) $M_H = 125.09 \pm 0.24$ GeV</p> <p>(mc3) $m_c(3 \text{ GeV}) = 0.986 \pm 0.013$ GeV</p> <p>(Mtau) $m_\tau = 1.77686 \pm 0.00012$ GeV</p> <p>(Mb) $M_b = 4.8$ GeV</p>
--	--

(1)

The symbols inside the brackets show the notation used in `RunDec` and `CRunDec`. The central values (without uncertainties) are implemented in the variable `NumDef` (both in `RunDec` and `CRunDec`).

3.1. Running and decoupling

3.1.1. α_s at high energies

As a typical example we consider the extraction of α_s from the two- to three-jet event ratio as, for example, performed by CMS [24]. For $\mu = 896$ GeV the value $\alpha_s(\mu) = 0.0889 \pm 0.0034$ has been obtained. Interpreting this result as the strong coupling in a six-flavour theory we can use

```

muin = 896;
as6mu = 0.0889;
as5Mz = ALH2A1L[as6mu, muin, {{6, Mt /. NumDef, 2*Mt /. NumDef}}, Mz /. NumDef, 5];

```

and obtain $\alpha_s^{(5)}(M_Z) = 0.1170$. Performing the decoupling for $\mu_{\text{dec}}^{(t)} = M_t$ instead of $\mu_{\text{dec}}^{(t)} = 2M_t$ does not change the provided four digits. The same is true if four-loop (instead of five-loop) accuracy is used for the running. If, however, `as6mu` is interpreted in the five-flavour theory one obtains $\alpha_s^{(5)}(M_Z) = 0.1199$ after using

```

AlphasExact[as6mu, muin, Mz /. NumDef, 5, 5]

```

Note that `ALH2A1L` implements the on-shell decoupling constants. In case the $\overline{\text{MS}}$ or scale-invariant (“SI”) version shall be used one has to combine `AlphasExact` and `DecAsDownMS` or `DecAsDownSI` to obtain $\alpha_s^{(5)}(M_Z)$.

As a further example we consider the recent determination of α_s from the double-differential inclusive jet cross section as reported by the CMS collaboration in Ref. [25]. The measured transverse momentum (p_T) interval is divided into several ranges and in each of them the strong coupling constant is extracted. In the last row of Tab. 5 of Ref. [25] one finds $\alpha_s^{(5)}(M_Z) = 0.1162_{-0.0062}^{+0.0070}$ which is obtained from an energy scale of $Q = 1508.04$ GeV. In Ref. [25] two-loop running with five active flavours has been used to obtain $\alpha_s^{(5)}(Q) = 0.0822_{-0.0031}^{+0.0034}$. Within `RunDec` this result can be obtained via

```

asmz      = 0.1162;
asmzplus  = asmz + 0.0070;
asmzminus = asmz - 0.0062;
Q         = 1508.04;

```

```
nloops = 2;
```

```

asQ      = AlphasExact[asmz, Mz /. NumDef, Q, 5, nloops];
delasQplus = Abs[asQ - AlphasExact[asmzplus, Mz /. NumDef, Q, 5, nloops]];
delasQminus = Abs[asQ - AlphasExact[asmzminus, Mz /. NumDef, Q, 5, nloops]];
truncuncert = Abs[asQ - AlphasExact[asmz, Mz /. NumDef, Q, 5, nloops-1]];
totuncertplus = Sqrt[truncuncert^2 + delasQplus^2];
totuncertminus = Sqrt[truncuncert^2 + delasQminus^2];

```

where we have introduced an additional uncertainty due to the truncation of the perturbative series. We obtain $\alpha_s^{(5)}(Q) = 0.0822_{-0.0032}^{+0.0035}$. Choosing five-loop accuracy, i.e. `nloops = 5`, leads to $\alpha_s^{(5)}(Q) = 0.0822_{-0.0031}^{+0.0034}$ and thus has only a minor effect on the result.

At energy scales above a few hundred GeV also the top quark is an active quark flavour and, in principle, one has to include decoupling effects when relating $\alpha_s^{(5)}(M_Z)$ to $\alpha_s^{(6)}(Q)$. With the help of `RunDec` this can be taken into account as follows

```
nloops = 5;
```

```

dec      = {{6, Mt /. NumDef, 2*Mt /. NumDef}};
asQ      = All2AlH[asmz, Mz /. NumDef, dec, Q, nloops];
delasQplus = Abs[asQ - All2AlH[asmzplus, Mz /. NumDef, dec, Q, nloops]];
delasQminus = Abs[asQ - All2AlH[asmzminus, Mz /. NumDef, dec, Q, nloops]];
truncuncert = Abs[asQ - All2AlH[asmz, Mz /. NumDef, dec, Q, nloops-1]];
totuncertplus = Sqrt[truncuncert^2 + delasQplus^2];
totuncertminus = Sqrt[truncuncert^2 + delasQminus^2];

```

It is furthermore possible to vary the decoupling scale between $M_t/f < \mu_{\text{dec}}^{(t)} < fM_t$ where f is often chosen between 2 and 4. Within `Mathematica` this can be realized for $f = 4$ via

```

f      = 4;
step   = 5;
tmp    = Map[ All2AlH[asmz, N[Mz /. NumDef], {{6, N[Mt /. NumDef], #}}, Q, nloops]&,
             Range[N[Mt /. NumDef]/f, f*N[Mt /. NumDef], step]];
scaleuncert = Max[ tmp ] - Min[ tmp ];

```

Using five-loop accuracy we arrive at $\alpha_s^{(6)}(Q) = 0.0840_{-0.0033}^{+0.0036}$ which differs significantly from the value for $\alpha_s^{(5)}(Q)$ given above. Here, the scale uncertainty (“`scaleuncert`”) is negligible. For `nloops = 2` we obtain $\alpha_s^{(6)}(Q) = 0.0839_{-0.0033}^{+0.0036}$ where 10% of the error is due to scale uncertainty.

