Contents

Chapter 1

Introduction

As of today, Quantum Mechanics and General Relativity are arguably the two most used fundamental frameworks which strive to fully describe reality. Although it is well known that their unification into a single theory which consistently describes the coupling between quantum mechanical systems and a supposed spacetime backdrop is a problem which has not been solved conclusively, they individually remarkably well describe observed phenomena within their respective domains.

QFT has thus advanced as the tool of choice to describe physics where no effects of strong gravitational fields, i.e. strong spacetime curvature, are to be expected. It serves as yet a more specific framework for building definite predictions with a particular model, i.e. a particular Lagrangian of the theory.

The most widely accepted QFT model today is the **Standard Model** (SM) with a Lagrangian for originally two times 18 plus either six or tree, depending on whether right-handed neutrinos are Majorana, fermionic (left- and right-handed quarks of different color and left- and right-handed leptons) and 12 bosonic (H, $W^{\pm}$, Z, and 8 gluons) fields. This choice is, however, made with the goal to describe reality with the most simple model and generically, there is freedom in choosing both, the parameters and the fields of a model, and thus the model itself. All efforts dedicated to "validating" the model eventually can at best observe no violations of the predictions.

To make such predictions for a given Lagrangian, however, the relevant parameters of the model have to be determined by a set of previous measurements first. And although the set of parameters is limited with only 26 physically relevant degrees of freedom (including the possibility of strong CP violation) in the SM, the circumstances under which those initial measurements may be obtained demand an enormous effort.

If we consider any experiment as a pair of input and output, it is clear that, first and foremost, there is intrinsically an uncertainty associated with both, the input and the output of the experiment and therefore, only statistical data can be gathered rather than definite values which were free of error.

On top of that, many measurements and almost all current ones concerning characterization of the Higgs particle are obtained at LHC. In analogy to how da Vinci's dissected the human body to learn about its anatomy, hadron colliders are more akin to a sledgehammer than a scalpel in this regard. The input of the experiment is not only afflicted by uncertainties it is not completely treatable with perturbation theory, either. Eventually, it is also, by definition, a transition experiment which is only accessible by quantum mechanical probability amplitudes.

Due to these two kinds of stochastic effects, the experiment can only ever provide samples from an underlying probability distribution which in turn is functionally related to the parameters of the model. Numerical fitting techniques such as the Matrix-Element-Method [17] which has been cited by many recent articles on Higgs analyses, are then employed to invert the relation such that the parameters can be obtained from the estimated probability distribution. With the obtained parameters, predictions for other experiments can be made to eventually validate the model.

In this thesis, the focus lies on a particular prediction which is used to assess the validity of the SM: The decay width of the Higgs boson. The goal and subject of this thesis is to provide the means to calculate more accurate theoretical predictions for comparison with experimental data. It addresses Integration-by-parts (IBP) reductions, the current bottleneck of many state-of-the-art perturbative calculations which is performed by reducing the required integrals to a set of fewer, s.c. master integrals.

First, it will be explained why the Higgs boson was introduced into the SM and a few alternatives are pointed out. After the general idea behind experiments for Higgs physics has been layed out, particular properties of the Higgs' decay width will be explained whence means for measuring the width shall be put into context. In section 3, the context which motivated the development of an improved IBP reduction algorithm will be recounted. Afterwards, the IBP method will be explained and its potential for improvement is analyzed. The remaining part of the thesis describes the new method in a generic framework for solving linear systems.

Chapter 2

Higgs Physics

2.1 The Higgs mechanism in the Standard Model

After the fermionic fields as the observed, fundamental constituents of matter, and force carrying vector bosons to mediate the strong, weak, and electromagnetic forces are introduced into the Lagrangian, the question arises why would one need an additional particle, although everything that can be observed should in theory be covered by those particles. And what is the "force" that it would correspond to.

However, if one were to construct the SM with all known particles, but without the Higgs and make predictions with what could approximately have been determined as parameters in such a model, various problems related to the masses of both, the fermionic and the bosonic fields would arise.

The simplest problem is that although massive fermionic fields were measured, no fermionic mass terms are a-priori allowed in the SM as a direct consequence of maximally broken parity by left- and right-handed fermions. That is, because they would violate an electroweak SU(2) symmetry, assuming the weak gauge bosons were in fact implemented in a corresponding manner.

Second, given that the Z- and W-Bosons are also measured to be massive, they, too, would require an a-priori mass term which leads to a propagator with bad UV behavior due to the additional $k_\mu k_\nu$ term and thus to a non-renormalizable theory, let alone no longer a gauge-invariant theory.

Third, and closely related to the second problem, the weak gauge bosons' masses would violate unitarity at tree level to the extent that higher-order corrections can no longer be considered perturbatively: At tree level, amplitudes would increase with the process' energy such that the unitarity theorem formulated by Froissard would be violated. In fact, it was claimed in [28] that tree-level unitarity is a necessary condition for unitarity of the theory.

It turns out that constructing a **spontaneous symmetry breaking** (SSB) sector in the SM allows mass terms for massive particles while at the same time circumventing above problems. For gauge boson masses this happens naturally as diagrammatic contributions, consequence of the construction rather than adding the according terms by hand. The modified Feynman rules then, for instance, lead to interferences which assert unitarity or scale the offending terms in the massive propagators to counter the non-renormalizable UV divergent behavior. The requirement of unitarity translates into s.c. sum rules, i.e. constraints about the couplings and masses [40].

In order to construct a gauge invariant mass terms for the fermions with the help of spontaneously broken field which compensates for the transformation of left-handed fields, the introduced field must transform with at least a two-dimensional representation of SU(2). Also, if charge is conserved, its vacuum expectation value must be invariant under the gauge transformation of QED. The **minimal sector** which satisfies the requirements is thus a scalar doublet. It breaks the four dimensional $\mathrm{SU}(2) \otimes \mathrm{U}(1)$ to a one dimensional U(1) which leaves one massive scalar while giving mass to three gauge bosons, the Z and Ws, as intended.

The minimal SSB sector, although it solves aforementioned problems about fermion- and gauge boson masses, is not the only possible mechanism in the SM to remedy the contradiction of the existence of masses and assuming it does not rule out further extensions.

Since 2012 after $7TeV$ and $8TeV$ runs at the LHC the ATLAS [3] and CMS [20] experiment have confirmed the existence of an excitation which fits the prediction of the Higgs particle with a mass of about $125GeV$.

However, it has already been proposed before that the electroweak symmetry is broken dynamically by non-perturbative interactions of the existing fermions. Particularly, a bound state of top quarks, the s.c. top condensate, and states bound by an additional force dubbed technicolor have been proposed [48]. In fact, although the original technicolor model was abandoned, its ideas have been carried over into modern theories which address further, contemporary problems.

2.1.1 Higgs as a gateway to BSM physics

If one assumes that EWSB is realized explicitly by the Higgs mechanism, then as a consequence such a model implies that extensions to the SM may become visible in observables whereas otherwise, in a different model, they would have not. For example, it turns out that the triple Higgs vertex even for more than the minimal SSB sector receives large corrections from a top-quark loop, although the mass of the top quark is such that naively, it could decouple by virtue of the Appelquist-Carazzone decoupling theorem. The decoupling theorem, however, is no longer valid in SSB scenarios [44, 39].

A great many other problems in particle physics suggest there is yet undiscovered content of the Lagrangian. To solve these problems, both, extensions and alternatives to the minimal EWSB Higgs mechanism, a.k.a. the SM Higgs, have been proposed which either include the minimal Higgs or a modification of it. These are the s.c. models with a non-minimal Higgs sector.

The central question regarding these extension is how to verify them or whether and how their predictions would deviate from those of the SM with a minimal Higgs sector. In the minimal Higgs sector, the fermion masses m_i, their couplings to the Higgs y_i and the Higgs mass or the vacuum expectation value v of the scalar field are related by

$$\frac{y_i}{m_i} = \text{ const } = \frac{1}{v} \tag{2.1}$$

With typical extensions, deviations from this relation will occur in one of various patterns. Sufficiently precise measurements may thus be able to rule out extension up to a certain limit. Vice versa, specific processes can be found which may indicate the necessity for extensions [41].

2.1.2 Higgs-mechanism BSM physics

The simplest extension of the SM Higgs are the s.c. N-Higgs-Doublet Models (NHDMs) of which the 2HDM is the most extensively studied one. Since the fields in those models are SU(2) doublets, the breaking of the symmetry always leads to three massive gauge bosons corresponding to the associated Goldstone bosons and $2^2N - 3$ Higgs fields.

While the 1HDM is the minimal Higgs sector for the SM, the 2HDM happens to be the minimal Higgs sector for a supersymmetric model where the structure of the Lagrangian demands two Higgs fields to provide gauge-invariant mass terms. The 2HDM is also proposed to explain baryogenesis and guarantee CP-symmetry of the QCD Lagrangian if there are CP-violating terms [13]. It has, for example, been shown that in the 2HDM model the additional fields do decouple and would thus agree with observations which confirm the 1HDM, if the additional Higgs is heavy enough [41].

All extended Higgs sectors naturally lead to more free parameters in the Lagrangian. However, depending on the particular extension, various requirements constrain the choice of possible parameters. For example, unitarity

[40], supersymmetry, and experimental bounds, like the measured gauge boson masses and the lack of **flavour changing neutral currents**[12, 29, 32]. The 2HDM model, for example, has been classified into Type I, II, X, and Y depending on which constraints are obeyed.

Instead of adding one or more additional Higgs SU(2) doublets, there are fairly popular models which add *triplets* or *singlets*. Most notably with regard to the 2HDM, the NMSSM (next to minimal super symmetric SM) adds yet another Higgs singlet to solve, among other things, a fine-tuning problem in the 2HDM supersymmetric model. Such a singlet affects little of the phenomenology and would, if it were of similar mass as the Higgs, be mistaken for such at the Large Hadron Collider (LHC) [8].

2.1.3 Non-Higgs-mechanism BSM physics

The aforementioned technicolor condensate has largely been abandoned since the discovery of a particle at the LHC which fits the description of the Higgs, particularly because of its disagreement with measured masses and couplings. However, many alternatives to the Higgs mechanism follow a similar, general approach in that the role of the Higgs is therein played by a **composite particle**. In these models, a bound state which is induced as a pseudo Goldstone boson by a **global symmetry breaking** replaces the Higgs; essentially the same mechanism by which the pion is generated.

The “little Higgs” models are a phenomenologically consistent version of this. They are tweaked such that the scale of broken symmetry whence new partner particles – though not supersymmetric partners – would occur, lies sufficiently high while maintaining the observed mass of the then effective peak in the spectrum at $125GeV$.

Lastly, higher dimensional theories such as of the Kaluza-Klein type wherein suitable boundary conditions are imposed on the fields can also serve to break the electroweak symmetry. For an overview over various composite Higgs models including higher dimensional ones see [4, 12].

2.2 Minimal Higgs sector and its implications

In the following, the minimal Higgs Sector as it is implemented in the SM shall be assumed. Although it does not ultimately address every problem that other approaches to EWSB are able to solve or mitigate, it is consistent with current observations and efforts dedicated to validating its correctness — or refute some of its extensions — are justified regardless of possible alternatives.

A crucial feature of the SM Higgs mechanism is that by virtue of SSB, the resulting **masses** and **couplings** of all, the Higgs, the bosons and the fermions to eachother altogether have only fewer degrees of freedom compared to when these parameters where introduced manually. In fact, once the scalar potential is fixed by a given μ and λ

$$V(\Phi) = -\mu^2(\Phi^\dagger\Phi) + \frac{1}{2}\lambda(\Phi^\dagger\Phi)^2 \tag{2.2}$$

the mass of the Higgs, its self interaction, the Gauge boson masses and their couplings to the Higgs are uniquely determined and so are the couplings of the fermions once their masses have been fixed, or vice versa [8]. In particular, the couplings of the SM Higgs to the massive particles is (inverse-)proportional to $v \equiv \mu/\sqrt{\lambda}$

$$g_{Hf\bar{f}} \sim \frac{m_f}{v} \tag{2.3}$$

$$g_{HVV} \sim v \tag{2.4}$$

In fact, it turns out that all couplings and masses but the very mass of the SM Higgs only depend on , the **vacuum expectation value** of the scalar field. From measurements of, for example, muon decay which determine G_F, the (effective) Fermi coupling via the W boson and thus the mass of the latter, v can be obtained, but the mass of the SM Higgs is equal to the mass of the scalar field $m_H = \mu$ and must thus be obtained from **independent measurements** of which in the low energy effective theories, which depend only on v, there are none.

However, bounds for the mass could be established before it was explicitly measured. From precision measurements of electroweak processes the Higgs mass could be determined to be in the range between ca. $20GeV$ to $700GeV$ [1]. As shown in figure 2.1, direct searches in the energetically available range have, prior to its detection, together with the requirement of vacuum stability ruled out $m_H < 114.4GeV$ [10]. The latter rules out a very light SM Higgs is because a lower Higgs mass, given fermionic masses and thus v, when $m_H = \sqrt{\lambda}v$ would correspond to a lower λ, possibly below the Planck-scale, which in turn leads to an unstable electroweak vacuum [8].

The need for unitarity, on the other hand, for example for partial wave scattering of longitudinally polarized W bosons, mandates highest bounds on $m_H < 1TeV$ [51]. Another argument which was frequently advanced to argue for upper bounds was that for SM Higgs masses above a threshold, perturbation theory would cease to converge [67].

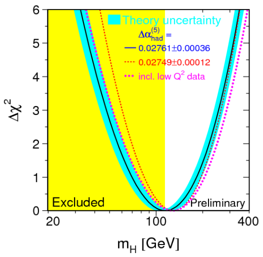

Figure 2.1: [1] Constraints of the Higgs mass as a χ^2 distribution from electroweak measurements where the Higgs mass affects the observables through higher-order corrections shown as the curve. The exclusion from direct searches at LEP is shown as the yellow band.

2.2.1 Hadron colliders

At a *pp*-collider such as the LHC, contrary to lepton colliders, the initial states of the process leading — through interaction — to the detected products can not be described perturbatively and, consequently, the process itself is partly non-perturbative.

Instead, the interaction of the inbound composite hadrons is modeled **probabilistically** from simpler processes at a hard scale involving only the proton's constituents, the s.c. **partons**. The probability of each of the simpler, partonic processes or, equivalently, effectively encountering a parton in a specific state during the scattering, is derived from a **parton distribution functions** (PDF) [26] illustrated in figure 2.2. The latter are in turn the result of probing the involved hadron in, foremost, deep inelastic scattering experiments [56].

The partons only carry a fraction of the momentum of their respective hadrons. The simple processes which we know how to calculate do thus occur with an unknown statistically distributed momentum of the inbound particles whose center of mass energy is known at an accuracy of $1GeV$ [68]. This is to be opposed to lepton colliders like the LEP where the whole process $l\bar{l} \rightarrow X$ can be modeled and also happens at more accurately measured beam energy which is known down to an accuracy of $1MeV$ [7].

Given the PDFs of the involved hadrons and a prescription for how to calculate the simpler processes which only involve partons, one may predict the observed and invisible events to be expected at a specific center of mass energy of the beam including various observables such as the **invariant mass** — i.e. squared four-momenta — of subsets of products or their **missing transversal momentum**. To evaluate the cross-section for an event in the

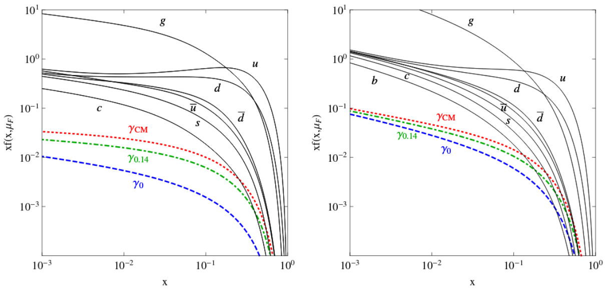

Figure 2.2: [61] Two PDFs of the proton, showing the probability of interacting with a specific parton depending on the fraction of momentum which the parton is carrying. The dependence on the momentum of the hadron is encoded in the factorization scale Q rather than expressing the probability in terms of the hadron's momentum.

hadronic collision, one thus convolves the partonic cross-sections with the PDF.

$$\sigma = \int x_1 x_2 \sum_{i_1,i_2} f_{i_1}(x_1, Q) f_{i_2}(x_2, Q) \sigma(i_1(x_1\sqrt{s}) \rightarrow i_2(x_2\sqrt{s}))$$

where the integrals over the momentum fraction x_i and sum over parton types i_1 and i_2 include all possible partonic processes which may lead to the considered event.

A typical comparison of theory and experiment of hadron collisions thus juxtaposes measured distributions of a combined variable, typically the number of event in a specific decay channel, over one or more properties of these such as the observed invariant mass, with the prediction. As for the discovery of the Higgs boson, shown in figure 2.3, it was the number of diphoton productions over the invariant mass that showed the predicted excess near the Higgs boson's mass which confirmed its existence most prominently [3].

When colliding at a specific beam energy, the observed invariant mass of the products of a partonic event will vary, according to the PDF, over the whole spectrum up to the beam energy. In lepton collisions, on the other hand, one produces only a single, known process with a specific energy at a time. The total momentum of all decay products is thus known, even if they are not observed up to initial state uncertainties in the beam energy known as beamstrahlung.

The PDF of the proton predicts that gluons and, second to that, light quarks — for low scales foremost the proton's valence quarks — are the

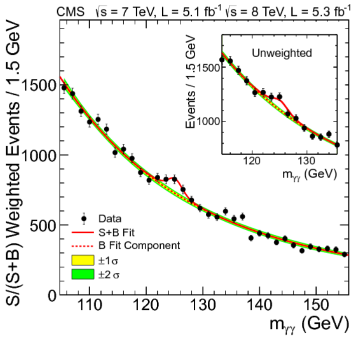

Figure 2.3: [23] The original plot from the LHC/CMS [20] experiment which shows the number of events observed around the expected peak of the Higgs. The solid red curve shows the theoretical prediction with an accordingly adjusted Higgs mass. The dashed red curve shows the same prediction without a Higgs.

dominant partons to interact in the hard process. Considering the associated partonic interaction, the dominant processes to SM Higgs production are **Gluon-Gluon-Fusion** (GGF), **Vector-Boson-Fusion** (VBF), **Higgs-Strahlung** processes (WH, ZH), and **associated production with top quarks** (TTH). Processes such as quark fusion of a sufficiently heavy flavours like charm and strange would contribute strongly from the involved vertices, but is suppressed due to lack of quark-antiquark presence in the proton's PDF [38].

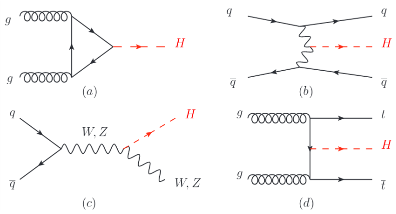

Figure 2.4: [54] The dominant production channels at the LHC in their order of importance. GGF (a), VBF (b), VH (c), and TTH(d).

At lepton colliders, the dominant processes are electroweak. At low energies, Higgs-Strahlung is the leading contribution but VBF, foremost W-boson fusion dominates towards higher energies.

2.3 Higgs decay and lifetime

2.3.1 Higgs detection

In principle, when evaluating measurements from experiment, every independent observable and the correlation among all observables is a valuable, more or less significant, piece of information which should contribute to making a statistic statement.

Also, there is typically more information associated with each event than just the type of its **decay product** and its invariant mass. On the most fundamental level, where just the momentum of an event could be measured, its time of detection and, secondary, location relate it to other events.

In a first step, the momenta of the decay products together with their temporal and spatial correlation allows to, with varying certainty depending on conditions such as the luminosity of the beam, classify and group multiple events as, for example, a single diphoton event. Doing so reduces the data while forging it into an interpretable form at the price of ultimately discarding information.

In order to turn experimental measurements into a form of data which can conclusively be interpreted with respect to a specific question — typically the validity of a model — this process of classifying, grouping, and discarding data is often repeated several times in what is referred to as (experimental) **cuts** and jet vetoing.

Applying cuts refers to considering only events which satisfy a specific bound, for example only events with an invariant mass in a given range or the products in which had a particular polarization. The term "vetoing" is commonly used with respect to jets where the events corresponding to the jet are ignored if the jet exhibits certain properties, such as a specific rapidity. When to be compared with theory, the computations may similarly be limited to only produce predictions within the limits of cuts so as to, for example, not waste resources for regions in which statistical significance would be low or skip comparison on regions which cannot be predicted appropriately altogether.

Eventually, one seeks to find a balance between mathematical simplicity and computational feasibility on the one hand and taking full advantage of the available information on the other.

VBF in Higgs production, for example, radiates the Higgs from a colorless exchange of a vector boson between the two quark lines. One can use this fact to isolate a class of events to be compared with equally isolated theoretical predictions. It can be shown [37], that jets originating from a VBF process are unlikely to be central. In theory, one could thus limit the analysis to processes with non-central jets [11].

2.3.2 Decay channels

The decay of the SM Higgs follows the same principles as its production, that is, as far as vertices are concerned, stronger coupling to heavier particles determines the likely decay channels. However, the **phase space** integral depends heavily on the product of the particles to the effect that the **branching ratios** of the Higgs are not simply sorted by the product's mass.

To obtain high statistical significance in experiments, one focuses on events the decay products in which have an invariant mass close to the Higgs-Mass because in these interactions, the Higgs mediated contribution is sufficiently dominant compared to other contributions and their interference, the s.c. background. In other words, one concentrates on the region around the Higgs induced peak in a diagram which plots the differential cross-section $d\sigma/dp^2$ where p is the momentum of the decay products.

The Higgs is predominantly produced from initial Gluon states and radiated from weak gauge bosons in quark-antiquark-interaction due to the particular form of the proton's PDF which does not provide sufficiently heavy flavors. Its decay, on the other hand, is not constrained by a PDF so that the dominating contribution is a LO decay of the Higgs into the heaviest, kinematically accessible quark-pair, which would be a $b\bar{b}$-pair. In general, the Higgs partial decay width to an on-shell fermion-antifermion pair follows at tree-level to [36]

$$\Gamma(H \to f\bar{f}) = \frac{G_F}{4\sqrt{2}\pi} N_c m_H m_f^2 \left(1 - \frac{(2m_f)^2}{m_H^2}\right)^{\frac{3}{2}} \tag{2.5}$$

to the effect that heavier fermions dominate the decay for heavier Higgs. Because the vector boson fields are massive, they have an additional degree of freedom provided by the Goldstone bosons or, equivalently, their massiveness, which implies that the polarization sum for on-shell particles contains a term of the form $p \otimes p$ and, consequently, an m_V^{-2} dependence, contrary to the m_f^{-1} dependence of fermions. Since overall, the phase space factor scales with m_H/m_V, The different mass dimension is then canceled by m_H^2 to the effect that the leading behaviour of the decay width on the Higgs mass

$$\Gamma(H \to VV) = \frac{G_F}{4\sqrt{2}\pi}\frac{1}{4}m_H^3\left(1 - 12\left(\frac{m_V}{m_H}\right)^2 + 72\left(\frac{m_V}{m_H}\right)^4 \right.$$
$$\left. -256\left(\frac{m_V}{m_H}\right)^6 + \left(\frac{m_V}{m_H}\right)^8 - \left(\frac{m_V}{m_H}\right)^{10}\right)^{\frac{1}{2}} \quad (2.6)$$

is increasing with m_H^3 rather than m_H. Since $m_V < m_H$, the squared term with negative sign dominates to the effect that the heavier Z has a lower branching ratio than $W^{\pm}$, as opposed to the fermionic case, where the leading term has a positive sign. Since the measured Higgs mass turns out to be below the threshold for vector boson production, the tree-level approximation does naturally not give valid results in that range. Corrected results for the branching ratios to off-shell products are shown in figure 2.5.

