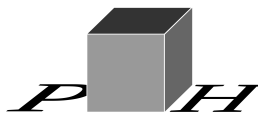
Computer algebra tools for quarkonium NLO computations

Vladyslav Shtabovenko
University of Siegen
Center for Particle Physics Siegen (CPPS)

Quarkonia as Tools 2023
in
Centre Paul Langevin, Aussois, France
9th of January 2023



Outline

Motivation

- Perturbative quantum field theory (QFT) calculations often mean evaluation of Feynman diagrams
- Higher order perturbative calculations are challenging but also necessary
- Increasing experimental precision (high luminosity LHC, . . .) calls for better theory input
- The challenges are often of technical rather than conceptual nature
- New advances require better algorithms and a lot of computer power
- Software tools are indispensable

A typical Feynman diagram calculation may involve many different building blocks ...

Diagrams
- Feynman rules
- Diagram generation
- Filtering
- Topologies

Amplitudes
- Expansions
- Dirac algebra
- Color algebra
- Tensor reduction
- Projectors

Loop integrals
- Partial fractioning
- IBP reduction
- Mappings between integrals
- Master integral evaluation

Miscellaneous
- Renormalization
- Phase space
- Matching
- Phenomenology
- ...

... many of which can be automatized completely or to some extent!

- Symbolic manipulations are usually automatized using **FORM** [Vermaseren, 2000; Kuipers et al., 2013] and **Mathematica**
- Other frameworks (e. g. **Maple**, **GiNaC** [Bauer et al., 2002], **Reduce** etc.) can be useful for different tasks
- Writing big codes from scratch can be time-consuming and error-prone
- More pragmatic approach: employ tools available on the market, combine them in a smart way
- Many calculations up to 1-loop are automatizable either fully or to some extent
- In the field of heavy quarkonia we have **HELAC-ONIA** [Shao, 2016], **MadGraph** /**MadOnia** [Alwall et al., 2014; Artoisenet et al., 2008], **FDC** [Wang, 2004], **HepLib** [Feng et al., 2023], **FeynCalc** [VS et al., 2020] + **FeynOnium** [Brambilla, Chung, VS, Vairo, 2020]
- However, often quarkonium calculations (especially beyond tree-level) are done by chaining different tools together using private codes.

Diagram generation

- Two main tools on the market: **FeynArts** [Hahn, 2001] and **QGRAF** [Nogueira, 1993]
- **FeynArts** is an all-in-one solution: diagram generation, visualization, Feynman rules insertion
  - Easy to use, works with **Mathematica**, new models can be added via **FeynRules** [Christensen & Duhr, 2009; Alloul et al., 2014]
  - Can become slow when generating and filtering large (> 100) numbers of diagrams
  - Often preferred in tree-level and 1-loop calculations
- **QGRAF** only generates the amplitudes but does not visualize them or insert Feynman rules
  - New models are easy to write, only need to specify vertices and types of fields (bosonic/fermionic)
  - Output can be formatted to be readable by any CAS
  - Visualizing diagrams requires additional tools (e. g. **GraphViz**)
  - Need additional codes for inserting Feynman rules
  - Extremely fast, even when producing millions of diagrams, preferred tool in the multi-loop community

Dirac and color algebra

- Theories with fermions ⇒ Dirac matrices in the amplitudes
- Dimensional regularization (loop integrals) implies that the algebra must be done in D dimensions
- Need to simplify chains with common indices/momenta and calculate traces in D dimensions

$$\gamma^\mu\gamma^{\nu_1}\dots\gamma^{\nu_n}\gamma_\mu = (d-4)(-1)^n\gamma^{\nu_1}\dots\gamma^{\nu_n} + 2(-1)^n\gamma^{\nu_3}\gamma^{\nu_2}\gamma^{\nu_1}\gamma^{\nu_4}\dots\gamma^{\nu_n} + 2\sum_{j=4}^{m}(-1)^{n-j}\gamma^{\nu_j}\gamma^{\nu_1}\dots\gamma^{\nu_{j-1}}\gamma^{\nu_{j+1}}\dots\gamma^{\nu_n},$$

$$\not{p}\gamma^{\nu_1}\dots\gamma^{\nu_n}\not{p} = (-1)^n p^2\gamma^{\nu_1}\dots\gamma^{\nu_n} + 2\sum_{i=1}^{n}(-1)^{i+1}\gamma^{\nu_1}\dots\gamma^{\nu_{i-1}}\gamma^{\nu_{i+1}}\dots\gamma^{\nu_n}\not{p}p^{\nu_i}$$