3.1.2. Compute $\alpha_s^{(5)}(M_Z)$ from $\alpha_s^{(3)}(m_\tau)$

As a further example where higher order corrections are crucial, we consider the extraction of $\alpha_s^{(5)}(M_Z)$ from τ lepton decays. In this case one has to take into account the effect of two flavour thresholds when running from m_τ to M_Z . Using `AlphasExact` and `DecAsUpSI` we can calculate $\alpha_s^{(5)}(M_Z)$ to n -loop accuracy with the help of

```

As5MZ[asin_?NumberQ, mu_?NumberQ, thr1_?NumberQ, thr2_?NumberQ, n_?IntegerQ] := Module[
  {as3c, as4c, as4b, as5b, as5, as5mz},
  as3c = AlphasExact[asin, mu, thr1, 3, n];
  as4c = DecAsUpSI[as3c, muc /. NumDef, thr1, 3, n];
  as4b = AlphasExact[as4c, thr1, thr2, 4, n];
  as5b = DecAsUpSI[as4b, mub /. NumDef, thr2, 4, n];
  as5mz = AlphasExact[as5b, thr2, Mz /. NumDef, 5, n];
  Return[as5mz];
];

```

This module can be used to obtain $\alpha_s^{(5)}(M_Z)$ including the uncertainties due to truncation of the perturbative series and the scale variation introduced by matching the three- and four-, as well as four- and five-flavour theory. Following Ref. [23] we calculate the truncation uncertainty by taking the difference between n -loop and $(n - 1)$ -loop results for $\alpha_s^{(5)}(M_Z)$. We obtain the scale uncertainties by varying the decoupling scale between $\mu_{\text{dec}}/3$ and $3\mu_{\text{dec}}$ around $\mu_{\text{dec}}^{(c)} = 3.0$ GeV for charm and $\mu_{\text{dec}}^{(b)} = m_b(m_b)$ for bottom. Using $\alpha_s^{(3)}(m_\tau) = 0.332 \pm 0.016$ [23], the scale invariant charm and bottom quark masses, and the following RunDec commands

```

mudecc      = 3.0;
mudecb      = mub /. NumDef;
asmtauerror = 0.016;
as5         = As5MZ[asMtau /. NumDef, Mtau /. NumDef, mudecc, mudecb, 5];
f          = 3;
step       = 1;
truncerr   = Abs[as5 - As5MZ[asMtau /. NumDef, Mtau /. NumDef, mudecc, mudecb, 5-1]];
tmp        = Map[ As5MZ[asMtau /. NumDef, Mtau /. NumDef, #, mudecb, 5]&,
  Range[mudecc/f, f*mudecc, step]];
scaleuncertc = Max[ tmp ] - Min[ tmp ];
tmp         = Map[ As5MZ[asMtau /. NumDef, Mtau /. NumDef, mudecc, #, 5]&,
  Range[mudecb/f, f*mudecb, step]];
scaleuncertb = Max[ tmp ] - Min[ tmp ];
totaluncert  = Sqrt[truncerr^2 + scaleuncertc^2 + scaleuncertb^2];
expuncert    = Abs[as5 - As5MZ[(asMtau /. NumDef)-asmtauerror, Mtau /. NumDef,
  mudecc, mudecb, 5]];

```

one obtains $\alpha_s^{(5)}(M_Z) = 0.1201 \pm 0.0019 \pm 0.0003$. The first uncertainty reflects the error of $\alpha_s^{(3)}(m_\tau)$ and the second one the truncation and scale uncertainties which amount to $\delta_{\text{trunc}} = 0.00005$, $\delta_b = 0.00003$ and $\delta_c = 0.00030$, respectively. It is clearly dominated by the scale uncertainty due to the decoupling of the charm quark. In Tab. 1 $\sqrt{\delta_{\text{trunc}}^2 + \delta_b^2 + \delta_c^2}$ is shown as a function of the number of loops used for the running of α_s . One observes a dramatic reduction after including three loops. Afterwards the improvements are significantly smaller than the uncertainty of $\alpha_s^{(3)}(m_\tau)$.

Fig. 1 shows the dependence of $\alpha_s^{(5)}(M_Z)$ on the decoupling scales for the charm (left) and bottom (right) quark mass using three-, four- and five-loop running. It is easily obtained with the help of **As5MZ** introduced above. In the case of the bottom quark one observes an almost $\mu_{\text{dec}}^{(b)}$ -independent behaviour at five loops. In fact, the variation of $\alpha_s^{(5)}(M_Z)$ in the considered range of $\mu_{\text{dec}}^{(b)}$ is below 0.03%. Also in the case of the charm quark the

# loops	$\delta = \sqrt{\delta_{\text{trunc}}^2 + \delta_b^2 + \delta_c^2}$
2	0.0054
3	0.0009
4	0.0005
5	0.0003

Table 1: Uncertainty induced by truncation (δ_{trunc}) and decoupling scale uncertainties (δ_b and δ_c) in the renormalization group running of α_s from m_τ to M_Z as a function of the number of loops used for the beta function.

