

reSolve - A Transverse Momentum Resummation Tool

F. Coradeschi^a, T. Cridge^{a,b}

^aDAMTP, CMS, University of Cambridge, Wilberforce road, Cambridge, CB3 0WA, United Kingdom

^bKITP, University of California, Santa Barbara, CA 93106, USA

Abstract

In this note, we introduce the new tool reSolve, a Monte Carlo differential cross-section and parton-level event generator whose main purpose is to add transverse momentum resummation to a general class of inclusive processes at hadron colliders, namely all those which do not involve hadrons or jets in the measured final state. This documentation refers to the first main version release, which will form the basis for continued developments, consequently it only implements the key features of those we plan to ultimately include. This article acts as a manual for the program; describing in detail its use, structure, validation and results; whilst also highlighting key aspects of the resummation formalism applied. It details the two classes of processes so far included; these are diphoton production and Drell-Yan production.

A main concept behind the development of the tool is that it is a hands-on white box for the user: significant effort has been made to give the program a modular structure, making the various parts which comprise it independent of each other as much as possible and ensuring they are transparently documented, customizable and, in principle, replaceable with something that may better serve the users needs.

reSolve is a new C++ program, based on an evolution of the private Fortran code 2gres, previously used for the calculations in refs. [1] and [2]; it is also influenced by the DYRes Fortran code of refs. [3] and [4]. This initial version calculates the low transverse momentum contribution to the fully differential cross-section for two main categories of processes; the inclusive production of two photons, and inclusive Drell-Yan production. In all cases resummation up to Next-to-Next-to-Leading Logarithm (NNLL) is included. We aim to extend the program to several more processes in the near future. The program is publicly available on Github.

Keywords: Resummation, transverse momentum, precision, differential distributions, diphoton, Drell-Yan

1. Program Summary

Program title: reSolve

Program obtainable from: <https://github.com/fkhorad/reSolve/>

Distribution format: tar.gz

Programming language: C++, fortran

Computer: Personal computer.

Operating system: Tested on Linux 4.13.0-38-generic, Linux 4.4.0-119-generic and on MAC-OS 10.11.6.

Word size: 64 bits.

External routines: minuit and Cuba

Typical running time: 0.1-1 seconds per Monte Carlo point per core used.

Nature of problem: Calculating the transverse momentum spectrum, including resummation, for a general process at hadron colliders.

*Corresponding author

Email address: t.cridge@damtp.cam.ac.uk (T. Cridge)

Solution method: Monte Carlo generation of the phase space points and phase space integration to allow the production of differential distributions, each phase space point itself has to be inverse Fourier transformed and double inverse Mellin transformed to allow the resummation, following the usual transverse momentum resummation impact parameter space formalism. reSolve operates up to Next-to-Next-to-Leading Order including Next-to-Next-to-Leading Logarithm resummation (NNLO+NNLL).

Restrictions: So far only diphoton production in the Standard Model (background, not including Higgs) and Drell-Yan production are included, nonetheless the program is designed to allow further extensions to additional processes, including by the user. The limitations on the processes possible to implement are that they must be hadron-hadron collisions producing a non-strongly interacting measured final state system. This first main implementation of reSolve calculates only the resummed part of the differential cross-section, which is dominant at low transverse momentum, this has to be matched with the usual finite contributions to obtain the spectrum over the whole transverse momentum range.

2. Introduction

Since the early days of the Large Hadron Collider, a large part of the experimental effort has been focused on direct searches for signals of New Physics at ever-increasing energy scales. However, given the lack of any clear evidence for new phenomena so far, the focus has been shifting, both theoretically and experimentally, to alternative strategies. Amongst these is what can be called the “precision physics at colliders” program; this involves an effort to extend both our experimental measurements and theoretical predictions of as many observables as possible, in the context of the Standard Model (SM) and beyond, to an ever increasing precision. This will allow us both to extend our knowledge and understanding of the properties of the model, and hopefully to eventually detect small deviations from the SM which will signal the presence of the as yet undiscovered New Physics which is expected to exist at higher scales.

An interesting observable for this precision strategy is the transverse momentum spectrum of generic final states. This is theoretically interesting to check our understanding of perturbative QCD and also important for the precise determination of observables such as the W mass [5] and many others, and will be a focus of ongoing and future measurements at 13TeV at Run 2 of the LHC and beyond.

The theoretical calculation of the transverse momentum (q_T) spectrum for generic systems of invariant mass M is challenging towards the low end of the spectrum, where $q_T \ll M$, due to the well-known presence of logarithmically-enhanced terms which appear to all orders in the perturbative expansion. In particular, once $\log^{-1}(M/q_T)$ is of order of $\alpha_s(M)$ (with α_s the strong coupling constant) the convergence of the perturbative series is completely ruined, and it is necessary to resum such large logarithmic contributions to all orders theoretically if we are to produce meaningful theoretical predictions at low transverse momenta. Indeed without this resummation, transverse momentum spectra diverge as $q_T \rightarrow 0$ (for instance see figure 1 in [1] for an explicit example of this divergence in the context of diphoton production); rendering both the total cross-section and differential transverse momentum distribution predictions without such resummation inaccurate at best, and often useless. The q_T resummation is an involved but well studied problem; in the present article, and in the program reSolve, we will follow the impact parameter space formalism of references [6, 7, 8, 9, 10, 11, 12, 13, 14, 15].

The aim of this work is to outline the development of a general tool, the reSolve program, which allows the implementation of q_T resummation to a large class of inclusive processes, of the generic form $h_1 h_2 \rightarrow F + X$, where h_1 and h_2 are hadrons, F is the measured final state, which can include an arbitrary number of *nonstrongly – interacting* particles (no jets or hadrons are allowed in F), and X denotes unresolved extra radiation. This therefore allows event generation and predictions for arbitrary observables at the fully differential level for such processes. reSolve is a new program, written completely from scratch in C++, based around an evolution of a previous private code 2gres, which was used for the calculations in refs. [1] and [2], and which is in turn related to the resummation codes DYres [3, 16] and Hres [17]. In this first main release, two processes are so far included, namely diphoton production ($F = \gamma\gamma$) and Drell-Yan production ($F = W, Z, Z/\gamma^*$); in both Next-to-Next-to-Leading-Logarithmic (NNLL) precision can be reached in the resummation. The diphoton process is interesting as it provides the SM background for Higgs production and decay in one of the experimentally cleanest channels. Hopes of understanding the nature of Electroweak Symmetry Breaking at a deeper level, and more generally of improving our knowledge of

Higgs boson physics, make the diphoton channel one of significant experimental focus, in both signal and background. The Drell-Yan process is similarly intriguing, used experimentally to measure the mass of the W boson, to constrain PDFs, and for many other purposes, it could also be extended to Beyond Standard Model W' and Z' models. In future iterations of reSolve further processes will be added, with Higgs production and decay (and signal-background interference) being amongst the easiest potential additions of the possible interesting candidates. The program has been designed in such a way so as to make the addition of new processes as straightforward as possible, taking full advantage of the substantial process-independence of the formalism, in order to also allow perspective users to add processes of their own interest. Indeed a description of how to add a process to reSolve is outlined in Section 4.3.

The outline of the paper is as follows; we begin in section 3 with the details of how to use the program – its input and output files, options for running with one or more PDF fits, parallelisation and using the incorporated histogramming program to convert the generated events into histogram data for the differential cross-section spectra. This section serves as a guide to users of the program. Following this, in section 4, we go into detail about the modular structure of the program, designed to make the addition of new processes straightforward, including providing an outline of how the code actually works. In section 5, a review of the theoretical impact parameter space resummation formalism is included, with several pointers towards how the formulae were practically implemented in the code, before section 6 provides further information on the validation of the reSolve program and results that can be generated by the program for both the diphoton and Drell-Yan processes. In addition its speed is briefly discussed. Finally, we end by signposting some of the future developments we intend to complete following this first version in section 7, before concluding in section 8. Appendix A provides a summary of how to use reSolve along with the sample input files included with the program, all included here for ease of reference; meanwhile Appendix B contains further information on the details of the theoretical formalism applied for the resummation.

3. Using the reSolve Program

We are presenting the new Monte Carlo tool reSolve. The objective of the program is to calculate differential cross-sections and generate events at parton-level in generic processes of the class $h_1 h_2 \rightarrow F + X$, where h_1 and h_2 are hadrons and F is a final state containing an arbitrary number of particles, but no strongly interacting ones, that is, no jets and no hadrons, including transverse momentum resummation; whilst X represents unresolved extra radiation. As of the current version, the processes implemented are $F = \gamma\gamma$ and $F = W^+, W^-, Z, Z/\gamma$. The structure of the program is however general, and most of the instructions that will follow will apply equally well to other processes and future versions. This paper will therefore serve as the main manual for users of the reSolve program.

3.1. Getting it running

- In order to run the reSolve program, first download the zipped tarball `reSolve.tar.gz`, this is available both on Github at <https://github.com/fkhorad/reSolve>, and with the published paper. Extract this to the desired working directory.
- Next enter the makefile, found in `code/src` and adapt any compilation flags as appropriate to your machine, note if you wish to use the Cuba integrator [18], rather than just the built-in `k_vegas` integrator provided with the code, you will need to tell the program where to find it. Interface codes for both `k_vegas` (our integrator) and Cuba are provided within the program in `code/src/main` and the relevant interface is automatically used once the integrator chosen is given in the input file, see section 3.2.
- Finally run ‘`make`’ to compile the program and produce the `./reSolve.out` executable.
- Running the program then involves simply entering in the terminal `./reSolve.out {path to input file}, e.g. ./reSolve.out input/Diphoton_NNLO_test_1.dat .`
- Note that the output working directory to which the output is directed is included in the input file. In order to avoid overwriting of data or corruption of events, reSolve will not run if there are already events (or a file ‘`reSolve_main.out.dat`’) in the specified folder. Therefore to re-run into the same folder, first either move or delete the event files so the working directory for the output (specified in ‘`workdir/`’ in the input file) is empty before running.

3.2. Input File

Sample input files are provided with the program, including all those used for figures and results in this paper, these are summarised in Appendix A.3. These are found in the directory labelled `input`, figure 1 shows the input file `Diphoton_NNLO_test_1.dat`. There are a number of different options within the input file and these are split into sections:

1. **Basic** - The first section, labelled “Basic” at the top of the input files includes general input. These are the `process`, which is 1 for diphoton resummation or 2 for Drell-Yan resummation - the only processes so far incorporated. If Drell-Yan is selected then one must also choose the specific Drell-Yan process via the `DYprocess` flag; 1 = W^+ , 2 = W^- , 3 = W^\pm , 4 = Z only, 5 = Z/γ^* . In addition, the `DYnarrowwidthapprox` flag allows the calculation of the on-shell only cross-section if it is set to 1. There is the `order` flag to indicate the order of the calculation (leading order (0), next-to-leading order (1) or next-to-next-to-leading order (2)) and the `resum_flag` to turn the resummation on(1) or off(0), this is useful as when adding a process to the `reSolve` program the main addition required is a new Born cross-section, this can then be tested by setting `order:0` and `resum_flag:0` to ensure it recovers the well-known Born total cross-section for the given process. `pdf_flag` allows for the PDFs input into the program to be changed, currently it is set to 82 indicating the MSTW (Martin-Stirling-Thorne-Watt) 2008 set [19] at NNLO - this is the only set naturally incorporated into the program, the program is nonetheless setup to make this easy to change between PDF sets. 80 offers the LO MSTW set and 81 the NLO MSTW PDF set. The MSTW PDF sets are read from ‘`Grids/`’ in the main program directory. `CM.energy` indicates the collision energy of the protons/anti-protons in the centre of mass frame in GeV , `verbosity` sets the amount of information to be output to terminal, in general we recommend this be kept set to 1 and higher values only used for debugging; `ih1/2` indicate whether beam 1 and beam 2 are proton or anti-proton beams; `save_events` is set to 0 if only the total cross-section is required, however to produce differential cross-sections the events must be saved and therefore `save_events` should be set to 1 to indicate the events will be saved in “easy” form or alternatively to 2 which is a “pseudo-lhe” form. Both allow determination of the overall transverse momentum spectrum, as well as other differential observables, rather than just the overall cross-section which is naturally output; finally `workdir` sets the working directory the events will be output to.
2. **Scales** - This section sets the three scales involved in the resummation formalism. First there is the usual factorisation scale μ_F (μ_F) encapsulating scale dependence as a result of the factorisation of the input and output processes of the collision, this is the scale to which the PDFs are evolved. Second there is the usual renormalisation scale μ_R (μ_R) dependence arising from the scale at which the α_s coupling and the partonic differential cross-section are evaluated. Finally there is also the scale μ_S (μ_S), which arises as a result of the truncation of the resummed expression at a given perturbative order, parametrisng the ambiguity stemming from the precise definition of the logarithmic terms which are being resummed. The setting of the scales here is however somewhat complicated, should you wish to set μ_F , μ_R , μ_S directly to fixed values throughout the resummation this is done here, in that case one must ensure that also the flags `muR_flag` and `muF_flag` are set to 0. However, rather than fixed scales one can set the values of μ_F^2 , μ_R^2 , μ_S^2 to fixed fractions of the qq^2 invariant mass of each generated event, to do this set `muR_flag` and `muF_flag` to 1 and μ_R , μ_F to the desired fraction of qq ; the resummation scale μ_S will be set to half the renormalisation scale μ_R in this case, as is the convention. In the input file in figure 1, the flags `muR_flag` and `muF_flag` are set to 1 and 0 respectively, therefore $\mu_R = qq$ and $\mu_S = \frac{qq}{2}$ whilst μ_F is fixed at $\mu_F = 113GeV$. Here one may also specify the parameter `mu_min`, which is the starting minimum scale from which the PDF fit factorisation scales are calculated, see Section 3.3 for more information.
3. **Integration** - This section deals with the inputs specific to the Monte Carlo phase space integration. `maxeval`, `nstart` and `nincrease` correspond to the approximate maximum number of Monte Carlo evaluations to perform total across all iterations¹, the number of evaluations to perform in the first iteration, and the number

¹This is not true in the `k Vegas` parallelised case, as described in Section 3.4, here the number of total iterations is set in this case via the number of iterations desired per core.

of additional evaluations in each subsequent iteration. Therefore the number of evaluations in the n th iteration is given by $n_{eval} = n_{start} + (n - 1)n_{increase}$ and the total number of evaluations across N iterations is $n_{tot} = Nn_{start} + \frac{1}{2}N(N - 1)n_{increase}$. The program will stop after a whole number of iterations once this number of evaluations n_{tot} exceeds `maxeval`. The `integrator_flag` determines which integrator algorithm is used. For the moment there are two possibilities, our own internal `k_vegas` Monte Carlo implementation is used for `integrator_flag= 1` or an external Cuba Monte Carlo implementation [18] is used for `integrator_flag= 2`. The internal `k_vegas` implementation is based on the original Lepage Vegas algorithm [20]. The interface and calling of Cuba will then be done automatically by the program once the directory to the extra Cuba integrator is provided in the `reSolve` makefile. Instead one may set `integrator_flag= 0`, events will then be read in from an event file with name given by the input variable “`randoms_file`”, this is useful for debugging or comparison with other programs. Note Cuba will automatically parallelise over the number of cores in the computer used whilst `k_vegas` will not. `multi_machine` sets whether you wish to use parallelisation with `k_vegas`, with 0 indicating not and 1 indicating parallelisation, it allows you to run different batches on different cores/computers and combine them all each iteration. Again, more information on how to perform parallelisation with the `k_vegas` integrator is given in Section 3.4. `seed` is used to set the seed for the randoms used for the Monte Carlo phase space integration by `k_vegas` or Cuba. This can be used to set the seeds for the randoms for the Monte Carlo integration based on time (-1), a random repeatable set of uniformly distributed seeds (0) or, if one is using parallelisation with `k_vegas`, one can set -2 to ensure each batch has a different seed, here the seed is set based on the machine tag but is deterministic and repeatable. This is as opposed to setting the seed flag to -1 to set the seed based on time, which will also ensure different randoms in each machine batch, but of course in a non-repeatable manner.

