Automated One-loop Computation in Quarkonium Process within NRQCD Framework

Feng Feng
Center for High Energy Physics, Peking University, Beijing 100871, China
E-mail: F.Feng@outlook.com

Abstract. In last decades, it has been realized that the next-to-leading order corrections may become very important, and sometimes requisite, for some processes involving quarkonium production or decay, e.g., $e^+e^- \rightarrow J/\psi + \eta_c$ and $J/\psi \rightarrow 3\gamma$. In this article, we review some basic steps to perform automated one-loop computations in quarkonium process within the Non-relativistic Quantum Chromodynamics (NRQCD) factorization framework\(^1\), and we give an introduction to some related public tools or packages and their usages in each step. We start from generating Feynman diagrams and amplitudes with FEYNArts for the quarkonium process, performing Dirac- and Color-algebras simplifications using FEYNCalc and FEYNCalcFormLink, and then to doing partial fractions on the linear-dependent propagators by APart, and finally to reducing the Tensor Integrals (TI) into Scalar Integrals (SI) or Master Integrals (MI) using Integration-By-Parts (IBP) method with the help of FIRE. We will use a simple concrete example to demonstrate the basic usages of the corresponding packages or tools in each step.

1. Generating Feynman Diagram and Amplitude
Before starting to generate the corresponding Feynman diagram, we need to replace the incoming or outgoing hadronic states with corresponding partonic states, e.g., we need to replace $J/\psi$ with a quark and anti-quark pair $c(p_1)\bar{c}(p_2)$ for color-singlet case, and the pair $c\bar{c}$ with an extra gluon $c(p_1)\bar{c}(p_2)g(k)$ for the color-octet case. We take the process $(e^+e^- \rightarrow J/\psi + \eta_c)$ as an example, for the color-singlet model, the Feynman diagrams we actually want to generate are such process:

$$e^+ + e^- \rightarrow \gamma^* \rightarrow c\left(\frac{p_3}{2} + q_3\right)c\left(\frac{p_4}{2} - q_4\right) + c\left(\frac{p_4}{2} + q_4\right)c\left(\frac{p_4}{2} - q_4\right).$$

Currently, there are two tools or packages to automatically generate the Feynman diagrams:

- **FEYNArts**\(^2\) is a Mathematica package for the generation and visualization of Feynman diagrams and amplitudes, it can be downloaded from [http://www.feynarts.de/](http://www.feynarts.de/).
- **QGraf**\(^3\) is a computer program with the programming language: FORTRAN 77, that was written to assist in large perturbative calculations, in the context of Quantum Field Theory. It can generate Feynman diagrams and represent them by symbolic expressions, it can be downloaded from [http://cfif.ist.utl.pt/~paulo/qgraf.html](http://cfif.ist.utl.pt/~paulo/qgraf.html).

\(^{1}\) The general description of the applications to NRQCD to the quarkonium processes was presented in the Plenary talk of this Workshop [1].
Since we use Mathematica as our computation environment, we give a demonstration of the usage of FeynArts to generate the Feynman diagrams and amplitudes for the process: \((e^+e^- \rightarrow \gamma^+ \rightarrow J/\psi + \eta_c)\). The basic steps are as follows:

- The underlying partonic process is:
  \[
  \gamma^* \rightarrow c\left(\frac{p_3}{2} + q_3\right)\bar{c}\left(\frac{p_3}{2} - q_3\right) + c\left(\frac{p_4}{2} + q_4\right)\bar{c}\left(\frac{p_4}{2} - q_4\right)
  \]

- Using CreateTopologies to generate the topologies for the case 1 \(\rightarrow\) 4:
  
  ```mathematica
  top=CreateTopologies[1,1->4,ExcludeTogologies->{WFCorrections, Tadpoles, V4onExt}];
  ```

  where we also exclude some topologies which don’t contribute in our case.