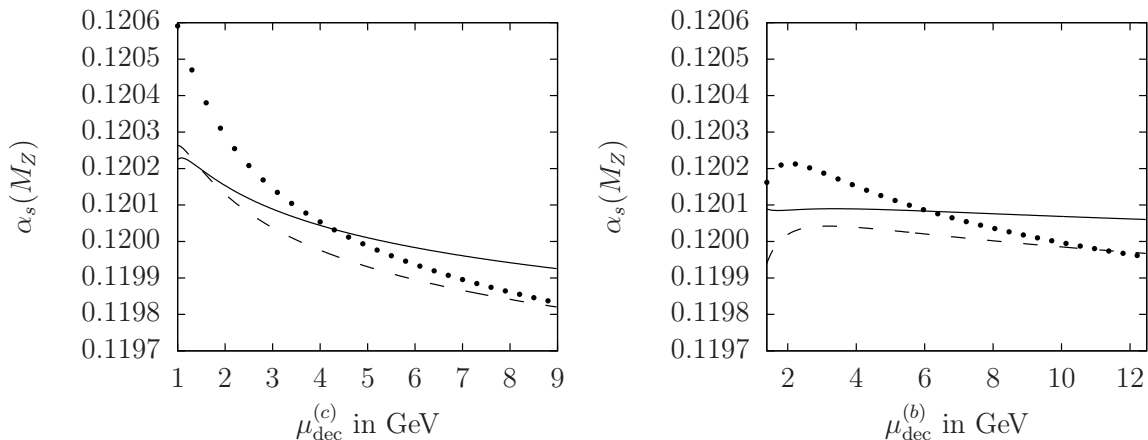


Figure 1: $\alpha_s^{(5)}(M_Z)$ calculated from $\alpha_s^{(3)}(m_\tau)$ with the charm (left) and bottom (right) quark decoupling scale varied between 1 GeV and 9 GeV for charm and $m_b/3 \approx 1.4$ GeV and $3m_b \approx 12.6$ GeV for bottom. The dotted, dashed and solid curves correspond to three-, four- and five-loop running.

curves become flatter when including higher order quantum correction. However, at the five-loop order the variation of $\alpha_s^{(5)}(M_Z)$ still amounts to about 0.3% which corresponds to $\delta_c = 0.00030$ as mentioned above.

3.1.3. Λ and α_s

RunDec and CRunDec implement two different routines which allow to obtain the QCD scale parameter $\Lambda^{(n_f)}$ for a given $\alpha_s^{(n_f)}(\mu)$. The numerical results are slightly different since one of the routines uses an explicit solution of the form $\Lambda^{(n_f)} = f(\alpha_s^{(n_f)})$ whereas the other looks for an implicit solution of $\alpha_s^{(n_f)} = f(\Lambda^{(n_f)})$. Using

```
Do[ expl[i] = LamExpl[asMz /. NumDef, Mz /. NumDef, 5, i];
    impl[i] = LamImpl[asMz /. NumDef, Mz /. NumDef, 5, i];
    diff[i] = Abs[expl - impl]
    ,{i,1,5} ]
```

we obtain the values shown in Tab. 2. As expected, the difference decreases with increasing

# loops	LamExpl	LamImpl	difference
1	88.35 MeV	88.35 MeV	0 MeV
2	209.86 MeV	227.51 MeV	17.65 MeV
3	210.10 MeV	209.54 MeV	0.56 MeV
4	209.78 MeV	209.53 MeV	0.25 MeV
5	209.80 MeV	209.87 MeV	0.07 MeV

Table 2: $\Lambda^{(5)}$ calculated from $\alpha_s^{(5)}(M_Z)$ using both the explicit and implicit routine. The difference decreases with increasing number of loops.

number of loops.³

In the next example we use the PDG value for the strong coupling constant (see Eq. (1)) and we calculate $\Lambda^{(n_f)}$ for $n_f = 3, 4, 5, 6$. To do so we provide the following routine, which requires as input the strong coupling constant and its uncertainty for n_f active flavours at the scale μ . Furthermore the number of loops has to be specified.

```
CalcLam[as_, aserr_, mu_, nf_, nloops_] := Module[{expl4, impl4, expl, impl,
  explplus, implplus, explminus, implminus, mean, mean4,
  truncuncert, diffuncert, expuncert},
  expl = LamExpl[as, mu, nf, nloops];
  impl = LamImpl[as, mu, nf, nloops];
  expl4 = LamExpl[as, mu, nf, nloops-1];
  impl4 = LamImpl[as, mu, nf, nloops-1];
  explplus = Abs[LamExpl[as + aserr, mu, nf, nloops] - expl];
  implplus = Abs[LamImpl[as + aserr, mu, nf, nloops] - impl];
  explminus = Abs[LamExpl[as - aserr, mu, nf, nloops] - expl];
  implminus = Abs[LamImpl[as - aserr, mu, nf, nloops] - impl];
  mean = (expl + impl)/2;
  mean4 = (expl4 + impl4)/2;
  truncuncert = Abs[mean - mean4];
  diffuncert = Abs[expl - impl]/2;
  expuncert = Max[explplus, implplus, explminus, implminus];
  Return[{mean, expuncert, truncuncert, diffuncert}];
];
```