4. Resummation - Here various settings for the general resummation implementation are set, these are process-independent. The maximum and minimum values of the invariant mass squared qq^2 , transverse momentum squared q_T^2 and rapidity η , are all set here via `QQ_Max`, `QQ_Min`, `QT_Max`, `QT_Min`, `eta_Max` and `eta_Min`. `gqnp` and `ggnp` are factors to account for non-perturbative corrections to the sudakovs, they factor in uncertainty from very low $q_T \sim \Lambda_{QCD}$ and are given in equation 34 in section 5.2. In addition, here one may set the variables `en_sec_multiplier` and `PDF_fussiness` which are related to the PDF fit files used. More detail is given in Section 3.3. Meanwhile further PDF fit options are available here; `PDF_fitonly` can be set to 1 to allow one to run the code just to obtain the PDF fit file. This can be useful if running parallel batches, if you start without the PDF fit file here all batches will attempt to evaluate it and this will, at best, increase the time taken to run the code (as it will wait for the slowest core to complete the fit). At worst it could lead to inconsistencies in the fit used.
5. Process Inputs - Penultimately, there are the process-specific inputs, those specific to the diphoton/Drell-Yan process.
 - (a) Diphoton - We first describe the diphoton process inputs. These include `boxflag` which allows the user to include (1) or not (0) the $gg \rightarrow \gamma\gamma$ box diagram (see figure 2), or even to only have this contribution to the process (`boxflag= 2`). There are then the diphoton cuts, these are: `crack1/2` which indicate if a crack in the rapidity sensitivity of the detector is present (often 1.37 to 1.56 for the LHC); `etaCut` which should be less than or equal to $Min(|eta_Min|, |eta_Max|)$; and `pT1cut` and `pT2cut`, which cut based on the q_T of the photons. It is required that the larger of the two photons’ transverse momenta is larger than `pT1cut`, i.e. $Max(q_T^{\gamma_1}, q_T^{\gamma_2}) > pT1cut$, and that the smaller of the two photons’ transverse momenta is larger than `pT2cut`, i.e. $Min(q_T^{\gamma_1}, q_T^{\gamma_2}) > pT2cut$. Finally `Rcut` is a cut placed on the opening angle of the two photons produced as two highly collimated photons may not be resolved experimentally, we require:
$$\Delta_R = \sqrt{(\eta_1 - \eta_2)^2 + (\phi_1 - \phi_2)^2} > Rcut.$$
 - (b) Drell-Yan - These process specific inputs similarly detail possible cuts required. First there are the usual general cuts described for the diphoton section - `crack1/2` and `pT1/2cut`. Then there are specific cuts for different Drell-Yan processes, `eta1cut` and `eta2cut` cut on the rapidity of the produced Drell-Yan leptons for neutral current Drell-Yan; in the case of charged current Drell-Yan the observables are different as the (anti-)neutrino is not observed, therefore the standard cuts are `etaecut` - the rapidity of the produced charged lepton, `pTecut` - the transverse momentum of the charged lepton produced, `pTmisscut` -

the missing transverse momentum of the event (assumed to be from the (anti-)neutrino), and `tmasscut` the transverse mass of the event as defined by $m_T = 2(|p_{T1}||p_{T2}| - p_{T1} \cdot p_{T2})$. The cut requirements are $|\eta_e| < \text{etaecut}$, $p_T^{(\nu)} > p_{T\text{misscut}}$, $p_T^{e^+} > p_{T\text{ecut}}$ and $m_T > \text{tmasscut}$.

6. Histogram - Finally, the input file may be supplemented with inputs to the histogrammer, which is now incorporated within reSolve and can determine the histogram files directly. For each differential cross-section desired, one line must be added of the form “histo: {variable} {additional info}” where the variables available are “qT”, “qq”, “eta”, “mT”, “pTmin” or “pTmax”; the additional information is either the number of bins followed by the lower bound of the first bin and the upper bound of the final bin, or the endpoints of each bin (to allow non-uniformly distributed bins). In addition, the histogrammer can also be used alone, without re-running the whole resummation, provided the user has the event files (from a previous run). In order to run the histogrammer alone include the line `hist_only:1`, set the usual `workdir` input line to the working directory of the event files to use for the histogrammer, and include the `histo` lines relevant for the histograms you wish to calculate. More information is given in Section 3.6.

3.3. Pdf Fits

In order to implement the b-space resummation formalism of [15] (and references therein), the PDFs must be defined in Mellin space at generic complex values, which requires that we perform an analytic fit for the PDFs (as a function of x , at fixed scale). This PDF fit is done at the very start of the program once and for all, before the integration. If a fixed scale or a narrow range of invariant masses are used, a PDF fit at a single scale is satisfactory, indeed this is what is traditionally done in such resummation programs. In this case, the single fit is done at the input fixed factorisation scale μ_F at a momentum fraction set via $(QQ_Max/CM_energy)^2$. However, in cases where the invariant mass squared region considered is broad and one has a dynamical factorisation scale, then one may wish to improve precision by running with multiple PDF fits at various scales between QQ_Min and QQ_Max . In order to do this set `muF_flag` and `mu_F` to 1 and set the variable `en_sec_multiplier` accordingly. It should be noted however that the first version of the program is currently significantly slower for multiple PDF fits, therefore we only recommend using this option in particular cases where very wide QQ regions are used or if very accurate predictions are required. This will be optimised in future versions. The energy scale at which the PDF fits are performed is determined by the invariant mass range and the `en_sec_multiplier` variable. The starting scale is taken as QQ_Min , but this can be reset by setting `mu_min` in the input file in the resummation part of the file, the program starts with the scale `Q.start= min(mu_min, QQ_min)`. By default the value of `mu_min` is 20GeV . The setting of the scales for the multiple PDF fits used is as follows, we denote `en_sec_multiplier` by E_n :

1. To determine the first value of the factorisation scale at which a PDF fit is performed, reSolve calculates $QQ_temp = Q_start \times E_n$, provided this is less than the QQ_Max then the scale the first PDF fit is performed at is then $QQ_Min \times \sqrt{E_n}$, i.e. the geometric mean of QQ_Min and the new scale QQ_temp . The program will then go on to perform another PDF fit, see step 3.
2. If however $QQ_temp > QQ_Max$ then reSolve performs just one fit at the geometric mean of the endpoints of the invariant mass range, $\mu_F = \sqrt{QQ_Min \times QQ_Max}$.
3. If in the previous steps, $QQ_temp < QQ_Max$ then reSolve will perform a PDF fit at a further scale. Once more, the program will take the scale of the previous fit and multiply it by E_n giving a new $QQ_tempnew = QQ_temp \times E_n$ and compare this new scale (which in the second fit case would now be $QQ_Min \times E_n^2$) with QQ_Max . Again if the new scale $QQ_tempnew < QQ_Max$ then the PDF fit will be performed at the new $QQ_tempnew$ scale, i.e. at $\mu_F = \sqrt{QQ_temp \times QQ_tempnew}$. $QQ_tempnew$ then becomes QQ_temp and we repeat this step 3 until $QQ_tempnew > QQ_Max$. Once $QQ_tempnew > QQ_Max$ then the fit is the last fit and is performed at $\mu_F = \sqrt{QQ_temp \times QQ_Max}$.

For example, with $QQ_Min = 80\text{GeV}$, $QQ_Max = 200\text{GeV}$ and $E_n = 1.5$, the first fit would be performed at $\sqrt{80 \times 120} = 97.98\text{GeV}$, $120\text{GeV} < QQ_Max$ so a second fit is performed. For this second fit, $QQ_temp = 120 \times 1.5 = 180\text{GeV}$, which is less than QQ_Max still so a third fit will be performed and the second fit is performed

1) Basic

2) Scales

3) Integration

4) Resummation

5) Process input
– Diphoton here

6) Histograms

```

# Basic
process: 1
resum_flag: 1
order: 2
pdf_flag: 82
CM_energy: 14000.
verbosity: 1 !Amount of output text
ih1 1 ! hadron type for beam1 (1=proton -1=antiproton)
ih2 1 ! hadron type for beam2 (1=proton -1=antiproton)
save_events: 1
workdir: Diphoton_NNLO_test1/
# Scales
mu_S: 1. ! resummation scale (GeV)
mu_R: 1. ! renormisation scale (GeV)
muR_flag: 1
mu_F: 113. ! factorisation scale (GeV)
muF_flag: 0
# Integration
maxeval: 5000 ! Approximate max number of integral iterations
nstart: 1000 ! Number of integral iterations in first evaluation
nincrease: 1000 ! Number of extra integral iterations in each further evaluation
integrator_flag: 2 !1=default, 2=cuba
multi_machine: 0
seed: 0
# Resummation
#pdf_fussiness=0.5
#en_sec_multiplier=2
gqnp: 2.666666667
gqnp: 6.0
QQ_Min: 80.
QQ_Max: 160.
QT_Min: 0.
QT_Max: 120.
eta_Min: -2.5
eta_Max: 2.5
# Diphoton
boxflag: 0 ! whether to include the gg->gamma gamma Box (0 = no, 1 = yes, 2 = Box only)
etaCut: 2.5
crack1: 1.37 ! etacut crack1
crack2: 1.37 ! etacut crack2
pT1cut: 40 !
pT2cut: 25 !
Rcut: 0.4 ! Rcut
#Histograms
histo: qT 0 0. | 1. 2. 3. 4. 6. 8. 10. 14. 18. 22. 28. 35. 45. 55. 70. 85. 100.
histo: qq 100 60. 180.
histo: eta 20 -2.5 2.5
histo: pTmin 100 20. 100.
histo: pTmax 100 20. 100.
histo: mT 100 0. 200.

```

Figure 1: Input file for reSolve, this file shown is the example file found in `input/Diphoton_NNLO_test_1.dat`. It is split up into appropriate sections - Basic, Scales, Integration, Resummation, Process-specific input (Diphoton in this case), and Histograms.

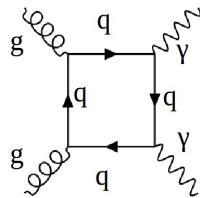


Figure 2: gluon gluon box diagram contribution to diphoton production

at $\sqrt{120 \times 180} = 146.97\text{GeV}$. For the third fit, $QQ_tempnew = 180 \times 1.5 = 270 > QQ_Max$, therefore this is the final fit and the fit is now performed at $\sqrt{QQ_tempprev \times QQ_Max} = \sqrt{180 \times 200} = 189.73\text{GeV}$.

PDF_fussiness allows nearby PDF fits to be used, for example setting it to 0.02 will ensure that PDF fits made at a scale within 2% of the desired μ_F are used rather than a timely, completely new fit being performed. By default PDF_fussiness will be set to 0.01 if no input is provided.

For validation plots for running the reSolve program with multiple PDF fits please see Section 6.3.

3.4. Parallelisation

With the input `integrator_flag`, the option to use either the built-in `k_vegas` Monte Carlo integration (1) or the external Cuba Monte Carlo integration (2) are possible. Whilst the Cuba implementation will by default parallelise over the number of cores of the machine used, the `k_vegas` implementation will automatically only use one core. However, the reSolve program has been designed to allow multiple batches of events to be run on different cores, and indeed on different machines, and then combined after each iteration before again being distributed across the machines and cores available. This can therefore be used to parallelise across all the cores in the machine used, like Cuba, or even to run across multiple machines. This can hasten the process of producing the events required for differential cross-section spectra greatly, depending upon the computer resources available.

In order to run batches in parallel across multiple machines/cores, one must first turn the `multi_machine` input to 1, as described in Section 3.2. In addition, in order to avoid producing the same events across multiple cores, one must set `seed` to -2, to allow the randoms' seeds to be set by converting the machine tag into a numerical seed different for each batch (in a deterministic and repeatable manner), or to -1, to set the randoms' seed based on time - which will be marginally different for the batch sent to each machine. For parallel running using `k_vegas`, unlike the standard running or Cuba running, the maximum number of iterations is not set by `maxeval` in the input file, rather it is set at the start of the parallelisation script in the variable `max_iter`, this is the number of iterations to run per core. The number of evaluations (phase space points) per iteration per core are set as usual via `nstart` and `nincrease` in the input file. The number of cores per machine is set via `max_cores` at the start of the parallelisation script. In order to parallelise across all the cores of just one machine, use the built in file `single_machine_parallel`. To run this, type into the terminal the call shown in figure 3.

```
single_machine_parallel {path to input file}
```

Figure 3: Terminal call to the `single_machine_parallel` file used to parallelise the in-built `k_vegas` vegas Monte Carlo implementation across all the cores of a computer, if this call does not work the machine used may require the user to prefix the call with `bash -x` in order to allow the terminal to determine it's a bash script. To run this, `integrator_flag` and `multi_machine` must both be set to 1 in the input file. Alternatively, the integrator Cuba will automatically parallelise across the cores of the machine, this requires `integrator_flag = 2`.

It is important to note that for parallel runs, in the case where one uses the `k_vegas` integrator, whether across the cores of one machine or across many machines, the numbers of integration evaluations at the start and the increase in the number of evaluations from one iteration to the next; `nstart` and `nincrease`; are then the numbers per core. Therefore each core used in parallel will, in total, undertake $n_{tot_per_core}$ evaluations, for a total of n_{tot} phase space points across all cores, given below in equations 1 and 2. In running the `single_machine_parallel` parallelisation script, the working directory for the input file used will be filled with event files of the form `events_lhe_{iter_number}.lhe`, each containing all the events from all the cores concatenated for that given iteration (lh here indicates the type of event output selected - `save_events` set to 2 in the input file). In addition, there are `reSolve_main_out_{core_number}.dat` files giving the final iteration result and accumulated results for the total cross-section for the specified core. Finally, the `reSolve_main_out_END_ITER.dat` file lists the total accumulated results across all cores for the total cross-section. Meanwhile, in the overall directory of the reSolve program, `nohup` files are produced, the files `nohup_{core_number}.out` contain the inputs and results for each core, the `nohup_iter_{iter_number}.out` files contain the Monte Carlo grids produced from each iteration.

$$n_{tot_per_core} = [\max_iter * nstart + 0.5 * n.increase * \max_iter * (\max_iter - 1)] \quad (1)$$

$$n_{tot} = \max_cores * [\max_iter * nstart + 0.5 * n.increase * \max_iter * (\max_iter - 1)]. \quad (2)$$

If instead of running across all the cores of one machine, one instead wants to reduce the runtime even further by parallelising across many machines, one must use the integrator option `k_vegas` - setting `integrator_flag` to 1 in the input file. Again `multi_machine` must also be 1. With these settings, one may then use the script `multi_machine_parallel_local` to undertake parallel runs across many computers, this file is shown in figure 4. This parallelisation script allows the use of multiple machines all of which are accessible via ssh on systems with or without a shared home directory in which to access input files and to output results to. The user must change the `exedir` line to the directory in which the reSolve program is installed on your machine, unless you are running the script from the reSolve working directory. To enter which machines you wish to run on, enter the machine names into `machines` at the top of the program. After saving the script, simply typing “`multi_machine_parallel_local {path to input file}`” into the terminal will set off a parallel run across the specified number of cores of all named machines, some machines may require this to be prefixed by `bash -x` to indicate the script is a bash script. The information is then combined at the end of each iteration to update the grid, before using all machine cores for the subsequent iteration, this continues until all iterations are complete. The maximum number of iterations to be performed is set at the top of the file, as is the number of cores to use per machine, in `max_iter` and `max_cores` respectively, these were also set at the top of the `single_machine_parallel` script. In the same way as was the case for the single machine parallelisation, in the multiple machine parallelisation the `max_iter` variable is used to set the number of iterations, with the `maxeval` input used for single core `k_vegas` running or Cuba running not relevant for `k_vegas` parallelisation. In the working directory for the input file, a file of the form `reSolve_main_out_{machine_name}_{core_number}.dat` is created for each core on each machine used, listing the overall total cross-section for that machine core for both the final iteration and the accumulated results across all iterations, meanwhile `reSolve_main_out_END_ITER.dat` lists the combined total cross-section across all machines, all cores for the final iteration and then the accumulated result across all machines, all cores and all iterations, an example of this file is given in figure 5. The output event files are also output into the input file working directory. In addition, `nohup_{machine_name}_{iteration_number}.out` files are created in the overall code working directory (one level above that of the input file’s working directory), which contain all the usual terminal output for each core on each machine used. A schematic of a parallel run across many machines is given in figure 6. In parallel, histogram data files will be automatically generated by the reSolve program as usual, nonetheless should the user wish to redo them, they can be re-binned or completely redone using the “`hist_only`” mode in exactly the same way as with running on one core or with Cuba across one machine, see Section 3.6. A description of the time taken to run in parallel compared with on one core is given in Section 6.4.