Last among the relevant decays are decays to massless particles via loops, just as in the production case. Their partial width increases, just as the above, with increasing Higgs mass.

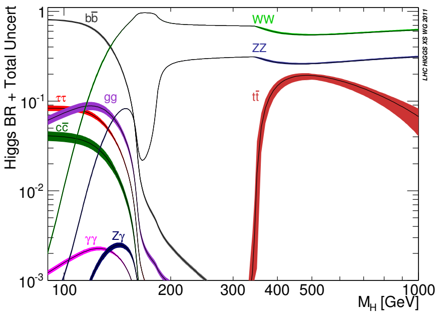

Figure 2.5: [34] Branching ratios of the SM Higgs in dependence of its mass to off-shell particles. The behaviour of each branching ratio is governed by three effects. First, the coupling of particles to the Higgs is proportional to their squared mass, which leads to the Higgs more likely decaying into heavier particles. Second, the momentum phase space for heavier particles is reduced because they will be produced at lower momentum similarly to multi-body decays. Third, since branching ratios describe the fraction of a particular decay among all decays, each branching ratio is also affected by how the partial decay widths into other particles behave.

The features of these individual decay channels lead to peculiar branching ratios and total decay width of the SM Higgs. If one combines all partial widths with state-of-the-art corrections to calculate the total width, it turns out that, as expected, it increases with increasing mass for a fixed set of decay channels. In fact, from $m_H > 500GeV$ upwards, the decay width is so enormously high that one could no longer speak of a peak in the spectrum. However, it should be noted that perturbation theory breaks down for large Higgs masses due to the Higgs self-coupling at such high masses [67]. On the other hand, at low values the width becomes very narrow. At the peak mass $m_H = 125GeV$, the width is predicted at about $4MeV$.

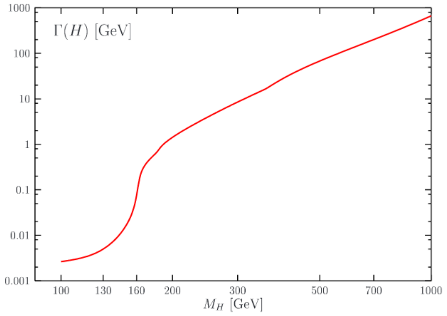

Figure 2.6: [36] The total decay width of the Higgs boson into any final state in dependence on its mass shown on a doubly logarithmic scale. The strong increase and subsequent bend occur at about the threshold for vector boson production.

In the change of branching ratios along different values of the Higgs mass one can identify that the gauge bosons dominate the decay towards larger masses due to the m_H^3 factor whereas the coupling to fermions is approximately ordered by masses for those fermions with about the same threshold.

2.4 Measurement of Higgs width

The complete process of production and decay of the SM Higgs receives, besides real and virtual corrections to the production and decay part as well as corrections connecting those two parts, corrections to the propagation of the Higgs itself.

Corrections which connect the production and decay part, i.e. virtual corrections which are neither restricted to the production nor decay individually, can be neglected because they are of higher loop and one can thus say that Higgs is produced on-shell and its production and decay decouple to the effect that they can be considered *separately*.

Corrections to the propagator of the Higgs as shown in figure 2.7, on the other hand, are responsible for the Breit-Wigner form of the peak which arises from this propagator. The full propagator

$$D = \frac{1}{p^2 - (m_H^{(0)})^2 + \Pi_H^{(0)}(p^2)} = \frac{1}{p^2 - (m_H^{(P)})^2 + \Pi_H^{(P)}(p^2)} \tag{2.7}$$

defined such that $\Re\Pi_H^{(P)}((m_H^{(P)})^2) = 0$ can be approximated near the pole as

$$D(p^2 \approx (m_H^{(P)})^2) \approx \frac{1}{p^2 - (m_H^{(P)})^2 + i m_H^{(P)} \Gamma_H} \tag{2.8}$$

with $\Gamma_H \in \mathbb{R}$, which is the Breit-Wigner distribution. This approximation discards every real part of $\Pi_H^{(P)}$ off the pole and discards the p^2 dependence of the imaginary part. Both effectively alter the form of the peak away from the Lorentzian shape which, however, is experimentally proven to be a good approximation.

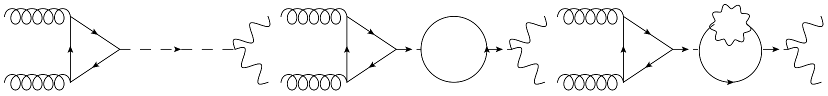

Figure 2.7: A series of virtual corrections to the Higgs propagator in GGF and subsequent decay into two Z bosons. The imaginary part of the propagator arises perturbatively at one loop from passing a threshold for the production of the particles in the loop, as given by the optical theorem or explicit calculation. Further loop corrections will necessarily always contribute an imaginary part. Therefore, the propagator of every unstable particle does have an imaginary part which increases with the number of possible decay channels.

The imaginary part of the self energy does arise from propagators the momentum of which is sufficiently high to produce on-shell particles according to the **optical theorem**. Since higher order loop corrections to the propagator involve diagrams with nested loops, such propagators ultimately always contribute an imaginary part in the full propagator. Also from the optical theorem it can be seen that adding further, possible decay channels for a particle to decay into increases the imaginary part and thus the decay width Γ_H in the approximate, full propagator.

Starting from a prediction for the decay width of any particle, one can conclude that if there were additional particles beyond those predicted that it could decay into, the width of its peak in the spectrum would increase.

Therefore, the width of a particle does not only allow to fit to a given model and thus verify it, but it also allows to detect indication for additional particles beyond a known and validated set in the model. In particular for the Higgs boson, the decay width allows to rule out BSM models the particle content of which would couple to the SM Higgs. The decay width of the Higgs is, besides its mass and coupling strengths to known particles as well as its CP properties, an important quantity for theoretical calculations and serves as a useful indicator for BSM searches.

As noted, in essence, every appropriate set of independent measurements of some physical observables can be used to infer an underlying parameter or effective parameter of the model. In that sense, countless methods exist to determine the decay width of a particle, some of which are more useful and effective than others depending on the circumstances. In broad terms, though, one can group these methods into several categories.

For long lived particles, a direct measurement of the particles **lifetime** τ which is inverse to its decay width $\Gamma = 1/\tau$ can serve as a reliable method to determine the latter. Among the several experiments to measure the lifetime, the one by Rossi and Hall to determine the μ lifetime (although thought to be a meson) is well known [60]. To be precise, the lifetime of the particle was in this case inferred from its decay rate, measuring how many of the muons would reach low altitude after production by cosmic rays in the upper atmosphere.

Alternatively, the lifetime of a particle can also be inferred from the average time-of-flight of the particle as it is done, for example, in time projection chambers and has recently been used for the SM Higgs to obtain a lower bound on its experimental decay width $\Gamma_H > 3.5 \times 10^{-9} MeV$ [47].

While lifetime measurements can always produce a lower bound for decay width, they cannot give a reasonable upper bound, let alone estimate for very *short lived* particles. In these cases, one relies on indirect methods which fit predictions to observations so as to fix the distributions' parameters.

The most straight-forward of these fits to determine the decay width and thus Breit-Wigner width of a particle, thus consists of fitting a Lorentzian curve to the cross section $\mathrm{d}\sigma/\mathrm{d}p^2$ of the process around the peak after subtracting the background. The background in turn, can either be calculated — given the relevant parameters were inferred from regions where the considered particle's contribution is suppressed — or approximated by interpolating across the peak if its sufficiently narrow.

The events obtained at ATLAS [25] and CMS [20] which are to be compared with theoretical predictions are, due to the nature of hadron-colliders, characterized by the decay products' invariant mass. The calorimeters which measure the energysof the decay products have at best a resolution of 1% to 2% for decay into two photons or four leptons. Reconstruction of the invariant mass for other decay channels is much worse with at best 10% accuracy [54, 21].

For a peak mass of around $125GeV$ and a predicted decay width of $\Gamma_H \approx 4MeV$ it is therefore not feasible to fit a Lorentzian curve to the shape of the observed spectrum for comparison with theory, because the result gives at best an upper limit of about $2GeV$ at the 95% confidence level when considering all statistical data [2].

Meanwhile, cross sections obtained at lepton colliders such as the proposed ILC [8] provide a much cleaner environment as far as the process is concerned and, additionally, provide extremely accurate information about the center of mass energy. For instance, the proposed ILC collider aims to provide a resolution of less than 0.01%, by a factor of 100 better than the most accurate calorimetric measurement at the LHC/CMS [43]. With such an energy resolution alone, at sufficient luminosities for, say, e^-e^+ collisions, the width of Higgs could indeed be measured by combining result from various decays with an accuracy of below 10% [8, 36].

2.4.1 Caola-Melnikov method

Although the LHC does not have a sufficient resolution to fit to the peak, there are other data besides the peak region at $125GeV$ which turn out to be sensitive to the Higgs decay width. A prominent method which has been proposed recently [19] uses data from the on-shell region at $125GeV$ and data from the off-shell region at around and above the threshold for on-shell Z production. It does, as generally preferable, attempt to correlate the measurements off the peak as exclusively to the Higgs width as possible. That is to say, the inference from the measured values back to the desired parameter, here the decay width, involves as few other model parameters — and thus assumptions — as possible.

In the region above the Z production the cross section for Higgs mediated Z production increases. As illustrated in figure 2.8, although the Higgs is off-shell in this region and thus reduces the contribution to the cross-section, the likelihood to produce it with such a high energy from the hadronic collision and the proximity of the Z peak to the decay eventually increase the cross section to a distinguishable amount above the background [45].

Besides the cuts on the invariant mass regions for on- and off-peak, events are typically further discriminated according to various criteria depending on the specific analysis [22, 46, 19]. Since only the decay products of the virtual Z-bosons are detected the decay has to be reconstructed which may

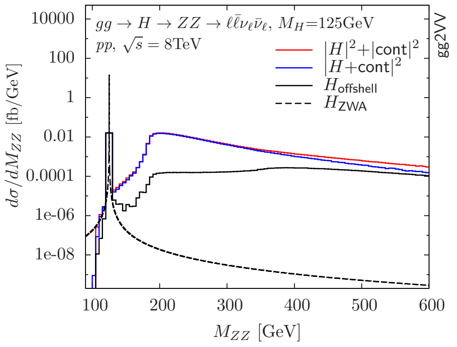

Figure 2.8: [45] The contribution of the Higgs mediated amplitude in the decay to $gg \to H \to 2l2\nu$ via two Z to the SM cross section shown in solid black or, including the interference, as the difference between the red and blue curve. As required by unitarity, the sum of the Higgs mediated amplitude and the interference is negative so as to reduce the cross section at larger center of mass energy.

thus impose further demands for accuracy. The decay of the bosons into four leptons is known to be a particular clear decay channel, especially because of the 1% - 2% resolution of e^- and μ invariant mass at the LHC. In the off-peak region, where the branching ratio to four leptons alone is not sufficient, a decay $H \to ZZ \to 2l2\nu$ with much worse energy reconstruction of about 20% is added into the consideration [46].

Caola and Melnikov (CM) consider approximations for the cross section on and off the peak for the decay to $e^+e^-\mu^+\mu^-$. With cuts on $p_\perp$, p^2 and $|\eta|$ for the electron and muons and a total $p^2 > 100GeV$ as well as additional conditions [57] one can restrict the considered events to those which actually originate from ZZ decay. The rate of these decays splits into the Higgs mediated processes, the background, and interference terms between those.

If one considers only the rate of Higgs mediated processes, it can be seen that the ratio of the approximate on-peak cross section with couplings g_i and g_f of the Higgs propagator

$$\frac{\mathrm{d}\sigma(pp \to H \to ZZ)}{\mathrm{d}m_{4l}^2} \propto \frac{g_i g_f}{(m_{4l}^2 - m_H^2)^2 + m_H^2\Gamma_H^2} \approx \frac{g_i g_f}{m_H^2\Gamma_H^2} \tag{2.9}$$

and the approximate off-peak cross section

$$\frac{\mathrm{d}\sigma(pp \to H \to ZZ)}{\mathrm{d}m_{4l}^2} \approx g_i g_f \tag{2.10}$$

allow to infer Γ_H and $g_i g_f$. The fact that this is only a crude approximation and that interference terms were neglected aside, it does, in a larger perspective, illustrate that there is indeed a strong correlation between data in the on- and off-peak region and the Higgs decay width. Caola and Menikov perform a proof-of-concept fit in their publication including interference contributions and subsequent work building upon their findings perform more scrutinous analyzes and comparisons to SM predictions from $m_H \approx 125 GeV$.

As far as model independence or, equivalently, dependence on other parameters of the theory such as $g_i g_f$ is concerned, one can conclude that if, for example, the above approximation were exact, direct inference of Γ_H could be made from dividing the differential on-peak cross section by the differential off-peak cross section. That is, given a histogram of the differential cross section with a certain energy resolution, every pair of on- and off-shell bins gives an estimate.

When the experiment deviates from the approximation, Γ_H can no longer be obtained by dividing only two values. More independent measurements are needed but the statistical- and thus error-correlation between, say, the on-peak divided by the off-peak cross-section deteriorates continuously as the approximation worsens. In the most general case, a numerical fit can be applied which considers all measurements simultaneously and reconstructs Γ_H with the best possible error. If the approximation were exact, the lack of correlation between Γ_H and other parameters given the data would be respected by the fit. Indeed, the method has been used at the LHC for $ZZ \to 2l2\nu$ and $ZZ \to 4l$ and concluded an upper bound of $\Gamma_H < 4.2\Gamma_H^{SM}$, where Γ_H^{SM} is the predicted value of $4.15 MeV$ [27].

Chapter 3

Two-loop amplitudes for $gg \to ZZ$

Since the relation between Γ_H at the on- and the off-peak cross section only illustrates the correlation, one ultimately, has to fit or compare theoretical predictions of the cross-section to the measured values. That requires to produce these predictions with at least the precision of the experimental determination in the first place. While techniques such as resummation allow to reach higher precision — or a sufficient precision at all — within fewer steps of perturbation theory, Feynman diagrams for perturbative amplitude predictions will eventually have to be calculated. The CM method applies to all production channels [27], foremost GGF [16] and VBF [15] which both have been considered in this context. The present work concentrates on predictions for GGF, which is the predominant production channel at the LHC.

After applying cuts to select events which are the most significant for analysis, those events and hence cross-sections are typically considered as a superposition of

- A signal from the process under investigation S — here, the Higgs GGF with subsequent decay in the four lepton channel.
- The SM background B — that is, other processes in the collision which lead to an accepted for lepton decay.
- The interference between the two I.

In terms of amplitudes, they correspond, in this order, to

$$|M_{fi}|^2 = |S + B|^2 = |S|^2 + |B|^2 + 2\Re(S^* B) \equiv |S|^2 + |B|^2 + I \tag{3.1}$$

Seen from the side of the SM without Higgs, the Higgs process plus interference contributions are in fact shown to be negative which reflects the fact that the Higgs unitarizes the S-matrix which is known to be non-unitary otherwise, because the amplitudes lie above the bound for unitarity [16, 28].

When considering a specific process perturbatively in QCD, one usually wants to predict the cross-section and thus absolute square of the amplitude, up to a certain order of the strong coupling constant g^n. For any given order n of a process with fixed initial and final states, the highest order of contributing Feynman diagrams is thus determined by the lowest order of contributing Feynman diagrams. For example, in $gg \to \cdots \to ZZ$ with Higgs mediated amplitudes being the signal, the lowest order Feynman diagram of the background is one-loop and of order α_s. Therefore, the highest order Feynman diagram in the signal which must be considered to calculate the interference cross-section up to order α_s^n is of order α_s^{n-1} and vice versa.
In collider experiments, the terminology of leading, next-to-leading order, and so forth, is used to classify amplitudes and cross-sections alike. The assigned order depends on the context. Formally, the order is given with respect to the order of a coupling constant and a set of processes. It can, for instance be any of the following examples, sorted by granularity:

- Amplitudes for a process with prescribed initial- and final states, e.g.: NLO w.r.t. $gg \to ZZ$ QCD corrections
- Amplitudes for a process with only partially prescribed initial- or final states, e.g.: NLO w.r.t. ZZ production QCD corrections
- All amplitudes of the model w.r.t. the perturbative series of the theory, e.g.: NLO in QCD

The leading order amplitude for a specific process then refers to the diagram with the lowest order in the coupling constant which contributes to the process. Note that referring to an order of the coupling constant α_S rather than $g = \sqrt{4\pi\alpha_S}$ is only possible if one knows in advance that no contributions of odd order in g occur. Apart from real corrections, this is naturally the case at hadron colliders but is not applicable if an order of the coupling constant w.r.t. QCD like mentioned in the third bullet is referred to.

Higgs production by GGF and VBF together with decay into four leptons has naturally received a lot attention. The $4l$ decay channel was, next to the $\gamma\gamma$ decay, the discovery channel of the Higgs and still is, due to its clean signature and good resolution, the most straight-forward detection method. The recently proposed CM method has since been applied by many groups to already existing data to constrain Γ_H.

The process in this area, however, depends strongly upon the ability to make accurate predictions from the theoretical model. Significant progress has been made concerning the analytic or numeric calculation of Higgs production by GGF. Results up to N^{3}LO QCD are known for the amplitude as described in [5] and references therein.

Cross sections for $gg \to ZZ$ are known exact at LO [16] whereas the estimate for Γ_H given in [27] made assumptions about NLO contributions based upon the relation between the known LO and NLO background amplitudes that should be validated by explicit calculation. For that, interference terms in $gg \to ZZ$ at NLO (α_S^3) are needed, which already require two-loop amplitudes for $gg \to H \to ZZ$ to interfere them with amplitudes of order α_S from the background as illustrated in figure 3.1.

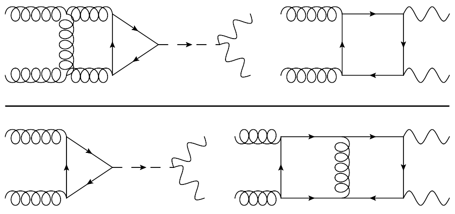

Figure 3.1: Two representative interference terms at NNLO $gg \to ZZ$ QCD at $\mathcal{O}(\alpha_S^6)$ from the Higgs mediated signal (left) and the background (right).

Recently, a set of LO and NLO background and signal amplitudes in an improved large mass expansion (LME) were calculated in [18] and combined to provide the first NLO accurate interference to validate the assumptions in [27] (the expansion of the signal amplitude was performed for consistency and to cross-check the results with the exact result). It was therein argued, that the improved LME gives good results, even above the $t\bar{t}$ mass threshold, based upon impressive accuracy illustrated by comparison at LO for the Higgs mediated processes.

But although the improved LME can, in principle, be applied to arbitrary high order in the heavy mass, there is currently no known method to bound the error due to how Padé-approximants [9] are involved in its construction[18]. Therefore, the correctness and accuracy of the approximation relies on estimates. In fact, the LME, which gives excellent results for the Higgs production and thus signal amplitude at NLO, where the t-quark loop is restricted to a $2 \to 1$ process, fails to converge with higher orders of $1/m_t$ for the $2 \to 2$ kinematics of NLO $gg \to ZZ$ background signal. The remaining error was estimated by scale variation to be about 20% when exceeding the threshold by $M_{ZZ} > 2m_t$ [18].

Albeit in agreement with [27] within the error, it is thus desirable to finally obtain exact two-loop amplitudes with *full* top-mass dependence. Not only will these results serve to validate assumptions of previous works and provide more accurate results; the generic methods thereby developed should also pave the way for hitherto intractable calculations.

There is notoriously no known catch-it-all algorithm which would be able to make fully automatic predictions of, say, cross-sections and almost every calculation needs manual intervention from the theorist to, most prominently, deal with singularities. But there are methods which automate part of the calculation and make it easier. The remaining sections in this chapter shall explain the ideas behind a set of well-known methods for the evaluation of multi-loop Feynman diagrams involved in the $gg \to ZZ$ interference at NLO with full top-mass dependence. In the following chapter, one of these methods known as Integration-by-parts, which is currently considered the limiting factor of the solution, is re-developed in a new, highly parallelized software framework and applied to the two-loop calculations of $gg \to ZZ$.

3.1 Integral reduction

When calculating amplitudes by evaluating Feynman diagrams in perturbative QCD, one finds that the number of possible diagrams involved in the calculation increases exponentially with the number of loops. Besides the number of diagrams one thus has to evaluate, the increasing complexity of the involved integrals makes it successively more difficult to find solutions to each diagram individually. The algebraic challenge to evaluate the integrals is supplemented by an increased complexity in the tensorial structure of the expression.

Section 3.1.1 shall exemplify one, albeit less practical and rather academic method which can be used to bring the original integrals into a more manageable form. In that sense, it merely shows a proof-of-concept for how integrals can be brought into the form — stripped of their tensorial structure — that will be assumed in the subsequent sections 3.1.2ff., where the actual solution of the integrals is performed.

3.1.1 Passarino-Veltman reduction

The amplitudes that are to be obtained are expressed in terms of Feynman diagrams which contain momentum integrals $\int \mathrm{d}k^4$ over conjugate spacetime, i.e. four-dimensional momentum space. Spacetime is thereby singled out and so is any Lorentz-tensor structure that the integral may contain, as opposed to tensor structures over other spaces — only color- or spin-space in the case of QCD.

Various methods to simplify the tensorial structure of integrals which arise in the raw amplitudes exist, the well-known Passarino-Veltman (PV) reduction [55] is a particularly simple and straight-forward universal algorithm. In its universality, it does, however, often produce results of impractical form with spurious singularities in the kinematic variables. Alternative methods, [33, 6, 35] to name just some examples, exist. They are tailored to produce more suitable results for specific situations. In fact, such specialized algorithms are in practice often preferred to PV reduction, despite the latter being simple and universal. This short exposition of the PV reduction should therefore only serve to illustrate the idea and the possibility of simplifying Feynman integrals in a particular manner.

The general idea behind PV reduction is to simplify the integral by knowledge of which quantities it does depend on. In general, the numerator contains operators which can be pulled out of the integral completely. The best example are γ operators on the spin-spacetime product space which can always be pulled out, because they only occur in the numerator, and contractions with external momenta p_i which do not occur in the denominator.

$$\begin{aligned}&\iint \mathrm{d}k_1 \dots \mathrm{d}k_l f(k_1, \dots, k_l, q_1, \dots, q_m, \gamma_1, \dots, \gamma_n, p_1, \dots, p_o) = \\ &\gamma_1^{\mu_1} \dots \gamma_n^{\mu_n} p_1^{\nu_1} \dots p_o^{\nu_o} \iint \mathrm{d}k_1 \dots \mathrm{d}k_l f_{\mu_1 \dots \mu_n \nu_1 \dots \nu_o}(k_1, \dots, k_l, q_1, \dots, q_m) \end{aligned} \quad (3.2)$$

From the fact that the integrand of the integral is geometrically invariant — though not in its components — under a Lorentz transformation of spacetime for all involved Lorentz quantities, one can infer that so must be the integral. In other words, it is a tensor which depends only on q_1 through q_m.