- Issues with γ^5: cannot simultaneously preserve $\{\gamma^\mu,\gamma^5\}=0$ and the cyclicity of the trace in D dimensions
- Different schemes for treating γ^5: NDR [Chanowitz et al., 1979], BMHV [Breitenlohner & Maison, 1977; 't Hooft & Veltman, 1972], Larin [Larin, 1993], ...
- See [Gnendiger et al., 2017] for a good overview
- Ready-to-use tools: **Tracer** [Jamin & Lautenbacher, 1993], **FormTracer** [Cyrol et al., 2017], **FeynCalc**, **Package-X** [Patel, 2015, 2017], **HepMath** [Wiebusch, 2015]

- QCD is a nonabelian SU(3) gauge theory ⇒ Feynman rules involve color algebra
- Calculations are usually done for a generic N, helps to understand the structure of the results
- Standard rules to eliminate common indices and do the traces

$$
\begin{aligned}
(AT^aBT^aC) &= \frac{1}{2}(AC)\,\mathrm{Tr}(B) - \frac{1}{2N}ABC \\
(AT^aB)\,\mathrm{Tr}(CT^aD) &= \frac{1}{2}(ADCB) - \frac{1}{2N}(AB)\,\mathrm{Tr}(CD) \\
\mathrm{Tr}(AT^aB)\,\mathrm{Tr}(CT^aD) &= \frac{1}{2}\,\mathrm{Tr}(ADCB) - \frac{1}{2N}\,\mathrm{Tr}(AB)\,\mathrm{Tr}(CD) \\
T^aT^b &= \frac{1}{2N}\delta^{ab} + \frac{1}{2}d^{abc}T^c + \frac{1}{2}if^{abc}T^c \\
if^{abc} &= 2\,\mathrm{Tr}(T^a[T^b,T^c]) \\
d^{abc} &= 2\,\mathrm{Tr}(\{T^a,T^b\}T^c)
\end{aligned}
$$

- Automatizable using e. g. **ColorMath** [Sjödahl, 2013], **FormTracer**, **FeynCalc**, **color.h** [van Ritbergen et al., 1999]

Loop integrals

- Upon simplifying Dirac and color algebra we get

$$i\mathcal{M} \to \sum_i c_i^{\mu_{j_1}\cdots\mu_{j_n}} I_{\mu_{j_1}\cdots\mu_{j_n}}$$

  - $c_i^{\mu_{j_1}}$ mostly made of Dirac matrices, polarization vectors and Levi-Civita tensors
  - $I_{\mu_{j_1}\cdots\mu_{j_n}}$ are tensor loop integrals
  - Naive (text book) approach: calculate each integral separately
  - Better: try to reduce them to a smaller set of *master* integrals
- But first we need to get rid of tensor indices!
- Two popular methods: projectors or tensor reduction
  - Projectors: work out projection operators P_i in Dirac and color space for all possible contributions

$$\mathcal{M} = \sum_k a_k I_k \Rightarrow I_k = \mathrm{Tr}(P_i \mathcal{M})$$

  - Tensor reduction: for each integral write down the ansatz with all possible tensor structures, then contract and solve a system of linear equations

$$I_{\mu_{j_1}\cdots\mu_{j_n}} = \sum_j r_j^{\mu_{j_1}\cdots\mu_{j_n}} I_j$$

- I_j are scalar integrals with scalar products in numerators and propagators raised to integer powers
- These scalar integrals are still not all independent!

- Common technique at 1-loop: Passarino-Veltman [Passarino & Veltman, 1979]
  - PaVe-workflow: tensor reduction $\rightarrow$ PaVe coefficients functions B_{11}, C_{001} etc.
    $\overset{\text{PaVe reduction}}{\rightarrow}$ PaVe scalar functions $A_0, B_0, C_0, D_0, ...$
  - Bottom line: Every 1-loop I_j (quadratic propagators!) mapped to a linear combination of $A_0, B_0, C_0, D_0, ...$
  - PaVe functions are well understood, can be evaluated analytically (**Package-X**) or numerically (**LoopTools** [Hahn & Perez-Victoria, 1999], ...)
- Multi-loop: no x-loop PaVe functions, proceed on a case-by-case basis
  - Main idea: reduce each x-loop scalar integral I_j to a smaller set of *master integrals* (MI)
  - Main technique: reduction via integration-by-parts identities (IBPs) [Chetyrkin & Tkachov, 1981; Tkachov, 1981]
  - IBP-reduction done via publicly available tools **FIRE**, [Smirnov & Chuharev, 2020], **Kira** [Klappert et al., 2021], **LiteRed** [Lee, 2014]
  - Profit: Calculate only the MI, instead of every I_j