Sample input files which work with parallelisation, either across many cores of one machine, or across many ssh-accessible cores, are included with the reSolve program. These sample parallelisable input files are in the input directory and called `Diphoton_Born_parallel_LHC.dat`, `yZ_Born_Tevatron_parallel.dat`, `Diphoton_NNLO_test_1_parallel.dat` and `yZ_NNLO_Tevatron_parallel.dat`. These are the same setups of the `Diphoton_Born_LHC.dat`, `yZ_Born_Tevatron.dat`, `Diphoton_NNLO_test_1.dat` and `yZ_NNLO_Tevatron.dat` files (used in the validation of the reSolve program in Sections 6.1.1, 6.2.1, 6.1.2 and 6.2.3 respectively) except adapted for `k_vegas` parallelisation.

In general, the parallelisation needed depends on the structure of the user’s computer network, this varies significantly from one user to another therefore the user may have to make small changes to the scripts as appropriate for their computer resources.

```
#!/bin/bash

# Multi-machine parallel launcher - via ssh - for reSolve using k_vegas
version=0.2.2

#### HARDCODED INPUTS ####
machines="endear encumber"
max_iter=10
max_cores=4
exedir="$PWD"
EXE="./reSolve.out"
#### END HARDCODED INPUTS ####
echo
echo "#####"
echo "Multi-machine parallel launcher for reSolve + k_vegas -- version $version"
echo "Execution can be regulated by several parameter -- edit the script to see"
echo "#####"
echo

if [ $# -eq 0 ] ; then
  echo "Usage: $0 INPUT FILE (with full path)"
  exit
fi
infile=$1
if [ $# -gt 1 ] ; then
  echo
  echo "WARNING - the script only accepts one command argument, a reSolve input file. Parameters beyond the first are being ignored."
  echo
fi

echo "#####"
echo "TAKE CARE: the \"machines\" string MUST be edited by hand!"
echo "Current value of string: $machines"
echo "#####"
echo

echo "Requested iterations: $max_iter"
echo
iter=1
while [ $iter -le $max_iter ] ; do

echo "Iteration $iter"

# Machines loop
PIDS=""
for pc in $machines ; do
  loc_cores='ssh $pc nproc --all 2>/dev/null'
  cores=$(( loc_cores > max_cores ? max_cores : loc_cores ))
  echo "Launching process in $pc on $cores cores"
# Cores loop
  ii=1
  while [ $ii -le $cores ] ; do
#    echo $ii
    tag=${pc:0:10}_$ii
    echo $tag
    nohup ssh $pc "cd ${exedir}; $EXE ${infile} $tag" >nohup_${tag}_${iter}.out 2>/dev/null &
    PIDS="$PIDS $!"
    ii=$(( ii+1 ))
  done
  last_node=$pc
done

echo "Waiting for child processes..."
wait $PIDS

# ITERATION END
echo "Finalising iteration"
nohup ssh ${last_node} "cd ${exedir}; $EXE ${infile} END_ITER" >nohup_iter_${iter}.out 2>/dev/null &
LASTPID=$!
wait $LASTPID

#
iter=$(( iter+1 ))
echo "Iteration over"
echo

done

echo "That should be it"

exit
```

Figure 4: The parallelisation program `multi_machine_parallel_local` for running across multiple machines, it is included with the program. To use this both `integrator_flag` and `multi_machine` must be set to 1 in the input file.

```

This is reSolve main out file
Integration by Vegas
----
Iteration no. 2      integral = 103.43
neval      = 120000    std dev = 0.939635
Accumulated results: integral = 103.43
                                std dev = 0.664422
                                chi**2 per it'n = 0

```

Figure 5: The accumulated results outputted into the file “reSolve_main_out_END_ITER.dat”, here the results are after two iterations on each of 3 cores across 2 different machines. These results were for the input file `yZ_Born_Tevatron_parallel.dat` used with the `multi_machine_parallel_local` script, this input file is the Z/γ^* setup listed later in table 2, with `multi_machine` and `integrator_flag` set to 1 to allow the `k_vegas` multiple machine parallelisation and with the iteration numbers in the input file altered to `nstart= 5000` and `nincrease= 15000`.

3.5. Output

The events produced by reSolve will be saved provided the flag “`save_events`” is set to 1 for the “easy” form or 2 for the “pseudo-lhe” form, contrastingly, if it is set to 0 only the total cross-section is output and is in the `reSolve_main_out.dat` file. Events are required if differential cross-sections are desired rather than just the total cross section, by default events are output and will be output in the “easy” form into your `workdir`, split into a different file for each iteration of the program. The events will all be automatically read by the histogrammer to determine the histograms specified in the input file. A sample output event file (in the “easy” form) is shown in Figure 7. Each event details the 4-momenta of the incoming partons, the 2 outgoing particle 4-momenta, the random values used to define the phase space point by the Monte Carlo and finally the event cross-section (in GeV) and event weight.

As for the output to terminal, with verbosity set to 1 the program will only output the result after each iteration. Higher verbosity settings provide more information but the primary use of such information is debugging and will slow down the program as many values may be output to terminal each iteration. Figure 8 gives the form of the output after each iteration.

The cross-section estimate given by the Monte Carlo iteration on iteration is $\sum_i \frac{1}{N} w_i f_i$, where w_i is the weight of the event, N is the total number of events and f_i is the cross-section estimate for that given phase space point. The error estimate meanwhile is given by the square root of the standard variance estimate in (3). The factor of $N - 1$ ensures we obtain the variance of the mean rather than of the f_i evaluations.

$$Var = \frac{1}{N-1} \sum_i [(f_i^2 w_i^2) - (f_i w_i)^2] \quad (3)$$

The Monte Carlo cross-section iterations and their error estimates are then combined by weighted average, where the weights are the inverse of the variance estimates.

The exact same process is carried out in the histogrammer included with the program, except the weighted averaging is now done bin by bin in order to produce a fully differential cross-section, in whichever observables are required, rather than just the total cross-section. The error given on the cross-section is consequently only approximate and should only be used as a judgement after a couple of iterations. After each iteration the grid used to weight the Monte Carlo events is updated, the chi-squared gives an indication of how well the grid approximates the integral, this is estimated via the difference between the weighted events and the cross-section estimate at that iteration (the mean), weighted according to the variance. This chi-squared should be divided by the number of degrees of freedom (which is equal to the iteration number minus 1) to understand how good the estimate is. This is produced as described in [20] and in the Cuba package [18]. The chi squared divided by the number of degrees in freedom (shown in brackets) should therefore decline iteration on iteration, again after the first few iterations are passed.

The overall accumulated results across all iterations are also output into the file ‘`reSolve_main_out.dat`’, or the file ‘`reSolve_main_out_END_ITER.dat`’ if ran across many machines.

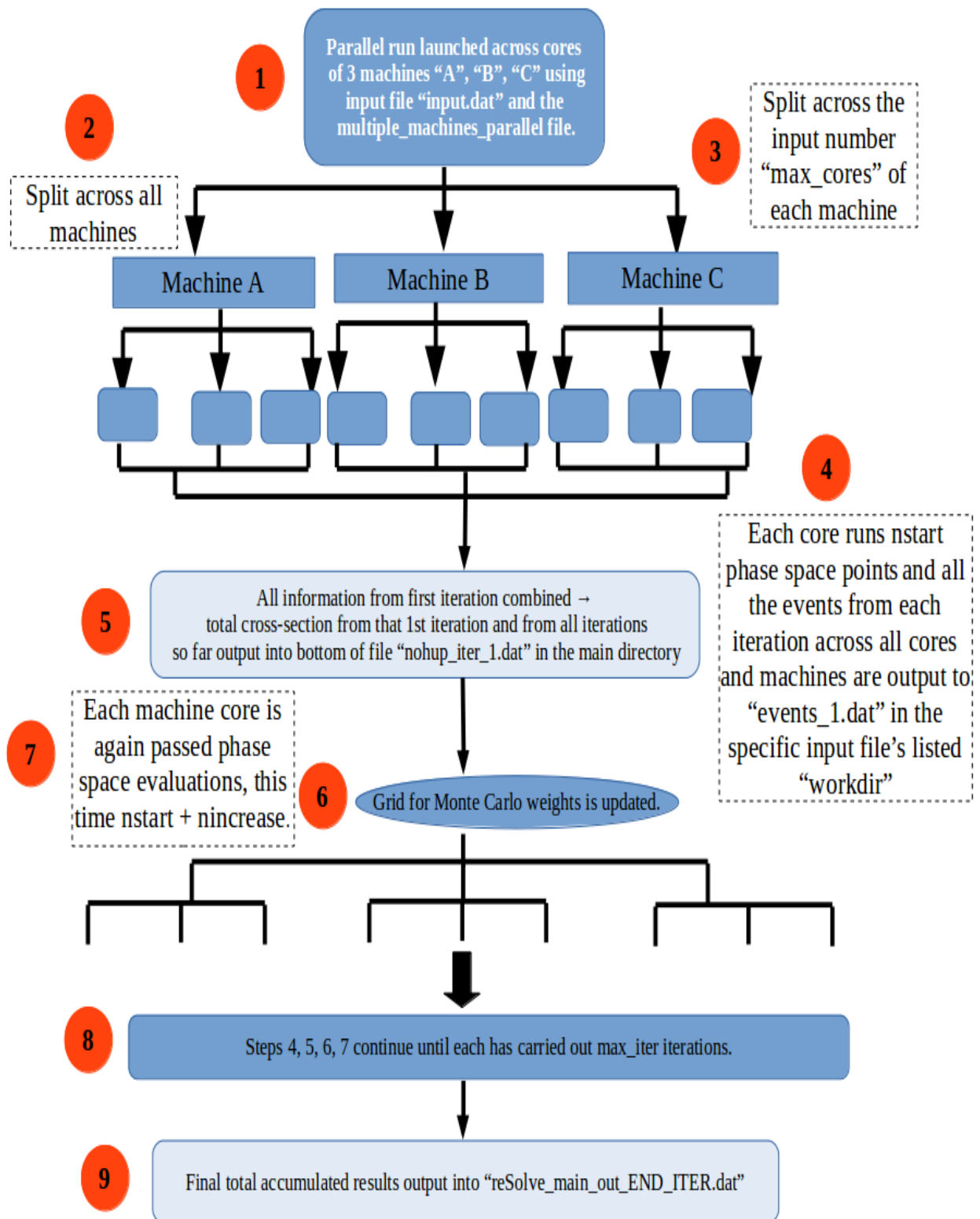


Figure 6: Flowchart demonstrating the running of a parallel run across the cores of many machines, the script `multi_machine_parallel_local` may be used to perform this.

```

events_10_22260.dat - Mousepad
File Edit View Text Document Navigation Help
221.685 43.7531 0 217.324
53.2027 43.7531 0 30.2688
182.884 95.4586 -2.64089 155.971
92.0041 -7.95242 2.64089 91.6217
0.0106011 0.531759 0.79519 0.990827 0.989647
0 2.21725e-06

34.1559 9.8524 0 32.7041
50.8687 9.8524 0 -49.9054
46.2087 44.4206 0.273046 12.7273
38.8158 -24.7158 -0.273046 -29.9287
0.00755263 0.0269638 0.458972 0.639316 0.00127327
0 3.85304e-07

293.296 45.8737 0 289.686
51.807 45.8737 0 24.0743
327.022 74.2908 0.0136457 318.471
18.0811 17.4566 -0.0136457 -4.7111
0.303792 0.584554 0.804554 0.0308931 0.49919
0 3.57263e-07

97.766 3.49978 0 97.7034
44.1473 3.49978 0 -44.0083
75.5124 -4.46644 -63.9378 39.9265
66.401 11.466 63.9378 13.7686
0.562877 0.00340235 0.57963 0.881055 0.729986
268.613 2.59316e-07

```

Figure 7: Output file of reSolve, this file is in the “easy” form. In this form each event is represented by six rows of information and followed by a blank row. The first two rows are the 4-momenta of the incoming partons, the next two rows are the outgoing photon four-momenta, the fifth row is then the set of 5 random values between 0 and 1 used to generate this event and its momenta. These randoms set the invariant mass squared - qq^2 , the transverse momentum squared - q_T^2 , the rapidity - η , and the θ and ϕ opening angles of the two photons in the diphoton centre of mass frame. Finally the last row gives the value of the cross-section here (in GeV) (0 if cut) and the weight of the event in the Monte Carlo.

```

Iteration 2: 30000 integrand evaluations so far
[1] 3.90839 +- 0.612087 chisq 0.0447161 (1 df)

```

Figure 8: Terminal output of reSolve after its second iteration. It lists the iteration number, the total number of integrand evaluations carried out, the cross-section in GeV with an error, and the chi-squared with number of degrees of freedom. If parallelisation and `k vegas` is used such information appears in output files after each iteration, rather than in the terminal.

3.6. Histogrammer

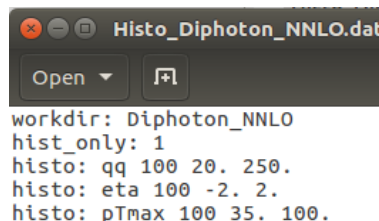
The program package also includes a histogrammer package, this can be used to read in the events produced by reSolve to produce the necessary differential cross-section invariant mass, transverse momentum or other differential cross-section histogram data.

The histogrammer has now been incorporated into the program in order to simplify the use of reSolve. With the histogrammer incorporated into the reSolve program, the user must simply include a few lines in the usual reSolve input file detailing the desired histograms and the binning. In the section “Histograms”, add a line of the form given in Figure 9, begin by indicating this is histogrammer input with `histo` and then follow by the variable for the histogrammer, the number of bins required and the start bin lower bound and final bin upper bound. This will then calculate the events in each bin for the number given of evenly spaced bins across the range specified. The option `{variable}` can be `qT`, `qq`, `eta`, `mT`, `pTmin` or `pTmax` for the transverse momentum spectrum, invariant mass spectrum, rapidity distribution, transverse mass distribution and distribution of the minimum/maximum transverse momentum of the (two) outgoing particles respectively. If, rather than evenly spaced bins, the user requires unevenly distributed bins, enter a “0.” where the number of bins is input, and instead proceed by entering the endpoints of every bin, this is useful in allowing finer bin spacings at the lower end of the transverse momentum spectrum, where resummation

is crucial. Histogram information will only be calculated for each variable specified, therefore if no lines of the form given in Figure 9 are included in the input file, the reSolve program will produce the events and total cross-section only. Of course, as had to be done in the previous beta version of reSolve, these events can then be used to determine the differential cross-sections required via a standalone histogram calculation, the histogrammer can be used without performing the whole resummation calculation again provided there are the required event files from previous runs of reSolve. In order to run just the histogrammer, for example should the user wish to re-bin the events or determine further histograms not initially specified in the input file, one must include the `hist_only` flag set to 1 (if it is not present it is set to 0 by default). In this case the resummation calculation and event generation will all be skipped and instead reSolve will read the event files specified in `workdir` and determine the histograms specified by the “`histo: {variable} {binning info}`” input lines. Figure 10 provides a sample input file where only the histogramming is done. It is important to note that the histogrammer produces the cross-section in each bin normalised by the bin width (rather than the total cross-section in the bin), this ensures the amplitude is independent of the binning used.

```
histo: {variable} {no. of bins} {start bin lower end} {final bin upper end}
```

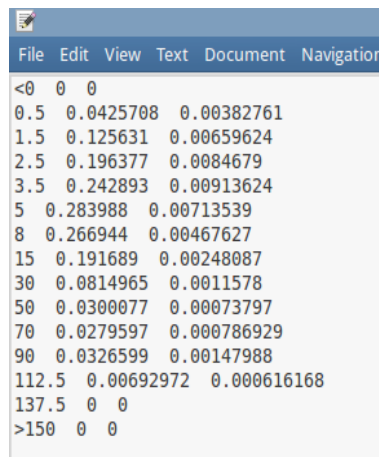
Figure 9: Line required for the histogrammer in the reSolve input file.



```
Histo_Diphoton_NNLO.dat
workdir: Diphoton_NNLO
hist_only: 1
histo: qq 100 20. 250.
histo: eta 100 -2. 2.
histo: pTmax 100 35. 100.
```

Figure 10: Sample input file to run reSolve so that it only does the histogram calculation, reading in previously generated events, here from the working directory `Diphoton_NNLO/` and determining the invariant mass (qq), rapidity (η) and maximum transverse momentum (pT_{max}) differential cross-section histogram data.