The uncertainties introduced through truncation of the perturbative series is taken to be the difference between n - and $(n - 1)$ -loop results. We use both `LamExpl` and `LamImpl` and take the mean value as the result and assign an uncertainty due to half of the difference of both methods. For the uncertainty induced by $\delta\alpha_s$ we cite the larger one obtained by the two methods. For $n_f = 5$ we obtain $\Lambda^{(5)}$ from

```
lam5 = CalcLam[asMz /. NumDef, 0.0011, Mz /. NumDef, 5, 5];
```

For the other values of n_f the code can be found in the supplementary files. Let us only mention that, in order to obtain $\alpha_s^{(n_f)}(\mu)$ we perform a decoupling of the top, bottom

³The one-loop relation trivially leads to identical results for `LamExpl` and `LamImpl`.

n_f	$\Lambda^{(n_f)}$	exp. uncertainty	δ_{trunc}	δ_{diff}
3	336 MeV	17 MeV	1 MeV	1 MeV
4	292 MeV	16 MeV	2 MeV	1 MeV
5	210 MeV	13 MeV	0 MeV	0 MeV
6	89 MeV	6 MeV	0 MeV	0 MeV

Table 3: $\Lambda^{(n_f)}$ calculated from $\alpha_s^{(5)}(M_Z) = 0.1181 \pm 0.0011$. The uncertainties due to truncation of the perturbative series and due to the two different methods are shown. For our analysis we use five-loop accuracy.

and charm quarks at the scales M_t , $m_b(m_b)$ and 3 GeV. The results for $\Lambda^{(n_f)}$ and the corresponding uncertainties can be found in Tab. 3. Our central values and uncertainties for $n_f = 4, 5, 6$ are identical to the ones from PDG [21]. For $n_f = 3$ a minor difference is observed which is due to the different decoupling scale chosen for the charm quark. In Ref. [21] $\mu_{\text{dec}}^{(c)} = 1.3$ GeV has been used which results in $\Lambda = 332 \pm 17$ MeV. We can reproduce this result using our code with four-loop accuracy. Since $\alpha_s^{(3)}(\mu_{\text{dec}}^{(c)} = 1.3 \text{ GeV})$ is already quite large we prefer to decouple the charm quark for $\mu_{\text{dec}}^{(c)} = 3$ GeV.

In a recent publication the ALPHA collaboration has presented results for $\Lambda^{(3)}$ and $\alpha_s^{(5)}(M_Z)$ [26]. In the following example we use $\Lambda^{(3)} = (332 \pm 14)$ MeV [26], compute $\alpha_s^{(5)}(M_Z)$ using two approaches, and compare to Ref. [26]. In the first one we compute $\Lambda^{(4)}$ and $\Lambda^{(5)}$ and finally extract $\alpha_s^{(5)}(M_Z)$. The corresponding RunDec commands read

```
Lambda3 = 0.332;
nloops = 5;
Lambda4 = DecLambdaUp[Lambda3, muc /. NumDef, 3, nloops];
Lambda5 = DecLambdaUp[Lambda4, mub /. NumDef, 4, nloops];
aslam = AlphasLam[Lambda5, Mz /. NumDef, 5, nloops];
```

We obtain $\alpha_s^{(5)}(M_Z) = 0.1180$ in excellent agreement with

$$\alpha_s^{(5)}(M_Z) = 0.1179 \pm 0.0010 \pm 0.0002 \quad (2)$$

from Ref. [26]. We can also reproduce the uncertainties which origin from the one of $\Lambda^{(3)}$ and the use of perturbation theory (see the `Mathematica` and `C++` code in the ancillary files.).

In the second approach one can use `AlphasLam` to obtain $\alpha_s^{(3)}$ and calculate $\alpha_s^{(5)}(M_Z)$ by running and decoupling as described in Section 3.1.2. Using five-loop accuracy, we obtain