- To set up the IBP-reduction we need to map each I_j to an integral family (also called topology)
- A topology is a set of propagators that form a basis (complete and linearly independent set)
- Example: 1-loop Box
  - The topology is made of 4 propagators that form a complete basis

$$\{q^2 - m_0^2, \quad (q+p_1)^2 - m_1^2, \quad (q+p_2) - m_2^2, \quad (q+p_3)^2 - m_3^2\}$$

  - Compact notation for integrals from this topology

$$G_{\text{box11}}(1,1,1,1), \quad G_{\text{box11}}(1,1,0,2), \quad G_{\text{box11}}(1,0,0,0), \quad G_{\text{box11}}(1,2,0,-1)$$

  - `box11` is the identifier of this topology, can be a number or any other string
  - The indices denote powers of the corresponding propagators

$$G_{\text{box11}}(1,2,0,-1) \equiv \int_q \frac{1}{[q^2 - m_0^2]} \frac{1}{[(q+p_1)^2 - m_1^2]^2} [(q+p_3)^2 - m_3^2]$$

- The reduction is done topology-wise: G's with more/higher indices $\Rightarrow$ G's with less/lower indices
- In principle, one *could* reduce each single integral separately, but this is *extremely* inefficient

- What if the corresponding propagators do not form a basis?
- For linear/quadratic propagators there can be two reasons:
  1) The basis is overdetermined due to linearly dependent propagators
  2) The basis is incomplete due to missing propagators
- Missing propagators: Just add some suitable scalar products to have the complete basis
- Heavy quarkonium calculations: linearly dependent propagators when switching to the center of mass kinematics
- Example: Integral with linearly dependent propagators

$$\int_l \frac{1}{[l^2]} \frac{1}{[(k_1+l-P/2)^2-m^2]} \frac{1}{[(l-P/2)^2-m^2]} \frac{1}{[(l+P/2)^2-m^2]}$$

Splits into two pieces when applying partial fraction decomposition

$$\frac{1}{2}\left(\int_l \frac{1}{[l^2]^2} \frac{1}{[(k_1+l-P/2)^2-m^2]} \frac{1}{[(l-P/2)^2-m^2]} + \int_l \frac{1}{[l^2]^2} \frac{1}{[(k_1+l-P/2)^2-m^2]} \frac{1}{[(l+P/2)^2-m^2]}\right)$$

- Tools for finding such relations: **MultivariateApart** [Heller & von Manteuffel, 2022], **$APART** [Feng, 2012], **FeynCalc**

- Starting point: amplitudes with many propagators and numerators. Need to
  - Identify present topologies and rewrite integrals as $G_{\text{topo}}(1,\dots) \Rightarrow$ easy
  - Minimize the number of topologies via suitable mappings $\Rightarrow$ not that easy
- Example: I_1 and I_2 fit into the same topology (actually, they are identical)
  - Unfortunately, this is not obvious in the propagator representation

$$I_1 = \int_q \frac{1}{[q^2 - m_1^2]} \frac{1}{[(q-p_1)^2 - m_2^2]} \frac{1}{[(q-p_2)^2 - m_3^2]}$$

$$I_2 = \int_q \frac{1}{[(q+p_1+p_2)^2 - m_1^2]} \frac{1}{[(q+p_2)^2 - m_2^2]} \frac{1}{[(q+p_3)^2 - m_3^2]}$$

  - There exists a suitable loop-momentum shift

$$I_1 \overset{q \to q + p_1 + p_2}{=\!=} I_2$$

- Finding such shifts in a *systematic* way is challenging.
- Enumerating all possible shifts by brute force is rather *hopeless*, especially beyond 1 loop ...

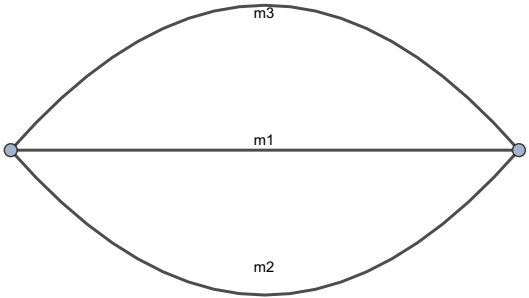
- More efficient: represent loop integrals through Feynman-parametric integrals or graphs
- Feynman parametric representation of an L-loop scalar Minkowskian integral [Nakanishi 1971, Zavialov 1979, Smirnov 2006]