Once the program has run, it will then produce all the events as before (provided the `save_events` flag is set to 1 or 2 not 0, and that `hist_only` mode is not being run) in addition to output files detailing the histogram plotting information for each of the desired differential distributions. With all of the above variables specified, the reSolve program will produce output files “`histo_0_qT.dat`”, “`histo_1_qq.dat`”, “`histo_2_eta.dat`”, “`histo_3_mT.dat`”, “`histo_4_pTmin.dat`”, “`histo_5_pTmax.dat`”. Each of these files lists the centre-points of each bin and the corresponding cross-section in that bin followed by a “0” column (the error in the position of the bin) and then an estimate of the error in that bin. A sample output file for the qT differential cross-section is given in figure 11. In addition, for all of the validation and results figures in this paper in Section 6, the histogram data used is provided with the paper.



```

<0 0 0
0.5 0.0425708 0.00382761
1.5 0.125631 0.00659624
2.5 0.196377 0.0084679
3.5 0.242893 0.00913624
5 0.283988 0.00713539
8 0.266944 0.00467627
15 0.191689 0.00248087
30 0.0814965 0.0011578
50 0.0300077 0.00073797
70 0.0279597 0.000786929
90 0.0326599 0.00147988
112.5 0.00692972 0.000616168
137.5 0 0
>150 0 0

```

Figure 11: Output file for q_T from the histogrammer included with the program.

4. Program Structure

The program structure is intended to be modular, with the different aspects of the calculation divided into different folders, see Figure 12. In principle several of these - Histogrammer, Integral, PDFfit, Resummation and Utility - can be used independently of the main code, this is important for the straightforward extension of the program to additional processes. The details of the calculations in each folder are detailed below in Section 4.1 and following this, an explanation of how the program works is given in Section 4.2. This is intended to allow users to understand how the program functions in order to both simplify its use and enable users to add their own processes should they wish.

4.1. Modular Structure

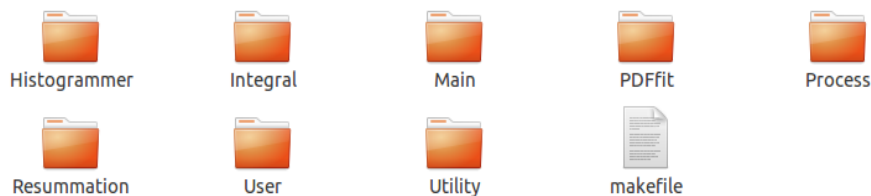


Figure 12: The folder structure of the reSolve code. It is modular, splitting up the different parts of the calculation.

- **Histogrammer** - This contains the files required to read in the event files generated by reSolve in the `workdir` and calculate the cross-section per bin width in the desired differential cross-sections. The histogrammer can also be used on previously generated events, without re-running the whole resummation calculation, by using the `hist_only` flag.
- **Integral** - This contains the files necessary to perform the Monte Carlo phase space integration, either using the in-house `k_vegas` integrator or the external Cuba integrator, it also contains the Fortran file which determines the 5 random values for the Monte Carlo event generation. The file `events_readin.cc` can be used to read in events from other programs, its use is largely in debugging and can be ignored with the program running normally.
- **Main** - This contains all the general files used to read in the input file, perform pre-processing, post-processing and interface with the Monte Carlo integration routines.

- **PDFfit** - This contains the routines used to evaluate the fit parameters for the PDF fits and then to link to the Fortran routine used to evaluate the PDF fits, as well as to output them in a form useful for the rest of the program.
- **Process** - This folder contains any process specific code. So far the program only calculates the resummed part of the diphoton or Drell-Yan differential transverse momentum spectra. Each process folder includes a file routine to read the process specific parts of the input file. The file `{process}_ps.cc` uses the randoms generated to determine the phase space point for each event - i.e. the invariant mass squared, transverse momentum squared and rapidity of the diphoton/Drell-Yan system as well as the θ and ϕ angles in the centre of mass frame. These set the particle four-momenta. These variables then determine the θ angle of the diphoton/Drell-Yan system with respect to the incoming partons as well as the phase space Jacobian and the relevant s, t, u Mandelstam variables. A cut file is also included, called to determine if a point generated by the Monte Carlo passes the cuts or not and therefore whether it is evaluated. The process dependent parts of the hard function calculation are calculated in the `{process}_hard.cc` file, including the Born cross-section and $H_q^{\gamma\gamma(1)}$ and $H_q^{\gamma\gamma(2)}$ coefficients in Drell-Yan scheme, see section Appendix B equations B.37 and B.38. Finally there is an integrand file `{process}_integrand.cc` which organises these aspects of the calculation and calls the generic resummation code in order to determine the cross-section for each event. The output “dumper” routines are then called to output the event information.
- **Resummation** - This folder contains the resummation routines which are process independent, and therefore will also be called for other processes. This includes the inverse Fourier and inverse Mellin transforms and the process independent parts of the hard function calculation, as well as the Sudakov factors. This aspect of the calculation is all directed by calling the “resummed” routine in `inv_fourier.cc`. More details on how the resummation calculation is performed are given in section 4.2 and section 5. The hard functions calculated within the program for now include only those required for the diphoton background and Drell-Yan processes calculations. The hard factors for gluon-gluon initiated processes are only included at leading order currently. This is all that is required for NNLL resummation for the diphoton process and even this is not required for Drell-Yan. For diphoton background calculations, gluon-gluon initiated processes begin at two orders beyond the qq initiated processes (which are evaluated here at next-to-next-to-leading logarithm). For other processes, for example Higgs signals, the gluon gluon contributions will need to be added in up to NNLL, this is left to future versions of reSolve.
- **User** - This is a folder where the user can call additional routines they may write for pre-processing, Monte Carlo or post-processing, as well as to read process-specific input. Currently, it contains routines to allow parallelisation of the `k_vegas` integrator across multiple cores and multiple machines, as well as the switch between the interfaces to either read events into the integrator or evaluate the phase space integral using `k_vegas` or `Cuba`.
- **Utility** - This folder contains auxiliary functions necessary for the program; including the `alphaS.f` Fortran routine for α_s evolution, the output “dumper” routines to output the events used in “easy” or “pseudo-lhe” form, the routines used for the Lorentz algebra, and initialisation routines associated with the PDFs.

This program structure is designed to modularise the program, this enables the straightforward extension of the program into other processes, indeed we intend to perform this further ourselves in the future. In order to add a new process one must simply replace the process sub-folder with equivalent files for the new process. Depending on the process, additional hard factors may need to be added to the hard function calculation in the resummation and potentially also the different orders included for the qq and gg processes as their relative order contributions depend upon the process. Currently, qq hard factors are included up to NNLL, whilst gg hard factors are only up to LL, this will be resolved in future versions as we extend the program to additional processes. A detailed guide on how to add a process to the reSolve program is given in Section 4.3.

In addition to the code structure given here, the input folder in the main reSolve directory contains all the input files used for validations and results described later in this paper in Section 6, in addition to a folder for the events produced in running each of these input files. These setups and the results generated are described in detail in Section 6.

4.2. Program Flow

As detailed in section 5, the resummation formalism of [21] requires the determination of the following integrals; phase space integral - evaluated via Monte Carlo, inverse Fourier transform from impact parameter (b) space to transverse momentum (q_T) space, and an inverse Mellin transform from Mellin space (N_1, N_2) to momentum fraction space (z_1, z_2) for each of the incoming partons. Figure 13 shows in detail the flow of the calculations performed, starting at `main.cc`, figure 14 then illustrates in more detail the resummation aspects of the calculation. The colours of the boxes indicate where in the program structure the various files and routines lie, with the key also given in figure 13.

Our program `reSolve` works as follows, this description and the figures are for the diphoton process, but the exact same sequence occurs for the Drell-Yan processes with the appropriate process-specific files in the Process sub folder interchanged:

1. The calculation begins in `main.cc`, this calls `InputPars.cc` to read in the various inputs provided in the input file and described in Section 3.2. From here `user.cc` is called to carry out required processing before the Monte Carlo integration.
2. This pre-processing includes calling `resu_preproc.cc`, this carries out various initialisations including those of the inverse Fourier transform, N-independent resummation parameters (via `resu_procindep.cc`), inverse Mellin transform contour and N-dependent resummation parameters (via `mellinspace_fns.cc`), and others. It also calls the PDF fitting routines and calls `pdfmellin.cc` to convert the PDF fits into Mellin space.
3. `User.cc` then moves onto the Monte Carlo aspect of the program, calling `k_vegas_interface.cc` or `cuba_interface.cc` if the input `integrator_flag` is 1 or 2 respectively. These programs call the random generator and pass any points evaluated to the Monte Carlo evaluation programs - either the in-house `k_vegas` or the external `Cuba`.
4. The Monte Carlo interface programs themselves then call the process specific files - first `diphoton_integrand.cc`, this is the integrand of the Monte Carlo integral. It calls `diphoton_ps` to convert the randoms generated into a phase space point; determining qq^2 , q_T^2 , η , θ_{CM} and ϕ_{CM} , whilst also evaluating the Jacobian for the phase space integral.
5. `diphoton_integrand.cc` next calls `diphoton_cuts.cc`, this has the task of reading in the cut information from the input file and determining whether the phase space point being considered passes the cuts or not. Provided the cuts are passed, `diphoton_hard.cc` is called, this evaluates the Born cross-section and other process-specific resummation variables; such as the H_q^1 , H_q^2 , H_g^1 and H_g^2 hard factors in the Drell-Yan scheme.
6. `reSolve` now moves onto the general resummation routines as `diphoton_integrand.cc` calls resummed in `inv_fourier.cc`. This section of the calculation is shown in more detail in figure 14.
7. First in the resummation routines, `inv_fourier.cc` determines the correction factor required to account for the fact that the PDFs are fit, this is done by evaluating the LO cross-section directly with standard PDFs and then with the PDF fit and using the ratio as a correction factor. This is all determined in `xsection.cc`.
8. `invbtoqt` is now called. The role of this routine is to call `intde2.cc` to perform the inverse Fourier transform back from b space to q_T space. The integrand of this inverse Fourier transform is the routine `invres`.
9. `invbtoqt` calls `invres` for a number of different points in impact parameter space, usually of the order of 20, depending on the precise details of the convergence. For each b value, `invres` must evaluate the double inverse Mellin transform used to perform the resummation. This is done via the routine `inversemellin_resummed`.
10. `inversemellin_resummed` in `inv_mellin.cc` organises the double inverse Mellin transform calculation, this calculation is built directly into the code. Variables b_* (`bstar` in the code) and b_{log} (`blog` in the code) are introduced to regulate singularities seen at large and small b respectively, see section 5.2:

$$b_* = \frac{b}{\sqrt{1 + b^2/b_{lim}^2}}, \quad b_{log} = \log \left[1 + \frac{q^2 b_*^2}{b_0'^2} \right], \quad \text{with} \quad b_{lim} = \frac{b_0'}{q} \exp \left(\frac{1}{2\alpha_s \beta_0} \right) \quad \text{and} \quad b_0' = 2 \exp \left[\gamma \frac{q}{\mu_S} \right]$$

(notice that the b_{log} variable is just \tilde{L} as defined in eq. (31) with the replacement $b \rightarrow b_*$)

11. First, the Sudakov form factors for soft gluon and soft quark emission are calculated by calling `sudakov.cc`. Then `GetResuPars` determines the C_1 , C_2 , anomalous dimensions and other N-dependent basis functions in Mellin space and evolves them to the resummation scale μ_S .
12. `hard_fns.cc` next determines the hard factors, incorporating the virtual diagram contributions into the resummation.
13. The sudakovs, hard factors and appropriate weights are used at each of 40 – 88 points along the contour in Mellin space, with the number of points depending on the rapidities of the two photons, this is done for each inverse Mellin transform. Note this contour is in the complex plane in order to allow for faster convergence.
14. The contributions at each point are then summed along the contour to calculate the double inverse Mellin transform.
15. Putting all this information together gives the inverse Mellin transformations, if these are called for each of around 20 b values this allows the determination of the inverse Fourier transform for each phase space point. Repeating the process for $\mathcal{O}(10^5 - 10^7)$ randomly distributed phase space points and including the effects of the Jacobian transformation between the randoms space volume and the phase space volume, reSolve thereby determines the overall cross-section. This information is printed out after each iteration; meanwhile all the events, their individual cross-sections and the randoms associated with their phase space points are output into output files (one per iteration), this information can be used to re-create the phase space variables and so produce histograms of the differential cross-section in invariant mass (qq), transverse momentum (q_T), rapidity (η), transverse mass (m_T), or maximum and minimum transverse momenta for the two photons ($p_T^{\{max/min\}}$).

The Monte Carlo phase space integration works by selectively refining the grid from which it draws the randoms iteration on iteration so as to maximise the sampling where the integrand is largest, it does this by importance sampling and stratified sampling [18] [20]. The result is each successive iteration should produce a more accurate estimate; in addition, the number of evaluations per iteration typically increases iteration on iteration (set by `nincrease`) and therefore the statistical fluctuations will also reduce.

4.3. Adding a Process to reSolve

Following the description of the structure of the reSolve program, and how it works, in the previous two sections (Sections 4.1 and 4.2 respectively), we now outline how to add a new process to the program. In order to produce the full differential cross-sections, in, for example, transverse momentum, reSolve adds q_T resummation to a generic process so as to accurately produce the low q_T part of the spectrum, which is often the technically most-challenging piece. The user must then add this transverse momentum resummed spectrum, produced by reSolve, to the usual finite piece and match them appropriately to obtain the complete spectrum over all q_T . The resummation part of the reSolve program is completely process-independent, therefore to add a new process one must simply undertake the following steps; where `{process}` appears in a routine, file or folder name below it is to be replaced by a suitable name for the added process.

- Set a process number for the added process, given 1 indicates diphoton and 2 indicates Drell-Yan, for the first added process take 3. In `User.cc` extend the else if sequence to include the added process if the process flag is set to 3, to do so copy the code section for either diphoton or Drell-Yan to produce the new calls for the new process. This includes calling a new routine `{process}_setup` which will be defined in the file “`{process}_input.cc`” which we will create later in the new sub folder “`Process/{process}_Res/`”.
- Enter the Process sub folder (see Figure 15), this is the sub folder which contains the process dependent routines. There are currently two sub folders within it, one for diphoton (`Diphoton_Res`) and one for Drell-Yan (`DrellYan_Res`), each contains files and routines `{process}_cuts.cc`, `{process}_hard.cc`, `{process}_input.cc`, `{process}_integrand.cc`, `{process}_ps.cc`, and corresponding header files, see Figure 16. The goal is to produce appropriate corresponding files for the new process. First create the new process sub folder `{process}_Res`. Next, we will begin with the aforementioned process-specific input file input routine `ReadInput_{process}` in `{process}_input.cc`, this reads the process-specific input from the input file “`{process}_input.cc`”. This contains two parts, first there is the `{process}_setup` routine which organises the program pre-resummation for the new process. To create this, the form of these files for diphoton or Drell-Yan can be copied, with appropriate changes in the number of random variables required for the Monte Carlo

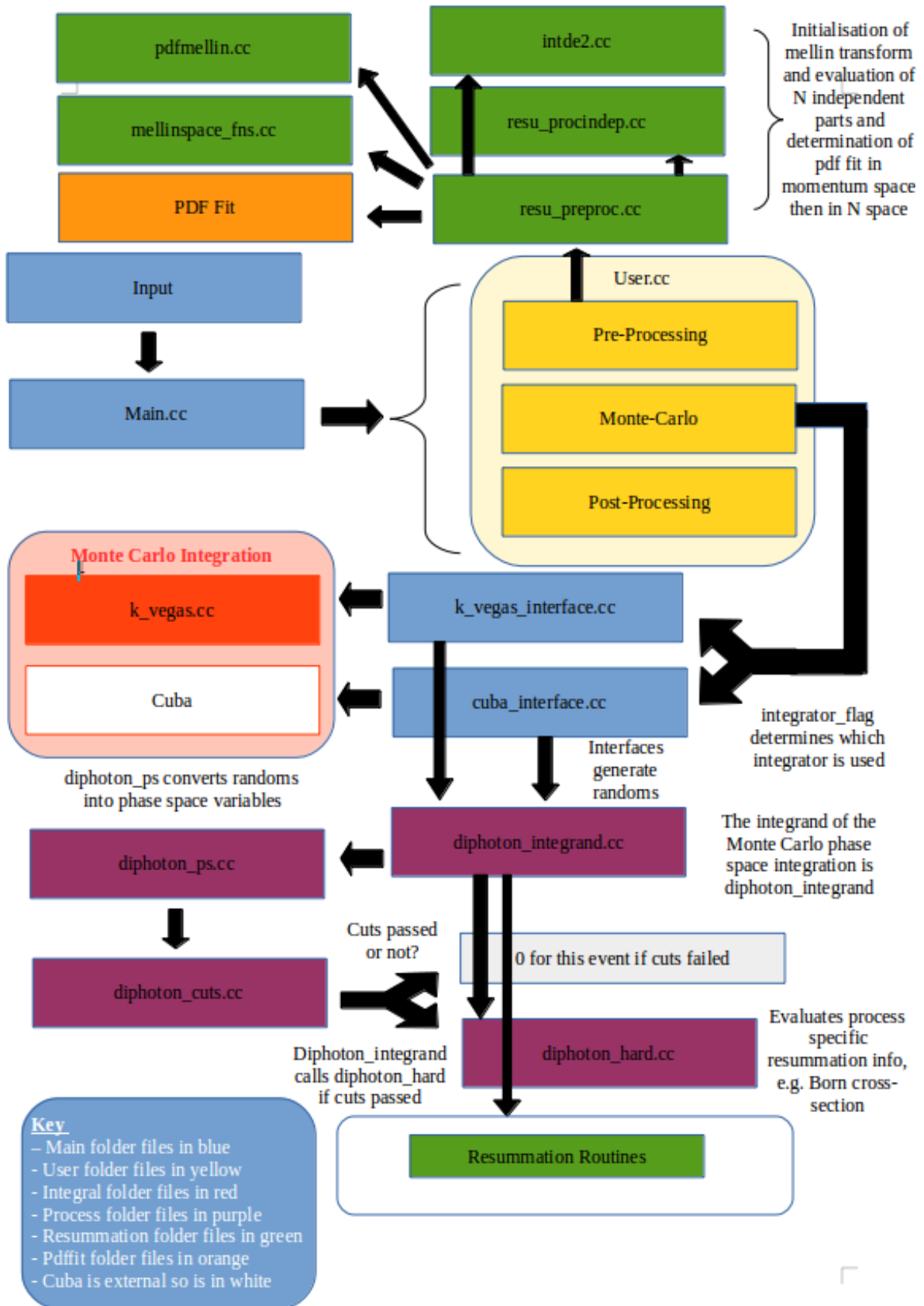


Figure 13: A flowchart demonstrating the different aspects included in the program and what is called when in the calculations. The different aspects of the program are coloured differently to indicate where they sit in the program folder structure. A zoom in of the resummation routines at the bottom of the flowchart is given in figure 14. The program functions analogously in the case of Drell-Yan processes.