If one asks, which quantities the inidices μ_1 through ν_o can be put on, then it turns out the only possible quantities is the intrinsic operator (the metric) g and q_i so as to guarantee valid transformation when the q_i are Lorentz-transformed. For example,

$$\begin{aligned}&\int \mathrm{d}k \frac{(q \cdot k)(p \cdot k)\not{k}}{(k+p)^2 - m^2} = \\ &\qquad q_\mu \gamma_\nu \int \mathrm{d}k \frac{k^\mu (p \cdot k) k^\nu}{(k+p)^2 - m^2} \equiv q_\mu \gamma_\nu (c_1 g^{\mu\nu} + c_2 p^\mu p^\nu) \end{aligned} \quad (3.3)$$

This relation provides enough equations to determine the unknown coefficients c_i by projecting the integral and the ansatz

$$\begin{aligned} g_{\mu\nu}\int \mathrm{d}k \frac{k^\mu (p\cdot k) k^\nu}{(k+p)^2-m^2} &= g_{\mu\nu}(c_1 g^{\mu\nu} + c_2 p^\mu p^\nu) \\ \int \mathrm{d}k \frac{k^2 (p\cdot k)}{(k+p)^2-m^2} &= c_1 4 + c_2 p^2 \end{aligned} \tag{3.4}$$

$$\begin{aligned} p_\mu p_\nu \int \mathrm{d}k \frac{k^\mu (p\cdot k) k^\nu}{(k+p)^2-m^2} &= p_\mu p_\nu (c_1 g^{\mu\nu} + c_2 p^\mu p^\nu) \\ \int \mathrm{d}k \frac{(p\cdot k)^3}{(k+p)^2-m^2} &= c_1 p^2 + c_2 p^4 \end{aligned} \tag{3.5}$$

which corresponds to an inhomogeneous linear system for the unknown vector c. The solution in case of the example is

$$\begin{aligned} c_1 &= \frac{1}{3}\left(\int \mathrm{d}k \frac{k^2 (p\cdot k)}{(k+p)^2-m^2} - \frac{1}{p^2}\int \mathrm{d}k \frac{(p\cdot k)^3}{(k+p)^2-m^2}\right) \\ c_2 &= -\frac{1}{3}\left(\frac{1}{p^2}\int \mathrm{d}k \frac{k^2 (p\cdot k)}{(k+p)^2-m^2} - 4\frac{1}{p^4}\int \mathrm{d}k \frac{(p\cdot k)^3}{(k+p)^2-m^2}\right) \end{aligned} \tag{3.6}$$

and illustrates a typical problem with PV reduction, namely the appearance of singularities from the inversion of the linear system e.g. p^2 in our case. In non-trivial examples with multiple external kinematic variables, these singularities appear through the s.c. Gram-determinant and are the foremost reason why the PV reduction is often not the method of choice to simplify integrals.

3.1.2 Integration-by-parts reduction

The integrals which are produced by the PV reduction are usually referred to as scalar integrals, where "scalar" should be understood with respect to spin-space. The primary achievement of PV reduction is thus to simplify the numerator of the integrand, although cancellations with the denominators, i.e. scalar propagators may occur accidentally. Pictorially speaking, arbitrary Feynman diagrams are thereby expressed in terms of Feynman diagrams with only scalar fields and the same topology. IBP reduction [24], on the other hand, gives relations between integrals of vastly different structure. Since PV reduction can be applied, one considers scalar integrals when using IBP identities.

In dimensional regularization, any integral of an integrand f can be made converge to the effect that its surface terms are negligible

$$\int \mathrm{d}^D k_1 \ldots \mathrm{d}^D k_l \frac{\partial}{\partial k_i^\mu} \left[p_j^\mu f(k_1, \ldots, k_l, p_1, \ldots, p_m) \right] = 0 \tag{3.7}$$

$$\int \mathrm{d}^D k_1 \ldots \mathrm{d}^D k_l \frac{\partial}{\partial k_i^\mu} \left[k_j^\mu f(k_1, \ldots, k_l, p_1, \ldots, p_m) \right] = 0 \tag{3.8}$$

Applying these equations to a single integral yields relations among various integrals and especially the second one with $i = j$ contains the original integral as well. Since an arbitrarily many of these identities can thus be generated, putting arbitrarily many integrals into relation to eachother, the essential question is which identities and integrals are best used to advantageously express a given integral in terms of others.

For a simple example, consider the massless one-loop vertex correction to a decay into two massive particles, given by the three point function C_0 (three external legs, two independent momenta)

$$C_0(p_1, p_2, m) = \int \frac{\mathrm{d}^D k}{(2\pi)^D} \frac{1}{((p_1 + k)^2 - m^2)((p_2 - k)^2 - m^2)k^2} \tag{3.9}$$

With $D(p, m) \equiv p^2 - m^2$, we can express every product of either p_1, p_2, or k with k as a sum of those D which can be identified in the denominator of the three point function

$$C_0(p_1, p_2, m) = \int \frac{\mathrm{d}^D k}{(2\pi)^D} \frac{1}{D(p_1 + k, m) D(p_2 - k, m) D(k, 0)} \tag{3.10}$$

and terms which do not depend on k.

$$S \equiv \{D(p_1 + k, m),\ D(p_2 + k, m),\ D(k, 0),\ \mathrm{const}_1, \ldots, \mathrm{const}_n\} \tag{3.11}$$

with $\partial/\partial k^\mu \mathrm{const}_i = 0$. Note that in general, however, this set of propagators and constants is not always sufficient and a set of scalar products needs to be added, e.g. pk or qk.

Applying the IBP relation $\int \mathrm{d}^D k \partial/\partial \cdot^\mu [k^\mu \ldots] = 0$ to any integral written in that manner will produce scalar products in the numerator while increasing the power of the D in the denominator. After replacing everything which appears in the numerator by a sum of elements in S, the final identity will necessarily be of the form

$$0 = \sum_i \mathrm{const}_i \int \frac{\mathrm{d}^D k}{(2\pi)^D} \frac{1}{D^{\alpha_1}(p_1 + k, m) D^{\alpha_2}(p_2 - k, m) D^{\alpha_3}(k, 0)} \tag{3.12}$$

The constants due to dimensional regularization, will typically depend on $\epsilon \equiv D/2 - 2$ which appears foremost from terms of the form $\partial/\partial k^\mu k^\mu = D$. Evidently, the operands in any identity obtained by subsequent application of the IBP relations will fit this schema or topology and may even lack one of the D, i.e. be in a subtopology. Combining the right set of identities one eventually obtains an identity which relates the three point function to simpler integrals, the s.c. and so-chosen master integrals

$$\begin{aligned}\epsilon m^2((p_1+p_2)^2-(2m)^2)C_0(p_1,p_2,m) = \\ (1-2\epsilon)m^2\int\frac{\mathrm{d}^Dk}{(2\pi)^D}\frac{1}{D(p_1+k,m)D(p_2-k,m)}+ \\ (1-\epsilon)\int\frac{\mathrm{d}^Dk}{(2\pi)^D}\frac{1}{D(k+p_1,m)}\end{aligned} \quad (3.13)$$

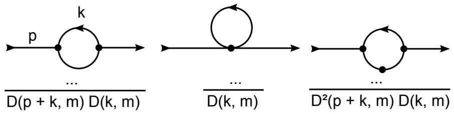

Figure 3.2: Although IBP identities are formulated as a mere mathematical technique, their intuitive description in terms of diagrams is often helpful in interpreting them. Numerators of the scalar integrals aside, just as one would not consider them if they were spurious remains of a previous reduction from spinor-space integrands, each integral corresponds to a specific Feynman diagram, all of which have the same topology. When one applies IBP identities to an integral and thus raises or lowers the power of some of the scalar propagators, it can be interpreted as "dotting" or "collapsing" the associated lines respectively.
Consequently, if a string of edges (one D) has been completely collapsed to the effect that its two end points merge, there is no way any "dotting" can separate them again. We obtain a permanent simplification and the simpler diagram is in a subtopology of the more complicated one. In particular, due to the action of the derivative, the power of denominators can be increased by at most one but decrease by more, depending on the cancellations that occur when decomposing scalar products in the numerator.

It is the very goal and achievement of this thesis to implement a fast, parallel solver which systematizes the utilization of IBP identities to obtain this kind of reduction, based upon work by Laporta [49] and Czakon [30].

As will be explained in section 4.2, the question in terms of which integrals the original integral, here C_0 should be expressed, does not have a conclusive answer. In fact, the above example illustrates an important feature which is related to this question, namely a trade of divergences. While C_0 is UV-convergent, it has a soft divergence due to the massless line, whereas the integrals on the right-hand-side of the equation have only massive lines and are UV divergent.

Indeed, divergences and the ability to regulate them cannot be considered separately from the mere structural simplicity of the associated diagrams. Since, eventually, the master integrals will have to be evaluated, a particular choice of expressing them may lead to structurally simpler diagrams some of which have divergences which are more difficult to treat analytically — contrary to our example. Therefore, although this thesis does describe a fully automated approach to the IBP reduction problem, it may frequently be favourable to express integrals differently than suggested by the algorithm, depending one one's methods to evaluate them. In any case it should be noted that the algorithm which is used to express integrals in terms of master integrals works independently of their choice.

3.2 Solution of master integrals by IBP

One partly analytic method to eventually evaluate master integrals which were returned from an IBP reduction does in turn use IBP reduction. It proceeds by calculating the master integral M_i at a simple point in its parameter space — i.e. by fixing its unbound variables such as external momenta p_i and masses m at convenient values — and then evolving the solution to an arbitrary parameters through a differential equation [59]. Typically, the parameters which are fixed and subsequently evolved are kinematic invariants, e.g. masses are fixed at 0.

The evaluation of the integrals at the boundary of the evolution is performed by Feynman-Parameters, Mellin-Barnes-Integrals, difference equations or various other methods, a generic selection of which (together with a summary of IBP reductions) can be found in [63].

The differential equation is obtained by differentiating the master integral with respect to the corresponding variables. Following the same argument as for the IBP generation, where differentiation with respect to the loop momenta was performed, it is clear that the differentiation of a master integral can yield integrals from the subtopologies but never from a different topology. It was shown by Petukhov and Smirnov in a rather convoluted paper [62] that every integral from a given topology of infinitely many integrals can be reduced to a finite set of master integrals. A fact originally not even anticipated in such generality [59]. Therefore, after in turn reducing the derivatives by IBP identities

$$\frac{\partial}{\partial m} M_i(\dots, m, \dots) = \sum_j c_{ij}(m, \epsilon) M_j(\dots, m, \dots) \tag{3.14}$$

one obtains a closed system of first order differential equations in each of the variables. In particular, derivatives for a differential equation in a mass parameter do not generate new terms in the numerator and thus only increase the power of denominators. If, for whatever reason, one does not succeed at closing the system, the only implication is that some master integrals will have to be calculated by other means while all others can then be evolved by their consequentially inhomogeneous differential equations.

As noted above, the choice of master integrals is not unique and especially for the generation of differential equations in a given variable m for master integrals, a specific suggestion for a s.c. canonical basis [42] has been made which is supposed to result in simple differential equations. In such a canonical basis, the equations take on the s.c. ϵ-form

$$\frac{\partial}{\partial m} M_i(\dots, m, \dots) = \epsilon \sum_j c'_{ij}(m) M_j(\dots, m, \dots) \tag{3.15}$$

where, in addition, it has been conjectured that c_{ij} be of Fuchsian form

$$c'_{ij}(m) = \sum_k \frac{c''_k}{m - m_k} \tag{3.16}$$

which would allow an analytic solution for the master integrals M_i in terms of polylogarithms. Recent advances in this area by Lee et al [52] have, however, also shown that such a canonical basis can not always be found because it would assert a positive answer to Hilbert's 21st problem about the existence of Fuchsian linear differential equations [52].

The unchallenged universal approach to solving the differential equation thus consists of a series expansion in ϵ and possibly other variables like m. Alternatively, numeric solutions may be obtained where possible, for example in the mass [14], though the stability of such an evaluation may present serious problems [31].

After the usual Laurent-series expansion in ϵ inherent to dimensional regularization which allows to discard parts of $\mathcal{O}(\epsilon)$ and above to simplify the problem without loss of information, it can be concluded that whence the differential equations are given in a specific, optimally Fuchsian form, the full solution is has reduced to solving a system of differential equations. A plethora of methods like, for example, direct integration of a higher order differential equation, for this purpose exist in the literature few of which, however, are tailored towards the peculiarly singular coefficients encountered in the context of QFT amplitudes.

Chapter 4

High-Concurrency IBP reduction

4.1 Graph terminology for Gauss-like algorithms

Before proceeding to the actual algorithms, a generic language for dealing with a **homogenous linear system** and describing constraints in its solution will be introduced in this section. In that formal treatment, it will become evident what are the fundamental operations of which a solution is composed and how various **solution strategies** differ from eachother.

Without loss, we assume that the linear system Ax = 0 under consideration has been stripped of all redundancies, i.e. is **linearly independent**. We also assume that it was indeed underdetermined, i.e. $\dim\ker A > 0$. We are then left with m equations for $n > m$ components of x. In other words, x can be parametrized by $n - m$ unknowns. If those unknowns are chosen to be components in a particular basis, we speak of a canonical parametrization with respect to that basis.

4.1.1 Choice of pivots

Clearly, the components for a canonical parametrization of the kernel can not be chosen arbitrarily. Pictorially and despite the lack of any metric, one might say that, for example, the components to basis vectors whose projections into the kernel are linearly dependent are not a suitable choice.

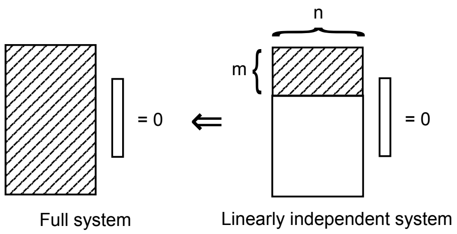

Figure 4.1: A homogenous linear system corresponds to a matrix equation wherein the unknowns are collected in the associated vector. Each row of the matrix represents one of the equations.
Each equation does either constrain the space of possible solutions or it does not. Every equation which does *not* constrain the solution beyond what is already constrained by the other equations, can be expressed as a linear combination of the latter. For n unknowns, the linear system may thus at most have n linearly independent equations in which case there is only the trivial solution and $\dim \ker A = 0$.
For our purposes, the linear systems under consideration have a non-vanishing kernel, i.e. there are fewer linearly independent equations than unknowns. We make a choice for a maximal set of linearly independent equations (shaded rows on the right) and truncate the matrix by removing the superfluous rows. The resulting system still expresses all constraints and the solution will therefore satisfy the original system (left) as well.

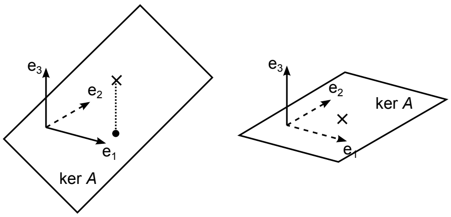

Figure 4.2: At the example of a 2-dimensional kernel in a 3-dimensional domain (with an arbitrary metric and euclidean for the purpose of illustration) it can be seen how one might choose a basis such that not all pairs of components are suitable choice to parametrize any point in the kernel.
On the left side, the kernel's is does not orthogonal to any basis vector and every pair of components can be chosen. On the right side, the plane is orthogonal to the e_3 basis vector, the associated component of which is thus required to be 0 and not suitable for parametrization.
In higher dimensional spaces one can generalize the construction and see how a set of basis vectors whose projection onto the kernel is linearly dependent does not constitute a valid choice. In the illustrated 3-dimensional case, the projection of one orthogonal vector is already linearly dependent. Note that the validity of any choice is independent of any choice of metric which only served to illustrate the idea.

In line with terminology of the well known Gauss algorithm, the components which are chosen to be parametrized by a canonical parametrization are called **pivotal components**, pivotal columns, or **pivots** in short.

To specify the criterion whether a set of components or columns P may be chosen as pivots, let us recall from linear algebra that regular column transformations on the matrix of A correspond to a basis change on the domain whereas regular row transformations correspond to a basis change on the image.

Since P corresponds to a set of basis vectors in the domain, any set of row transformations will leave the validity of P as a choice unchanged. If the matrix, by row transformations, can be brought into a form where the columns P can be concatenated into a unit matrix, a possible canonical parametrization by the remaining columns has been found and we speak of the parametrized form. Particularly, this choice assures that none of the remaining components is required vanishing, as can be seen by counting the degrees of freedom.

In conclusion, the choice of pivots, which contains information about the nature of the kernel, is done as information about the kernel is obtained. Depending on the particular algorithm, this may be at the very end after all row transformations, when the kernel is already identified or incrementally.

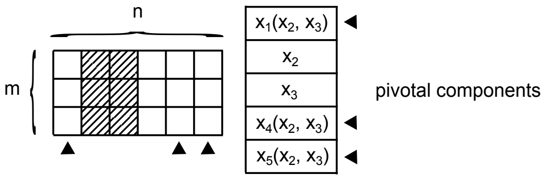

Figure 4.3: An example for a system of 5 unknowns with 3 equations wherein the second and third component, corresponding to the second and third column of the matrix, have been chosen as parameters and the remaining 3 columns are the chosen pivots.

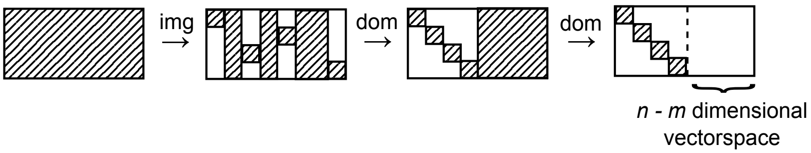

Figure 4.4: Row transformations bring the initial, linearly independent system (left) into a form where m columns can be concatenated into a unit matrix by permuting the domain basis (left middle to right middle). In this form, a parametrization can immediately be read off. With another basis transformation of the domain in which the pivotal basis vectors are subtracted from the other so as to turn the dangling right block of the matrix zero (white in the right picture), we can find a basis, which is an extension of the kernel's basis and pivotal components are required vanishing.

4.1.2 Choice of pivotal rows

A typical incremental choice which pertains to **Gauss-like algorithms** such as Laporta's algorithm involves assigning a unique row, the pivotal row, and thus element of the matrix to each pivot - a process called pivoting. The defining property of what is called a Gauss-like algorithm in this context is that in these algorithms, the matrix is brought into parametrized form by successively eliminating all elements of each rows which is the pivot of another row by subtracting the corresponding pivotal rows.

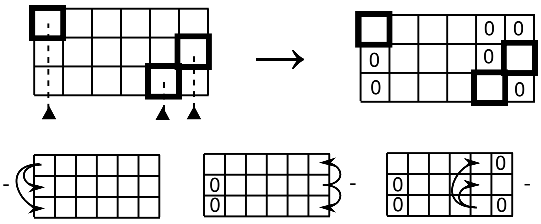

Figure 4.5: Assuming a valid choice, each row is assigned a unique pivot. As shown in the bottom row, the pivotal elements outside of the assigned row can then be eliminated by successively subtracting the according row of the pivot. For example, the first row may be subtracted accordingly from the 2nd and 3rd to turn the elements in the first column to 0 and the process may then be repeated with the 2nd and 3rd row.

As will be seen, the choice for each pivot during the solution is made when it is sure that the associated row can be brought into a form where all other pivotal columns in that row vanish while the column of that very pivot does not.

4.1.3 Graph representation

With a choice of pivots and associated pivotal rows, the matrix can be represented by a pivot graph which reflects both the intention and the progress of a Gauss-like algorithm. Every component is a node. Every coefficient forms the weight of an edge pointing from the pivot associated with its row to a component associated with its column. The weight of self-edges, that is coefficients which are in the pivotal column of a pivotal row, is increased by one.

If edges of vanishing weight are omitted in the graph, this prescription implies that if and exactly if the system is in parametrized form, all edges to pivots are eliminated.

Regular row transformations translate into transformations of the graph

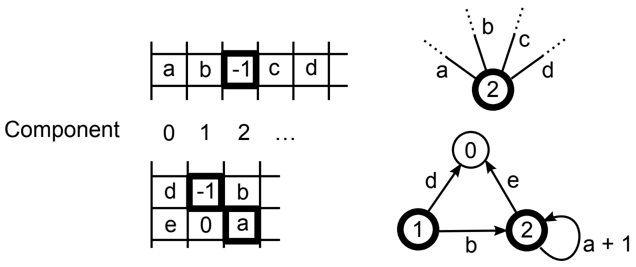

Figure 4.6: Two examples of graphs corresponding to parts of linear systems. In most of the examples to be given, only parts of a linear system will be shown and nodes as well as edges to these nodes will be omitted for clarity. Because of linearity, all operations henceforth discussed can directly be applied when more than the shown edges and nodes are involved. They act on all edges individually and the existence of an edge does not change how other edges are affected by an operation. In other words, any transformation acts on a **primary edge**, denoted as a thick line, and **other edges**. Only the action on one of the other edges is given and can be applied as such to all further, other edges.
Shown above is the basic mapping from part (including its pivot) of a row of a matrix to the according node and edges of the graph. Shown below that is an example of how two rows build a simple graph. Components which are not pivots and edges to them will often be suppressed in the following so that pivots will sometimes not exhibit any outgoing edges in the depiction whereas in reality, all their edges are leading to non-pivotal components.
Components which are not pivots, such as the 0th component in the example, do not have any outgoing edges because their is no associated row. If the linear system is indeed underdetermined and linearly independent, pivots, on the other hand, must have outgoing edges.

and can be build from a minimal set of three operations designated **relay**, **collect** and **normalize**, which allow to construct operations in a non-redundant manner.

Normalization and associated redundancy

Normalizing a pivot corresponds to eliminating *one* **self-edge**, i.e. turning the associated coefficient to -1.

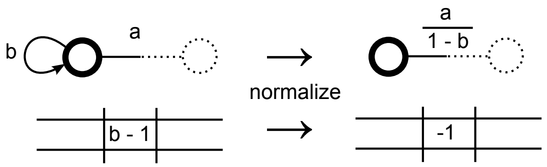

Figure 4.7: Normalization transformation sets the self-coefficient to 1 and therefore the weight of the self-edge to 0 such that it omitted. As a consequence of normalization, other outgoing edges (including further self edges) are divided by 1 minus the eliminated weight.

A row transformation as it would typically be performed manually involves divisions, multiplications and sums. If such a row transformation is performed multiple times with the same row to modify multiple other rows, the involved division is the same multiple times. Although the performance of such operations does eventually depend on how they are implemented, it is at least conceivable that disentangling the operations such that

$$x_1 = a_1 b/c$$

$$x_2 = a_2 b/c$$

is effectively calculated as

$$y = b/c$$

$$x_1 = a_1 y$$

$$x_2 = a_2 y$$

which corresponds to a prior normalization, does at least not worsen performance. In essence, and as can be seen from these equations, we have isolated a duplicate expression to compensate for redundancies.