$$\left(\frac{e^{\varepsilon\gamma_E}}{i\pi^{d/2}}\right)^L \int \frac{\left(\prod_{i=1}^L d^d k_i\right)}{P_1^{m_1}\dots P_N^{m_N}} = \frac{(-1)^{N_m}\Gamma(N_m - \frac{Ld}{2})}{\prod_{j=1}^N \Gamma(m_j)} \int_0^\infty \prod_{j=1}^N dx_j x_j^{m_j-1}\, \delta\left(1 - \sum_{i=1}^N x_i\right) \frac{\mathcal{U}^{N_m - \frac{(L+1)d}{2}}}{\mathcal{F}_M^{N_m - \frac{Ld}{2}}}$$

with N quadratic/eikonal propagators P_i and $N_m = \sum_{i=1}^N m_i$ with $m_i \geq 0$

- Euclidean case: remove $(-1)_m^N$, replace $\mathcal{F}_M$ with $\mathcal{F}_E$
- **Symanzik polynomials $\mathcal{U}$ and $\mathcal{F}$** encode most of the integral properties (nice summary in [Bogner & Weinzierl, 2010])
- Main advantage: ambiguity under shifts of loop momenta eliminated!
  - Employ a function of $(\mathcal{U}, \mathcal{F})$ and m_i to characterize the given loop integral (topology)?
  - Find mappings between different topologies?
  - In principle, yes! But things are not so simple ...

- Introduce the characteristic polynomial P
- Popular choices: $P = \mathcal{U} \times \mathcal{F}$ or $P = \mathcal{U} + \mathcal{F}$
- P depends on the Feynman parameters x_i and is not unique!
- Ambiguity under shifts of loop momenta $\Rightarrow$ ambiguity under permutations of x_i
- A new P' from P by permuting x_i (e. g. $x_1 \leftrightarrow x_2$) still describes the same loop integral
- Need to find some *canonical ordering* of the Feynman parameters x_i in the given P
- Possible solution: Algorithm invented by Alexey Pak [Pak, 2012]
- Rough idea: Write P as a matrix, find the canonical form by swapping/sorting rows and columns

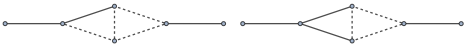


$$= -\Gamma(3-d) \int_{x_i} \underbrace{(x_1x_2 + x_1x_3 + x_2x_3)}_{\mathcal{U}}^{3-\frac{3d}{2}}$$
$$\underbrace{(x_1x_2 + x_1x_3 + x_2x_3)(m_1^2x_1 + m_2^2x_2 + m_3^2x_3)}_{\mathcal{F}}^{d-3}$$

- Pak algorithm: canonical ordering of x_i + symmetries between the corresponding lines.
- Very detailed description in the PhD thesis of Jens Hoff [J. S. Hoff, 2015]
- Technical implementation in **Mathematica** straightforward
  - Automatic calculation of $\mathcal{U}$ + $\mathcal{F}$ in **UF.m** (now part of **FIESTA** [Smirnov et al., 2022] and **FIRE**)
  - Many of Pak's ideas implemented in **TopoID** [J. Hoff, 2016], `https://github.com/thejensemann/TopoID`
- Pak algorithm + UF-parametrization pave way to
  - Topology identification
  - Mappings between master integrals
  - Derivation of Feynman-parametric representations for analytic calculations
  - Analytic regularization [Panzer, 2015a, 2014, 2015b]
  - And many more!
- Corresponding routines available in **FeynCalc**, **tapir** [Gerlach et al., 2023], **pySecDec** [Borowka et al., 2018], **LiteRed**

- Suppose that you are done with the IBP-reduction [Chetyrkin & Tkachov, 1981; Tkachov, 1981] of your loop integrals, what are the next steps?
  - Find mappings between master integrals from different integral families
  - Visualize your unique master integrals
  - Calculate them analytically/numerically?

- With Pak algorithm at our disposal finding mappings between master integrals is straightforward
- Example (using **FeynCalc**): 2-loop self-energies with one or two massive lines



```
In[1]:= topos = {FCTopology[prop2Lv1, {FAD[p1], FAD[{p1 + q1, m}], FAD[p2], FAD[p2 + q1], FAD[p1 - p2]}, {p1, p2}, {q1}, {}, {}],
          FCTopology[prop2Lv2, {FAD[p1], FAD[{p2, m}], FAD[{p2 + q1, m}], FAD[p1 + q1], FAD[-p1 + p2]}, {p1, p2}, {q1}, {}, {}]};
```