$$\alpha_s^{(5)}(M_Z) = 0.1177 \pm 0.0009 \pm 0.0002 \pm 0.0003, \quad (3)$$

where the last two uncertainties are due to truncation of the perturbative series and the scale uncertainty. Both the central value and the three uncertainties are obtained with the help of

```
Lambda3 = 0.332;
```

```

Lambda3plus = Lambda3 + 0.014;
Lambda3minus = Lambda3 - 0.014;
mudecc      = 3.0;
mudecb      = mub /. NumDef;
nloops      = 5;
f = 3;

As3Lambda[lamin_?NumberQ, scale1_?NumberQ, scale2_?NumberQ, n_?IntegerQ]
:= Module[{as31, as32},
  as31 = AlphasLam[lamin, scale1, 3, n];
  as32 = AlphasExact[as31, scale1, scale2, 3, n];
  Return[as32];
];

As5MZfromLambda[lamin_?NumberQ, scale1_?NumberQ, thr1_?NumberQ,
  thr2_?NumberQ, n_?IntegerQ] := Module[{as3, as5},
  as3 = As3Lambda[lamin, scale1, thr1, n];
  as5 = As5MZ[as3, thr1, thr1, thr2, n];
  Return[as5];
];

as5 = As5MZfromLambda[Lambda3, mudecc, mudecc, mudecb, nloops];
expuncert = Max[
  Abs[as5 - As5MZfromLambda[Lambda3plus, mudecc, mudecc, mudecb, nloops]],
  Abs[as5 - As5MZfromLambda[Lambda3minus, mudecc, mudecc, mudecb, nloops]] ];
truncuncert = Abs[as5 - As5MZfromLambda[Lambda3, mudecc, mudecc, mudecb, nloops-1]];
step = 1;
tmp = Map[ As5MZfromLambda[Lambda3, #, mudecc, mudecb, nloops]&,
  Range[mudecc/f, f*mudecc, step]];
scaleuncertlam = Max[ tmp ] - Min[ tmp ];
tmp = Map[ As5MZfromLambda[Lambda3, mudecc, #, mudecb, nloops]&,
  Range[mudecc/f, f*mudecc, step]];
scaleuncertc = Max[ tmp ] - Min[ tmp ];
tmp = Map[ As5MZfromLambda[Lambda3, mudecc, mudecc, #, nloops]&,
  Range[mudecb/f, f*mudecb, step]];
scaleuncertb = Max[ tmp ] - Min[ tmp ];
totalscaleuncert = Sqrt[scaleuncertc^2 + scaleuncertb^2 + scaleuncertlam^2];

```

We have introduced `as5MZfromLambda` which calculates $\alpha_s^{(3)}(\mu)$. It is based on `as5MZ`, which has been introduced in Section 3.1.2 to compute $\alpha_s^{(5)}(M_Z)$ from $\alpha_s^{(3)}(m_\tau)$. Note that the central values in Eqs. (2) and (3) agree within the second uncertainty of Eq. (2) which is due to the use of perturbation theory when relating $\Lambda^{(3)}$ to $\Lambda^{(5)}$. Note, however, that the corresponding uncertainty in our second method, which is the one we recommend for such calculations, amounts to $\sqrt{0.0002^2 + 0.0003^2} \approx 0.0004$ and is twice as large.

As a last example in this subsection we use $\Lambda^{(3)}$ from Ref. [26] and compute $\alpha_s^{(3)}(m_\tau)$ with the help of the following commands

```

astau      = AlphasLam[Lambda3, Mtau /. NumDef, 3, 5];
astau4     = AlphasLam[Lambda3, Mtau /. NumDef, 3, 4];
astaplus   = Abs[AlphasLam[Lambda3plus, Mtau /. NumDef, 3, 5] - astau];
astaminus  = Abs[AlphasLam[Lambda3minus, Mtau /. NumDef, 3, 5] - astau];

```

```
truncuncert = Abs[astau - astau4];
expuncert   = Max[astaplus, astauminus];
```

We obtain $\alpha_s^{(3)}(m_\tau) = 0.3119 \pm 0.0073 \pm 0.0040$, where the first uncertainty is due to the one of $\Lambda^{(3)}$ and the second one due to the truncation of the perturbative series. For comparison, we also show the value obtained from computing in a first step $\alpha_s^{(3)}(3 \text{ GeV})$ and the subsequent running from 3 GeV to m_τ . We obtain $\alpha_s^{(3)}(m_\tau) = 0.3125 \pm 0.0074 \pm 0.0019$. The results are consistent with each other. However, the uncertainties from the truncation of the perturbative series differ by a factor two which indicates that α_s is relatively large. It seems to be advantageous to use the relation between Λ and α_s at higher scales and use the renormalization group running to arrive at $\mu = m_\tau$.

3.1.4. Running and decoupling for m_b

For the decay of the Standard Model Higgs boson to bottom quarks it is necessary to evaluate the $\overline{\text{MS}}$ bottom quark mass at an energy scale which is of the order of M_H . The uncertainty of the decay rate depends crucially on the accuracy of m_b and thus it is important to consistently propagate the uncertainties from low to high energies. In Ref. [22] one finds the following result for the bottom quark mass

$$m_b^{(5)}(10 \text{ GeV}) = \left(3610 - \frac{\alpha_s - 0.1189}{0.002} \cdot 12 \pm 11 \right) \text{ MeV}, \quad (4)$$

which translates to

$$m_b^{(5)}(M_H) = 2771 \pm 8|_{m_b} \pm 15|_{\alpha_s} \text{ MeV}. \quad (5)$$

The corresponding `Mathematica` commands to obtain the central value are given by

```
mb10 = 3.610 - 12/1000*((asMz /. NumDef) - 0.1189)/0.002;
mu10 = 10;
nloops = 5;
as10 = AlphasExact[asMz /. NumDef, Mz /. NumDef, mu10, 5, nloops];
asMh = AlphasExact[asMz /. NumDef, Mz /. NumDef, Mh /. NumDef, 5, nloops];
mMS2mMS[mb10, as10, asMh, 5, nloops]
```

The code which calculates the uncertainties and the `C++` code can be found in the supplementary files.

As a further example let us consider the bottom quark mass at the scale $\mu = M_t$ in the six-flavour theory. With the help of

```
mL2mH[mb10, as10, mu10, {{6, Mt /. NumDef, 2*Mt /. NumDef}}, Mt /. NumDef, 5]
```

one obtains $m_b^{(6)}(M_t) = 2670 \text{ MeV}$.