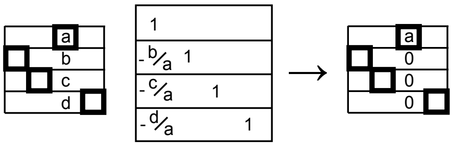

Figure 4.8: Normalization as an individual operation can be identified as a common sub-operation of multiple substitutions. In the above example, 3 substitutions as they would usually be performed in the classic Gauss algorithm are encoded in the transformation matrix (left). They each substitute the first row with the according factor from the three lower rows. Since $1/a$ is applied three times to each coefficient of the first row, it is most likely beneficial to perform this calculation only once, explicitly.

Relay and associated redundancy

Relaying an edge to a pivot corresponds to the deliberate elimination of this edge or coefficient by substitution of that pivot. In the graph, the edge which is relayed is dissolved and is replaced by edges which now lead to the components reachable from the head of the old edge.

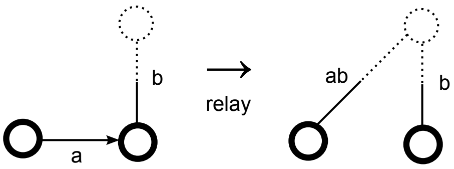

Figure 4.9: Relaying an edge "forwards" that edge along all other edges leading out from the node at its head. If there are multiple edges leading out from the node's head, the relayed edge is distributed or "splits" and is relayed along each edge separately. The coefficient of the resulting edges is the original coefficient multiplied with the coefficients of the edges that were relayed along.

In Gauss-like algorithms, each relay is supposed to eliminate an edge permanently such that at most $(m-1)m$ steps are necessary to parametrize the system. If an edge which was deliberately eliminated by relaying it reappears through the relay of another edge, we say that the edge is reintroduced.

While the number of necessary steps to fully parametrize the system is bound from above my $(m-1)m$, an algorithm may however, be more or less fortunate about its choice of pivots and order of operations. Which particular series of operations leads to a lower necessary number and which leads to a higher one depends on the values of the coefficients.

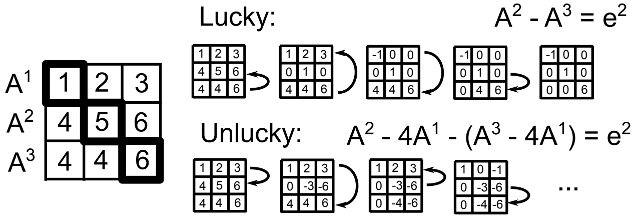

Figure 4.10: For the simplest case, consider how the pivotal columns of a system with 3 rows can be brought into the desired form. Since the three pivotal columns together form a linearly independent 3-by-3 system, there is exactly one linear combination of two rows which, subtracted from the remaining row, yields the row with all desired elements eliminated.
Eventually, it is the goal of any Gauss-like algorithm to find this precise combination. But an algorithm may reach the result more or less effectively depending on both, the choice of pivots and the order in which the algorithm considers the rows. In the given example, the second row can be brought into the desired form by subtracting only the third row because there is an additional cancellation. Next, the first row can be brought into the desired form by subtracting the third row when again, an additional cancellation occurs. That way, a "lucky" algorithm reaches the parametrized form in only 4 steps.
An "less lucky" algorithm, on the other hand, absent a-priori knowledge about the system, could attempt an equally valid set of subtractions which would also lead to a result but not benefit from additional cancellations. It could need up to $m^2 - m$, the number of off-diagonal elements, operations.

Since, after a reintroduction, the system is not necessarily in the same state as before, i.e. the values of the coefficients may have changed, this may then lead the algorithm to an indeed quicker solution despite or even because of the reintroduction.

However, it seems reasonable to not rely on this positive and very unlikely side-effect of an otherwise completely undesirable situation and rather forbid reintroductions wholesale. Both, the Gauss algorithm and the algorithms presented here do this.

Reintroductions can occur if, after relaying an edge between two nodes, there remains a directed path between those two nodes. They are a particularly bad case of what is commonly known as **fill-in** in Gauss-like algorithms. That is, creating edges or coefficients which were previously zero.

Preventing fill-in and thus reintroduction, which is generally an NP-complete problem known as minimal chordal completion, can easily be achieved by the following recursive treatment if the graph is **acyclic**. When an edge is relayed, all offending edges on the head pivot which would cause fill-in are relayed first.

If, however, the graph is cyclic, reintroductions on directed cycles in the graph cannot be resolved by this easy prescription because it would provoke infinite repetitions. The most immediate form of such a reintroduction is along a cycle of size 1, that is the self edge of a pivot which is not normalized. Therefore, any relay of an edge requires that the pivot to which this edge points is properly normalized. This automatically captures the redundancy from duplicate expressions which were mentioned in the context of normalization and is one reason why the different notation of self-edges is justified.

The Gauss-algorithm prevents reintroductions by defining an arbitrary, partial ordering between connected pivots and thus defining partitions of directed, acyclic subgraphs of the graph, the lesser and the greater directed subgraphs. The lesser subgraphs are then made vanish by successively relaying all their edges, from the smallest pivot through the greatest until only the greater subgraph remains, on which the procedure is repeated in reverse order. This corresponds to the recursive treatment for acyclic graphs described above when *everything* should be solved, contrary to when only a certain pivot should be solved and the recursion touches only the relevant edges.

In Laporta's algorithm and the derivative of it used here, reintroduction is prevented by relaying all edges to a given pivot, before relaying any other edges. If self-edges are properly eliminated, this obviously makes reintroduction impossible, since there is no way a pivot may be reintroduced. Again, this prescription by itself does not suffice to reduce fill-in.

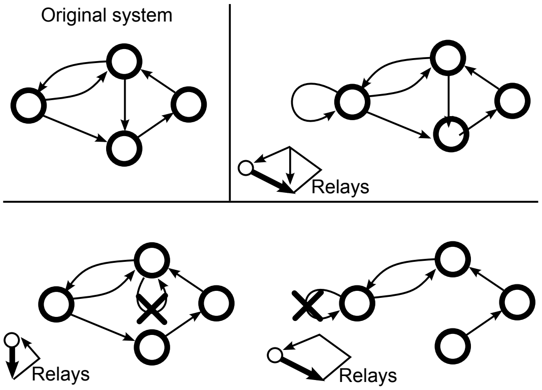

Figure 4.11: The evolution of an edge in a graph through relays can be depicted by drawing a tree, starting with the edge (the stem) and advancing along the nodes that the edge is relayed at. In such a tree, loops are closed when the leaf of a tree goes back to the tree. When the leaf touches the lower end of the stem, the resulting loop is a self-edge and can be removed by normalization. If, however, the leaf touches the higher end of the stem, the loop corresponds to a reintroduction.
In the above example, the designated edge at the bottom left of the graph should be eliminated. In the top row, where this edge is relayed several times, a reintroduction on a directed cycle occurs in addition to the resolvable self-edge. In the bottom row, we see that if the middle edge, which eventually effects said reintroduction, is eliminated first by relaying it, then the former edge can safely be relayed without reintroduction.
For example, generally, one possible method to eliminate an edge without reintroduction is to relay just before a reintroduction would occur and then eliminate the edge which would cause it. Using this method recursively will eventually solve any system without reintroductions. However, this method would not effectively minimize re-fill.

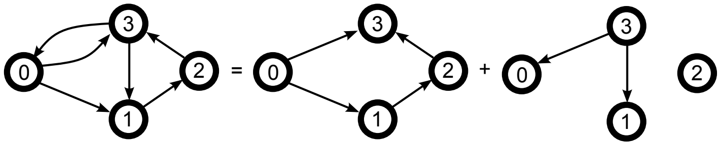

Figure 4.12: The original graph is split into a "lesser"- and a "greater" subgraph which are then successively made vanish.

Collection and associated redundancy

Collecting on a pivot means combining multiple edges to the same component into a single edge. Collections are not always necessary to prevent redundancies and could be omitted after a single relay, albeit without any apparent gain. They are, however, necessary on further relays where omitting them would indeed induce a duplicate expression and thus redundancy.

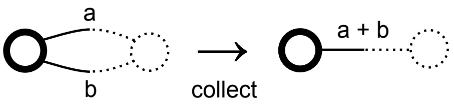

Figure 4.13: Collection combines a set of edges into one edge. Other edges are left unaffected.

Besides the deliberate elimination of edges by relays, edges may also disappear through collections, when their weight cancels to zero. In consequence, collects are to be performed before normalizations and before it is decided whether an edge has to be relayed.

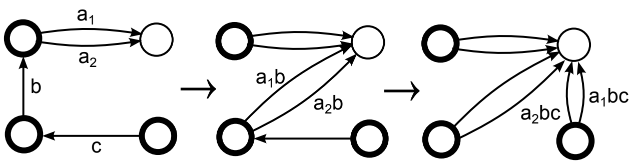

Figure 4.14: The relay of the edge with weight b above does not lead to a redundancy if a_1 and a_2 are independent coefficients because the resulting edges have no common subexpression. A second relay on those two edges, however, then creates a redundancy.

4.2 Systematic IBP identities solution

The method of obtaining IBP identities can in principle be applied not only to the integrals whose solution we want to obtain in terms of simpler integrals, but also repeatedly to those integrals in turn appearing in the identities and even to at first unrelated integrals, whose associated identities will eventually contain integrals whose value we seek.

However, the infinite set of integrals which are coupled by IBP identities can be quantified so as to establish an explicit limit for how many and which IBP identities are considered for the reduction. Generically, every integral that is either to be reduced or results from application of IBPs can be brought into the form

$$\int \mathrm{d}^D k_1 \dots \mathrm{d}^D k_n \frac{\prod_{(i,j)\in N_{(p\cdot k)}} (p_i \cdot k_j)^{\beta_j} \prod_{(i,j)\in N_{(k\cdot k)}} (k_i \cdot k_j)^{\gamma_y}}{\prod_j D_j^{\alpha_j}}$$

to have a unique, canonical way of identifying them. This happens by decomposing every scalar product that may initially appear in the numerator in terms of propagators D_i and a finite set of s.c. **irreducible scalar products** $N_{(p\cdot k)} \cup N_{(k\cdot k)}$. To determine this definition uniquely, the set of propagators D_i shall be taken from the propagators that appear in the integral which should be reduced and one fixed choice for irreducible scalar products is made so as to allow this decomposition.

Essentially, this identification with a canonical representation could be considered as an auxiliary set of identities which identify integrals belonging to the same representative in pairs. Since they are only equalities which do not involve multiple integrals they can, however, be applied a-priori by bringing them into normalized form.

For a given range of α, β, and γ, this yields a corresponding set of IBP identities and thus a linear system wherein integrals are the unknowns. This range, determining the set of identities, has to be chosen manually. Suggestions for these values do exist. However, if one expects a better reduction from increasing the set of identities, more identities should be generated.

4.2.1 Laporta's algorithm

In essence, besides the order of how eliminations are performed, Gauss-like algorithms differ only by their pivoting strategy. Gauss' algorithm, as it is taught in linear algebra classes or school proceeds in an arbitrary row order and picks an arbitrary pivot in each row, as induced by an arbitrary permutation of the bases. As a refined version of this, Gauss' algorithm for numerical solutions proceeds also in an arbitrary row order but specifically picks the largest pivot in each row for numerical stability. The pivoting strategy suggested by Laporta [50, 49] to solve the symbolic system of IBP identities is even more specific as it proceeds in a prescribed row order and makes deliberate choices for the pivot of each row.

From the origin of IBP identities, which relate Feynman diagrams with different topologies, it can be seen that they exhibit a particular structure which relates only a certain set of integrals to eachother.

Heuristically speaking, the identities obtained by applying IBP to a topologically more complicated integral are likely to contain topologically more complicated integrals, too. A sensibly correlated strict order of complexity, a higher sum of powers of the denominators α_i, seconded by numerators β_i, γ_i and auxiliary criteria is then used to construct a suitable pivoting strategy.

1. number of denominators n
2. sum of denominator powers less the number of denominators $\sum \alpha_i - n$
3. sum of numerator powers $\sum \beta_i + \sum \gamma_i$

and, to define a unique order between integrals which are not yet differentiable by the above, in order of which powers and which propagators appeared.

In terms of this order of complexity, the very goal of the reduction can be formulated as wanting to express an integral in terms of a set of least-complex integrals. That is to say, the least complex integrals are the most desirable — though not necessarily a possible — choice for parameters to parametrize the kernel. They are typically called the master integrals. From this, the primary rule of pivoting strategy is to choose the most complex possible component in a given identity as pivot, so that the less complex ones eventually remain. Note however, that this order — contrary to Gauss' algorithm — does not serve to prevent reintroductions.

In Laporta's algorithm, reintroduction is prevented by the complete relay of all edges to a pivot, henceforth referred to as relaying a pivot.

At the same time, the order of relays is chosen such that pivots are eliminated in the order that they are found. In conclusion, when a new identity is considered, all pivots therein referred to are immediately relayed, i.e. eliminated. The pivot then chosen is similarly relayed on the already considered system and future identities yet to be added.

With a choice for which pivot to choose in each identity, there is still freedom in choosing the order in which identities are considered. Considering the relay of a specific pivot, the order of identities can be chosen to make it unlikely that this pivot occurs in the previous identities, by considering identities in overall increasing complexity. That is, considering identities in order of the complexity of the integral from which they are obtained.

The choice of the most complex integral as a pivot is made after all previous pivots in this identity were relayed so as to guarantee a valid choice of pivots and in turn parametrization of the kernel. If there were no deviations from the desired mechanism by which earlier identities do not contain any references to the later pivots, the resulting choice of pivots would immediately result in an acyclic graph to which we know a straightforward solution exists.

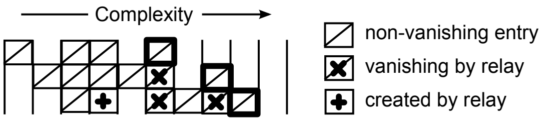

Figure 4.15: If we arrange components in order of complexity increasing to the right, the first identity or first row which was generated by applying IBP to a less complex integral, will also have less complex components and be thus further to the left. The most complex, i.e. right-most of the non-vanishing columns will then be chosen as the pivot in this row.
When the next identity or second row is generated by applying IBP to a more complex integral, the resulting components will also be more complex. After relaying the previously chosen pivot in that identity, the most complex column among the remaining components is chosen as the next pivot.

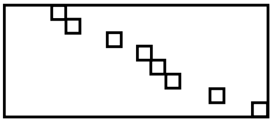

Figure 4.16: With columns ordered by complexity, successful application of the described algorithm would lead to a series of pivots which is strictly right advancing with only zeroes to their right. Also, the choice of pivots would be unaffected by the performed relays as each time the most complex integral from an identity is chosen as pivot. In other words, the resulting graph would be purely acyclic and also we would immediately have obtained information about the kernel without any calculation.

In reality, however, especially amongst multiple IBP identities from the same or similarly complex integrals, the identities considered earlier do frequently contain integrals which are chosen as pivots for later identities. In those cases, not only are the additional relays required to prevent reintroduction, but the identity's pivot may initially vanish so that it is only normalizable after an appropriate set of relays.

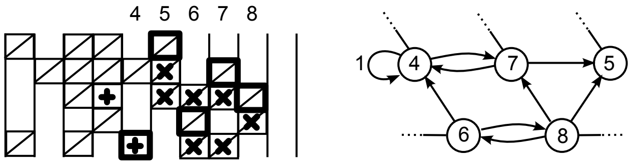

Figure 4.17: Because the proposed mechanism is only approximate, the situation is not as optimal as suggested and the components of successively generated IBP identities are often *not* strictly advancing to the right. Several things can happen when this is the case.

1. A generated identity may a priori only contain components which are left to the current right-most pivot (last row).
2. The relay of previously assigned pivots may eliminate the right-most component in the generated identity (previous to last row).
3. What remains as the right-most non-vanishing component in the generated identity may actually have been created only through the relay of other pivots (last row).

When the second or third of these things happen, the resulting graph is no longer acyclic, as can been seen in the example above. On top of that, if the third thing happens, the pivot is not normalizable until an appropriate set of relays has been performed.

Czakon's algorithm

The algorithm developed by M. Czakon in 2005 addresses the problem that the naive application of Laporta's algorithm to the relevant graphs of our problem will quickly render steps as computationally heavy as to no longer be viable. It also enhances the algorithm by using additional linear equations, such as symmetries or otherwise provided by the user, beyond the IBP identities. Together, these equations shall be referred to simply as identities.

In essence, the drastically bad performance of the original algorithm can be attributed to two causes:

First, coefficients resulting from the prescribed substitutions grow extremely fast. Although the coefficients present in the end result are of moderate computational size, they *easily* become of the order of 10^{10} characters in their expanded form during intermediate steps. While this may in principle suggest that a very specific method of calculation could reach the same results more efficiently, no such method is known, let alone realized by Laporta's ad-hoc algorithm.

> Without cancellations, it is conceivable that the size of coefficients on average no less than adds up, depending on which operation is involved and how the results are represented internally.
> If, for example, rational functions are internally represented in distributed form, i.e. as a sum of rational functions without parentheses, dividing two sums will result in an expression which is about the combined size of the operands whereas multiplying them will blow the resulting size out of proportion. On the other hand, without "lucky" cancellations, it is unlikely that a result will ever be smaller than the size of its operands, let alone smaller than one of the operands.

Second, the first effect is aggravated by the fact that said substitutions are performed wholesale across all generated identities, the set of which is only limited by a manual, rough estimate. The solution of a particular integral, i.e. writing it in terms of master integrals, however, only concerns a subset of the generated identities.

That is because the whole generated system is a set of linear equations, some of which are linearly dependent among eachother.

Generically, Laporta's algorithm detects such linear dependencies when the prescribed substitutions cancel all coefficients. This follows by induction since when an identity is generated, the previously generated set is by construction linearly independent in the subspace spanned by the *pivotal basis vectors*. Relaying all previously assigned pivots then corresponds to subtracting a unique combination of the previous identities such that the according components vanish. Therefore, if the current identity or row *is* linearly dependent and all components can be made vanish, the particular unique combination will achieve that.

Eventually, although those calculations were then needed to detect the linear dependence and thus effect the removal of the equation, they were eventually superfluous as far as the solution of a particular integral is concerned.

On top of that, the solution of a particular integral, after the initial system was reduced to N linearly independent equations, may not even involve the full set of the remaining $N-1$ identities. Instead the system can and does frequently **decouple** such that not only the linearly independent subset of the identities is needed, but even only a subset thereof.

Whereas the first issue of **intermediate expression-swell** is intrinsic to the generic solution of the associated linear system, the increased effects of it due to the second issue can and were largely removed by Czakon by substituting all symbols with numerical values in a first iteration. Doing so reveals linear dependencies and allows one to perform the reduction to N linearly independent identities with practically no expression swell. Only in extremely rare cases were numerical values **cancel** whereas symbolically they wouldn't, the numeric solution will remove an identity which would otherwise have contributed to the reduction.

After this first numerical pass where redundant identities were dropped and pivots assigned, the linearly independent identities, their pivot assignments, and the order in which those assignments took place are stored in s.c. "identity databases" for later processing. In a **second iteration** based upon these data, the actual symbolic solution is performed. At this stage, the system whose solution is to be obtained is already fully known and available, contrary to Laporta's algorithm were it is successively generated while substitutions are being performed.

> As explained above, an algorithm may be more or less "lucky" in how it performs relays which corresponds to exploiting additional cancellations. If the goal is to relay all edges of a particular pivot like that of the third row in the initial example, a "lucky" algorithm may end up with fewer operations. That said, there is always a risk that an algorithm performs calculations which could have been forgone. What is thus referred to as the work that is saved by Czakon's algorithm is not those superfluous operations from an "unlucky" order of relays but those operations which are in fact irrelevant to the solution of a particular pivot as it is performed by the algorithm.

Instead of pivoting the system incrementally during the solution, the pivots and their associated rows are already determined and make it possible to solve a particular integral symbolically without considering decoupled parts of the system. To obtain this solution, the same relays that have been performed numerically and would have been performed in the complete, symbolic process, are now applied to the pivots whenever the solution concerns them. Thus, the solution **recurses** through the graph, starting at the pivot of the sought integral. All edges to less complex pivots are then relayed, after these pivots have been treated in the same fashion.

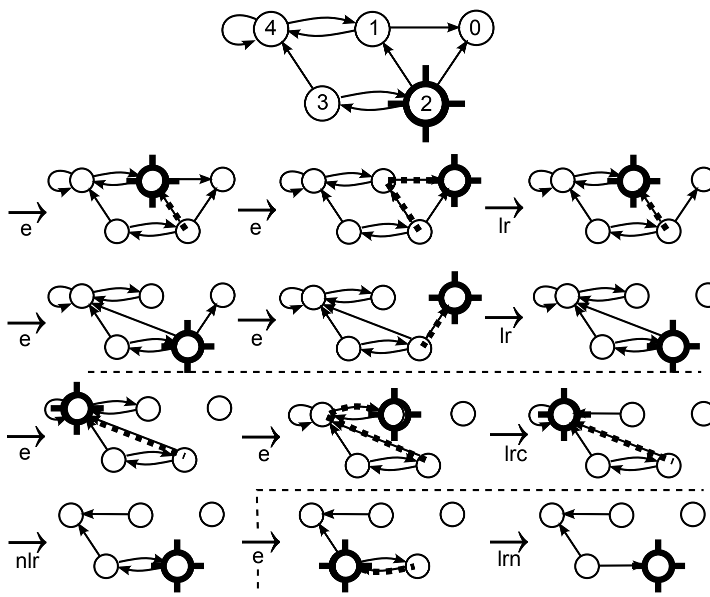

Figure 4.18: Czakon's algorithm starts from a pivot graph which was generated in the numeric pass, which in turn followed Laporta's algorithm. The naming of pivots was now changed to reflect the order in which pivots were assigned.
The goal of the algorithm is to solve for a specific pivot. In the example given here, pivot 8, should be expressed in terms of non-pivotal components, i.e. the integral should be expressed in terms of master integrals.
In each step, the current target and trace of the recursion are indicated by a crosshair and the dotted trail respectively. It is denoted between the steps whether the algorithm enters the recursion ("e") or leaves the recursion ("l") and performs the relay ("r"), collect ("c") or normalization ("n").
Above the dashed line, the algorithm attempts to recovers the state in which pivot 8 was assigned. It does so by determining the relays that were performed and reproducing them, but not so before the involved pivots were in turn treated so as to recover the state in which they were relayed. After each dashed separation, one of the remaining edges to other pivots is eliminated by recursing to pivots which were assigned *after* pivot 8.
If every subgraph containing pivot 8 is acyclic, then every time all earlier n pivots are relayed, exactly n identities which are linearly independent in the subspace spanned by the n pivotal basis vectors are used. Therefore, the resulting coefficient is uniquely that obtained in the numeric pass. If however, any subgraph is cyclic, later pivots may enter the consideration to the effect that the self-coefficient may differ from the numeric run.