- Using InsertFields to insert the fields in the corresponding model to the topologies we have just generated in the last step:
  
  ```mathematica
  tmp=InsertFields[top, {V[1]}->{F[3],-F[3],F[3],-F[3]}, Model->"SMQCD",
  ExcludeParticles->{V[1|2|3|4],S[_],F[4]}, InsertionLevel->{Clases}];
  ```

- Selecting or removing other unwanted diagrams using DiagramSelect or DiagramDelete, e.g.,
  
  ```mathematica
  all=DiagramDelete[tmp, 3...4, 13, 14, 25...25, 33...34, 42...43];
  ```

- Using CreateFeynAmp to generate Feynman amplitudes for each diagram:
  
  ```mathematica
  amp=CreateFeynAmp[all, Prefactor->1]/.{FourMomentum[Incoming,1]->p3+p4, 
  FourMomentum[Outgoing,1]->p3/2+q3, FourMomentum[Outgoing,2]->p3/2-q3, 
  FourMomentum[Outgoing,3]->p4/2+q4, FourMomentum[Outgoing,4]->p4/2-q4, 
  MQU[___]->mu, MQD[___]->md, EL->e, GS->GStrong};
  ```

- Exporting the amplitudes to output file for later processing in FeynCalc[4].

2. Simplifying Dirac- and Color-Algebra

To perform the Dirac- and Color-algebra simplifications, we adopt the covariant spin projectors techniques[5, 6, 7] for the \(q\bar{q}\) production:

\[
\begin{align*}
  v(p)\bar{u}(p) & \rightarrow \frac{1}{4\sqrt{2}E(E + m_c)}(\not{p} - m_c)[\gamma_5, \not{\epsilon}^*](\not{P} + 2E)(\not{p} + m_c) \\
\end{align*}
\]

where \(\gamma_5\) and \(\not{\epsilon}\) correspond to spin-singlet and spin-triplet respectively, and for \(q\bar{q}\) decay, the projector reads:

\[
\begin{align*}
  u(p)\bar{v}(p) & \rightarrow \frac{1}{4\sqrt{2}E(E + m_c)}(\not{p} + m_c)(\not{P} + 2E)[\gamma_5, \not{\epsilon}](\not{\bar{p}} - m_c) \\
\end{align*}
\]

where \(E\) is defined by:

\[
E = \sqrt{m_c^2 + \left(\frac{p - \bar{p}}{2}\right)^2}
\]

Several packages or tools can be used to simplify the Dirac- and Color-algebra:

- FORM[8, 9, 10, 11, 12, 13] is a Symbolic Manipulation System written in C language, it can be downloaded from http://www.nikhef.nl/~form/.
- FeynCalc[4] is a Mathematica package for algebraic calculations in elementary particle physics, it can be downloaded from http://www.feyncalc.org/.
- FeynCalc/FormLink[14] is developed to combine high-performance of FORM and user-friendliness of FeynCalc, it can be downloaded from http://www.feyncalc.org/formlink/.
Note that there is also another package FormCalc[15] which uses Form from Mathematica. The difference between FeynCalc/FormLink and FormCalc is the way in which Mathematica and Form communicate with each other. FormCalc basically uses the method of input and output files, while FeynCalc/FormLink uses the piping method. The basic idea of FormLink is:

- **FormLink** creates two unnamed pipes: r# and w#.
- **FormLink** starts Form process with the command line: `form -pipe r#, w# init`, where `init` is just a initial Form file with extension .frm.
- **Form** sends its Process ID(PID) to **FormLink** in w#, and when **FormLink** receives the PID, it will responses two comma-separated PID to **Form**, the first one is the same as **Form** PID, and the last one corresponds to the PID of **FormLink**.
- **Form** start running the init.frm file with the following codes:

```plaintext
Off Statistics;
#ifndef 'PIPES_'
#message "No pipes found";
.end;
#endif
#if ('PIPES_' <= 0)
#message "No pipes found";
.end;
#else
#procedure put(fmt, mexp)
#toexternal 'fmt', 'mexp'
#toexternal "THE-END-MARK#"
@endprocedure
#setexternal 'PIPE1_';
#toexternal "OK"
#fromexternal .end
```

The key statement is `#fromexternal`, when **Form** runs into this instruction, it will be blocked until the **Form** code has been sent from **Mathematica** through **FormLink**, and then **Form** will continue to execute the code which has been just sent.