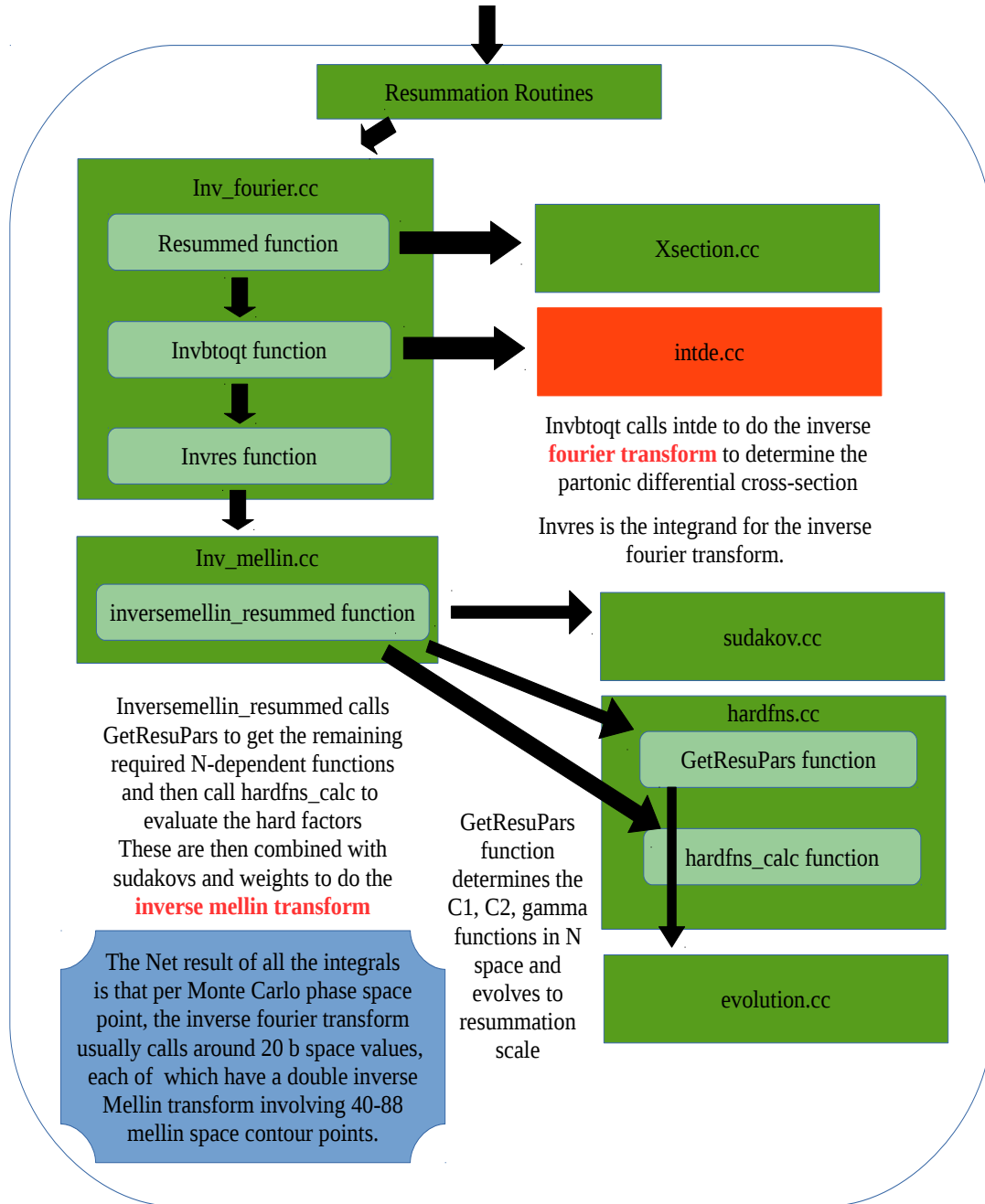


Figure 14: A flowchart providing more detail on the resummation aspect of the program, which is the main part of the calculation. This highlights how both the inverse Fourier and double inverse Mellin transforms are performed. This part of the program is process independent.

(`ndim`) and other variables such as the number of particles (`npart`). This setup routine will call the routine `{process}_ReadInput`, again the basic form of this will be analogous to the diphoton and Drell-Yan cases, with the new relevant cuts for the process under consideration; for example diphoton has an “R cut” based on the angular separation of the two photons, whilst Drell-Yan may have a transverse mass cut and “pTmiss” cut. Create also the corresponding header file, including the class `{process}_input` used to pass this input information to the cut-checking routine later.

- Next create the `{process}_integrand.cc` file and corresponding routine. This routine is called from `User.cc` via the `k vegas_call` or `cuba_call`, and coordinates the main calculations for this process. Once more, the general form of these files for the diphoton or Drell-Yan case may be copied. First, the routine calls a phase-space generating routine `{process}_ps` in order to generate the randoms and phase space for each process event, this is contained in the process specific file `{process}_ps.cc`. Next the cuts relevant to the process phase space are checked by calling the `{process}_cuts` routine in the `{process}_cuts.cc` file, if the cuts are passed (“false” means not cut here, i.e. cuts passed) then the Born-level cross-section is determined via the routine `sigmaj{process}calc` in the file `{process}_hard.cc`. The relevant information is then loaded into the `resuminfo` object containing all the necessary information needed for the process-independent resummation part of the calculation, which is implemented by calling the “resummed” routine in inverse Fourier. This determines the overall cross-section for each event, including resummation up to NNLL, finally the events are then output in whichever form is indicated in the input file.
- This `{process}_integrand` routine directed the resummation calculation by calling the relevant phase space generating, cut-checking and Born cross-section determining process-specific routines; these are `{process}_ps`, `{process}_cuts` and `{process}_hard` respectively and must be created for the new added process. First consider the phase space generation; this routine reads in randoms generated in the rest of the code and uses them to set the relevant parameters for the process phase space, these are parameters such as the invariant mass squared (q^2), transverse momentum squared (q_T^2), η of the diphoton/Drell-Yan system and θ_{CM} and ϕ_{CM} angles describing the plane in which the diphotons/Drell-Yan system are emitted within the centre of mass frame of the colliding partons. The Jacobian for the transformation between these random variables, whose values are between 0 and 1, and the phase space variables, is given by “`randsjacob`”. A kinematics routine is then called to determine the 4-momenta and angular separation of the relevant particles (e.g. the two photons for diphoton, two leptons for Z Drell-Yan, etc) in the lab frame in order to allow later application of the cuts. Various other variables such as the factorisation, resummation and renormalisation scales are also set here; this will be repeated for the new process and so can be copied; all this information on phase space variables, kinematics and other variables is then passed into the `resuvars` object to be passed to the relevant sections of the resummation code as and when necessary.
- The `{process}_cuts.cc` file and routine’s overall structure may again be duplicated from the diphoton and Drell-Yan examples. There are two broad types of cuts contained, first generic kinematical constraints are applied to cut if either $|\eta|$ or q_T become too large that the resummation formalism is no longer valid. These are the “`gencuts`” and should be kept for all processes. Secondly, there are the process specific, phase space cuts, “`pscuts`”, which are checked via the `PS{process}cuts_1` routine. This routine will have to be written anew for each added process, nonetheless it is straightforward. The process-specific cut information read in from the input file (via the `ReadInput_{process}` routine discussed earlier) is passed in via a `{process}_input` object, whilst the phase space and event information is passed via a “`PSpoint`” object. With this information, the relevant phase space kinematic parameters can be determined for each event and tested against the cuts.
- Finally, the `{process}_hard.cc` file is where most potential difficulty lies in ensuring consistency of factors. The `sigmaj{process}calc` routine uses the input process information and event phase space point parameters to determine the Born-level cross-section for the added process. This is then loaded into the `sigmaj` vector array to be used elsewhere in the code, for example in computation of the hard factors in the resummation code. Process-specific hard factors are also calculated within this file (although not in the Drell-Yan case as the Drell-Yan scheme has been used - see Section 5.3 for further information).

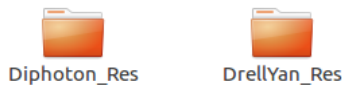


Figure 15: The Process subfolder, which contains (nearly) all the process-dependent aspects of the resummation program. A new subfolder within it will be created for each additional process. See Figure 16 for the routines included

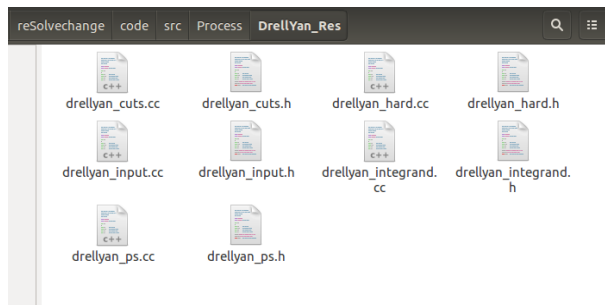


Figure 16: The `DrellYan_Res` subfolder (contained in the Process subfolder), which contains the Drell-Yan specific routines and links with the generic resummation parts of the `reSolve` program. An analogous subfolder and routines therein exists for the diphoton process and should be recreated for any processes added.

- It is also necessary to add a new section in the file `hardfns.cc` in `Resummation/`, there are sections of code here which detail the contributions from gluon-gluon initiation, quark-gluon initiation, quark-quark initiation to the hard factors. In the interests of the speed of evaluation of the program, only the non-zero contributions for each process are explicitly summed, for example for diphoton or Z/γ Drell-Yan only $q\bar{q}$ is summed over as the final quarks at the stage of the diphoton production must be the same, whilst for W^\pm Drell-Yan the contributions are $q\bar{q}'$ and, as a result of the CKM matrix, can occur with q and q' of different generations, such as $u\bar{s}$. Aside from determining which contributions are non-zero and summing these appropriately, one may copy the diphoton and Drell-Yan code here. It is worth noting that the structure of the theoretical formalism here is process-independent, one must just sum over all contributions (including zero contributions) for each process, the small process dependence introduced in `reSolve` is purely a pragmatic one, to avoid wasting time summing many zero contributions in the time-critical part of the program.

All of the remainder of the program should remain exactly as it is, the calculation of the hard factors, sudakovs, determination of the inverse Mellin transforms, the inverse Fourier transform from impact parameter space, the Monte Carlo phase space integration and everything else required will be calculated automatically by the program. In this way `reSolve` takes advantage of the generality of the b-space Mellin-space resummation formalism of [6, 7, 8, 9, 10, 11, 12, 13, 14, 15], which is described in Section 5 and in Appendix B of this paper. However, in the current version only the hard factors for qq initiated processes are included up to NNLL, those for gg initiated processes are only included at LL, therefore processes requiring these hard factors beyond LL will need the additional higher order hard factors to be added. We intend to resolve this in the immediate future, indeed it is necessary for the adding of the Higgs diphoton signal process and should not be of great difficulty.

5. Transverse momentum resummation theory and formalism

The central feature of reSolve is the addition of transverse momentum (q_T) resummation to a rather general class of inclusive hadronic processes, of which the processes included in the current version, $h_1 h_2 \rightarrow \gamma\gamma$ and $h_1 h_2 \rightarrow W^\pm/Z$ (with the h_i here being either a p or a \bar{p}) are just an example. We thus devote this section to a brief review of the specific q_T resummation formalism used in the program.

The physical motivation for q_T resummation lies in the well-known fact that, when measuring transverse momenta q_T which are much softer than the typical hard scale of a process μ , the appearance of large logarithmic terms (of the generic form $\alpha_s^n (\log(\mu/q_T))^m$, with $m \leq 2n$, where α_s is the strong coupling constant) spoils the convergence of the perturbative series. In these cases, contributions to all orders in α_s must be resummed in order to recover reliable predictions for observable quantities. There are different approaches which can be followed to achieve this resummation; the formalism used in reSolve is the impact parameter (b) space formalism of refs. [6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. In the following we will not give any derivation of the final formulae used in the code, but just state them and comment on how they are actually implemented in reSolve. We will make an effort to distinguish between what is inherent in the formalism and what is a choice related to our specific implementation. We mainly follow the notation used in ref. [15].

The class of processes we will consider are those of the form $h_1 h_2 \rightarrow F + X$, where h_1 and h_2 are hadrons and the observed final state F is made up of an arbitrary number of particles, with the only constraint that they be not strongly interacting (that is, no jets or hadrons are allowed in F). The X denotes (as usual) unresolved extra radiation. Our master formula [15] for the *fully differential* cross-section $d\sigma$ at low q_T for the $h_1 h_2 \rightarrow F + X$ process is then:

$$\frac{d\sigma_{res}^F(p_1, p_2, M^2, \mathbf{q}_T, y, \mathbf{\Omega})}{dM^2 d^2 \mathbf{q}_T dy d\mathbf{\Omega}} = \int \frac{d^2 b}{(2\pi)^2} \int_{x_1}^1 \frac{dz_1}{z_1} \int_{x_2}^1 \frac{dz_2}{z_2} W^F(\mathbf{b}, z_1, z_2, \dots) \equiv \frac{M^2}{s} \times \quad (4)$$

$$\left[d\hat{\sigma}_{c\bar{c}}^{F, LO} \right] \int \frac{d^2 b}{(2\pi)^2} e^{i\mathbf{b}\cdot\mathbf{q}_T} S_c(M^2, b_0^2/b^2) \int_{x_1}^1 \frac{dz_1}{z_1} \int_{x_2}^1 \frac{dz_2}{z_2} \left[H^F C_1 C_2 \right]_{c\bar{c}; a_1 a_2} f_{a_1/h_1}(x_1/z_1, b_0^2/b^2) f_{a_2/h_2}(x_2/z_2, b_0^2/b^2).$$

Eq. (4) is given as a function of the hadron momenta p_1 and p_2 , the total invariant mass (M^2 , also denoted q^2 or qq^2 elsewhere), transverse momentum (a 2-dimensional vector \mathbf{q}_T , with $q_T = |\mathbf{q}_T|$, and similarly $b = \mathbf{b}$) and rapidity y of the final state F , as well as on any additional variables (collectively denoted by $\mathbf{\Omega}$) internal to F that may be needed to fully define F phase space. The indices a_1, a_2 and c are flavour indices (\bar{c} denotes the anti-parton of c) and a sum over them is implicit. The f_{a_i/h_i} are usual PDFs. $b_0 = 2e^{-\gamma_E}$ (with $\gamma_E = 0.5772\dots$), and the partonic momentum fractions x_1 and x_2 are fixed and given by

$$x_1 = \frac{M}{\sqrt{s}} e^{+y}, \quad x_2 = \frac{M}{\sqrt{s}} e^{-y}. \quad (5)$$

As noted, eq. (4) is an approximation that holds for low values of q_T : more explicitly, this means that the expression is valid for $q_T/M \ll 1$ up to corrections formally of $O(q_T^2/M^2)$ (in other words, the hard scale of the process is taken to be of order M). To keep in mind this restriction, a suffix *res* (for ‘‘resummed’’) is appended to the differential cross-section symbol. To recover a description valid for arbitrary values of q_T , some prescription for a matching with a high- q_T description for the same process (which in turn can be obtained via a usual fixed order expansion) must be provided. There are several different approaches to defining such a matching, with an additive approach typically being used in the b -space formalism [21]. Note however, that no matching is required, if only the lower end of the q_T spectrum is to be analysed. No matching scheme is implemented in the current version of reSolve, rather reSolve is aimed at computing the resummed (i.e. low q_T) part of the differential q_T spectrum, which is often the most challenging piece.