- We have a list of master integrals that are not all unique

```
In[2]:= glis = {GLI[prop2Lv1, {0, 1, 1, 1, 1}], GLI[prop2Lv2, {0, 1, 1, 1, 1}], GLI[prop2Lv2, {1, 0, 1, 1, 1}],
          GLI[prop2Lv2, {1, 1, 0, 1, 1}], GLI[prop2Lv2, {1, 1, 1, 0, 1}]}
```

- **FeynCalc** identifies three mappings between our masters

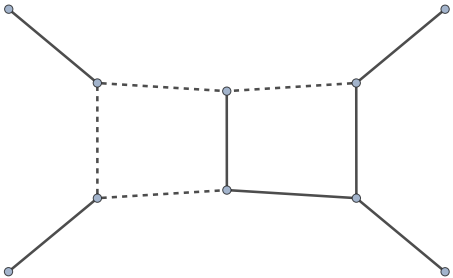
```
In[3]:= FCLoopFindIntegralMappings[glis, topos][[1]]
```

$$\text{Out[3]=}\ \{G^{\text{prop2Lv2}}(1,0,1,1,1)\to G^{\text{prop2Lv1}}(0,1,1,1,1),G^{\text{prop2Lv2}}(1,1,0,1,1)\to G^{\text{prop2Lv1}}(0,1,1,1,1),G^{\text{prop2Lv2}}(1,1,1,0,1)\to G^{\text{prop2Lv2}}(0,1,1,1,1)\}$$

- Working principle based on `FindRules` from **FIRE**
- Could also use **LiteRed**'s `FindSymmetries`

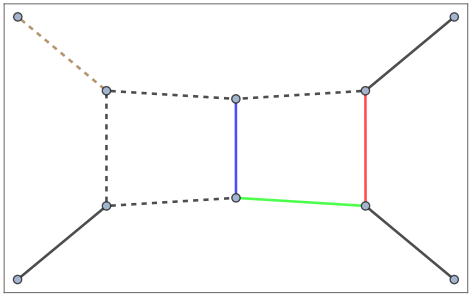
- Given a propagator representation of a loop integral, how to obtain its graph representation?
- Can be done using **AZURITE** [Georgoudis et al., 2017], **PlanarityTest**, [Bielas et al., 2013], **LiteRed** and also **FeynCalc**
- Two-step approach: `FCLoopIntegralToGraph` reconstructs the graph, `FCLoopGraphPlot` plots it.

```
In[4]:= FCLoopIntegralToGraph[FAD[{p1, m1}, {p2, m2}, {Q1 + p1, m3}, Q2 - p1, Q1 + p1 + p2, Q2 - p1 - p2, Q2 + Q3 - p1 - p2], {p1, p2}];
        FCLoopGraphPlot[%]
```



- The plot can be styled to make it more visually appealing

```
In[5]:= FCLoopIntegralToGraph[FAD[{p1, m1}, {p2, m2}, {Q1 + p1, m3}, Q2 - p1, Q1 + p1 + p2, Q2 - p1 - p2, Q2 + Q3 - p1 - p2], {p1, p2}];
        FCLoopGraphPlot[%, GraphPlot -> {MultiedgeStyle -> 0.35, Frame -> True}, Style -> {
        {"InternalLine", _, _, mm_ /; ! FreeQ[mm, m1]} -> {Red, Thick},
        {"InternalLine", _, _, mm_ /; ! FreeQ[mm, m2]} -> {Blue, Thick},
        {"InternalLine", _, _, mm_ /; ! FreeQ[mm, m3]} -> {Green, Thick},
        {"ExternalLine", Q1} -> {Brown, Thick, Dashed}}]
```



- At NLO we have plenty of options for the evaluation of 1-loop master integrals (quadratic propagators)
  - Analytically via **Package-X**
  - Numerically using **LoopTools**, **Collier** [Denner et al., 2017], **OneLOop** [van Hameren, 2011], **QCDLoop** [Carrazza et al., 2016], ...
- Beyond 1-loop things are not that simple
  - Numerical evaluation using **FIESTA** and **pySecDec**
  - Analytic results can be difficult to get, unless already available in the literature.
    - Plethora of different methods, impossible to discuss in one talk
    - Probably the most promising technique are differential equations [Kotikov, 1991a, 1991b, 1991c; Bern et al., 1994; Remiddi, 1997; Gehrmann & Remiddi, 2000; Henn, 2013]

Summary and Outlook

Summary

- Loop calculations is a vibrant field with new ideas being developed on a monthly basis
- Useful for heavy quarkonia, where often at least 1-loop accuracy is needed to match the experimental precision
- As far as 1-loop virtual corrections are concerned, all necessary tools for analytic results are publicly available
- It only takes dedication and willingness to learn how to use those programs