Let us demonstrate the basic usage of FeynCalc/FormLink with a simple example, i.e., the trace of six Dirac gamma matrix, first we calculate the trace with FeynCalc:

```plaintext
<<HighEnergyPhysics’fc’
Tr[GS[p1, p2, p3, p4, p5, p6]]
```

It is also quite simple to perform the trace with FeynCalcFormLink, first we prepare the expression in FeynCalc syntax, i.e.,

```plaintext
exp = DiracTrace[GS[p1, p2, p3, p4, p5, p6]]; 
```

note that we use DiracTrace instead of Tr to prevent the evaluation of the trace, then we just use FeynCalcFormLink to calculate the expression exp:

```plaintext
FeynCalcFormLink[exp]
```

FeynCalcFormLink first translate the exp in FeynCalc syntax to Form code, for this simple case, the translated Form code is as follows:

```plaintext
Vectors p1,p2,p3,p4,p5,p6;
Format Mathematica;
L resFL = (g_(1,p1)*g_(1,p2)*g_(1,p3)*g_(1,p4)*g_(1,p5)*g_(1,p6));
trace4,1;
.contract 0;
.sort;
#call put("%E", resFL)
#fromexternal
```
Scalar integrals: Generally, those n-point tensor integrals can be reduced to much simpler loop integrals, n-point one-loop integrals can be characterized by the so-called n-point tensor integrals, e.g., result to \texttt{FeynCalc} starts sending the result back to \texttt{Mathematica}, and \texttt{FeynCalcFormLink} will translate the result to \texttt{FeynCalc} syntax.

\texttt{SUNSimplify} in \texttt{FeynCalc} can be used to perform the simplification on the color-algebra, e.g., \texttt{SUNsimplify[SUNT[a,b,a,b]]} to get the result

\[ T^a T^b T^a T^b = -\frac{1}{2} C_F (C_A - 2 C_F). \] (5)

Before we are going to do the loop momentum integrals, we can use another technique, the method of region expansion[16], to greatly simplify our calculations. Usually, we expand the relative momentum \( q \) between quark and anti-quark in quarkonium state after performing the loop integration, and then project the \( S_-, P_- \) or \( D \)-waves. We can also expand the \( q \) before the loop integration, as long as only the hard region is concerned according to the method of region expansion[16]. So if the NRQCD factorization is valid, it will be safe to use the method of region expansion to compute the short-distance coefficients, which correspond to the hard region.

3. Passarino-Veltman Reduction for the Tensor Integrals

The generic one-loop integral looks like:

\[ T^{\mu_1 \cdots \mu_p} \equiv \frac{(2 \pi \mu)^{4-d}}{i \pi^2} \int d^d k \, \frac{k^{\mu_1} \cdots k^{\mu_p}}{D_0 D_1 D_2 \cdots D_{n-1}} \] (6)

where \( D_i = (k + r_i)^2 - m_i^2 + i \varepsilon, \ r_i = \sum_{k=1}^n \ p_k \ (i = 1, \cdots, n-1), \ r_0 = 0 \) and \( r_{ij} = r_i - r_j \). These one-loop integrals can be characterized by the so-called n-point tensor integrals, e.g.,

\[ B^{\mu}(r_{10}^2, m_0^2, m_1^2) = \frac{(2 \pi \mu)^{4-d}}{i \pi^2} \int d^d k \, k^{\mu} \prod_{i=0}^1 \frac{1}{(k + r_i)^2 - m_i^2} \] (7)

\[ C^{\mu\nu}(r_{10}^2, r_{12}^2, r_{20}^2, m_0^2, m_1^2, m_2^2) = \frac{(2 \pi \mu)^{4-d}}{i \pi^2} \int d^d k \, k^{\mu} k^{\nu} \prod_{i=0}^2 \frac{1}{(k + r_i)^2 - m_i^2} \]

\[ D^{\mu\nu\rho}(r_{10}^2, r_{12}^2, r_{23}^2, r_{30}^2, r_{13}^2, m_0^2, m_1^2, m_2^2, m_3^2) = \frac{(2 \pi \mu)^{4-d}}{i \pi^2} \int d^d k \, k^{\mu} k^{\nu} k^{\rho} k^{\sigma} \prod_{i=0}^3 \frac{1}{(k + r_i)^2 - m_i^2} \]

Generally, those n-point tensor integrals can be reduced to much simpler loop integrals, n-point scalar integrals:

\[ A_0(m_0^2) = \frac{(2 \pi \mu)^{4-d}}{i \pi^2} \int d^d k \, \frac{1}{k^2 - m_0^2} \] (8)

\[ B_0(r_{10}^2, m_0^2, m_1^2) = \frac{(2 \pi \mu)^{4-d}}{i \pi^2} \int d^d k \, \prod_{i=0}^1 \frac{1}{(k + r_i)^2 - m_i^2} \]

\[ C_0(r_{10}^2, r_{12}^2, r_{20}^2, m_0^2, m_1^2, m_2^2) = \frac{(2 \pi \mu)^{4-d}}{i \pi^2} \int d^d k \, \prod_{i=0}^2 \frac{1}{(k + r_i)^2 - m_i^2} \]