The physical meaning of eq. (4) is nicely illustrated by fig. 17, which also gives an intuitive understanding of how the various pieces of the formula arise.

Several comments are needed in order to fully appreciate the structure of eq. (4), as a number of its ingredients are written in a rather implicit form. S_c is the Sudakov form factor, which is *universal*, that is, process-independent (the two terms will be used interchangeably throughout this section) and given by [11]

$$S_a(\mu_2^2, \mu_1^2) = \exp \left\{ - \int_{\mu_1^2}^{\mu_2^2} \frac{dq^2}{q^2} \left[A_a(\alpha_s(q^2)) \log \frac{\mu_2^2}{q^2} + B_a(\alpha_s(q^2)) \right] \right\}, \quad (6)$$

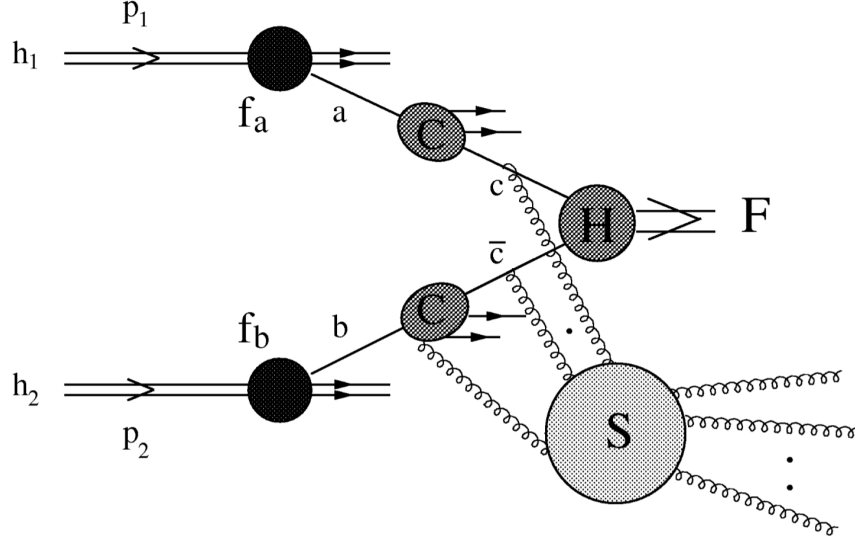


Figure 17: Pictorial version of eq. (4): a parton of momentum fraction x_1 is extracted from hadron h_1 in the upper leg, then it splits (z_1) via a collinear partonic sub process $a \rightarrow c$ at the upper C blob. A specular process happens at the lower leg: in the end the momenta that enter the hard process H are $x_1 z_1 p_1$ and $x_2 z_2 p_2$, where $p_{1,2}$ are the momenta of the initial hadrons. Soft partons can be emitted anywhere (except inside H itself) and contribute to the Sudakov form factor S_c .

where μ_1, μ_2 are any two scales. The functions A_a and B_a depend only on α_s and admit the perturbative expansion

$$A_a(\alpha_s) = \sum_{n=1}^{\infty} \left(\frac{\alpha_s}{\pi}\right)^n A_a^{(n)}, \quad B_a(\alpha_s) = \sum_{n=1}^{\infty} \left(\frac{\alpha_s}{\pi}\right)^n B_a^{(n)}, \quad (7)$$

and the coefficients up to $A_a^{(3)}$ and $B_a^{(2)}$ are explicitly known. The expressions for these and other resummation coefficients will be collected, along with the references they are originally collected from, in Appendix B. The Sudakov form factor (6) resums to all orders, in eq. (4), logarithmically-enhanced contributions of the form $\log(M^2 b^2)$ (the region $q_T/M \ll 1$ corresponds, in impact parameter space, to $Mb \gg 1$). To see this explicitly, the $(dq^2)/q^2 = d \log q^2$ integral in eq. (6) must be carried out; the most straightforward way to do this is to use the α_s evolution equation to express $\alpha_s(q^2)$ in terms of $\alpha_s(M^2)$ via the α_s evolution equation

$$\begin{aligned} \frac{d \log(\alpha_s(q^2))}{d \log q^2} &= \beta(\alpha_s(q^2)) \equiv - \sum_{n=0}^{\infty} \beta_n \alpha_s(q^2)^{n+1} \\ \Rightarrow \alpha_s(q^2) &= \frac{\alpha_s(M^2)}{l} - \left(\frac{\alpha_s(M^2)}{l}\right)^2 \frac{\beta_1}{\beta_0} \log l + \dots, \end{aligned} \quad (8)$$

where $l = 1 + \beta_0 \alpha_s(M^2) \log(q^2/M^2)$. This way, the scale q^2 only appears through the logarithmic term $\alpha_s(M^2) \log(q^2/M^2)$ in the integrand and, carrying out the integration, one then obtains an expression of the form

$$- \int_{b_0^2/b^2}^{M^2} \frac{dq^2}{q^2} \left[A_a(\alpha_s(q^2)) \log \frac{\mu_2^2}{q^2} + B_a(\alpha_s(q^2)) \right] = \left(\frac{\alpha_s(M^2)}{\pi}\right)^{-1} \bar{g}^{(1)} + \left(\frac{\alpha_s(M^2)}{\pi}\right)^0 \bar{g}^{(2)} + \left(\frac{\alpha_s(M^2)}{\pi}\right)^1 \bar{g}^{(3)} + \dots, \quad (9)$$

where the coefficients $\bar{g}^{(n)}$ contain contributions to all orders in the combination $\alpha_s \log(b^2/b_0^2 M^2)$, which is formally of $\mathcal{O}(1)$, but that are suppressed by growing powers of the small parameter $\alpha_s(M^2)/\pi$. The first order in this expansion is usually called Leading Log (LL), the following one Next-to-Leading Log (NLL), and so on. The explicit expressions of the first three terms are

$$\bar{g}^{(1)} = \frac{A^{(1)}}{\beta_0} \frac{\lambda + \log(1 - \lambda)}{\lambda}, \quad (10)$$

$$\begin{aligned} \bar{g}^{(2)} = & \frac{B^{(1)}}{\beta_0} \log(1 - \lambda) - \frac{A^{(2)}}{\beta_0^2} \left(\frac{\lambda}{1 - \lambda} + \log(1 - \lambda) \right) \\ & + \frac{A^{(1)}\beta_1}{\beta_0^3} \left(\frac{1}{2} \log^2(1 - \lambda) + \frac{\log(1 - \lambda)}{1 - \lambda} + \frac{\lambda}{1 - \lambda} \right), \end{aligned} \quad (11)$$

$$\begin{aligned} \bar{g}^{(3)} = & -\frac{A^{(3)}}{2\beta_0^2} \frac{\lambda^2}{1 - \lambda} - \frac{B^{(2)}}{\beta_0} \frac{\lambda}{1 - \lambda} + \frac{A^{(2)}\beta_1}{\beta_0^3} \left(\frac{\lambda(3\lambda - 2)}{2(1 - \lambda)^2} - \frac{(1 - 2\lambda)\log(1 - \lambda)}{(1 - \lambda)^2} \right) \\ & + \frac{B^{(1)}\beta_1}{\beta_0^2} \left(\frac{\lambda}{1 - \lambda} + \frac{\log(1 - \lambda)}{1 - \lambda} \right) + A^{(1)} \left(\frac{\beta_1^2}{2\beta_0^4} \frac{1 - 2\lambda}{(1 - \lambda)^2} \log^2(1 - \lambda) \right. \\ & \left. + \log(1 - \lambda) \left(\frac{\beta_0\beta_2 - \beta_1^2}{\beta_0^4} + \frac{\beta_1^2}{\beta_0^4(1 - \lambda)} \right) \right), \end{aligned} \quad (12)$$

where the $\log(M^2 b^2)$ terms are contained in the variable λ :

$$\lambda = \frac{1}{\pi} \beta_0 \alpha_s(M^2) \log(M^2 b^2 / b_0^2). \quad (13)$$

These are also the terms currently implemented in the program, which thus can perform resummations up to NNLL.

It is important to note that the Sudakov form factor (6) is not the only source of b -dependence – and thus of large logarithms – in eq. (4). One more contribution comes from the PDFs, which are also functions of b through their scale dependence. This dependence can be extracted, and the PDFs evaluated at the usual (hard) factorization scale $\mu_F \sim M$ via the standard PDF evolution equation, whose integral solution reads

$$f_{a/h}(x, b_0^2/b^2) = \int_x^1 dz U_{ab}(x/z; b_0^2/b^2, \mu_F^2) f_{a_1/h_1}(z, \mu_F^2), \quad (14)$$

and the kernels $U_{ab}(x/z; b_0^2/b^2, \mu_F^2)$ also contribute to the all-order resummation of logarithmic $\alpha_s \log(b^2/b_0^2 \mu_F^2) \sim \alpha_s \log(b^2/b_0^2 M^2)$. The U_{ab} evolution kernels are also implemented in the program in a dedicated routine.

As already stated, the Sudakov factor is universal, as of course are the PDFs. The process dependence is in fact completely contained in the last two remaining factors which build up eq. (4), the first of which, $[d\hat{\sigma}_{c\bar{c}}^{F, \text{LO}}]$, is essentially trivial, being simply the Born-level partonic cross-section for the process $c\bar{c} \rightarrow F$. More explicitly

$$[d\hat{\sigma}_{c\bar{c}}^{F, \text{LO}}] = \frac{d\hat{\sigma}_{c\bar{c}}^{F, \text{LO}}}{M^2 d\mathbf{\Omega}}(x_1 p_1, x_2 p_2, \mathbf{\Omega}, \alpha_s(M^2)). \quad (15)$$

This term obviously contains no b dependence. Notice that since all particles in F are colourless, the process at parton-level can only be initiated by opposite-coloured partons, that is, either $q_i \bar{q}_j$ or gg , which is the reason for the label $c\bar{c}$ rather than a more general cc' .

The remaining process-dependent factor in eq. (4) is $[H^F C_1 C_2]_{c\bar{c}; a_1 a_2}$, and its explicit expression is different depending on whether the Born-level partonic process is initiated by $q\bar{q}$ or gg . In the former case, its explicit form is

$$[H^F C_1 C_2]_{q\bar{q}; a_1 a_2} = H_q^F(x_1 p_1, x_2 p_2, \mathbf{\Omega}, \alpha_s(M^2), \mu_R) C_{q a_1}(z_1, \alpha_s(b_0^2/b^2)) C_{\bar{q} a_2}(z_2, \alpha_s(b_0^2/b^2)), \quad (16)$$

where the H_q^F and C_{qa} admit a perturbative expansion similar to the ones in (7):

$$\begin{aligned} H_q^F(x_1 p_1, x_2 p_2, \mathbf{\Omega}, \alpha_s(M^2), \mu_R) &= 1 + \sum_{n=1}^{\infty} \left(\frac{\alpha_s}{\pi}\right)^n H_q^{F(n)}(x_1 p_1, x_2 p_2, \mathbf{\Omega}, \mu_R) \\ C_{qa}(z, \alpha_s) &= \delta_{qa} \delta(1-z) + \sum_{n=1}^{\infty} \left(\frac{\alpha_s}{\pi}\right)^n C_{qa}^{(n)}(z) \end{aligned} \quad (17)$$

(note that a dependence of the coefficients H_a and C_{qa} on the specific flavours of the quark q_i and anti quark \bar{q}_j never arises from QCD corrections thanks to flavour symmetry, since quark mass effects are neglected in eq. (4). In H_a however such a dependence can in principle still arise due to, for instance, charge effects). The coefficients $C_{qa}^{(n)}$ are all known up to order 2 and are process-independent. The explicit expressions for the $C_{qa}^{(n)}$ are given in Appendix B. The $C_{qa}(z, \alpha_s)$ also contribute one last piece of b -dependence, since they are evaluated at $\alpha_s(b_0^2/b^2)$. To make this dependence more explicit, we can use the general relation

$$h(\alpha_s(\mu_1^2)) = h(\alpha_s(\mu_2^2)) \exp \left[\int_{\mu_1^2}^{\mu_2^2} \frac{dq^2}{q^2} \beta(\alpha_s(q^2)) \frac{d \log(h(\alpha_s(q^2)))}{d \log(\alpha_s(q^2))} \right], \quad (18)$$

valid for any function of α_s . With this, we can rewrite

$$C_{qa}(z, \alpha_s(b_0^2/b^2)) = C_{qa}(z, \alpha_s(M^2)) \exp \left[\int_{b_0^2/b^2}^{M^2} \frac{dq^2}{q^2} \beta(\alpha_s(q^2)) \frac{d \log(C_{qa}(z, \alpha_s(q^2)))}{d \log(\alpha_s(q^2))} \right] \quad (19)$$

(this is not a matrix equation: it holds element by element for C_{qa}), so that each C_{qa} coefficient is now expressed as a function of α_s at the *perturbative* scale M^2 and an integral of the same form as (6), which gives the final contribution to the large $\log(M^2 b^2)$ logarithm resummation. Meanwhile, the process dependence comes only from the ‘‘hard factor’’ $H_q^{F(n)}$, which is however b -independent and does not contain any large logarithms.

In the case of a gg -initiated process, the term $[H^F C_1 C_2]_{gg; a_1 a_2}$ has a richer structure, due to correlations [14] produced by the evolution of the colliding partons a_1, a_2 into gluons. The general considerations are however the same. Explicitly we have:

$$\begin{aligned} [H^F C_1 C_2]_{gg; a_1 a_2} &= \sum_{h_1 \lambda_1 \lambda_2} H_g^{F(h_1 \lambda_1)(h_2 \lambda_2)}(x_1 p_1, x_2 p_2, \mathbf{\Omega}, \alpha_s(M^2), \mu_R) \\ &C_{ga_1}^{h_1 \lambda_1}(z_1, p_1, \mathbf{b}, \alpha_s(b_0^2/b^2)) C_{ga_2}^{h_2 \lambda_2}(z_2, p_2, \mathbf{b}, \alpha_s(b_0^2/b^2)), \end{aligned} \quad (20)$$

where $h_{1,2}$ and $\lambda_{1,2}$ are helicity indices. The terms $C_{ga_i}^{\lambda_i h_i}$ can be decomposed into *helicity conserving* and *helicity flipping* components

$$C_{ga_i}^{\lambda_i h_i}(z_i, p_i, \mathbf{b}, \alpha_s) = C_{ga_i}(z_i, \alpha_s) \delta^{\lambda_i, h_i} + G_{ga_i}(z_i, \alpha_s) D^{(\lambda_i)}(p_i, \mathbf{b}) \delta^{\lambda_i, -h_i}, \quad i = 1, 2; \quad (21)$$

here $D^{(\lambda_i)}(p_i, \mathbf{b})$, which embodies the full \mathbf{b} and p_i dependence of the $C_{ga_i}^{\lambda_i h_i}$ coefficient, is in fact a simple phase factor

$$D^{(\lambda_i)}(p_i, \mathbf{b}) = -e^{\pm 2i \lambda_i (\varphi(\mathbf{b}) - \varphi_i)}, \quad (22)$$

where $\varphi(\mathbf{b})$, φ_1 , φ_2 are the azimuthal angles of \mathbf{b} , p_1 and p_2 respectively, and the overall sign of the exponent is ‘‘+’’ if $i = 1$ and ‘‘-’’ if $i = 2$. Again, the $H_g^{F(h_1 \lambda_1)(h_2 \lambda_2)}$, C_{ga} and G_{ga_i} admit perturbative expansions

$$\begin{aligned} H_g^{F(h_1 \lambda_1)(h_2 \lambda_2)}(x_1 p_1, x_2 p_2, \mathbf{\Omega}, \alpha_s(M^2), \mu_R) &= 1 + \sum_{n=1}^{\infty} \left(\frac{\alpha_s}{\pi}\right)^n H_g^{F(h_1 \lambda_1)(h_2 \lambda_2)(n)}(x_1 p_1, x_2 p_2, \mathbf{\Omega}, \mu_R) \\ C_{qa}(z, \alpha_s) &= \delta_{qa} \delta(1-z) + \sum_{n=1}^{\infty} \left(\frac{\alpha_s}{\pi}\right)^n C_{qa}^{(n)}(z), \quad G_{ga}(z, \alpha_s) = \sum_{n=1}^{\infty} \left(\frac{\alpha_s}{\pi}\right)^n G_{ga}^{(n)}(z), \end{aligned} \quad (23)$$

where it is significant to notice that the helicity flipping coefficients G_{ga} have no $O(\alpha_s)^0$ term. Both the $C_{ga}^{(n)}$ and the $G_{ga}^{(n)}$ are, as their analogues C_{qa} , universal and known up to order 2 (with their explicit expressions given in Appendix B), and can be expressed as functions of $\alpha_s(M^2)$ times exponential terms which contribute to large log resummation. The process dependence is, in this case, embodied in the hard factor $H_g^{F(h_1, \lambda_1)(h_2, \lambda_2)}$ which is, as in the $q\bar{q}$ case, b -independent.