4.3 Quicksolve and CKS algorithm

The parallelization of the solution is achieved, apart from various auxiliary tools, by two layers of Quicksolve (QS), henceforth referred to as the frontend and the backend. We will first describe the general idea of what constitutes these two layers and how they work together in order to perform the solution.

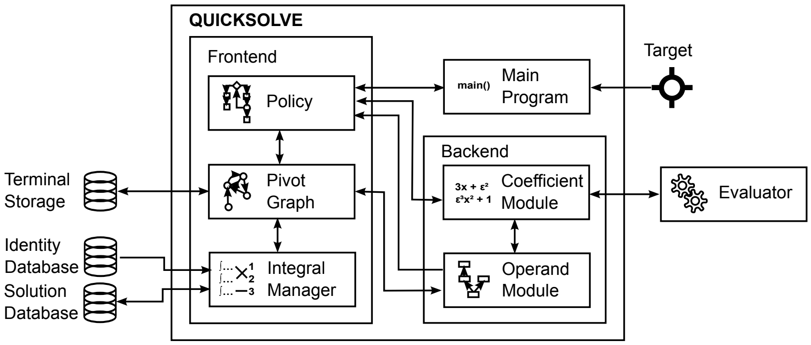

Figure 4.19: An overview over the complete program QS. As an input, one or multiple integrals to be reduced, the targets corresponding to the initial pivots of the reduction, are given (top right) together with an identity database (bottom left). The frontend manages the loading and storing of data about the solution and implements the logic by which the solution is performed. The backend performs the actual, parallel processing and relays the ultimate calculation of coefficients to an external evaluator.
The figure should serve as an informative reference for later sections only. The meaning of the individual components will be explained in the following and in detail in the appendix.

The technical details of the algorithms are relegated to the appendix and should be read to fully understand the mechanism of parallelization. However, the essential idea of the greater algorithm's **policy** can be understood without internals of the system and is explained in the subsequent chapter.

4.3.1 The general idea

The **frontend** implements what is considered the policy of the solution. The policy describes the logic by which is decided which operations, foremost normalize, relay, and collect are performed on the system and in which order.

The policy itself has no explicit notion of parallel execution. It is intrinsically serial and, during many phases, relies on the result of previous operations in order to decide how to proceed. For reference, what has been outlined as the original and thus serial algorithm in the previous section is one, perfectly valid choice for a policy in a fully parallel version of QS.

In fact, the Czakon's algorithm was used as QS' policy during early versions. Only later, as will be explained, a different policy, named CKS for Czakon-Kirchner-Sodhi was developed, which, however, is still free of any notion of parallelism.

The distinction between the frontend and the backend is such that the frontend does not actually perform any of the operations which, instead, are the responsibility of the backend. The frontend thus communicates or "dispatches" operations to the backend.

If, for the sake of making a decision, the frontend requires knowledge of the result of any such operation, it can again query the backend. It then depends on whether the backend can already provide the requested result. If the result is not yet ready, the frontend may, depending on its policy, either wait for the result or proceed differently.

The backend, receiving orders from the frontend does indeed *have* notion of parallel execution. At any point, the backend knows which of the scheduled operations are independent of eachother and can thus be evaluated in parallel. With a given amount of s.c. worker threads, it will evaluate thus many independent operations in parallel, eventually enabling it to provide their results.

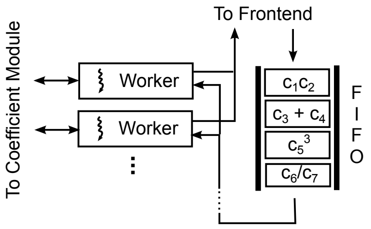

Figure 4.20: A simplified view of the internals of the Operand module in the backend. Operations which the frontend has dispatched for calculation and which are determined to be calculable are queue in a first-in-first-out (FIFO) working storage and consumed by worker threads. The worker threads evaluate the operations in parallel with support from the Coefficient module and return the result when ready.

At this point it is worth noting a shortcoming of the system sketched so far:

That is, the backend does not prioritize the dispatched operations. They are, once independent, evaluated in a first-come-first-served order. Prioritization could, generally speaking, have two positive consequences:

First, if results become available in a different order, this may, depending on the policy, affect the progress of the frontend and in turn result in different operations to be dispatched to the backend. Since the backend doesn't have notion of the frontend's policy, it would be that policy's responsibility to figure out how to prioritize for a positive effect.

As of writing this report, the implemented policy does not intend to make use of any such prioritization however, a more advanced policy may make further considerations which suggest prioritizing specific operations.

Second, given a set of already dispatched operations, if it is predictable that there are times during which there are more independent operations than workers and other times with fewer independent operations, the backend may possibly make internal use of prioritization in order to balance the workload.

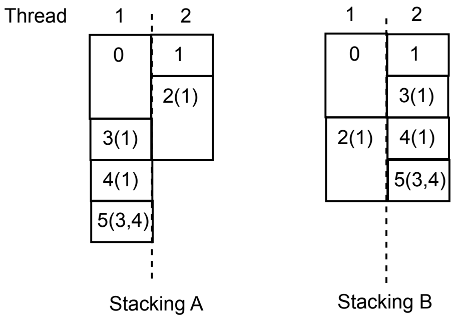

Figure 4.21: The operations with numbers 0 through 5 depicted above depend on the operations which are mentioned in parentheses. If there are two available worker threads, a particular stacking which is permitted by the dependencies leads to quicker evaluation and reduces the chance of idle time. As can be seen from the example, a thread can only fall idle when a dependency chain ends, otherwise any evaluated operations is followed by at least another operation.

Absent a mechanism of prioritization the evaluation in the backend decouples from the policy in the frontend. Even with an intrinsically serial logic in the policy, the amount of time spent in the policy reduces with respect to the fully serial case without a backend. The dispatch of any operation to the backend returns immediately where in the fully serial case without a backend the frontend must delay all further calculations until the operation has been evaluated. In essence, what is improved by this separation is that a more granular logic for when the policy actually awaits a result is implement.

Whereas in the traditional, serial process every calculation included the dispatch, the evaluation, the associated waiting, and the retrieval of the result, the three stages are now separated and only performed when necessary.

4.3.2 The policy

The **pivoting strategy** during the generation of the identities does not succeed at producing the ideal, fully acyclic graph wherein each identity so-considered contains the most complex integral yet, but it still produces a largely acyclic system. However, although the cyclicity of the graph is small, a vast majority of identities will turn out initially unnormalizable. In order to find a solution method, it is therefore advisable to closely follow the order of relays that were performed in the numeric pass with Czakon's algorithm. That way, if the relays from the numeric pass are properly reproduced, pivots will be normalizable when they are relayed.

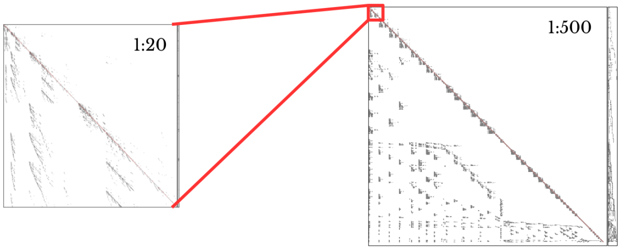

Figure 4.22: The matrix structure corresponding to the system of identities for the reduction of the massless 2-Loop amplitude $gg \to t\bar{t}$ considered in the conclusion 5. On the right, the whole system of about 125,000 linearly independent identities in total. The column order has been chosen such that the pivots are on the diagonal and sorted by order of assignment from top-to-bottom and from left-to-right. The components corresponding to ca. 400 master integrals are shown in the narrow right column.
On the left, a small subsystem is shown at higher resolution together with the occurring master integrals. As expected, the system is extremely sparse and largely acyclic. The structure arising from Laporta's algorithm can be recognized in the partly blocked structure. Red dots on the diagonal signify initially unnormalizable pivots.

This section describes the policy, i.e. the core algorithm which is used to traverse the system to solve the system of identities for one or multiple given integrals. In consistency with the remaining document, technical details about the implementation can be found in a dedicated section in appendix B.2.

Starting from a specific pivot as the target for reduction, the policy recursively reproduces the relays that affected every pivot involved in the solution, like it was explained for Czakon's algorithm. As a consequence of complete relay and contrary to for instance Gauss' algorithm where the relay of edges on any given pivot has to follow the prescribed order, the order in which edges are relayed on a pivot becomes irrelevant as far as reintroduction is concerned. To reproduce the exact set of operations from the numeric pass, however, the precise order of relaying edges on each pivot would have to be known and reproduced during the recursion.

More strongly connected clusters in the system which resulted from similarly complex integrals or IBP identities are the origin of the cyclicity of the graph. As a positive effect of their existence, they also increase the number of loops involving any of their pivots besides its self-loop. Therefore, resolving these loops through relays will eventually modify the self-edge. That explains why deviating from the precise order of relays among the edges of a specific pivot does lead to different operations but only very rarely leads to an unnormalizable pivot after all relays have been performed.

Eventually, it turns out that ordering relays in the described recursive manner is a sufficient requirement for an effective solution. An additional algorithm then detects cases where, despite it being unlikely, pivots turn out unnormalizable and attempts to resolve them.

A policy complying to this requirement and not relying on any other information from the numerical pass besides the set of identities and their associated pivotal columns has to determine the validity and necessity of normalization and relay respectively, on its own. To do this, the CKS algorithm performs yet another numeric solution the operations of which, this time, are directly associated to the actual symbolic solution. In short, the CKS algorithm performs the solution on a completely numeric system. The operations therein executed are then in parallel replayed on the actual, symbolic system.

Just like it would have been during a symbolic solution, there are certain points in the numeric solution where the algorithm waits for the result of previous operations. Most prominently, normalization is only performed once it has been determined that the pivot is normalizable.

Another wait may occur while examining the coefficient weights of the edges from a pivot. When deciding which edge, if any, to relay, that is the missing information from the previous numeric run, the values of the coefficients are required.

At any given pivot, during any step of the solution, there is a set of s.c. suitable edges which are considered for relay. Whether an edge is suitable depends on various conditions, foremost on the order of complexity of its head

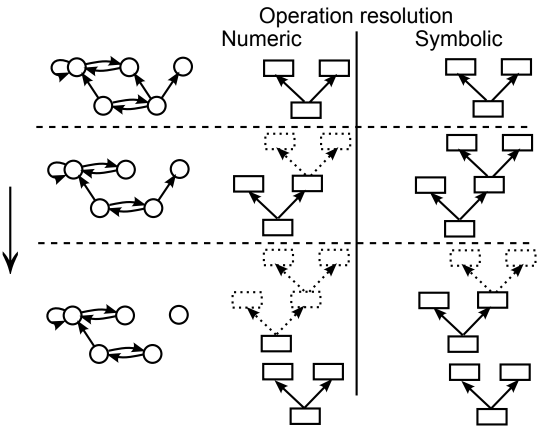

Figure 4.23: Illustration of the dual policy, where the state of the pivot graph is shown on the right and a simplified representation of the operations and their dependencies, explained in detail in appendix A.2, on the right. The dotted operations have been successfully completed, while the operations drawn in solid are new or unevaluated. As shown, the numeric instance of the system which is used to make most decisions about substitutions runs ahead of the symbolic, slower system.

and tail. If it happens, that the suitable edges' weights are not yet evaluated by the backend, the policy waits for any of them to become available. Once an edge is available, the algorithm proceeds in one of several possible ways.

- If the weight is non-vanishing, the algorithm recurses to the referenced pivot and, after returning, eventually relays the edge.
- If, however, the weight vanishes, i.e. vanishes numerically, it is at first taken note of and then the search for further suitable edges continues.

Ultimately, after all edges were considered and none found non-vanishing, vanishing edges are dealt with, depending on the set option `QsPivotGraph::elimination`:

- **Waited elimination**: All numerically vanishing edges are determined symbolically by waiting for the symbolic operand tree to catch up accordingly. If they are confirmed to be zero, they deletion is effected. Else, a warning is emitted to indicate that a numeric cancellation has occurred.
- **Optimistic elimination**: All numerically vanishing edges are assumed to be symbolically zero, deleted numerically and symbolically discarded (through a call to `qs_operand_discard` which is explained in appendix A.2.2). This will eventually verify this assumption when the symbolic value is available. In the meantime, the policy can proceed.
- **No elimination**: Edges are deleted where they are already known symbolically and otherwise kept with a zero coefficient.

Waited elimination is the recommended option during early stages of the solution where cancellations from collect are seen to be likely and coefficients small, such that the symbolic evaluation can catch up sufficiently quick. Optimistic elimination is useful during stages where the complexity of the coefficients is such that numeric cancellations are sufficiently unlikely but there are still as many as to cause a notable amount of cancellations. No elimination can safely be used when one finds that optimistic elimination runs into a case of a numeric cancellation and also when there are few cancellations, but no elimination will severely affect performance if there are a lot of cancellations.

Suitable edges for relay

Originally, the set of suitable edges to be considered for relay is the same as for Czakon's algorithm. It is determined by the assignment-order of the current pivot, i.e. the tail, the order of the pivot at the head of the edge and by whether the pivot at the head of the edge was already relayed previously.

In order to deal with the rare cases that a pivot turns out non-normalizable at the point where it is supposed to be relayed, the solution tracks a value designated “despair” and each pivot holds a counter designated “consideration”. The consideration value counts how often the algorithm reached the associated pivot by recursion from another pivot. If all pivots are properly normalizable, this value is either 0, if the pivot is currently not of the walk through the graph, or 1 if it is.

If normalization fails, the algorithm increases the despair value and extends the set of suitable edges by those whose head pivot has a consideration count smaller than the current despair. It then attempts further relays until a non-vanishing self edge can be restored. This mechanism aims to resolve cases where the resolution of a non-normalizable pivot requires further of such resolutions, eventually leading back to itself.

Chapter 5

Closing remarks

5.1 Status of two-loop $gg \to ZZ$ reduction

Based upon the work done in [18], the two-loop diagrammatic contributions of $gg \to ZZ$ are evaluated to obtain NLO accurate interference contributions to the cross-section. As pointed out, PV reduction is not the tool of choice for this calculation. However, the tensor structure of the involved integrals can be resolved before IBP reduction, by projecting the signal and background amplitude onto the respective other according to

$$\begin{aligned}
\langle |M^{\alpha_1 \cdots}|^2 \rangle \propto \sum_{\lambda_1 \ldots} \left| \epsilon^{\mu_1}(\lambda_1) \ldots \left(S^{\alpha_1 \cdots}_{\mu_1 \ldots} + B^{\alpha_1 \cdots}_{\mu_1 \ldots} \right) \right|^2 = \\
\sum_{\lambda} (\epsilon_1^{\mu}(\lambda))^* \epsilon_1^{\nu}(\lambda) \ldots \left(S^{\alpha_1 \cdots}_{\mu_1 \ldots} + B^{\alpha_1 \cdots}_{\mu_1 \ldots} \right)^* \left(S^{\alpha_1 \cdots}_{\nu_1 \ldots} + B^{\alpha_1 \cdots}_{\nu_1 \ldots} \right) = \\
\cdots + \left(S^{\alpha_1 \cdots}_{\mu_1 \ldots} \right)^* \left(P^{\mu_1 \nu_1} \ldots \right) B^{\alpha_1 \cdots}_{\nu_1 \ldots} + \ldots
\end{aligned} \tag{5.1}$$

where P are the projection operators onto the physical polarization space and * denotes the complex conjugate. Clearly, the method is also applicable to bispinor-space which, however is not present with no external fermion lines. With an explicit representation of the P the indices can be contracted to the effect that only scalar integrals have to be evaluated, much as if the PV reduction mechanism had been applied. After this reduction, 1992 integrals need to be evaluated.

The generation of identities according to Laporta's algorithm, i.e. the initial numeric run, is performed using the part of the original software-suite

[30], including the generation of auxiliary identities from, for example symmetries. 267 master integrals corresponding to the choice of pivots made in the initial run are obtained so that their derivatives with respect to the t mass can be calculated to obtain a system of differential equations. The derivatives contain a total of 819 integrals which also need to be reduced.

It turns out that each free parameters in the integrals requires a tremendous amount resources in computation time and memory consumption compared to if the scale is removed. In order to speed up computation time, the t mass is therefore expressed in terms of the Z mass.

The reduction was first started with the original, unparallelized version of the software [30] long before the development of QS began. When QS finally matured and entered production, the reductions of the differential equations and the amplitudes had almost run in serial for *two* years.

At the time of writing, QS has, in its final version, run comparably little time on the reduction with an estimated net of about one months due to frequent interruptions for code changes and other circumstances. In this time, the reduction of amplitudes has almost finished. The output of the program shows that only two more symbolic (baked) operations have to be performed, where each of the later operations turned out to take of the order of *weeks*. Clearly, towards the end there remains little to be done for the parallel evaluation. The reduction of the system for the differential equations, on the other hand, has not been started with the current policy described herein, yet.

Given these circumstances, the performance of QS could not be conclusively analyzed with the application to the two-loop $gg \to ZZ$ amplitudes, although it can be observed that, when running for a long, consecutive period, the parallelization does indeed kick in as expected.

In the course of development, a smaller testcase for the two-loop $gg \to tt$ amplitudes with massless quarks was used. It should be noted, that this case is considerably different from the actual, practical case in that the evaluations are as simple as to no longer constitute the main work of the program. The overhead needed for traversing the system in parallel becomes significant. Therefore the statistics gathered with an overall, nonetheless excellent overall speedup of about 4 on 4 parallel CPUs, cannot be readily transferred to large, realistic scenarios. Only the actual evaluation and benchmarking of such scenarios will eventually provide answers to this question.

5.2 Development and deploy of QS

The software and the various tools that come with it, e.g. the subsampling renderer used to visualize the system in figure 4.22 and validation tools, were developed in the programming language C and are publicly available as source code. Until official references to the repository which respect the licensing of accompanying software libraries are published, a copy of the source code can be obtained through the author. It should however be noted, that the original software suite [30] which performs the generation of identities and associated databases for use with QS, is not included in QS and outside of the domain of this thesis or its author.

5.3 Outlook

Within the time constraints of this thesis, no results for the two-loop $gg \to ZZ$ amplitude with full t mass dependence could be obtained, but the improved new method will most likely be first in the race to provide these results. The paramount achievement of this thesis is not restricted to the specific process at hand. Indeed, the original software has arguably been the fastest *general* algorithm for IBP reduction of arbitrary many-loop amplitudes and the associated, increasingly huge, symbolic system, *despite* it being serial. The new parallel method can, depending on the actual system, achieve speedups of the order of tens, if not hundreds — though specific figures will have to be found through actual benchmarks — and has thus pushed the boundary of what is possible even further.

More importantly, since we know that the subject of IBP reduction is undergoing constant improvement by the community with new suggestions for how to mitigate the critical, intermediate expression-swell, the algorithm and software are structured such that these improvements can easily be incorporated for algorithms which do solve the symbolic system based on linear reduction. Not all methods for IBP reduction do follow this approach. In fact, various algorithms from abstract algebra have been suggested to tackle IBP reduction from a different perspective; see [66, 64] and [53] for two typical examples.

Currently, none of these algebraic methods are however as universal or reliable as to be able to apply them to every IBP reduction. On the other hand, incremental improvements to the linear reduction to master integrals seems to be a promising approach. Foremost, a specific choice of master integrals, the generation of a specifically tailored set of identities, and appropriate approximations are great candidates to simplify the solution. All of which can readily be implemented in or on top of the existing infrastructure with almost no effort.

In conclusion, we will be awaiting the results from the $gg \to ZZ$ amplitude, particularly the reduction of the system of differential equations which is not running with the newest, highly concurrent version of QS, yet. In the meantime, it will be interesting to apply the method to other problems, try out different policies and benchmark the system thoroughly.

Appendices

Appendix A

The backend

The backend's sole purpose is to deal with abstract operands. Subsets of these operands are iteratively combined by a single operation of the underlying field into a new operand. The new operand shall then, depending on the operation, be referred to as follows:

- Multiplicative `QsOperand` from `QsOperands` $a_1, \dots, a_n$: for $n > 1$: $a_1 \dots a_n$ for $n = 1$: a_1
- Additive `QsOperand` from `QsOperands` $a_1, \dots, a_n$: for $n > 1$: $a_1 + \dots + a_n$ for $n = 1$: a_1
- Multiplicative Inverse `QsOperand` from `QsOperands` $a_1, \dots, a_n$: for $n > 1$: $a_1/(a_2 \dots a_n)$ for $n = 1$: $1/a_1$
- Additive Inverse `QsOperand` from `QsOperands` $a_1, \dots, a_n$: for $n > 1$: $a_1 - a_2 \dots - a_n$ for $n = 1$: $-a_1$

An operand `QsOperand` itself is a handle to a value `QsCoefficient` which may either already be known or not.

Starting out from `QsOperands` of known values, they are combined into new `QsOperands` which are initially not yet known by value. Step by step this builds a tree of `QsOperands`, the Operand-tree which describes both them and their dependencies.

We differentiate between two subtypes of `QsOperands`: `QsIntermediates` and `QsTerminals`. In fact, `QsTerminals` are the only type which have an evaluation associated with them and become known by value. `QsIntermediates` are used to express intermediate steps during the combination of `QsTerminals`

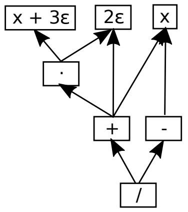

Figure A.1: A simplified example of the operand tree as it will be denoted in the following. The leaves of the tree correspond to `QsOperand`s which are necessarily known by value. Subsequent `QsOperand`s which are formed by combining existing `QsOperand`s under a given operation are denoted by nodes which are here labelled with the corresponding operation.

into a new `QsTerminal` when there is more than one kind of operation involved.

Consider, for example, the case where, starting from four `QsTerminals` a_1 through a_4 which are already known by value, we want to obtain the value in a new `QsTerminal` a_5

$$a_5 = a_1 a_2 + a_3 a_4$$

We could achieve this by first creating two multiplicative `QsTerminal`s

$$b_1 = a_1 a_2$$

$$b_2 = a_3 a_4$$

and then combine those into the result by an additive `QsTerminal`

$$a_5 = b_1 + b_2$$

`QsIntermediate`s replace `QsTerminal`s in order to express that we are not interested in the actual values of b_1 and b_2, by making b_1 and b_2 `QsIntermediate`s. Eventually, there will only be one evaluation, namely that of a_5 and the whole expression

$$a_1 a_2 + a_3 a_4$$

will be fed into the evaluation mechanism at once without obtaining intermediate results explicitly. This provides the evaluation mechanism ab initio with the information that no explicit results need to be provided for the subexpressions and possibly allows for better performance. For the particular evaluation mechanism that is, at the point of writing, in use in QS, it indeed turns out beneficial to feed it the whole expression at once.

The traversal of the Operand-tree is referred to as "upstream" when moving over an edge from a `QsOperand` to one of the `QsOperand`s which are operands of the former and "downstream" otherwise.