3.2. Quark mass relations

3.2.1. Relation between threshold and $\overline{\text{MS}}$ quark masses

In this subsection we provide `RunDec` (and `CRunDec`) commands which allow to check the numbers in Tab. 3 of Ref. [13]. To do so we use the newly implemented routines for the conversion between threshold and $\overline{\text{MS}}$ quark masses. In addition we allow for an uncertainty in the threshold mass and $\alpha_s^{(5)}(M_Z)$ and compute the resulting uncertainty of the $\overline{\text{MS}}$ quark mass in the output. The `RunDec` commands which fulfill this task are given by

```
xerr = 0.002;
mbSI = mub /. NumDef;
mtOS = Mt /. NumDef;
mcSI = muc /. NumDef;

mThr2mSI[mThr_,merr_,asmz_,aserr_,nl_,nloop_,scheme_] := Module[
  {as, asp, asm, muf, mSI, mcentral, msoserr, expuncert, muncert, asuncert},
  as[mu_] := Switch[nl,
    5, AlphasExact[asmz, Mz /. NumDef, mu, 5, 4],
    4, AlphasExact[DecAsDownSI[AlphasExact[asmz, Mz /. NumDef, 2*mbSI, 5, 4],
      mbSI, 2*mbSI, 4, 4], 2*mbSI, mu, 4, 4],
    3, AlphasExact[DecAsDownSI[AlphasExact[DecAsDownSI[AlphasExact[
      asmz, Mz /. NumDef, 2*mbSI, 5, 4], mbSI, 2*mbSI, 4, 4],
      2*mbSI, 3.0, 4, 4], mcSI, 3.0, 3, 4], 3.0, mu, 3, 4]
  ];
  asp[mu_] := Switch[nl,
    5, AlphasExact[asmz + aserr, Mz /. NumDef, mu, 5, 4],
    4, AlphasExact[DecAsDownSI[AlphasExact[asmz + aserr, Mz /. NumDef,
      2*mbSI, 5, 4], mbSI, 2*mbSI, 4, 4], 2*mbSI, mu, 4, 4],
    3, AlphasExact[DecAsDownSI[AlphasExact[DecAsDownSI[AlphasExact[
      asmz + aserr, Mz /. NumDef, 2*mbSI, 5, 4], mbSI, 2*mbSI, 4, 4],
      2*mbSI, 3.0, 4, 4], mcSI, 3.0, 3, 4], 3.0, mu, 3, 4]
  ];
  asm[mu_] := Switch[nl,
    5, AlphasExact[asmz - aserr, Mz /. NumDef, mu, 5, 4],
    4, AlphasExact[DecAsDownSI[AlphasExact[asmz - aserr, Mz /. NumDef,
      2*mbSI, 5, 4], mbSI, 2*mbSI, 4, 4], 2*mbSI, mu, 4, 4],
    3, AlphasExact[DecAsDownSI[AlphasExact[DecAsDownSI[AlphasExact[
      asmz - aserr, Mz /. NumDef, 2*mbSI, 5, 4], mbSI, 2*mbSI, 4, 4],
      2*mbSI, 3.0, 4, 4], mcSI, 3.0, 3, 4], 3.0, mu, 3, 4]
  ];
  muf := Switch[nl, 5, 80.0, 4, 2.0, 3, 2.0];

  mSI[m_, asnl_, err_] := Switch[scheme,
    "1S", m1S2mSI[m, {}], asnl, nl, nloop, 1+err],
    "PS", mPS2mSI[m, {}], asnl, muf, nl, nloop, 1+err],
    "RS", mRS2mSI[m, {}], asnl, muf, nl, nloop, 1+err],
    "RSp", mRSp2mSI[m, {}], asnl, muf, nl, nloop, 1+err];

  mcentral := mSI[mThr, as, 0];
  msoserr := Abs[mSI[mThr, as, xerr] - mcentral];
  If[merr != 0,
    muncertplus := Abs[mcentral - mSI[mThr + merr[[1]], as, 0]],
```

```

    muncertplus = 0;
];
If[Length[merr] > 1,
  muncertminus := Abs[mcentral - mSI[mThr - merr[[2]], as, 0]],
  muncertminus = muncertplus;
];
If[aserr != 0,
  asuncert := Max[ Abs[mcentral - mSI[mThr, asp, 0]],
                  Abs[mcentral - mSI[mThr, asm, 0]] ],
  asuncert = 0;
];
Return[{mcentral, muncertplus, muncertminus, asuncert, msoserr}];
];

```

Using this routine we can compute the Tab.3a of [13] with the help of

```

Do[ mfromPS = First[mThr2mSI[168.049, 0, asMz /. NumDef, 0, 5, i, "PS"]];
  mfrom1S = First[mThr2mSI[172.060, 0, asMz /. NumDef, 0, 5, i, "1S"]];
  mfromRS = First[mThr2mSI[166.290, 0, asMz /. NumDef, 0, 5, i, "RS"]];
  mfromRSp = First[mThr2mSI[171.785, 0, asMz /. NumDef, 0, 5, i, "RSp"]];
  Print[i, "      ", mfromPS, " ", mfrom1S, " ", mfromRS, " ", mfromRSp];
  ,{i,1,4} ];

```

The remaining three tables are obtained with similar code (see ancillary files).