\[ D_0(r_{10}^2, r_{12}^2, r_{23}^2, r_{30}^2, r_{13}^2, m_0^2, m_1^2, m_2^2, m_3^2) = \frac{(2 \pi \mu)^{4-d}}{i \pi^2} \int d^d k \, \prod_{i=0}^3 \frac{1}{(k + r_i)^2 - m_i^2} \]
We take a rank 4 tensor integral $D^{\mu\nu\rho\sigma}$ as an example, the tensor integral $D^{\mu\nu\rho\sigma}$ can be expressed as follows according to the Lorentz invariance:

\[
D^{\mu\nu\rho\sigma} = (g^{\mu\nu}g^{\rho\sigma} + g^{\mu\rho}g^{\nu\sigma} + g^{\mu\sigma}g^{\nu\rho})D_{0000} + \sum_{i,j,k,l=1}^{3} r_i^\mu r_j^\nu r_k^\rho r_l^\sigma D_{ijkl}\]

(9)

where the $D_{0000}$, $D_{00ij}$ and $D_{ijkl}$ are some Lorentz scalar coefficients which can be expressed in terms of the n-point scalar integrals: $A_0$, $B_0$, $C_0$ and $D_0$. Such procedure is called Passarino-Veltman Reduction(PaVe-Reduction).

These coefficients can be achieved with the function PaVe in FeynCalc, e.g., the

\[
D_{0000}(r_{10}^2, r_{12}^2, r_{23}^2, r_{30}^2, r_{20}^2, r_{13}^2, m_0^2, m_1^2, m_2^2, m_3^2)
\]

in FeynCalc is expressed as:

\[
\text{PaVe}[0,0,0,0,\{r_{10}^2,r_{12}^2,r_{23}^2,r_{30}^2,r_{20}^2,r_{13}^2\},\{m_0^2,m_1^2,m_2^2,m_3^2\}]
\]

where the first part in the argument of PaVe function $\{0,0,0,0\}$ is the subscript of the corresponding $D_{0000}$ coefficient, and the remaining are the same as those in the argument of $D_{0000}$ coefficient. It should be noted that the factor involving renormalization scale $\mu$, i.e., $(2\pi\mu)^{4-d}$ has been dropped out in FeynCalc.

Now to perform the PaVe-Reduction, we just use PaVeReduce,

\[
\text{PaVeReduce}[\text{PaVe}[0,0,0,0,\{1,2,3,4,5,6\},\{1,1,1,1\}]]
\]

the output looks like

\[
\begin{align*}
-135C_0(1, 2, 5, 1, 1, 1) & - 5751C_0(1, 4, 6, 1, 1, 1) & - 88691C_0(2, 3, 6, 1, 1, 1) \\
2401 & + 192080 & + 2650704 \\
-5755C_0(3, 4, 5, 1, 1, 1) & + 1587D_0(1, 2, 3, 4, 5, 1, 1, 1) & + 51B_0(1, 1, 1) \\
633864 & - 38416 & + 1960 \\
+ 907B_0(2, 1, 1) & + 1052B_0(3, 1, 1) & - 32340 \\
27048 & + 99176 & - 1617 \\
& 347B_0(4, 1, 1) & - 5B_0(5, 1, 1) \\
& 50B_0(5, 1, 1) & - 181B_0(6, 1, 1) \\
& 22540 & + 72
\end{align*}
\]

(10)

where we take some special numerical values for the argument of $D_{0000}$. We can see that the coefficient $D_{0000}$ is now expressed in terms of n-point scalar integrals: $B_0$, $C_0$ and $D_0$.

Since the PaVe-Reduction is based on solving linear equations, generally it will encounter some problems when the Gram determinant equals 0, which happens if we expand the relative momentum $q$ before loop integration, due to taking the derivative over $q$. So we need some more general method called Integration-By-Parts (IBP) reduction to perform the reduction of tensor integrals.