It is useful to further clarify the explicit form of the hard factors H_q^F and $H_g^{F(h_1, \lambda_1)(h_2, \lambda_2)}$: in fact, apart from the complication arising from helicity correlation in the latter, their structure is very similar. In both cases, these terms include the contributions from all *purely virtual* corrections to the $c\bar{c} \rightarrow F$ partonic process, up to a specified order. The explicit definition of the H_q^F and $H_g^{F(h_1, \lambda_1)(h_2, \lambda_2)}$ factors can be given in terms of a *UV-renormalised* and *IR subtracted* helicity scattering amplitude, $\tilde{\mathcal{M}}_{c\bar{c} \rightarrow F}^{h_1 h_2}$. The UV renormalisation is carried out by standard means in a specified renormalisation scheme, typically $\overline{\text{MS}}$ (this is the case for reSolve). The *UV-renormalised* scattering amplitude $\mathcal{M}_{c\bar{c} \rightarrow F}^{h_1 h_2}$ is then transformed to $\tilde{\mathcal{M}}_{c\bar{c} \rightarrow F}^{h_1 h_2}$ via a subtraction operation, which can be written to all orders as

$$\tilde{\mathcal{M}}_{c\bar{c} \rightarrow F}^{h_1 h_2}(x_1 p_1, x_2 p_2, \Omega, \mu_R) = \left(1 - \tilde{I}_c(\epsilon, M^2, \mu_R)\right) \mathcal{M}_{c\bar{c} \rightarrow F}^{h_1 h_2}(x_1 p_1, x_2 p_2, \Omega, \mu_R, \epsilon), \quad (24)$$

where $\tilde{I}_c(\epsilon, M^2, \mu_R)$ is an universal subtraction operator and μ_R the renormalisation scale. Here we supposed $\mathcal{M}_{c\bar{c} \rightarrow F}^{h_1 h_2}$ to be evaluated in dimensional regularization, and $\epsilon = d - 4$ is the corresponding dimensional regularization parameter. $\mathcal{M}_{c\bar{c} \rightarrow F}^{h_1 h_2}$ will be in general ϵ dependent because, even though it is renormalised, it will still contain divergences of IR origins. For the sake of brevity, we will not report the specific form (and perturbative expansion) here, referring to the reader to the original paper [15]. The important point is that the subtraction in eq. (24), exactly cancels all ϵ poles in $\mathcal{M}_{c\bar{c} \rightarrow F}^{h_1 h_2}$, leaving the left-hand side of eq. (24) finite.

In terms of $\tilde{\mathcal{M}}_{c\bar{c} \rightarrow F}^{h_1 h_2}$, the hard factors H_q^F and $H_g^{F(h_1, \lambda_1)(h_2, \lambda_2)}$ can now be written as

$$H_q^F = \frac{|\tilde{\mathcal{M}}_{q\bar{q} \rightarrow F}^2|}{|\mathcal{M}_{q\bar{q} \rightarrow F(0)}|^2}, \quad H_g^{F(h_1, \lambda_1)(h_2, \lambda_2)} = \frac{[\tilde{\mathcal{M}}_{q\bar{q} \rightarrow F}^{h_1 h_2}]^* \tilde{\mathcal{M}}_{q\bar{q} \rightarrow F}^{\lambda_1 \lambda_2}}{|\mathcal{M}_{q\bar{q} \rightarrow F(0)}|^2}, \quad (25)$$

where the squared matrix elements have an implicit summation over helicity indices, that is

$$|\tilde{\mathcal{M}}_{q\bar{q} \rightarrow F}^2| = \sum_{h_1 h_2} [\tilde{\mathcal{M}}_{q\bar{q} \rightarrow F}^{h_1 h_2}]^* \tilde{\mathcal{M}}_{q\bar{q} \rightarrow F}^{h_1 h_2} \quad (26)$$

and $\mathcal{M}_{q\bar{q} \rightarrow F(0)}^{h_1}$ is the Born scattering amplitude (the tilde is omitted since the Born term is finite by itself).

Eq. (25) makes the relation between the gluon-initiated process hard factor $H_g^{F(h_1, \lambda_1)(h_2, \lambda_2)}$ and the quark-initiated one H_q^F explicit and easy to understand: if we could turn off the spin flipping coefficients G_{ga} , substituting eq. (25) in eqs. (16) and (20) respectively would yield exactly specular expressions. By contrast, in the actual full expression for $[H^F C_1 C_2]_{gg; a_1 a_2}$, extra contributions involving the G_{ga} appear. It is interesting to note [14], though we will not show it here, that whenever an azimuthally-averaged observable (such as the q_T spectrum) is considered, any cross-terms involving both C_{ga} and G_{ga} vanish to all orders. Even in those cases, however, an extra contribution involving two G_{ga} terms always survives, though it is suppressed (since G_{ga} has no order 0 contribution) by at least two extra powers of $\alpha_s(\mu_R^2)$ with respect to the leading term in $[H^F C_1 C_2]_{gg; a_1 a_2}$.

5.1. The resummation scale

The factorization of logarithmic terms in eqs. (6) and (19) involves some degree of arbitrariness (see for instance [21]). The argument of the formally large logarithms can always be rescaled as

$$\log(M^2 b^2) = \log(\mu^2 b^2) + \log(M^2 / \mu^2), \quad (27)$$

where μ is any scale such that $\mu \sim M$. In order to parametrize this arbitrariness, an extra scale μ_S can be introduced [21]. The large logarithms are then defined to be

$$L \equiv \log \frac{\mu_S^2 b^2}{b_0^2}, \quad (28)$$

and eqs. (6) and (19) are modified accordingly, in a straightforward way. The function of the μ_S is to estimate the intrinsic uncertainty in the resummation formula (4); in this sense, the scale is analogous to the usual renormalisation and factorization scales μ_R and μ_F . By purely empirical considerations, the central value of the resummation scale is set to be $\mu_R/2$ by default in reSolve, even though of course the user can set it to any value he or she chooses from the input file.

5.2. The \mathbf{b} and z_1, z_2 integrals

The evaluation of the differential cross-section $\frac{d\sigma_{res}^F}{dM^2 d^2\mathbf{q}_T dy d\Omega}$ involves four integrations, over \mathbf{b} , z_1 and z_2 . The dependence on the azimuthal angle φ_b is actually simple enough that the corresponding integration can always be done analytically, reducing the number of integrals which have to be dealt with numerically to three. In particular, for $q\bar{q}$ -initiated processes, the $d\varphi_b$ integration just implies the replacement:

$$\int \frac{d^2\mathbf{b}}{(2\pi)^2} e^{-i\mathbf{q}_T \cdot \mathbf{b}} \rightarrow \int_0^\infty \frac{b}{2\pi} J_0(q_T b), \quad (29)$$

where J_0 is the 0th-order Bessel function, since nothing in the main integrand W^F depends on φ_b and

$$J_0(x) = \int_0^{2\pi} \frac{d\varphi}{2\pi} e^{\pm ix \cos \varphi}. \quad (30)$$

In the gg -initiated case, due to the presence of azimuthal correlations, after the $d\varphi_b$ integration the integral is split into two terms still involving Bessel functions, the first proportional to $J_0(q_T b)$ and the second to $J_2(q_T b)$; we refer the reader to [14] for details.

The surviving integration over $b = |\mathbf{b}|$ is the most delicate one from both a numerical and theoretical point of view. Numerically, it involves a semi-infinite integrand over a rapidly oscillating (asymptotically) function. To perform this integral in reSolve, we use an external package based on ref. [22] which is specifically designed in order to deal with this kind of integral.

From a theoretical point of view, the issue is that the main function $W^F(\mathbf{b}, \dots)$ actually becomes ill-defined both at very small and very large values of b . Very small values $\mu_S b \ll 1$ are actually spurious from the point of view of resummation, since they correspond in transverse momentum space to large q_T values $q_T/\mu_S \gg 1$ which are outside the range of validity of eq. (4). To neatly cut-off these contributions, and thus to make the integrand well-defined in the small b region as well as to reduce the impact of unjustified resummed logarithms in the large q_T region, we implement, following ref. [21], the replacement:

$$L \rightarrow \tilde{L} \equiv \log\left(\frac{\mu_S^2 b^2}{b_0^2} + 1\right), \quad (31)$$

everywhere inside the b -dependent coefficients. We see that $\tilde{L} \rightarrow L$ in the resummation region $\mu_S b \gg 1$, so the replacement is legitimate, while $\tilde{L} \rightarrow 0$ in the q_T/μ_S region where resummation is not justified. The replacement is also useful in implementing a matching procedure at intermediate and large values of q_T , see for instance ref. [21] for details.

When approaching the opposite region, $\mu_S b \gg 1$, the functions $\bar{g}^{(n)}$ in eqs. (10)-(12) become singular, namely at $\bar{\lambda} \rightarrow 1$ ($\bar{\lambda} \equiv (1/\pi)\beta_0\alpha_s(\mu_S^2) \log(\mu_S^2 b^2/b_0^2)$). In terms of the impact parameter, this gives:

$$b^2 \rightarrow b_L^2 = \frac{b^2}{\mu_S^2} \exp\left(\frac{\pi}{\beta_0\alpha_s(\mu_S^2)}\right), \quad (32)$$

which in turns corresponds to $b_L \sim 1/\Lambda_{QCD}$, where Λ_{QCD} is the scale of the Landau pole in QCD. This singularity signals the onset of non-perturbative phenomena at very small transverse momentum scales $q_T \sim \Lambda_{QCD}$, where even formula (4) breaks down. This kind of singularity is a common feature of all-order resummation formulae of soft

gluon contributions (see also [21]), and has to be regularised. In the current reSolve implementation, we follow the prescription suggested in ref. [23], which freezes the integration over b below a fixed upper limit via the substitution:

$$b \rightarrow b_* = \frac{b}{\sqrt{1 + b^2/b_{lim}^2}}, \quad (33)$$

with $b_{lim} = \frac{b'_0}{q} \exp\left(\frac{1}{2\alpha_s\beta_0}\right)$ and $b'_0 = 2 \exp\left[\gamma \frac{q}{\mu_s}\right]$.

Admittedly, if non-perturbative contributions are sizeable, they should be properly included. In reSolve, we include them according to the simple parametrization suggested in [21], multiplying the b -space integrand function by a non-perturbative factor, S_{NP} , which includes a Gaussian smearing:

$$S_{NP} = \exp\left(-g_{NP}^c b^2\right), \quad (34)$$

where $c = q, g$, that is, different factors can be used for $q\bar{q}$ - and gg -initiated processes. Variations of the g_{NP}^c (these variables are called `ggnp` and `gqnp` in the code) can then be used to estimate the impact of non-perturbative corrections.

Finally, let us comment briefly about the z_1 and z_2 integrals. Following refs. [21, 24], we don't do these directly, but rather switch to Mellin space, defining

$$W_{N_1, N_2}^F(\mathbf{b}, \dots) = \int_0^1 dx_1 x_1^{N_1-1} \int_0^1 dx_2 x_2^{N_2-1} W^F(\mathbf{b}, \dots) \quad (35)$$

where remember x_1 and x_2 were defined in eq. (5). The double Mellin transformation turns the $dz_1 dz_2$ convolution integrals in eq. (4) into simple products [24], making them trivial to perform. At the end, the W_{N_1, N_2}^F function is transformed back to the physical x_1, x_2 space using a simple and rapidly converging Gaussian quadrature. The only downside of this approach is that the Mellin transforms of the PDFs evaluated at arbitrary complex values of the Mellin variables N_1 and N_2 are needed to fully define W_{N_1, N_2}^F . Since the PDF collaborations typically provide the $f_{a/h}$ functions as numerical grids in momentum fraction (as opposed to Mellin) space, the PDF Mellin moments have to be calculated inside reSolve itself, which requires a fit of the PDFs themselves to an analytic form whose Mellin transform is known or calculable. The strategy used in this case by reSolve was described in more detail in section 3.3.

5.3. The resummation scheme

The various coefficients $A_a^{(n)}$, $B_a^{(n)}$, C_{ab}^n and $H_c^{(n)}$ in eq. (4) are not actually uniquely defined. Indeed, the formula is invariant under the transformations [15]

$$\begin{aligned} H_c^F(\alpha_s, \dots) &\rightarrow H_c^F(\alpha_s, \dots)(h_c(\alpha_s))^{-1}, \\ B_c^F(\alpha_s) &\rightarrow B_c(\alpha_s) - \beta(\alpha_s) \frac{d \log h_c(\alpha_s)}{d \log \alpha_s}, \\ C_{ab}(z, \alpha_s) &\rightarrow C_{ab}(z, \alpha_s)(h_c(\alpha_s))^{1/2}, \\ G_{gb}(z, \alpha_s) &\rightarrow G_{gb}(z, \alpha_s)(h_g(\alpha_s))^{1/2}, \end{aligned} \quad (36)$$

where $h_c(\alpha_s) = 1 + O(\alpha_s)$ are arbitrary perturbative functions (with $h_q(\alpha_s) = h_{\bar{q}}(\alpha_s)$). A specific choice of the coefficients is dubbed a *resummation scheme*. The most often used scheme is the so-called *hard scheme*, which is defined by the fact that, order by order in perturbation theory, the coefficients $C_{ab}^{(n)}(z)$ for $n \geq 1$ do not contain any $\delta(1-z)$ term. An alternative possibility is using the freedom in eq. (36) to set $H_q^{F(n)} = 0$ and $H_g^{F'(n)} = 0$ for $n \geq 1$ in two specific processes, a $q\bar{q}$ -initiated F and a gg -initiated one F' . This can be dubbed the *F-F' scheme*. For reasons related to its historical development, the current version of reSolve is defined in the *DY-H scheme*, meaning that we set $H_q^{(n)}$ to 0 for the Drell-Yan process (DY) and $H_g^{(n)}$ to 0 for Higgs production (H). We plan to switch to the arguably more physical *hard-scheme* in one of the upcoming releases in the near future.