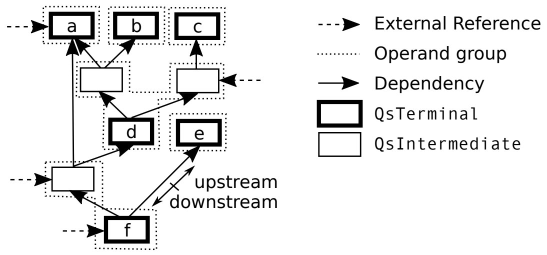

Figure A.2: Complete depiction of an examplary Operand-tree. `QsTerminal`s are labelled with names a through f and informal "operand group" boundaries are indicated by dotted frames. Here and later external references to `QsOperand`s are indicated by dashed arrows. Note that at most one `QsOperand` may depend on any `QsIntermediate`. For simplicity, `QsIntermediate`s will therefore be omitted in the following, which amounts to replacing each operand group by its associated `QsTerminal`.

Walking upstream (downstream) from any `QsOperand`, those `QsTerminal`s which are reachable by only traversing `QsIntermediate`s in between are referred to as the immediate dependencies (dependers).

The values which are or are not yet associated with `QsTerminal`s are of `QsCoefficient` type. When all immediate dependencies of a `QsTerminal` are available by value as a `QsCoefficient`, the respective `QsTerminal` becomes ready for evaluation. Once evaluated, it is then also available by value as a `QsCoefficient` enabling the evaluation of its immediate dependers and so on.

`QsOperand`s are part of the Operand module and are agnostic of the internal structure of `QsCoefficient`, how values are represented and the way in which the evaluation is performed. These details concern the Coefficient module which eventually implements the way how those values are handled. However, while the Operand module does not require any knowledge of how exactly `QsCoefficient`s are implemented and evaluated, the Coefficient module, during evaluation, somehow requires knowledge of the whole expression as it is encoded through `QsIntermediate`s.

In order to achieve this without introducing a recursive dependency of the Coefficient module on the internals of the Operand module, the Coefficient module defines an abstract `QsCompound` type which, similarly to a `QsOperand`, is a family `QsCoefficients` and `QsCompounds` together with an associated operation.

When all immediate dependencies of a `QsTerminal` A, which is not yet known by value, are known by value, the Operand module may evaluate the latter. In order to do so, it passes a pointer to the relevant part of A declared as a `QsCompound` to the Coefficient module, together with a callback to what is called the discoverer function. The Coefficient module can use the discoverer function in order to examine the passed-in `QsCompound`, determine its operation and its contents, which are either `QsCoefficients` or in turn other `QsCompounds`.

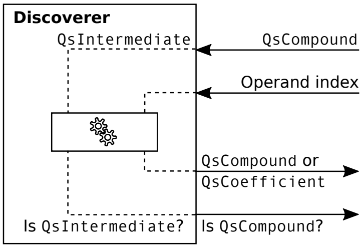

Figure A.3: Schematics of the discoverer mechanism. The discoverer callback, which is provided by the Operand module, is called by the Coefficient module in order to recursively parse the content of a `QsTerminal`'s "operand group" (see section A.2.1) which will have to be evaluated. The Coefficient module assumes a nested structure of `QsCompounds`, corresponding to `Expressions` — and in turn to `QsIntermediates` for all but the top level — from the Operand module's perspective.

That way, the Coefficient module may explore and walk through the involved `QsOperands` with only a notion of a generic compound type in addition to its own `QsCoefficient` type.

A.1 Coefficient module

What is considered a value of a `QsTerminal` from the perspective of the Operand module is a `QsCoefficient` and represented by a text from the perspective of the Coefficient module. It is evaluated using the FERMAT CAS as evaluator. If another evaluator were used, a `QsCoefficient` may internally be represented by another type of data. Eventually, the specific representation of the values does not matter to the Coefficient module as long as it knows how to combine them under operations, pass them to the evaluator and retrieve their result. In that sense, the Coefficient module acts as a translation layer between the Operand module and the particular choice of evaluator. The representation's contents are never actually interpreted by any part of QS.

A.1.1 Interface

The Coefficient module provides, besides the `QsCoefficient` itself, handles to instances of the evaluator, `QsEvaluator`s. One `QsEvaluator` represents a thread-safe handle to an evaluator and is therefore what each worker-thread creates one instance of to allow for parallelization.

Depending on the particular implementation and choice of evaluator, various options may be set or may be required for each evaluator. For convenience, these are collected into `QsEvaluatorOption`s objects which are then passed during creation of the evaluator.

As described, the Coefficient module also declares the abstract `QsCompound` type and `QsCompoundDiscoverer`, the discoverer function.

A.1.2 Implementation

What follows is the actual description of the Coefficient module as it was implemented using FERMAT as evaluator. The FERMAT evaluator provides options to register symbols, which is mandatory for all symbolics occurring in the involved expressions. It also provides an option to choose the name of the executable which will be used and an option to re-initialize that binary every n evaluations for various purposes.

Initialization

The initialization which occurs during creation of a `QsEvaluator` and during re-initialization, if that option was requested, is handled by `init_fermat`. The function forks out an instance of the binary, opening pipes to and from it for communication. Finally, the symbols which were registered in the associated `QsEvaluatorOption`s are also registered with FERMAT. Depending on whether they were registered with a substitution value, this choice is also reflected when they are registered in FERMAT by using either the `&J` directive or the `:=` assignment.

Evaluation

Each evaluator has an associated `QsCompoundDiscoverer` which is passed during creation. For evaluation, the evaluator is passed a `QsCompound` on which the discoverer is applied. The discoverer can then be called for the supposed n-th element of the `QsCompound`, returning either NULL if there are not that many elements or a pointer to what is either a `QsCompound` or a `QsCoefficient.` An additional boolean returned by the discoverer then indicates whether the pointer is a `QsCompound` or a `QsCoefficient`.

As the evaluator discovers the `QsCompound`, it pipes the respective values, operations and parenthesis as strings into FERMAT.

The submission is terminated by submitting a token which will appear in the output and allow identification of the end of the result when it is returned.

Retrieval

For retrieval, the evaluator awaits the appearance of the token in the output, cleans the output of various artifacts such as linebreaks and stores the resulting string back into a new `QsCoefficient.` If the input pipe breaks before the token is received, failure is assumed. The evaluator then first re-initializes before returning NULL to indicate to the caller that the evaluation failed and must be repeated.

A.2 Operand module

The Operand module provides, apart from aforementioned limitation that no priorities may be assigned, a generic framework for calculations and finds, for a given set of calculations, all independent calculations to perform them in parallel. As described, the Operand module does not care about the particular way the `QsTerminals`' values — the `QsCoefficients` — are implemented. It only keeps track of when they become available and provides a `QsCompoundDiscoverer` callback for the Coefficient module to pass the necessary information about how the operands are combined by operations.

A.2.1 Interface

The central object provided by the Operand module are the two `QsOperand`-derived types `QsTerminal` and `QsIntermediate.` Given any set of those, they can be combined into either a new `QsIntermediate` by `qs_operand_link` or into a new `QsTerminal` by `qs_operand_bake.`

Whereas linking two operands into a new `QsIntermediate` serves only as a means to build expressions which involve more than one type of operation, baking them into a new `QsTerminal` can be considered the one true operation that is performed. In essence, whatever linkage between operands has been established through `QsIntermediate`s, this information is relevant to exactly one `QsTerminal` by which the `QsIntermediate`s are used. One may thus establish an informal notion of an "operand group" as depicted in figure A.2 consisting of one `QsTerminal` and all `QsIntermediate`s involved therein.

This notion is also reflected by the fact that the system does not allow for any `QsIntermediate` to be used in more than one `QsOperand.` Doing so would imply that a particular expression which is not explicitly evaluated to a value is found in two places and therefore suggest a redundant calculation. Ultimately, this does not prevent one from linking and baking `QsOperand`s in a redundant manner in the sense that their results could be obtained with fewer resources by more intelligently combining them. However, it prevents the most obvious kind of redundancy where a particular subexpression would occur in multiple evaluations.

When all immediate dependencies of a `QsTerminal` are known by value, the `QsTerminal` will be ready for evaluation. Parallel evaluation of `QsTerminal`s is handled by `QsAEF` objects. These Asynchronous Evaluation Frameworks each maintain their set of worker threads and associated `QsEvaluators`. Each `QsTerminal`, during creation, is assigned to one particular `QsAEF` which will be responsible for its evaluation.

Once a `QsTerminal` becomes ready for evaluation, it is transferred to its assigned `QsAEF`'s queue from where one of the `QsAEF`'s worker threads can pick it up to begin evaluation.

As the user of the Operand module, in our particular case the frontend of QS, makes use of `QsOperand`s, their values will sometimes be required. Both, linking and baking `QsOperand`s into `QsIntermediate`s and `QsTerminal`s respectively, return immediately and for `QsTerminal`s before the operations have actually be evaluated.

A `QsTerminalGroup` is a container into which a set of `QsTerminal`s can be put and which provides an interface to obtain their values and possibly wait for their successful evaluation. If a set of `QsTerminal` is put into a `QsTerminalGroup`, waiting on the `QsTerminalGroup` with `qs_terminal_group_wait` blocks the caller until at least one `QsTerminal` is known by value. A call to `qs_terminal_group_pop`, on the other hand, returns the first `QsTerminal` which is already known by value and removes it from the `QsTerminalGroup` or returns NULL, if no `QsTerminal` was found ready.

The Operand module imposes no limit on how quickly new QsOperands may be created by either linking or baking others. While the evaluation of QsTerminals is still pending and new QsOperands are being created, the Operand-tree may in principle grow indefinitely. Depending on its particular structure, in particular how many independent QsTerminals it contains at any time, concurrent knowledge of the values of many QsTerminals may be required. In practice, where large QsCoefficients are encountered, this requires a mechanism for storing those QsTerminals to disk.

Both, the tracking of memory and the storing to as well as the retrieval from disk are handled by callbacks which are collected in a QsTerminalMgr object. Besides the mandatory QsAEF associated with a QsTerminal during creation, a QsTerminalMgr object may then be associated with the QsTerminal. The QsTerminalMgr specifies callbacks to be invoked...

- when the evaluation of the QsTerminal finishes and memory for the value is required, QsTerminalMemoryCallback.
- when the QsTerminal is no longer needed, QsTerminalDiscarder.
- for storing the QsTerminal, QsTerminalSaver.
- for loading the QsTerminal, QsTerminalLoader.

Besides those callbacks every QsTerminalMgr refers to a QsTerminalQueue object which is a container in which the currently loaded QsTerminals are kept track of and from which QsTerminals are taken for storage if memory is required. Lastly, since storage and loading of QsTerminals requires some means of identifying them, the QsTerminalMgr also specifies the size of the data structure which serves as an identification and which is passed as a QsTerminalMeta pointer during creation of each QsTerminal.

For the purpose of memory management, these identifications don't need to have any meaning and only be unique.

When using the Operand module, the user creates a set of QsTerminalQueues and QsTerminalMgr. Baked QsTerminals which are not associated with a QsTerminalMgr will necessarily always be kept in memory and so are QsTerminals without an identification, i.e. whose QsTerminalMeta pointer is NULL. Otherwise, when their evaluation finishes and memory for their QsCoefficient value is allocated, the user is informed of the allocation through the callback.

In response, the user may decided to cause the value of one or more evaluated QsTerminals from a given QsTerminalQueue to be written back to disk by calling qs_terminal_queue_pop on that queue, which will eventually lead to one QsTerminal to be passed to its associated callback for storing.

A.2.2 Implementation

The central object in the implementation of the Operand module is the Operand-tree. Because there is concurrent access to the Operand-tree from multiple threads and operations at one node of the tree cause information to propagate along the tree both up- and downstream, the correct order of locks and atomic operations often becomes quite convoluted. The main lock for the very purpose of the Operand-module is the `QsTerminal` lock which locks the type change from a `QsTerminal` which is not yet known by value to a `QsTerminal` which is known by value. The latter shall informally be referred to as a "result" and the associated boolean flag is `QsTerminal::is_result`. Almost all other locks are related to memory management.

First, the main semantic structure for creation of the Operand-tree by `QsOperand`s, tracking dependencies between them and causing their evaluation will be described. Then the auxiliary structures which allow for efficient memory management will be explained.

Operand dependency tracking

The Operand-tree builds without any central instance but instead only from the `QsOperand`s that are being created by the user; it is thus only a concept. Each `QsOperand` constitutes a node in this tree and the relation to the `QsOperand`s from which it is built, meaning either linked or baked, constitutes a directed edge. Because there is no central intelligence to govern the Operand-tree, the involved `QsOperand`s use reference counting to determine whether they are still required or can be destroyed, when anything in the tree changes.

References to a `QsOperand` are semantically properly minimal. That means the only references are external ones established by the user of the Operand module and internal ones from downstream neighbours in the graph, which actually require the particular `QsOperand` for their evaluation. No other references shall be acquired or held internally by the Operand module as depicted in figure A.2, where each arrow effectively amounts to a reference.

Operand creation Both, `QsTerminal`s and `QsIntermediate`s share the common notion of an expression `Expression` which is a list of those `QsOperand`s which the respective `QsOperand` was built.

Since `QsIntermediate`s' only purpose is to encode combinations of operations, `QsIntermediate`s are essentially fully defined by their `Expression` and the common `QsOperand` structure with reference counting. Although the set of immediate dependencies of a `QsIntermediate` can easily be discovered by traversing upstream, they are cached in `cache_tails` for quicker access. When a `QsIntermediate` is linked from another `QsIntermediate`, that cache is removed on the latter. However, when a `QsTerminal` is baked from a `QsIntermediate`, the cache is kept for reference from the `QsTerminal`, because the `QsTerminal` doesn't have such a cache itself.

The creation of a `QsIntermediate` trivially consists of building `cache_tails` and composing the `Expression`.

`QsTerminals`, besides storing their individual properties that were specified during creation, i.e. a `QsAEF` and a `QsTerminalMgr` are a Union of a `BakedExpression` and `TerminalData`. When not yet evaluated, the `BakedExpression` is used, otherwise the `TerminalData` holds the value and other information that is only required when the `QsTerminal` is a result. Further, every `QsTerminal` keeps a list of its immediate dependers.

A `BakedExpression` is an `Expression` with additional data which are only required for `Expressions` which are eventually baked in a `QsTerminal.` That is, a reference to a `QsTerminalGroup` container which the parent `QsTerminal` may possibly be part of and an atomic counter indicating how many immediate dependencies of the `QsTerminal` are not yet available by value, the **dependency count** DC.

The creation of `QsTerminal` proceeds by iteratively registering itself as an immediate depender in the immediate dependencies and incrementing the dependency count. Whenever a `QsTerminal` becomes known by value, the dependency count on all its immediate dependers is decreased by one. If the dependency count reaches zero on any `QsTerminal`, it is added to the queue from which worker threads may pick it up and evaluated it. Therefore, in order to prevent the dependency count from prematurely reaching zero during creation, while other dependencies are yet to be registered, it is initially set to one and finally decreased again, when all dependencies have been registered.

Operand evaluation When the dependency count of a `QsTerminal` reaches zero, it is added to the end of the queue of the `QsAEF` that it was associated with. Calling `qs_aef_spawn` on a `QsAEF` creates a new worker thread of the worker function which will monitor the queue and pick up `QsTerminals` from there.

When evaluating a `QsTerminal`, the worker thread first assures that the values of all `QsTerminals` which will eventually be involved in the evaluation are loaded in memory by issuing a recursive call to `manage_tails`. `manage_tails` causes all values to be loaded and increases a reference count on them to prevent them from being written back to disk while they are needed. The `QsTerminal's BakedExpression` is passed as `QsCompound` to the evaluator for discovery and evaluation.

When the result is available, the `QsTerminal::is_result` flag is changed and the type change from a `QsTerminal` not known by value to a `QsTerminal` known by value is completed. If the `QsTerminal` is part of a `QsTerminalGroup`, the `QsTerminalGroup` will be notified of the change. Eventually, the dependency count on the `QsTerminal`'s dependers is decreased.

Since the Operand module properly minimizes references, all references to the `QsTerminal` may be dropped as soon as the change of `QsTerminal::is_result` is completed. After this point, the `QsTerminal` must not be referred to since it may already have been destroyed. This becomes relevant later on when memory management will be explained but already during decrement of the dependency count, which still requires the information which are the dependers of the `QsTerminal.`

Therefore, the `BakedExpression`, which contains this information, is not destroyed with the `QsTerminal` but only later by the worker thread.

Memory management

The tracking of available memory is done not by the Operand module but instead in the callbacks. When there, because memory consumption exceeds a limit, or elsewhere, it is decided that memory usage must be reduced, the user may call `qs_terminal_queue_pop` on one of the `QsTerminalQueue`s which contains `QsTerminal`s with currently loaded values.

This will, if the `QsTerminalQueue` contains any of such `QsTerminal`s which is not currently required to be in memory, store the value of one `QsTerminal` by calling the callback for storage. In order to determine which `QsTerminal` that will be, a metric designated the dependency-value **D-Value** is assigned to every `QsTerminal` as shown in figure A.4.

To indicate that a `QsTerminal` is required, `qs_terminal_acquire` is called on a `QsTerminal.` This prevents the `QsTerminal` from being removed from memory until a corresponding call to `qs_terminal_release` is performed and loads the `QsCoefficient` into memory, if it is not yet loaded.

The D-Value of a `QsTerminal` describes how soon the `QsTerminal` will be required in memory. A D-Value of zero means that there currently is no known `QsOperand` whose evaluation would require the `QsTerminal`'s value. For D-Values other than zero, the greater the D-Value the later the `QsTerminal`'s value will likely be required.

More specifically, the D-Value of a `QsTerminal` is defined as the minimal accumulated dependency count ADC of all the `QsTerminal`'s immediate dependers plus one. The accumulated dependency count of a `QsTerminal` is in turn recursively defined as the sum of all the `QsTerminal`'s immediate dependencies' ADCs plus one. The ADC of a `QsTerminal` which is known by value is defined as zero.

While the dependency count DC specifies how many immediate dependencies are still unevaluated, the ADC therefore specifies how many `QsTerminal`s still have to be evaluated before the current `QsTerminal` can be evaluated. The D-Value of a specific `QsTerminal`, being the minimum of all immediate dependencies' ADCs therefore specifies how may `QsTerminal`s have to be evaluated before reaching an evaluation for which that specific `QsTerminal`'s value is required.

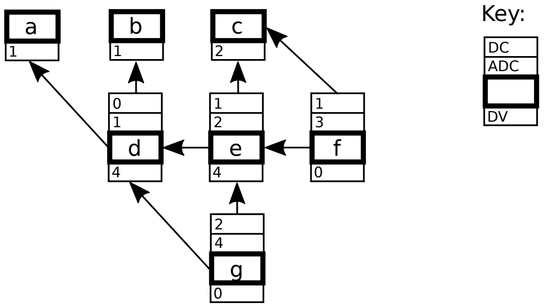

Figure A.4: The DC, ADC, and D-Value of various `QsTerminals` during one consistent state of the system. Consistency herein implies that all information has been fully propagated throughout the whole graph. In reality, the modification of any of these values following the successful evaluation of a `QsTerminal` upstream will propagate through the graph *peu-à-peu* to the effect that the numbers are momentarily inconsistent. Note that `QsTerminals` *a* through *c* which are results do not carry an DC or ADC, their ADC is implicitly assumed to be zero. `QsTerminals` *f* and *g*, on the other hand, are currently not depended upon by any other operands so that their D-Value is zero.

When deciding which `QsTerminal` to store back to disk in order to free memory, a `QsTerminal` with the highest D-Value is taken or the first `QsTerminal` that is found to even have D-Value zero.

ADC calculation The ADC of new `QsTerminals` in a large Operand-tree may grow very quickly and easily overflow multiple words in even moderate cases. However, the higher the ADC of a `QsTerminal`, the less likely it is relevant for any calculation. In fact, for the structure of the Operand-tree that is generated by solving a linear system of the type relevant to this work and a reasonable memory limit, there is almost always a `QsTerminal` with

D-Value zero in the `QsTerminalQueue` which can immediately be chosen to be stored. If, in an extreme case, no `QsTerminal` of D-Value zero could be found, the `QsTerminal` with the numerically highest D-Value will be taken. If there is a `QsTerminal` all of whose immediate dependers' ADC overflow, such a `QsTerminal` will be chosen.

The mistake in doing so is thus to not differentiate between the theoretically precise D-Values among set of `QsTerminal`s which are all very unlikely to be required soon. It therefore has little to no impact on performance.

That being said, while overflown ADCs are rarely relevantly involved in memory management decisions, they still have to be correctly kept track of once they would no longer overflow or even reach zero. In conclusion, the more likely a `QsTerminal's` value will be required soon, the more important a correct D-Value. In consequence, the higher an ADC, the less important it is for it to be precise.

This allows for two approximations. The first being a rather complete lack of precision at very high ADCs, that is, beyond the overflow bound there is only one state rather than distinct values: Overflow. This is a necessity if the ADC should be kept in a fixed precision datatype of reasonable size. The second approximation is a gradual increment in precision as the ADC decreases. Preferably, the ADC becomes precise below a lower bound and is precise when it reaches zero. This approximation is not a necessity, but allows for much more lenient handling in favor of performance.

Without the second approximation, any finished evaluation bringing a `QsTerminal's` ADC from one to zero, would propagate to every single `QsTerminal` downstream. For large Operand-trees, this may be a lot of work. Worse, it would require immense mutual exclusion locks on downstream `QsTerminal`s to prevent them from being destroyed while the propagation takes place.

With the second approximation in place, decrements of the ADC may or may not be propagated during every step of the recursion at will, as long as the respective `QsTerminal` at which the recursion takes place correctly remembers whether it did or did not propagate. To remember it, a `QsTerminal` not only holds its own ADC, but also a list of contributions of its own ADC to the ADCs of its immediate dependers. When a `QsTerminal` propagates a decrement further downstream, the contribution is decreased by the same value that is propagated. Else, the contribution is untouched.

As a simple rule, the decision whether to propagate an ADC decrement downstream is based on how much the ADC that contributes in an immediate depender deviates from the current ADC. If the deviation exceeds a certain limit, the decrement is propagated, else it is not. That limit, in turn, increases

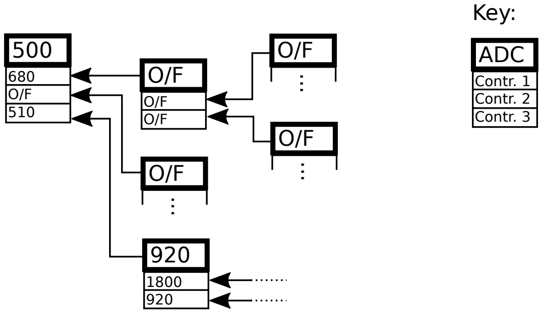

Figure A.5: Exerpt from set of ADCs corresponding to `QsTerminals` with "O/F" indicating an overflowing ADC. Shown in the main box is the ADC of the respective operand. Below it the value at which it contributes to its respective dependers. The system shown here therefore depicts an **inconsistent** state where reduction of the first ADC to 500 was not yet propagated downstream. Instead, for example the top branch of downstream is still in a state corresponding to the ADC being 680 and later iterations of the propagation will eventually remedy this inconsistency.

with the depth of the recursion and is arbitrarily set to $2^{2(\text{Recursion Depth})}$

In other words, a single decrement originating from the evaluation of a specific `QsTerminal` is damped downstream and propagates less and less likely into more distant nodes. But when many distant decrements together would sum up to a certain limit, a propagation eventually takes place.