4. Partial Fraction and IBP Reduction

Let us consider a general Feynman integral, here we adopt the notation as in [17],

\[
F(a_1, \cdots, a_n) = \int \cdots \int \frac{d^dk_1 \cdots d^dk_h}{E_{a_1}^{k_1} \cdots E_{a_n}^{k_h}}
\]

(11)

where $k_i$, $i = 1, \cdots, h$, are loop momenta and the denominators $E_r$, $r = 1, \cdots, n$, are either quadratic or linear with respect to the loop momenta $k_i$ of the graph. Irreducible polynomials in the numerator can be represented as denominators raised to negative powers.
The basic idea of IBP reduction [18] is that, we know the integration of such derivative is 0, i.e.,
\[ \int \cdots \int d^d k_1 d^d k_2 \cdots \frac{\partial}{\partial k_i} \left[ \frac{p_j}{E_1^{n_1} \cdots E_m^{n_m}} \right] = 0 \] (12)
where \( k_i \) are the loop momenta, and \( p_j \) are the momenta which can internal or external, so with different \( k_i \) and \( p_j \), we can get a list of equations which can be expressed as follows:
\[ \sum \alpha_i F(a_1 + b_{i,1}, \cdots, a_n + b_{i,n}) = 0 \] (13)
By solving these equations, we can express the complicated loop integrals in terms of much simpler ones, which we call Master Integral (MI).

There are many packages or tools in the market which can be used to perform the IBP reduction, e.g.,
- AIR[19] is MAPLE package, which can be downloaded from [http://www.phys.ethz.ch/~pheno/air/](http://www.phys.ethz.ch/~pheno/air/).
- FIRE[17] is MATHEMATICA package, which can be downloaded from [http://science.sander.su/FIRE.htm](http://science.sander.su/FIRE.htm).
- REDUCE[20] is written in C, which can be downloaded from [http://reduze.hepforge.org/](http://reduze.hepforge.org/).
- LiteRed[21] is another MATHEMATICA package, which can be downloaded from [http://www.inp.nsk.su/~lee/programs/LiteRed/](http://www.inp.nsk.su/~lee/programs/LiteRed/).
- Many other private codes.

There is a precondition to perform the IBP reduction, i.e., the propagators should be linear independent, so we need another MATHEMATICA package APART[22] to perform partial fraction on the propagators.

Let us take a simple physical loop integral to demonstrate the usage of APART and FIRE,
\[ \exp = \frac{(k \cdot p_1) (k \cdot p_2)}{k^2 [(k + p_1)^2 - m^2][(k + p_2)^2 - m^2]} \] (14)
the linear independent variables involving loop momentum are \( k^2, k \cdot p_1, k \cdot p_2 \), which can be expressed in FEYNCALC:
\[ \text{xs = FCI/\@\{ SP[k], SP[k, p1], SP[k, p2] \} } \]
then to perform partial fraction on the loop integral is ready with APART:
\$\text{APart[exp, xs]}$
the result looks like:
we can see that there are at most three propagators in each term, and these propagators in each term are linear independent now.

Finally we can use FIRE to perform IBP reduction, we take the tensor integral framed with red box in Eq. (15) as an example, such integral can be expressed as $F[\{-1, 1, 2\}]$ with $F$ defined by:

$$F[\{l, m, n\}] = \int \frac{d^4 k}{(2\pi)^4} \frac{(k\cdot p_2)^{-l}}{(m^2 - k^2 - 2k\cdot p_1 - p_1^2)^m(-m^2 + k^2 + 2k\cdot p_2 + p_2^2)^n} \quad (16)$$

The basic usage of FIRE is like this:

Replacement = \{p1^2 \rightarrow m^2, p2^2 \rightarrow m^2, p_1 p_2 \rightarrow \text{SP}[p_1, p_2]\};
Internal = \{k\};
External = \{p_1, p_2\};
Propagators = \{k\cdot p_2, -2k\cdot p_1 - k^2 + m^2 - p_1^2, 2k\cdot p_2 + k^2 - m^2 + p_2^2\};
PrepareIBP[];

startinglist = \{IBP[k, k], IBP[k, p_1], IBP[k, p_2]\}/.Replacement;

Prepare[];
Burn[];

first we input the internal and external momenta, and provide the independent propagators in Propagators, then prepare the IBP equations with startinglist, finally Burn in FIRE, and now it is ready to get the result for $F[\{-1, 1, 2\}]$, just use the $F$ function:

$$F[\{-1, 1, 2\}] = \frac{(d - 2)G(\{0, 0, 1\})}{8(m^2 - p_1\cdot p_2)} + \frac{(d - 2)G(\{0, 1, 0\})}{8(m^2 - p_1\cdot p_2)} + \frac{1}{4}(4 - d)G(\{0, 1, 1\}) \quad (17)$$