As a further example let us assume that the top quark PS mass has been extracted from experimental data (e.g. from the threshold cross section of a future electron positron linear collider) to

$$m_t^{\text{PS}} = 168.049 \pm 0.100 \text{ GeV}. \quad (6)$$

Using $\alpha_s^{(5)}(M_Z)$ from Eq. (1) and `mThr2mSI` leads to

$$m_t^{\overline{\text{MS}}} = 163.508 \pm 0.095_{\delta m_t^{\text{PS}}} \pm 0.043_{\delta \alpha_s} \text{ GeV}. \quad (7)$$

It is interesting to note that an uncertainty of 0.0011 in $\alpha_s^{(5)}(M_Z)$ induces an uncertainty of about 40 MeV into the $\overline{\text{MS}}$ top quark mass.

In Ref. [27, 28] the PS bottom quark mass has been determined to $m_b^{\text{PS}} = 4.532_{-0.039}^{+0.013}$ GeV using QCD sum rules. With the help of

```

mbPS          = 4.532;
delmbPSplus   = 0.013;
delmbPSminus  = 0.039;
as             = asMz /. NumDef;
alsuncert     = 0.0013;
mThr2mSI[mbPS, {mbPSplus, mbPSminus}, asMz /. NumDef, alsuncert, 4, 4, "PS"];

```

we obtain $m_b(m_b) = 4.209_{-0.036}^{+0.014}$ GeV, in good agreement with the result given in [28]
 $m_b(m_b) = 4.203_{-0.034}^{+0.016}$ GeV.

3.2.2. $\overline{\text{MS}}$ -OS relations including light quark mass effects

Light quark mass effects in the $\overline{\text{MS}}$ -OS relation of the top and bottom quark masses can be computed with the help of the following commands

```
as6[mu_] := All2A1H[asMz /. NumDef, Mz /. NumDef,
                  {{6, Mt /. NumDef, 2*Mt /. NumDef}}, mu, 5];
as5[mu_] := AlphasExact[asMz /. NumDef, Mz /. NumDef, mu, 5, 5];

mut = 163.0;
Mt4 = mMS2mOS[mut, {}, as6[mut]*XXX, mut, 6, 4];
Mtb = mMS2mOS[mut, {mub /. NumDef}, as6[mut]*XXX, mut, 6, 4];
MtbC = mMS2mOS[mut, {{mub /. NumDef, mub /. NumDef}, {mc3 /. NumDef, 3.0}},
               as6[mut]*XXX, mut, 6, 4];
delb = Expand[(Mtb - Mt4)];
delc = Expand[(MtbC - Mtb)];

Mb4 = mMS2mOS[mub /. NumDef, {}, as5[mub /. NumDef]*XXX, mub /. NumDef, 5, 4];
Mbc = mMS2mOS[mub /. NumDef, {mc3 /. NumDef, 3.0},
               as5[mub /. NumDef]*XXX, mub /. NumDef, 5, 4];
delc = Expand[(Mbc - Mb4)];
```

The label XXX has been introduced to separate the different loop orders and the mass effects are contained in the quantities `delc` and `delb`. For the top and bottom quark mass we have

$$\begin{aligned}
M_t &= (163 + 7.509_{1l} + 1.603_{2l} + 0.495_{3l} + 0.195_{4l} \\
&\quad + 0.008_{2lb} + 0.002_{2lc} + 0.010_{3lb} + 0.004_{3lc}) \text{ GeV}, \\
M_b &= (4.163 + 0.398_{1l} + 0.206_{2l} + 0.160_{3l} + 0.136_{4l} + 0.007_{2lc} + 0.015_{3lc}) \text{ GeV}, \quad (8)
\end{aligned}$$

where the subscripts are self-explanatory. In all cases the mass corrections at three loops are larger than at two loops which suggests that there is no convergence in the mass correction terms. The situation is improved after decoupling the heavy quarks and using $\alpha_s^{(3)}$ as an expansion parameter (see, e.g., Ref. [29]).

4. Summary

This article describes the improvements implemented in versions 3 of the programs `RunDec` and `CRunDec`. This concerns in particular the five-loop corrections to the running of strong coupling constant and the quark masses, the four-loop correction to the decoupling of heavy quarks from the running, the four-loop corrections to the relation between the $\overline{\text{MS}}$ and on-shell heavy quark mass, and various relations between the $\overline{\text{MS}}$ and on-shell heavy quark mass to so-called threshold masses. Furthermore, light quark mass effects are implemented in the $\overline{\text{MS}}$ -on-shell relation to three loops. We provide a large number of examples which exemplify the use of both the `Mathematica` and `C++` version. It is straightforward to apply modifications and include them in other codes.

Acknowledgements

We thank Konstantin Chetyrkin for providing analytic results for the five-loop QCD beta function and quark anomalous dimension. This work is supported by the BMBF through Grant No. 05H15VKCCA.