Initially, the ADC of a `QsTerminal` is composed during its creation by collecting the ADCs of its immediate dependencies and making note of the respective contributions. While the `QsTerminal` is already registered as an immediate depender in any of its immediate dependencies, the ADC still increases as other dependencies are registered. However, any previously registered `QsTerminal` or any `QsTerminal` upstream thereof, may finish evaluation while creation is still going on. It is in that situation that a decrement of an ADC propagates downstream to an ADC which is currently still being incremented because dependencies are added. In all other cases, that is, once the `QsTerminal` is fully created, the ADC is strictly decreasing.

Conversely, an ADC may begin to overflow exactly once, during creation, and stop overflowing exactly once, during decrement. Decrement from overflowing state into non-overflowing state are not effected as long as construction is going on for efficiency and because a strictly decreasing ADC during re-calculation is required for precedence order.

Whereas overflow is readily detected and applied during creation at each

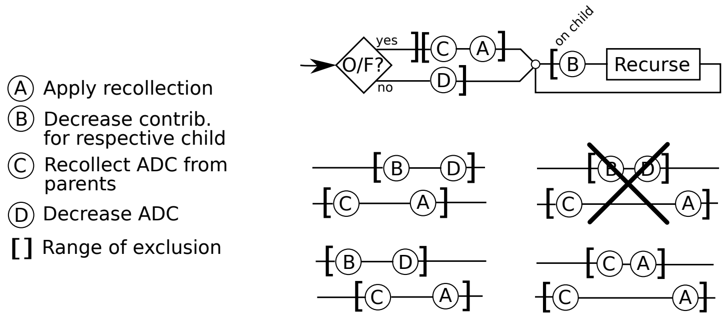

Figure A.6: The recursive algorithm for propagating ADC decrements through the Operand-tree. The shown exclusion lock corresponds to the `QsTerminal` which is currently manipulated by the recursion or one of its children where indicated.
Four examples of how a pair of threads, progressing from left to right, may end up ordering the operations. From left to right, top to bottom, they demonstrate:

- The upper thread may modify the contribution *after* was it collected by the lower thread. In that case, the decrement associated with the modiciation is applied after the collected value was written out, resulting in a consistent result. This situation does not need to be prevented.
- The case were the collecting thread attempts to write out a collected value which is no longer correct due to the decrement which occured in the meantime is ruled out between those two threads, because any successful decrement implies that previously a recollection has occurred.
- The case where the recollected value is applied and would thus overwrite a properly decremented one is prevented by locking *at least* around C and BD.
- The lower thread attempts to apply a recollected value although the upper thread has already written out a more recent recollected value. This situation is prevented by extending the lock not only around C but instead around CA.

increment, detecting the return from overflow to an actual value is not immediately possible by just considering each decrement. Absent any information beyond the fact that the ADC did overflow, it is necessary to reconsider all contributions in order to determine whether it is still overflowing after a specific decrement. Every decrement propagated to an ADC which is overflowing thus triggers an attempt to re-calculate the latter from scratch by recollecting the contributions from all its immediate dependencies.

This recollection is also the reason why the ADC can not replace the DC as an indicator for whether all dependencies of a `QsTerminal` have been fulfilled. If re-calculation is initiated in a recursion originating from the final decrement of an ADC from one to zero in an evaluated `QsTerminal`, re-calculation needs to recollect contributions from the finished `QsTerminal.` It is not possible to assume that, because the `QsTerminal` is evaluated, the contribution during recollection sum to zero. Because if there are multiple paths between the evaluated `QsTerminal` and the re-calculated one, the decrement may possibly have only propagated on some of them.

Therefore, the evaluated `QsTerminal` still needs to provide all information about its contributions so that they can be recollected. Since only `QsTerminals` which are not yet flagged by `QsTerminal::is_result` can provide this information, the evaluated `QsTerminal` can not yet change type.

On the other hand, adding a `QsTerminal` to the queue for worker threads requires that all its immediate dependencies are indeed already flagged by `QsTerminal::is_result`. Therefore, when the ADC is propagated and reaches one on the depender, we may not act as if the DC had reached zero because the `QsTerminal` from which the decrement originated is not yet a result.

Since the DC and the ADC are tracked separately and only `QsTerminals` which are not yet known by value have an ADC, decrement of the DC may only be performed, both during creation of the `QsTerminal` and after evaluation in the worker thread, after the decrement of the ADC has been performed. Otherwise a different thread may change type of a `QsTerminal` downstream before the recursion of the ADC decrement has reached it.

D-Value calculation Whenever the ADC of a `QsTerminal` decreases above zero, the D-Value of all the `QsTerminal`'s immediate dependencies is updated. Since the D-Value only ever decreases due to decreasing ADCs, the update consists merely of comparing the new ADC to the D-Value and updating the latter if the ADC is found to be smaller.

Contrarily, the increment of an ADC or the respective `QsTerminal`'s unregistration as an immediate depender may cause the D-Value to increase and re-calculation is necessary. Increments are ruled out; any ADC first contributes after the respective `QsTerminal` was fully created whence the ADC is strictly decreasing.

To allow for re-calculation of the D-Value, every `QsTerminal` keeps a list of immediate dependers. Re-calculation than recollects their ADCs and finds the minimum of them.

Immediately after `QsTerminal` is evaluated and its ADC is decremented to zero, the `QsTerminal` is removed from the list of immediate dependers. In other words it is un-registered as an immediate depender and its ADC, by definition, no longer contributes to the D-Value of its dependency. Since this may happen concurrently for multiple dependers of a specific `QsTerminal` together with regular updates of that `QsTerminal's` D-Value, two mechanisms are required.

First, redundant or obsolete re-calculations should be prevented or aborted. Second, consistency between the recollection of ADCs and regular updates should be achieved such that it is established whether an update did or did not obsolete a recollection.

Contrary to ADC recollection, where consistency is achieved by using the fact that the ADC is strictly decreasing after recollection and thus give general precedence to lower values, D-Value recollection has no such constraint and must be handled differently.

Redundancy or obsolescence of recollection occurs if a first thread removes a depender from a `QsTerminal`, proceeds with re-calculation and, while it recollects the ADCs, a second thread removes a depender and starts re-calculation. The first thread's re-calculation is then either obsolete or redundant, depending on whether the it took notion of the second thread's removal. That is, whether it already iterated over the second thread's `QsTerminal` before it was removed.

To prevent such situations, each D-Value has an associated field `DValue::current_removal` which indicates the `QsTerminal` which was last removed from the list of dependers. If, during recollection, a thread finds that the indicated removal no longer equals the `QsTerminal` from which it was initiated, it aborts, knowing that it was obsoleted by another thread. Note that in this case pointer comparison is valid because the `QsTerminal` whose pointer is compared is assumed not yet to be destroyed and its memory reused.

In order to deal with regular updates occurring during the recollection without initiating yet another recollection, an auxiliary value, designated the D-Value on hold, is introduced. When recollection begins, the D-Value on hold is set to zero and all regular updates occurring during recollection update the D-Value on hold rather than the regular D-Value. When a thread successfully

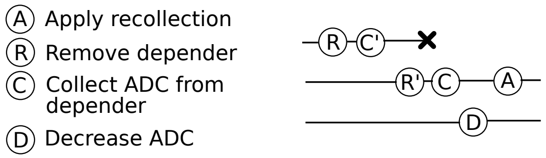

Figure A.7: An example of how the thread at the top runs a recollection which becomes obsolete due to the removal of depender d' by the second thread, which was already considered at C'. The first thread aborts recollection because it finds that a different depender d' is listed as `current_removal`. Also illustrated is a third thread which decreases the ADC of a depender after it has been collected but before the collection was applied. The decrement is then effected by the recollecting thread during application A.

finishes recollection, the recollected D-Value is compared against the D-Value on hold and the minimum of both is applied to become the new D-Value.

Eventually, there is no synchronization constraint on the D-Value. Modification of any ADC need not immediately be reflected by the related D-Values. Instead, if re-calculations occurs, the D-Value may only reflect the changed ADCs when another thread finishes re-calculation. The lack of synchronization, however, is fully intended to not delay any other operations, since even approximate or delayed D-Values are suitable for determining which `QsTerminal`s are to be written to disk for memory management.

Operand discard

The Operand module and therefore backend here described assumes that all `QsOperand`s once created do bear meaning to the user. Every calculation thus dispatched is eventually calculated. This assumption leaves room for improvement and is closely related to the lack of prioritization. For example, a simple yet realistic situation may be that two `QsTerminal`s, both of which result from a large upstream tree with large ADCs are combined in a multiplicative Operand. If the user has indicated no further use of the first of them by dropping their last reference and the second evaluates to zero while the first still has a large ADC, the first `QsTerminal`'s value becomes irrelevant and so may various `QsOperand`s upstream to it, as illustrated in figure A.8.

In more general terms, if the last reference on a `QsTerminal` which is not yet known by value is dropped, its evaluation becomes superfluous. And so may become evaluations of other `QsTerminal`s upstream, an internal reference to which may be dropped in consequence to the first `QsTerminal` being destroyed.

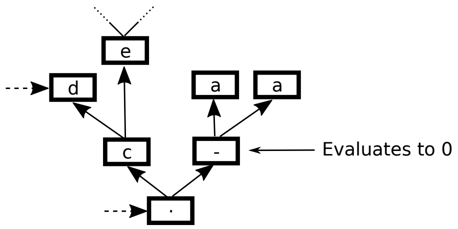

Figure A.8: Multiplication with an operand which turns out to be zero, here from the right branch, will render everything downstream from the multiplication which is not externally referenced superfluous. In the given example, *e* and everything downstream of it which is not externally referenced could be discarded.

Together with a method of prioritization, if such cases occur with a relevant frequency, handling them in a sophisticated manner can help to find and avoid unnecessary calculations. For example, if the `QsTerminal` which evaluated to zero were prioritized, discarding the other `QsTerminal` could occur as early as possible, saving the superfluous calculations.

The current Operand module does make the assumption that whoever dispatched the operations already knows that all of them will be required. Originally, it therefore leads to an error to drop the last reference to a `QsOperand` which is either a `QsIntermediate` which was not used in any other `QsOperand`, or to drop the last reference to a `QsTerminal` which is not yet known by value.

As has been described in the main text, the implemented policy in the frontend does require to be able to discard operands. Specifically if they are known to be irrelevant. For that situation, `qs_operand_discard` allows to prematurely drop the external reference to a `QsOperand` without failure. It does, however, not prevent the evaluation of the `QsOperand`, which will still be performed. What it does is lift the burden off the user to wait for its evaluation and only then drop the reference. Instead, the discard is made note of in the respective `QsTerminal` and the last reference is automatically dropped after evaluation. For convenience, a flag exists to indicate whether it should be asserted that the `QsTerminal` evaluated to zero after it becomes known by value.

A.3 Locks

Due to the heavily concurrent nature of the Operand module, many operations and variables need to be atomic or protected by locks and other synchronization methods. The more complicated locking logic concerns the ADC, D-Value and memory management. In other situations, the locking is fairly straight forward and traditional, such as for the worker threads and their queue. This section shall give a brief overview over the involved locking primitives and their purpose. Especially the convoluted locks for ADC calculation however, can not be sensibly explained in their entire interaction and should be understood on a per-problem basis, by inspecting the relevant sections of the code.

A.3.1 Overview

The Operand module uses POSIX Threads and their associated primitives, i.e. spinlock, mutex and condition variable. Spinlocks are typically lightweight synchronization primitives which can be implemented by a single atomic integer type and which can be locked and unlocked in virtually no time. They are, particularly due to memory contraints when part of a `QsOperand` of which there are to be expected of the order 1e8 to 1e10 many, the preferred locking primitive.

Mutexes and read-write-locks are used when spinlocks are either formally unsuitable because a condition variable requires a mutex by definition, or when locks are either expected to last long or such that finer distinction between reading and writing them leads to fewer blocks. Both, contrary to spinlocks, are typically implemented such that when a lock can not be acquired, the thread will wait in a manner which allows its resources to be used for other operations, rather than wait in a loop ("spin") for the lock to become available. This appears to prove useful for example for the mutual exclusion imposed on access to the database which is used to store and load the `QsTerminals`' values. Reads and writes may take a noticeable amount of time while the lock itsself occurs relatively seldomly compared to other locks.

In general, the existence and purpose of the used locks does not strictly follow one-lock-one-purpose. Because we are interested in minimizing memory usage, some locks, like for example `QsTerminal::lock` are used for multiple, closely related purposes when there is no interference between them. When this is the case, it will be pointed out in the description below.

A.3.2 List of locks

`QsTerminal::lock` (RW-Lock) Protects the type change between being evaluated and known by value and being not yet known by value. It is fairly often read and held for ADC and D-Value calculation because those involved operations depend on whether the `QsTerminal` is already evaluated or not.

The lock is also used to indicate creation of a `QsTerminal` when it is write-locked in order to prevent the ADC to be returned from overflow while it may still increase, as described above. For consistent DCs and ADCs it is necessary that a decrement of either of them is only applied to those immediate dependers which were affected by the associated increment. The lock therefore also wraps the registration of immediate dependers in a `QsTerminal`.

`QsTerminalGroup::lock` **(Mutex)** Associated with the neighbouring condition variable and is logically redundant since the condition is on the `QsTerminalGroup`'s reference count by its members, which is atomic.

`ADCData::adc_contribution_lock` **(Spinlock)** Protects access to any of the contributions of a `QsTerminal`'s ADC and to the ADC itself. It serves to guarantee consistency of the contributions even when re-calculations are performed, as described above.

`TerminalData::lock` **(Spinlock)** Makes the transfer of a `QsCoefficient` from memory, when it is stored in `TerminalData::coefficient`, to disk, i.e. a database, together with adjustment of the reference count atomic. The reference count indicates whether the `QsTerminal`'s value may or may not be removed from memory and is controlled through `qs_terminal_acquire` and `qs_terminal_release`.

`DValue::lock` **(Spinlock)** General purpose lock for changes to the D-Value. In particular, protects access to the `DValue::current_removal` and makes accesses to it and the D-Value on hold atomic.

`QsTerminal::dependers_lock` **(Spinlock)** Locks changes to the list of dependers `QsTerminal::dependers`. This lock is acquired when dependers are registered during their creation and when they are removed. Contrary to `QsTerminal::lock` which also must be held during registration, this lock is the only one required to be held during removal.

`QsTerminalQueue::lock` **(Mutex)** Locks against removal and addition of `QsTerminals` to the `QsTerminalQueue`. It is also held when iterating over the `QsTerminalQueue` to find a candidate for `qs_terminal_queue_pop`. It may be possible to decrease the amount of locking during these operations, possibly making the `QsTerminalQueue` itself lock-free and only lock the `QsTerminals` which are in the container. In practical application, where `QsTerminals` with

D-Value zero are quickly found at the beginning of the queue, the required locks are typically short.

`QsAEF::n_terminals_lock` **(Mutex)** Associated with the condition variable `QsAEF::n_terminals_change` with the condition of `QsAEF::n_terminals` to track the number of `QsTerminal`s which are ready for evaluation but not yet evaluated. During creation the `QsAEF`, a limit for `QsAEF::limit_terminals` can be specified and, when reached, will block the calling thread until enough `QsTerminal`s in that `QsAEF` have been evaluated.

`QsAEF::operation_lock` **(Mutex)** Associated with the condition variable `QsAEF::operation_change` governs the addition and removal of `QsTerminal`s which are ready for evaluation to and from the `QsAEF`'s queue.

A.3.3 Memory management locking

`QsCoefficient` values of `QsTerminal`s are stored to and retrieved from disk or secondary storage in general by callbacks that are specified in form of a `QsTerminalMgr`. Although no particular assumption about how this is done is made from the perspective of the Operand module, it is reasonable to assume some sort of mutual exclusion is eventually needed in the callbacks. Typically, if everything is written to a common database on disk, storing `QsCoefficient`s is mutually exclusive and exclusive with respect to reading. In any case, the storing and loading of one particular `QsTerminal`'s `QsCoefficient` value would be mutually exclusive, but depending on how granual the backing store is designed, the mutual exclusion may possibly be coarser than that.

Since the Operand module may not know about what particular exclusion limits are needed, the locking scheme is implemented outside of the Operand module's scope and the Operand module may also forgo the most granual, per `QsTerminal` lock which will automatically be covered by a possibly coarser scheme in the callbacks.

That said, `QsTerminalData::lock` does not exist to protect access to `QsTerminalData::coefficient`, which is assumed to be protected by the exclusion in the callbacks. It is however necessary, in order to make accesses to `QsTerminalData::coefficient` and `QsTerminalData::refcount`, the reference count atomic but does not extend around the process of storing or retrieving the actual data. This is also the only sane choice to be made with generic callbacks where all locks should be released before leaving the realm of one's responsibility after which point unpredictable things may happen.

A.4 Distributed parallel evaluation

The backend, as it was described, works in a memory-shared parallelized fashion. In the particular version at hand, where an external process is used to perform the evaluation, the worker threads could have been replaced by an unthreaded version which uses non-blocking communication with the external processes. While this would, due to the inherently parallel nature of processes, still result in parallel evaluation, explicit threading would already be required in order to modify the Operand-tree and its associated values, e.g. the ADC and D-Value efficiently.

A fully threaded version allows to easily use different implementations of the Coefficient module, including those which perform the evaluations without forking to a new process. Yet, evaluations are intrinsically memory distributed because they take place in different processes. The Operand module only relies on a specific protocol for communicating with the evaluator process. Since it does not matter how this process obtains the results, that process may as well run on a different CPU.

This is in fact the approach used to parallelize QS to more than the number of cores available on a single computer. As for our particular application where the expressions grow to the size of many hundreds of Megabytes and the time to calculate them easily becomes hours or days, the calculation is out-sourced to other computers via LAN which quickly becomes feasible with typical network speed. With the supplementary wrapper to FERMAT, `distributed_fermat`, priorities may be assigned to different hosts and precedence be given to local evaluation. `QsEvaluator` for this purpose supports and expects an option in `QsEvaluatorOptions` which designates the binary to be used for evaluation. This feature is made use of in the implemented policy, to be described below, where two different `QsAEF`s with different associated `QsEvaluatorOptions` and thus different binaries are employed.

Together with robust handling of failures as described above, this allows one to utilize an arbitrary set of desktop computers, provided they are in principle available, and thus increase the number of evaluators to the dozens or hundreds, depending on the amount of parallelization permitted by the Operand-tree in the first place. Eventually, this will turn out as a practical and very efficient approach to parallelizing the system though clearly, all memory still resides on the principal computer which therefore may require tremendous resources. However, even for the demand of resources, distributed solutions may easily be found without unnecessarily extending the Operand module by further functionality.

Appendix B

The frontend

The backend is mostly a general-purpose module for parallel, heavy symbolic operations and can, apart aforementioned limitations on prioritization and discarding, be applied to a variety of problems. The frontend, on the other hand, is specifically crafted towards the solution of identities.

Just as the frontend is specialized for the solution of a particular problem compared to the backend, it may again be divided into a more general framework for solving symbolic matrices with a Gauss-like algorithm, the pivot graph, and a more specific layer for solving identities, the actual policy.

However, for practical reasons and as will be detailed immediately below, although representing a more general concept, the pivot graph is designed around the policy and offers some features implemented for that very policy.

B.1 The pivot graph

The pivot graph `QsPivotGraph` topologically represents the system's pivot graph. QS begins the solution with the initial pivot graph which of the pivoted system of identities from Czakon algorithm. That is, it starts from the pregenerated set of identities provided by the user in databases. `QsPivotGraph` then modifies the pivot graph and stores the modified, preferably conclusively solved version back to disk as identities as well.

Every component of the pivot graph is identified by a `QsComponent` which is an integer. When loading identities into the pivot graph or storing the results back to database, the translation between the verbose identification of each pivot as an integral and the identification as a `QsComponent` is handled by the loading and storing layer, that is `QsIntegralMgr` which keeps a bijective mapping of integrals to integers.

B.1.1 Interface

QsPivotGraph provides generic functions to perform the canonical transformations relay, collect, and normalize on the graph. The coefficients, that is the edges of the pivot graph are realized by QsOperands which makes all of those operations non-blocking.

In order for the policy to obtain the actual values of the edges to make decisions, qs_pivot_graph_terminate_nth is used, to bake any QsOperand into a QsTerminal. QsPivotGraph does not fully hide the underlying Operand module from the user. Edges are returned as QsOperand as QsTerminals so the user may, for example, add them into a QsTerminalGroup.

Besides representing the system's pivot graph, QsPivotGraph implements the memory management callbacks and, consequently, handlers to load and store data from and to disk.

The loads and stores are eventually performed by either the QsPivotGraph itself or the QsIntegralMgr, depending on what is loaded or stored.

The initial identities and the identities of the final state of the pivot graph are actually loaded and stored by further callbacks in QsIntegralMgr which acts as a unifying layer between the frontend and a particular representation on disk. Besides the mapping of verbose integral descriptions to unified QsComponents and vice versa, the QsIntegralMgr also chooses the database files from which the initial identities are read and those into which the solution is written.

QsCoefficients which are to be stored from within the Operand-tree for memory management and later retrieved on behalf of the Operand module are handled directly in the QsPivotGraph's callbacks and stored in a QsDb database known to the QsPivotGraph.

As mentioned, QsPivotGraph is tailored towards the policy by which it is employed. In particular, QsPivotGraph keeps two coefficients, i.e. QsOperands, per edge which are are calculated in exactly the same fashion with the slight difference, that their evaluations are performed by two different QsAEFs. As will be explained about the policy, one coefficient is a numeric representation and one is a symbolic representation. Clearly, from transformations of the pivot graph, there then emerge two completely separated operand trees for the numerical and the symbolic coefficients.

When the QsPivotGraph is created it is therefore passed the two QsAEFs associated with the respective duplicate of each coefficient. It is passed the QsIntegralMgr callbacks for loading and saving the identities. And it is

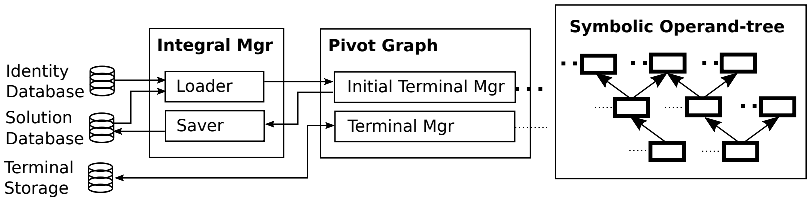

Figure B.1: Relation of the three databases, the `QsIntegralMgr`, the `QsPivotGraph` and the symbolic Operand-tree. As indicated by either the thick dotted lines, the very first operands are provided by the `QsIntegralMgr` from either the initial identity databases or already available solution databases. As indicated the thin dotted lines, operands which are created during the solution of the system as temporary objects are backed by the separate Terminal Storage which is accessed directly from the Pivot Graph through the memory management callbacks.

passed a database object `QsDb` for storing and loading `QsCoefficients` for memory management as well as the supposed memory limit.