We can apply such procedure to each loop integral in Eq. (15) to get the finally IBP reduced result:

$$\Rightarrow \left| \begin{array}{c} (k\cdot p_1)(k\cdot p_2) \\ k^2[(k + p_1)^2 - m^2][(k + p_2) - m^2]^2] \\ \frac{(D - 2)}{16(m^2 - p_1\cdot p_2)} \end{array} \right| + \left| \begin{array}{c} (D - 2) \\ \frac{1}{16(m^2 - p_1\cdot p_2)} \\ + \frac{1}{8}(4 - D) \end{array} \right| \left| \begin{array}{c} \frac{1}{(-k^2 - 2k\cdot p_1)(k^2 + 2k\cdot p_2)} \\ \frac{1}{k^2(k^2 + 2k\cdot p_2)} \end{array} \right| \quad (18)$$

where we set $p_1^2 = p_2^2 = m^2$ to simplify the result, and $\|\cdot\|$ is defined by

$$\|\exp\| = \int \frac{d^D k}{(2\pi)^D} \exp$$

It can be seen that the original tensor integral has been reduced to much simpler scalar integrals or master integrals, and we can apply such procedure to each tensor integral in each Feynman diagram, and get the final expression expressed in terms of scalar integrals, which can be calculated, analytically or numerically, by any other means.

**Acknowledgments**

The author would like to thank Rolf Mertig for the collaboration in the FORMLINK and FORMLINKFeynCalc project, and to Prof. Yu Jia, Postdoc Wen-Long Sang and Hai-Rong Dong for many useful discussions in the related works.

Finally, Feng Feng would like to commemorate his beloved mother and father.
References

[1] Braguta V V, Likhoded A K, Luchinsky A V and Poslavsky S V 2013 Analytical calculation of heavy quarkonia production processes in computer (Preprint arXiv:1309.3093 [hep-ph]).

[2] Hahn T 2001 Generating Feynman diagrams and amplitudes with FeynArts 3 Comput. Phys. Commun. 140 418 (Preprint hep-ph/0012260).

[3] Nogueira P 1993 Automatic Feynman graph generation J. Comput. Phys. 105 279.

[4] Mertig R, Bohm M and Denner A 1991 FEYN CALC: Computer algebraic calculation of Feynman amplitudes Comput. Phys. Commun. 64 345.

[5] Bodwin G T and Petrelli A 2002 Order $\alpha_s^4$ corrections to S wave quarkonium decay Phys. Rev. D 66 094011 (Preprint hep-ph/0211085).

[6] Bodwin G T, Garcia T i Tormo and Lee J 2010 Factorization in exclusive quarkonium production Phys. Rev. D 81 114014 (Preprint arXiv:1003.0061 [hep-ph]).

[7] Kuipers J, Ueda T, Vermaseren J A M and Vollinga J 2012 FORM version 4.0 (Preprint arXiv:1203.6543 [cs.SC]).

[8] Vermaseren J A M 2000 New features of FORM (Preprint math-ph/0010025).

[9] Vermaseren J A M 2010 FORM development PoS CPP 2010 012 (Preprint arXiv:1101.0511 [hep-ph]).

[10] Benefice M and Smirnov V A 1998 Asymptotic expansion of Feynman integrals near threshold Nucl. Phys. B 522 321 (Preprint hep-ph/9711391).

[11] Smirnov A V 2008 Algorithm FIRE – Feynman Integral REDuction, JHEP 0810 107 (Preprint arXiv:0807.3243 [hep-ph]).

[12] Chetyrkin K G and Tkachov F V 1981 Integration by Parts: The Algorithm to Calculate beta Functions in 4 Loops Nucl. Phys. B 192 159.

[13] Anastasiou C and Lazopoulos A 2004 Automatic integral reduction for higher order perturbative calculations JHEP 0407 046 (Preprint hep-ph/0404258).

[14] Studerus C 2010 Reduce-Feynman Integral Reduction in C++ Comput. Phys. Commun. 181 1293 (Preprint arXiv:0912.2546 [physics.comp-ph]).

[15] Lee R N 2012 Presenting LiteRed: a tool for the Loop InTEgrals REDuction (Preprint arXiv:1212.2685 [hep-ph]).

[16] Feng F 2012 $Apart: A Generalized Mathematica Apart Function Comput. Phys. Commun. 183, 2158 (2012) (Preprint arXiv:1204.2314 [hep-ph]).