References

- [1] K. G. Chetyrkin, J. H. Kühn and M. Steinhauser, *Comput. Phys. Commun.* **133** (2000) 43 [arXiv:hep-ph/0004189].
- [2] B. Schmidt and M. Steinhauser, *Comput. Phys. Commun.* **183** (2012) 1845 doi:10.1016/j.cpc.2012.03.023 [arXiv:1201.6149 [hep-ph]].
- [3] Y. Schroder and M. Steinhauser, *JHEP* **0601** (2006) 051 doi:10.1088/1126-6708/2006/01/051 [hep-ph/0512058].
- [4] K. G. Chetyrkin, J. H. Kühn and C. Sturm, *Nucl. Phys. B* **744** (2006) 121 doi:10.1016/j.nuclphysb.2006.03.020 [hep-ph/0512060].
- [5] T. Liu and M. Steinhauser, *Phys. Lett. B* **746** (2015) 330 doi:10.1016/j.physletb.2015.05.023 [arXiv:1502.04719 [hep-ph]].
- [6] P. A. Baikov, K. G. Chetyrkin and J. H. Kühn, *JHEP* **1410** (2014) 076 doi:10.1007/JHEP10(2014)076 [arXiv:1402.6611 [hep-ph]].
- [7] T. Luthe, A. Maier, P. Marquard and Y. Schroder, arXiv:1612.05512 [hep-ph].
- [8] P. A. Baikov, K. G. Chetyrkin and J. H. Kühn, arXiv:1702.01458 [hep-ph].
- [9] P. A. Baikov, K. G. Chetyrkin and J. H. Kühn, arXiv:1606.08659 [hep-ph].
- [10] F. Herzog, B. Ruijl, T. Ueda, J. A. M. Vermaseren and A. Vogt, arXiv:1701.01404 [hep-ph].
- [11] T. Luthe, A. Maier, P. Marquard and Y. Schroder, arXiv:1701.07068 [hep-ph].
- [12] P. Marquard, A. V. Smirnov, V. A. Smirnov and M. Steinhauser, *Phys. Rev. Lett.* **114** (2015) no.14, 142002 doi:10.1103/PhysRevLett.114.142002 [arXiv:1502.01030 [hep-ph]].
- [13] P. Marquard, A. V. Smirnov, V. A. Smirnov, M. Steinhauser and D. Wellmann, *Phys. Rev. D* **94** (2016) no.7, 074025 doi:10.1103/PhysRevD.94.074025 [arXiv:1606.06754 [hep-ph]].
- [14] M. Beneke, *Phys. Lett. B* **434** (1998) 115 doi:10.1016/S0370-2693(98)00741-2 [hep-ph/9804241].
- [15] A. H. Hoang, Z. Ligeti and A. V. Manohar, *Phys. Rev. D* **59** (1999) 074017 doi:10.1103/PhysRevD.59.074017 [hep-ph/9811239].
- [16] A. H. Hoang, Z. Ligeti and A. V. Manohar, *Phys. Rev. Lett.* **82** (1999) 277 doi:10.1103/PhysRevLett.82.277 [hep-ph/9809423].
- [17] A. H. Hoang and T. Teubner, *Phys. Rev. D* **60** (1999) 114027 doi:10.1103/PhysRevD.60.114027 [hep-ph/9904468].
- [18] A. Pineda, *JHEP* **0106** (2001) 022 doi:10.1088/1126-6708/2001/06/022 [hep-ph/0105008].
- [19] N. Gray, D. J. Broadhurst, W. Grafe and K. Schilcher, *Z. Phys. C* **48** (1990) 673. doi:10.1007/BF01614703
- [20] S. Bekavac, A. Grozin, D. Seidel and M. Steinhauser, *JHEP* **0710** (2007) 006 doi:10.1088/1126-6708/2007/10/006 [arXiv:0708.1729 [hep-ph]].
- [21] C. Patrignani *et al.* [Particle Data Group], *Chin. Phys. C* **40** (2016) no.10, 100001. doi:10.1088/1674-1137/40/10/100001
- [22] K. G. Chetyrkin, J. H. Kühn, A. Maier, P. Maierhofer, P. Marquard, M. Steinhauser and C. Sturm, *Phys. Rev. D* **80** (2009) 074010 doi:10.1103/PhysRevD.80.074010 [arXiv:0907.2110 [hep-ph]].
- [23] P. A. Baikov, K. G. Chetyrkin and J. H. Kühn, *Phys. Rev. Lett.* **101** (2008) 012002 doi:10.1103/PhysRevLett.101.012002 [arXiv:0801.1821 [hep-ph]].
- [24] S. Chatrchyan *et al.* [CMS Collaboration], *Eur. Phys. J. C* **73** (2013) no.10, 2604 (2013) doi:10.1140/epjc/s10052-013-2604-6 [arXiv:1304.7498 [hep-ex]].
- [25] V. Khachatryan *et al.* [CMS Collaboration], [arXiv:1609.05331 [hep-ex]].
- [26] M. Bruno *et al.*, *PoS LATTICE 2016*, 197 (2016) [arXiv:1701.03075 [hep-lat]].

- [27] M. Beneke, A. Maier, J. Piclum and T. Rauh, Nucl. Phys. B **891** (2015) 42 doi:10.1016/j.nuclphysb.2014.12.001 [arXiv:1411.3132 [hep-ph]].
- [28] M. Beneke, A. Maier, J. Piclum and T. Rauh, arXiv:1601.02949 [hep-ph].
- [29] C. Ayala, G. Cvetič and A. Pineda, JHEP **1409** (2014) 045 doi:10.1007/JHEP09(2014)045 [arXiv:1407.2128 [hep-ph]].