All pivots are augmented by a `QsMetadata` structure the contents of which are up to the policy to choose. Contrary to the Operand module which is intended for generic use, the `QsPivotGraph` is, for practical reasons, not as strictly separated and thus the `QsMetadata` structure is embedded directly in the data of the `QsPivotGraph`, rather than held a pointer to. The user of `QsPivotGraph` can obtain a pointer to each pivot's `QsMetadata` by a call to `qs_pivot_graph_meta` which, through the return of a NULL pointer, would also indicate that the designated component is not a pivot.

A special deletion technicality is implemented in `qs_pivot_graph_delete_nth.` As all `_nth` functions, the desired edge is indicated by its index in the list of edges rather than by giving its head component. Typically, a policy needs to consider all edges from a given pivot by looping over them. When an edge somewhere in the middle of the list is deleted, consecutive indices must change. In order for a policy to continue the loop unaffected and neither miss any edge nor consider an edge twice, it should, when deleting an edge, pass the index of the current iteration. The elements in the array are then re-ordered such that repeating the iteration at the current index will work as expected.

B.1.2 Implementation

Since the heavy work is lifted by the Operand module, the `QsPivotGraph` consists merely of structures representing the topology of the pivot graph and those necessary for memory management. It holds an array `QsPivotGraph::components` of pointers to Pivot structures wherein each `QsComponent` is associated with its according index in the array. Components which are not pivots are thus NULL in the array.

Each Pivot specifies the edges from it to other components as an array of References, each element of which specifies the head of the edge as well as the two `QsOperand`s, coefficient for the symbolic coefficient and numeric for the numeric.

Since numeric coefficients are tiny compared to symbolic coefficients, only symbolic coefficients are memory managed. For their management, there are two `QsTerminalMgr`s with one common `QsTerminalQueue`. Which of the `QsTerminalMgr`s is assigned to a symbolic `QsTerminal` depends on the origin of the latter. If it corresponds to a coefficient that resulted from baking other `QsOperand`s, the normal `QsTerminalMgr` `QsPivotGraph::memory::mgr` is assigned, which stores and retrieves `QsCoefficient`s from the known `QsDb` database.

If, however, it corresponds to a coefficient that was specified by the initial identities, storing it to backing store would be a waste of resources. Instead, no action is taken since the data is already available in the initial databases. This is done by the initial `QsTerminalMgr` `QsPivotGraph::memory::initial_mgr`.

For symbolic `QsTerminal`s which are handled by the initial `QsTerminalMgr`, the `QsTerminalMeta` identification is of type CoefficientId and designates the edge in the initial system, that is the tail and head component or the identities and its summand respectively.

For the baked `QsTerminal`s which are handled by the normal `QsTerminalMgr`, `QsPivotGraph` implements a mechanism to generate unique identifiers of type `CoefficientUID`. They are of incrementing integer type and wrap around at the upper bound of the particular type. The number of assigned `CoefficientUID`s in the lower half and the upper half of the allowed range is tracked and identifiers discarded when no longer needed, to assert that no `CoefficientUID` is assigned twice after at least one wrap has occurred.

B.2 Implementation of the policy

The policy is implemented in a single function which is recursively called. It can be subdivided into three main sections which will be described in detail in the following three chapters. It is called with the `QsComponent` identifying the current target pivot and an auxiliary structure containing various data which need to be passed along, such as the `QsPivotGraph` in use or the option for elimination.

Each function call corresponds to either the relay of an edge or an attempted normalization with return. In other words, the relaying of multiple edges from a single pivot is done by proper tail recursion to forgo yet another loop, whereas the recursion to other pivots is realized by improper recursion

The initial invocation of the function for the pivot which is to be solved specifies a despair of one. Since at that point, all pivots' consideration count is zero, this causes the algorithm to consider all edges from that very pivot to any other pivot suitable, to the effect that not only the edges to pivots with lower order of complexity are eliminated, but all edges, as it should be.

Search for edges The first section is responsible for work related to looping over all edges of the current pivot. It attempts to find the one, if any, that is supposed to be relayed. During and after the loop, the elimination of vanishing edges is effected as described above.

Edges which are not yet numerically known are added to a `QsTerminalGroup`. To not senselessly add everything, they are added incrementally as they are found to be not ready. The loop thus constitutes a repetition of considering new edges and checking whether any of the already considered ones has become ready. If there are no more edges to consider, the repetition waits on the `QsTerminalGroup` with `qs_terminal_group_wait`.

With optimistic elimination, numerically vanishing edges are removed at the very moment they are found zero. Otherwise, their symbolic `QsOperand` is added to a second `QsTerminalGroup`.

Eventually, if a suitable edge for relay has been found, the contents of that second `QsTerminalGroup` are ignored and discarded. Only if no further edge is to be relayed, indicating the improper recursion will end with the associate normalization, the numerically vanishing edges may be symbolically waited for.

Relay with tail recursion If an edge for relay has been found the algorithm recurses to the associated pivot with despair zero. When it returns, that pivot is guaranteed to be normalized and thus ready for relay. While the algorithm recurses, `QsMetadata::touched` associated with the current pivot is assumed to remain false. If the recursion, due to an unnormalizable pivot, happens to recurse back into the current pivot, the algorithm will we able to detect this and skip any further actions, assuming that whatever conclusion it has drawn about the edges was invalidated by now. If, however, `QsMetadata::touched` successfully remains false, all pivots are collected and it is attempted to relay the next edge by tail recursion.

Normalize with return If no edge for relay could be found in the first section, the recursion will return to its improperly recursive caller after normalizing the current pivot. If it finds that the pivot is non-normalizable the despair is increased by one and further relays on the current pivot are commenced by proper tail-recursion. This hopefully, eventually leads to the pivot being normalized, at which point the recursion returns all the way up.

Bibliography

[1] "A Combination of preliminary electroweak measurements and constraints on the standard model". In: (2004). arXiv: hep-ex/0412015 [hep-ex].

[2] Georges Aad et al. "Measurement of the Higgs boson mass from the $H \to \gamma\gamma$ and $H \to ZZ^* \to 4\ell$ channels with the ATLAS detector using 25 fb^{-1} of pp collision data". In: *Phys. Rev.* D90.5 (2014), p. 052004. DOI: 10.1103/PhysRevD.90.052004. arXiv: 1406.3827 [hep-ex].

[3] Georges Aad et al. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Phys. Lett.* B716 (2012), pp. 1–29. DOI: 10.1016/j.physletb.2012.08.020. arXiv: 1207.7214 [hep-ex].

[4] E. Accomando et al. "Workshop on CP Studies and Non-Standard Higgs Physics". In: (2006). DOI: 10.5170/CERN-2006-009. arXiv: hep-ph/0608079 [hep-ph].

[5] Charalampos Anastasiou et al. "High precision determination of the gluon fusion Higgs boson cross-section at the LHC". In: *JHEP* 05 (2016), p. 058. DOI: 10.1007/JHEP05(2016)058. arXiv: 1602.00695 [hep-ph].

[6] C. Anastasiou et al. "The Tensor reduction and master integrals of the two loop massless crossed box with lightlike legs". In: *Nucl. Phys.* B580 (2000), pp. 577–601. DOI: 10.1016/S0550-3213(00)00251-0. arXiv: hep-ph/0003261 [hep-ph].

[7] L. Arnaudon et al. "Accurate determination of the LEP beam energy by resonant depolarization". In: *Z. Phys.* C66 (1995), pp. 45–62. DOI: 10.1007/BF01496579.

[8] D. M. Asner et al. “ILC Higgs White Paper”. In: *Proceedings, Community Summer Study 2013: Snowmass on the Mississippi (CSS2013): Minneapolis, MN, USA, July 29-August 6, 2013.* 2013. arXiv: `1310.0763 [hep-ph]`. URL: `http://inspirehep.net/record/1256491/files/arXiv:1310.0763.pdf`.

[9] G. Baker and P. Graves-Morris. *Padé Approximants.* 1996.

[10] R. Barate et al. “Search for the standard model Higgs boson at LEP”. In: *Phys. Lett.* B565 (2003), pp. 61–75. DOI: `10.1016/S0370-2693(03)00614-2`. arXiv: `hep-ex/0306033 [hep-ex]`.

[11] Vernon D. Barger, R. J. N. Phillips, and D. Zeppenfeld. “Mini - jet veto: A Tool for the heavy Higgs search at the LHC”. In: *Phys. Lett.* B346 (1995), pp. 106–114. DOI: `10.1016/0370-2693(95)00008-9`. arXiv: `hep-ph/9412276 [hep-ph]`.

[12] Gautam Bhattacharyya. “A Pedagogical Review of Electroweak Symmetry Breaking Scenarios”. In: *Rept. Prog. Phys.* 74 (2011), p. 026201. DOI: `10.1088/0034-4885/74/2/026201`. arXiv: `0910.5095 [hep-ph]`.

[13] G. C. Branco et al. “Theory and phenomenology of two-Higgs-doublet models”. In: *Phys. Rept.* 516 (2012), pp. 1–102. DOI: `10.1016/j.physrep.2012.02.002`. arXiv: `1106.0034 [hep-ph]`.

[14] Michele Caffo et al. “The Master differential equations for the two loop sunrise selfmass amplitudes”. In: *Nuovo Cim.* A111 (1998), pp. 365–389. arXiv: `hep-th/9805118 [hep-th]`.

[15] John M. Campbell and R. Keith Ellis. “Higgs Constraints from Vector Boson Fusion and Scattering”. In: *JHEP* 04 (2015), p. 030. DOI: `10.1007/JHEP04(2015)030`. arXiv: `1502.02990 [hep-ph]`.

[16] John M. Campbell, R. Keith Ellis, and Ciaran Williams. “Bounding the Higgs width at the LHC using full analytic results for $gg- > e^-e^+\mu^-\mu^+$”. In: *JHEP* 04 (2014), p. 060. DOI: `10.1007/JHEP04(2014)060`. arXiv: `1311.3589 [hep-ph]`.

[17] John M. Campbell, Walter T. Giele, and Ciaran Williams. “The Matrix Element Method at Next-to-Leading Order”. In: *JHEP* 11 (2012), p. 043. DOI: `10.1007/JHEP11(2012)043`. arXiv: `1204.4424 [hep-ph]`.

[18] John M. Campbell et al. “Two loop correction to interference in $gg \to ZZ$”. In: *JHEP* 08 (2016), p. 011. DOI: `10.1007/JHEP08(2016)011`. arXiv: `1605.01380 [hep-ph]`.

[19] Fabrizio Caola and Kirill Melnikov. "Constraining the Higgs boson width with ZZ production at the LHC". In: *Phys. Rev.* D88 (2013), p. 054024. DOI: 10.1103/PhysRevD.88.054024. arXiv: 1307.4935 [hep-ph].

[20] S. Chatrchyan et al. "The CMS experiment at the CERN LHC". In: *JINST* 3 (2008), S08004. DOI: 10.1088/1748-0221/3/08/S08004.

[21] Serguei Chatrchyan et al. "Energy Calibration and Resolution of the CMS Electromagnetic Calorimeter in pp Collisions at $\sqrt{s} = 7$ TeV". In: *JINST* 8 (2013). [JINST8,9009(2013)], P09009. DOI: 10.1088/1748-0221/8/09/P09009. arXiv: 1306.2016 [hep-ex].

[22] Serguei Chatrchyan et al. "Measurement of the properties of a Higgs boson in the four-lepton final state". In: *Phys. Rev.* D89.9 (2014), p. 092007. DOI: 10.1103/PhysRevD.89.092007. arXiv: 1312.5353 [hep-ex].

[23] Serguei Chatrchyan et al. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". In: *Phys. Lett.* B716 (2012), pp. 30–61. DOI: 10.1016/j.physletb.2012.08.021. arXiv: 1207.7235 [hep-ex].

[24] K. G. Chetyrkin and F. V. Tkachov. "Integration by Parts: The Algorithm to Calculate beta Functions in 4 Loops". In: *Nucl. Phys.* B192 (1981), pp. 159–204. DOI: 10.1016/0550-3213(81)90199-1.

[25] The ATLAS Collaboration and G Aad. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3.08 (2008), S08003. URL: http://stacks.iop.org/1748-0221/3/i=08/a=S08003.

[26] John C. Collins. "What exactly is a parton density?" In: *Acta Phys. Polon.* B34 (2003), p. 3103. arXiv: hep-ph/0304122 [hep-ph].

[27] *Constraints on the Higgs boson width from off-shell production and decay to ZZ to llll and llvv.* Tech. rep. CMS-PAS-HIG-14-002. Geneva: CERN, 2014. URL: https://cds.cern.ch/record/1670066.

[28] John M. Cornwall, David N. Levin, and George Tiktopoulos. "Derivation of Gauge Invariance from High-Energy Unitarity Bounds on the s Matrix". In: *Phys. Rev.* D10 (1974). [Erratum: Phys. Rev.D11,972(1975)], p. 1145. DOI: 10.1103/PhysRevD.10.1145, 10.1103/PhysRevD.11.972.

[29] Nathaniel Craig, Jamison Galloway, and Scott Thomas. "Searching for Signs of the Second Higgs Doublet". In: (2013). arXiv: 1305.2424 [hep-ph].

[30] M. Czakon. “IdSolver”. unpublished.

[31] M. Czakon. “Tops from Light Quarks: Full Mass Dependence at Two-Loops in QCD”. In: *Phys. Lett.* B664 (2008), pp. 307–314. DOI: 10.1016/j.physletb.2008.05.028. arXiv: 0803.1400 [hep-ph].

[32] Sacha Davidson and Howard E. Haber. “Basis-independent methods for the two-Higgs-doublet model”. In: *Phys. Rev.* D72 (2005), p. 035004. DOI: 10.1103/PhysRevD.72.099902,10.1103/PhysRevD.72.035004. arXiv: hep-ph/0504050 [hep-ph].

[33] Ansgar Denner and S. Dittmaier. “Reduction schemes for one-loop tensor integrals”. In: *Nucl. Phys.* B734 (2006), pp. 62–115. DOI: 10.1016/j.nuclphysb.2005.11.007. arXiv: hep-ph/0509141 [hep-ph].

[34] A. Denner et al. “Standard Model Higgs-Boson Branching Ratios with Uncertainties”. In: *Eur. Phys. J.* C71 (2011), p. 1753. DOI: 10.1140/epjc/s10052-011-1753-8. arXiv: 1107.5909 [hep-ph].

[35] Ganesh Devaraj and Robin G. Stuart. “Reduction of one loop tensor form-factors to scalar integrals: A General scheme”. In: *Nucl. Phys.* B519 (1998), pp. 483–513. DOI: 10.1016/S0550-3213(98)00035-2. arXiv: hep-ph/9704308 [hep-ph].

[36] Abdelhak Djouadi. “The Anatomy of electro-weak symmetry breaking. I: The Higgs boson in the standard model”. In: *Phys. Rept.* 457 (2008), pp. 1–216. DOI: 10.1016/j.physrep.2007.10.004. arXiv: hep-ph/0503172 [hep-ph].

[37] R. S. Fletcher and T. Stelzer. “Rapidity gap signals in Higgs production at the SSC”. In: *Phys. Rev.* D48 (1993), pp. 5162–5167. DOI: 10.1103/PhysRevD.48.5162. arXiv: hep-ph/9306253 [hep-ph].

[38] H. M. Georgi et al. “Higgs Bosons from Two Gluon Annihilation in Proton Proton Collisions”. In: *Phys. Rev. Lett.* 40 (1978), p. 692. DOI: 10.1103/PhysRevLett.40.692.

[39] B. Grzadkowski, P. Krawczyk, and S. Pokorski. “NATURAL RELATIONS AND APPELQUIST-CARAZZONE DECOUPLING THEOREM”. In: *Phys. Rev.* D29 (1984), pp. 1476–1487. DOI: 10.1103/PhysRevD.29.1476.

[40] J. F. Gunion, H. E. Haber, and J. Wudka. “Sum rules for Higgs bosons”. In: *Phys. Rev.* D43 (1991), pp. 904–912. DOI: 10.1103/PhysRevD.43.904.

[41] Howard E. Haber and Yosef Nir. “Multiscalar Models With a High-energy Scale”. In: *Nucl. Phys.* B335 (1990), pp. 363–394. DOI: 10.1016/0550-3213(90)90499-4.

[42] Johannes M. Henn. “Multiloop integrals in dimensional regularization made simple”. In: *Phys. Rev. Lett.* 110 (2013), p. 251601. DOI: `10.1103/PhysRevLett.110.251601`. arXiv: `1304.1806 [hep-th]`.

[43] K. Hiller et al. “ILC beam energy measurement based on synchrotron radiation from a magnetic spectrometer”. In: *Nucl. Instrum. Meth.* A580 (2007), pp. 1191–1200. DOI: `10.1016/j.nima.2007.07.030`.

[44] Shinya Kanemura et al. “Higgs coupling constants as a probe of new physics”. In: *Phys. Rev.* D70 (2004), p. 115002. DOI: `10.1103/PhysRevD.70.115002`. arXiv: `hep-ph/0408364 [hep-ph]`.

[45] Nikolas Kauer and Giampiero Passarino. “Inadequacy of zero-width approximation for a light Higgs boson signal”. In: *JHEP* 08 (2012), p. 116. DOI: `10.1007/JHEP08(2012)116`. arXiv: `1206.4803 [hep-ph]`.

[46] Vardan Khachatryan et al. “Constraints on the Higgs boson width from off-shell production and decay to Z-boson pairs”. In: *Phys. Lett.* B736 (2014), pp. 64–85. DOI: `10.1016/j.physletb.2014.06.077`. arXiv: `1405.3455 [hep-ex]`.

[47] Vardan Khachatryan et al. “Limits on the Higgs boson lifetime and width from its decay to four charged leptons”. In: *Phys. Rev.* D92.7 (2015), p. 072010. DOI: `10.1103/PhysRevD.92.072010`. arXiv: `1507.06656 [hep-ex]`.

[48] Stephen F. King. “Dynamical electroweak symmetry breaking”. In: *Rept. Prog. Phys.* 58 (1995), pp. 263–310. DOI: `10.1088/0034-4885/58/3/001`. arXiv: `hep-ph/9406401 [hep-ph]`.

[49] S. Laporta. “High precision calculation of multiloop Feynman integrals by difference equations”. In: *Int. J. Mod. Phys.* A15 (2000), pp. 5087–5159. DOI: `10.1016/S0217-751X(00)00215-7,10.1142/S0217751X00002157`. arXiv: `hep-ph/0102033 [hep-ph]`.

[50] S. Laporta and E. Remiddi. “The Analytical value of the electron (g-2) at order alpha**3 in QED”. In: *Phys. Lett.* B379 (1996), pp. 283–291. DOI: `10.1016/0370-2693(96)00439-X`. arXiv: `hep-ph/9602417 [hep-ph]`.

[51] Benjamin W. Lee, C. Quigg, and H. B. Thacker. “The Strength of Weak Interactions at Very High-Energies and the Higgs Boson Mass”. In: *Phys. Rev. Lett.* 38 (1977), pp. 883–885. DOI: `10.1103/PhysRevLett.38.883`.

[52] Roman N. Lee. “Reducing differential equations for multiloop master integrals”. In: *JHEP* 04 (2015), p. 108. DOI: `10.1007/JHEP04(2015)108`. arXiv: `1411.0911 [hep-ph]`.

[53] Andreas von Manteuffel and Robert M. Schabinger. “A novel approach to integration by parts reduction”. In: *Phys. Lett.* B744 (2015), pp. 101–104. DOI: 10.1016/j.physletb.2015.03.029. arXiv: 1406.4513 [hep-ph].

[54] K. A. Olive et al. “Review of Particle Physics/Higgs Boson Physics”. In: *Chin. Phys.* C38 (2014), p. 090001. DOI: 10.1088/1674-1137/38/9/090001.

[55] G. Passarino and M. J. G. Veltman. “One Loop Corrections for e+ e- Annihilation Into mu+ mu- in the Weinberg Model”. In: *Nucl. Phys.* B160 (1979), pp. 151–207. DOI: 10.1016/0550-3213(79)90234-7.

[56] Ringaile Placakyte. “Parton Distribution Functions”. In: *Proceedings, 31st International Conference on Physics in collisions (PIC 2011): Vancouver, Canada, August 28-September 1, 2011.* 2011. arXiv: 1111.5452 [hep-ph]. URL: https://inspirehep.net/record/954990/files/arXiv:1111.5452.pdf.

[57] *Properties of the Higgs-like boson in the decay H to ZZ to 4l in pp collisions at sqrt s =7 and 8 TeV.* Tech. rep. CMS-PAS-HIG-13-002. Geneva: CERN, 2013. URL: https://cds.cern.ch/record/1523767.

[58] V. Ravindran, J. Smith, and W. L. van Neerven. “NNLO corrections to the total cross-section for Higgs boson production in hadron hadron collisions”. In: *Nucl. Phys.* B665 (2003), pp. 325–366. DOI: 10.1016/S0550-3213(03)00457-7. arXiv: hep-ph/0302135 [hep-ph].

[59] Ettore Remiddi. “Differential equations for Feynman graph amplitudes”. In: *Nuovo Cim.* A110 (1997), pp. 1435–1452. arXiv: hep-th/9711188 [hep-th].

[60] Bruno Rossi and David B. Hall. “Variation of the Rate of Decay of Mesotrons with Momentum”. In: *Phys. Rev.* 59 (1941), pp. 223–228. DOI: 10.1103/PhysRev.59.223.

[61] Carl Schmidt et al. “CT14QED parton distribution functions from isolated photon production in deep inelastic scattering”. In: *Phys. Rev.* D93.11 (2016), p. 114015. DOI: 10.1103/PhysRevD.93.114015. arXiv: 1509.02905 [hep-ph].

[62] A. V. Smirnov and A. V. Petukhov. “The Number of Master Integrals is Finite”. In: *Lett. Math. Phys.* 97 (2011), pp. 37–44. DOI: 10.1007/s11005-010-0450-0. arXiv: 1004.4199 [hep-th].

[63] V. A. Smirnov. *Feynman integral calculus.* 2006.

[64] Vladimir A. Smirnov. “Applying Gröbner Bases to Solve IBP Relations”. In: *Feynman Integral Calculus*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 251–262. ISBN: 978-3-540-30611-5. DOI: `10.1007/3-540-30611-0_14`. URL: `http://dx.doi.org/10.1007/3-540-30611-0_14`.

[65] M. Spira et al. “Higgs boson production at the LHC”. In: *Nucl. Phys.* B453 (1995), pp. 17–82. DOI: `10.1016/0550-3213(95)00379-7`. arXiv: `hep-ph/9504378 [hep-ph]`.

[66] O. V. Tarasov. “Reduction of Feynman graph amplitudes to a minimal set of basic integrals”. In: *Acta Phys. Polon.* B29 (1998), p. 2655. arXiv: `hep-ph/9812250 [hep-ph]`.

[67] M. J. G. Veltman. “Second Threshold in Weak Interactions”. In: *Acta Phys. Polon.* B8 (1977), p. 475.

[68] J Wenninger. *Energy Calibration of the LHC Beams at 4 TeV*. Tech. rep. CERN-ATS-2013-040. Geneva: CERN, May 2013. URL: `http://cds.cern.ch/record/1546734`.