5.4. Phase-space definition of the final state F

The definition of the full phase space of the final state F , obviously needed for a full calculation, implies a subtlety. In fact, the four variables M , \mathbf{q}_T and y uniquely define the *total* 4-momentum of F . The Ω variables then define the individual momenta of the particles in F (for instance, in the specific case of the diphoton production process, the azimuthal and polar angles θ_{CM} and φ_{CM} of one of the photons in the diphoton center-of-mass are used to fully define both photons momenta). At this point, however, in order to obtain the $d\hat{\sigma}_{c\bar{c}}^{F,LO}$ and H_c^F coefficients, also the momenta of the initial state partons c , \bar{c} are needed. As formally the initial state partons have vanishing transverse momenta, this creates a mismatch between initial and final state total momentum. This is related to the fact that the all-order emissions which are resummed in (4) are factorized in the limit $q_T \rightarrow 0$, so that the “hard” matrix elements they multiply (which give rise to $d\hat{\sigma}_{c\bar{c}}^{F,LO}$ and H_c^F) nominally have $q_T = 0$. This is another ambiguity which is unavoidable in any transverse momentum resummation calculation, and in fact only has an impact when q_T is sizeable, that is, when the resummed approximation to the differential cross-section is losing validity anyway. From the practical point of view of the code, various approaches can be used. Arguably the simplest is to forgo momentum conservation altogether and simply set $q_T = 0$ when defining initial state momenta. In reSolve, we follow the slightly more involved (and arguably more physical) strategy of refs. [25] and [1], where the initial state partons are assigned a fictitious transverse momentum proportional to q_T in order to preserve momentum conservation.

6. Validation and Results

In order to validate this completely new code, great care and attention was used during its development to compare the outputs of as many of the various routines included as possible against similar routines and calculations in private programs. These private programs include the `2gres` code (version 2.2), which has been used in the production of results for previous papers in refs. [1] and [2], and the `DYRes` code (version 1.0) [3, 16]. In addition to these checks, extensive physical checks were performed - ensuring the correct events were cut, ensuring no negative cross-sections were produced for any event (which would be a sign of running into regimes where our transverse momentum resummation formulation was invalid), analysing the sudakovs and hard factors produced, and many other checks. A selection of these checks are presented here, for both the diphoton case and the Drell-Yan case. We conclude this section by commenting on the speed of the program, which is often a limiting factor for such Monte Carlo reliant programs.

6.1. Diphoton Results

For the validation of the program for the diphoton process ($process = 1$ in the input file), first of all we ensure that the program produces the correct output for the Born-level process by comparison against known results, this is described in Section 6.1.1. Following this, we compare differential cross-sections in both invariant mass and in transverse momentum, for two main test files, with the private `2gres` (version 2.2) program. These are detailed in Section 6.1.2. Finally, we demonstrate the validation of `reSolve` for a setup for which we compare against the `2gres` program and experimental data, this is described in Section 6.1.3.

6.1.1. Born Check

The first validation is the comparison against known results in previous papers, in particular in Table 1 in [26] for the Born cross-section. This reference details the production cross section for diphotons plus jets given a typical set of kinematical cuts applied in ATLAS and CMS analyses [27] [28]. These inputs are $\sqrt{s} = 14\text{TeV}$, $2\mu_S = \mu_R = qq$, $\mu_F = 113\text{GeV}$, $qq_{min} = 20\text{GeV}$, $qq_{max} = 250\text{GeV}$, q_T is unimportant as this is at leading-order (LO), $-2.5 < \eta < 2.5$, no gg box as we are at leading order, $etacut = 2.5$, no “crack” in the detector, $pT1cut = 40\text{GeV}$, $pT2cut = 25\text{GeV}$ and $Rcut = 0.4$. The input file used is provided with the code as `input/Diphoton_Born_LHC.dat`. We can only use this as a test of the LO result as beyond LO not only the resummed piece of the cross-section is required, but also the finite piece. Nonetheless for LO we obtain a total cross-section of 5.708 ± 0.008 pb whilst the value given in the paper was 5.712 ± 0.002 pb, therefore we are consistent.

6.1.2. Validation of `reSolve` against `2gres` code

Differential cross-sections in both invariant mass and transverse momentum were produced for two main test files and compared with the private code `2gres`, this has been previously used for similar transverse momentum resummation calculations for diphotons. It should be noted that a small error was found in the old `2gres` code around the Jacobian and subsequently corrected before undertaking this comparison. The test files chosen are the `Diphoton_NNLO_test_1.dat` and `Diphoton_NNLO_test_2.dat` provided with the program and with the paper. The inputs of these two tests are summarised in Table 1. These test files reflect common cuts, invariant mass ranges and transverse momentum ranges used for diphoton measurements, at 14TeV and 8TeV for `test1` and `test2` respectively. Therefore this comparison is of results produced by the new `reSolve` program in the expected regions of application.

Figure 18 shows the comparison between `reSolve` and the previous private program `2gres` for the `test1` inputs with 500,000 events, with figure 18a showing the invariant mass spectrum and figure 18b showing the transverse momentum spectrum. Excellent agreement is seen in both cases with the two programs agreeing within the errors shown. First of all consider the invariant mass spectrum, the invariant mass region for the `test1` inputs is $QQ_{Min} = 80\text{GeV}$ to $QQ_{Max} = 160\text{GeV}$, this is exactly the region over which we have non-zero cross-section, demonstrating the cuts are being implemented correctly. Meanwhile, the shape of the distribution is as expected, rising sharply above 80GeV once within the invariant mass region and peaking at the lower end of the invariant mass range. This occurs because as the invariant mass increases the phase space for the production of a higher diphoton invariant

Test file	test_1	test_2
Process	1	1
Order	2	2
CM.energy (TeV)	14	8
μ_S, μ_R, μ_F (GeV)	qq/2, qq, 113	qq/2, qq, 85
QQ_Min, QQ_Max (GeV)	80, 160	50, 110
QT_Min, QT_Max (GeV)	0, 120	0, 100
η _Min, η _Max	-2.5, 2.5	-2.37, 2.37
gg box (boxflag)	No (0)	No (0)
etaCut	2.5	2.37
crack1, crack2	1.37, 1.37	1.37, 1.37
pT1cut, pT2cut (GeV)	40, 25	40, 30
Rcut	0.4	0.4

Table 1: The two test files used for validation for the diphoton process in reSolve both against the private code 2gres and internally, ensuring different numbers of PDF fits, different integrators and different numbers of iterations all produce consistent results. The files are the Diphoton_NNLO_test_1.dat and Diphoton_NNLO_test_2.dat provided with the program.

mass decreases. Meanwhile in figure 18b we obtain the characteristic transverse momentum spectrum shape, with the spectrum approaching zero at $q_T = 0$ GeV, peaking sharply just above 0 GeV in the region where transverse momentum resummation is most important, and again reducing as the q_T increases. Furthermore we see a slight rise from 65 GeV peaking around 80 GeV, this is a kinematical effect caused by the q_T cuts applied on each photon being 40 GeV and 25 GeV. The q_T spectrum is then correctly cut off at the $q_T = 120$ GeV upper bound. Again the agreement between the new public reSolve program and the previous private 2gres code is excellent within the errors. Moreover, for this test1 input file the total cross-section is also in agreement; reSolve obtains $7.68 \pm 0.03 pb$, whilst 2gres obtains $7.67 \pm 0.03 pb$.

Figure 19 illustrates a similar comparison for the test2 inputs, again with 500,000 events. Figure 19a shows the invariant mass spectrum comparison, whilst figure 19b is the transverse momentum spectrum comparison. For test2 the invariant mass range is 50 GeV to 110 GeV as given in Table 1, this region is clearly visible in figure 19a. The transverse momentum spectrum in figure 19b again shows the correct behaviour, going towards 0 at $q_T = 0$ GeV, peaking sharply just above 0 and then falling away and cutting off at the edge of the q_T region at 110 GeV. Again the peak around 80 GeV results from an increase in the phase space at this point, due to the cuts applied, as illustrated in the invariant mass spectrum in figure 19a. Once more the total cross-section is also in good agreement between the two programs, with reSolve obtaining $2.54 \pm 0.01 pb$ and 2gres $2.56 \pm 0.01 pb$.

Another check that can be performed, is to compare the spectra seen with different width bins for a given input file, such a comparison is presented briefly here and validates the events and the histogram routine, see figure 20. Figures 20a, 20b, 20c illustrate the invariant mass spectrum obtained for the test1 input file for 50, 100 and 200 bins respectively spread evenly over the region $0 \rightarrow 180$ GeV. Given the events appear in a mass range 80 GeV to 160 GeV this means there are approximately 22, 44 and 89 bins over this region for the respective binnings. All three plots of course demonstrate the same overall shape given they are different presentations of the same generated events. In general the behaviour of the histogrammer here is exactly as expected, with more consistent size errors seen in figure 20a which has the largest bins. Figures 20b and 20c show yet narrower bins and as a result we start to see statistical variations in the shape of the distribution, by figure 20c in particular the errors on the bins are becoming larger as a result of fewer events per bin, and as a result we see occasional bins - such as that around 88 GeV -

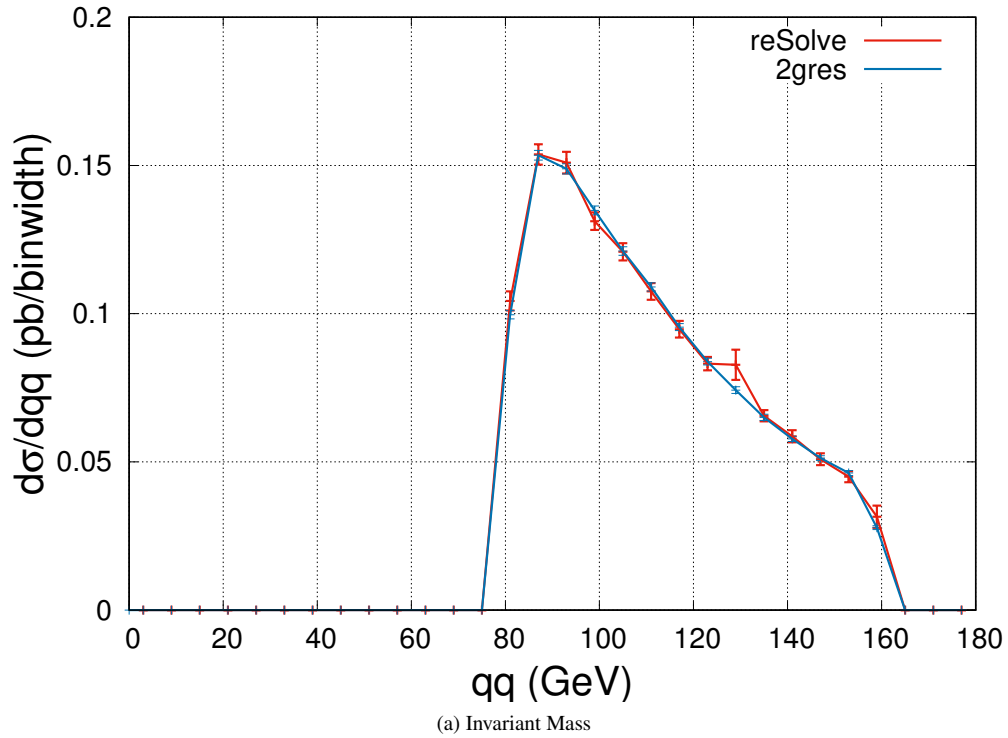
which show large discrepancies from the surrounding bins. To determine the optimal bin size, one must balance the competing needs for small enough bins not to smear out any interesting behaviour in the spectra, and for large enough bins that there are sufficient events per bin to make the statistical fluctuations and errors per bin palatable, these themselves can be reduced by longer run times. This behaviour is all exactly as one would expect, providing some confirmation of the events generated and, in particular, of the histogram package included within the reSolve program.

6.1.3. ATLAS Validation

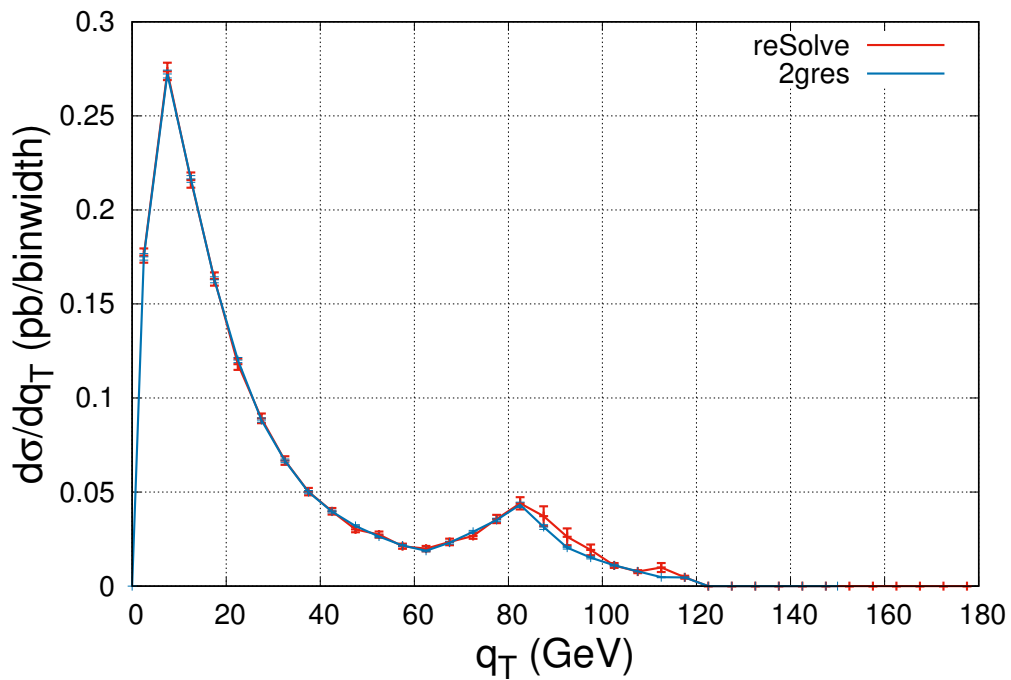
An additional validation that was performed was to use reSolve to reproduce the events and corresponding total cross-section, invariant mass spectrum and transverse momentum spectrum for the inputs listed in [2]. These inputs are provided in the reSolve program package in ‘‘input/Diphoton_Atlas_A.dat’’ and are: diphoton process at NNLO+NNLL, MSTW NNLO PDFs (PDF_flag = 82), centre of mass energy 8TeV, pp collisions; square factorisation, resummation and renormalisation scales set proportional to qq^2 - i.e. $\mu_F^2 = \mu_R^2 = qq^2$ and $\mu_S^2 = \frac{qq^2}{4}$; 2,000,000 iterations with $nstart = nincrease = 10000$, $0 < QQ < 500$ GeV, $0 < QT < 150$ GeV, $-2.37 < \eta < 2.37$, gg box included, $etacut = 2.37$, $crack1 = 1.37$, $crack2 = 1.56$, $pT1cut = 40$ GeV, $pT2cut = 30$ GeV and $Rcut = 0.4$. As a result of the large invariant mass range considered, 5 PDF fits are used across the allowed range to improve accuracy. For these inputs the 2gres program has been previously validated against experimental data.

The total cross-section produced by the reSolve program for these inputs was 6.188 ± 0.013 , compared with 6.18 ± 0.02 from 2gres, this therefore indicates very good agreement. The invariant mass and transverse momentum spectra also are consistent and are given in figure 21 and figure 22 respectively. The q_T plot shows agreement at both the low q_T end and higher q_T end, with the position and height of the peak in the spectrum agreeing within the errors, meanwhile as one increases q_T the differential cross-section reduces as expected. The spectrum peaks again slightly around 80GeV as a result of increasing phase space available beyond this q_T , as demonstrated in the invariant mass spectrum raising rapidly above 80GeV, this is a threshold effect caused by the cuts - with cuts on pT1 and pT2 of 40GeV and 30GeV invariant masses of less than 80GeV are difficult to attain.

As well as validation against just the resummed part of the transverse momentum differential cross-section, we added the finite piece - as previously calculated by 2gNNLO [26] - in order to validate against the total transverse momentum differential cross-section. We then compared our reSolve results with the matched finite piece added (with Monte Carlo error only and only for the resummed part of the differential cross-section) against those of 2gres (with errors shown indicating scale variation) and ATLAS experimental results [29]. The comparison plot is shown in figure 23. The figure demonstrates the excellent agreement between the reSolve program and the data previously calculated with 2gres, the main difference coming in the 80GeV - 100GeV region where we observe a bump in the resummed cross-section (see for example figure 22), the difference here is because the previous 2gres program had its effect - indeed the new reSolve program now shows better agreement with the experimental data in this region than the 2gres code did previously. The bump, as described previously, is a threshold effect arising from the cuts applied. There are small differences between the reSolve and 2gres predictions and the experimental results at intermediate transverse momenta, we expect these are within errors once all errors - including those from the pdfit, scale variation, Monte Carlo, matching and other sources are taken into account; in any case the current version of reSolve determines only the resummed piece of the differential cross-section and so it is the excellent agreement at low transverse momenta which is the focus of our validation.

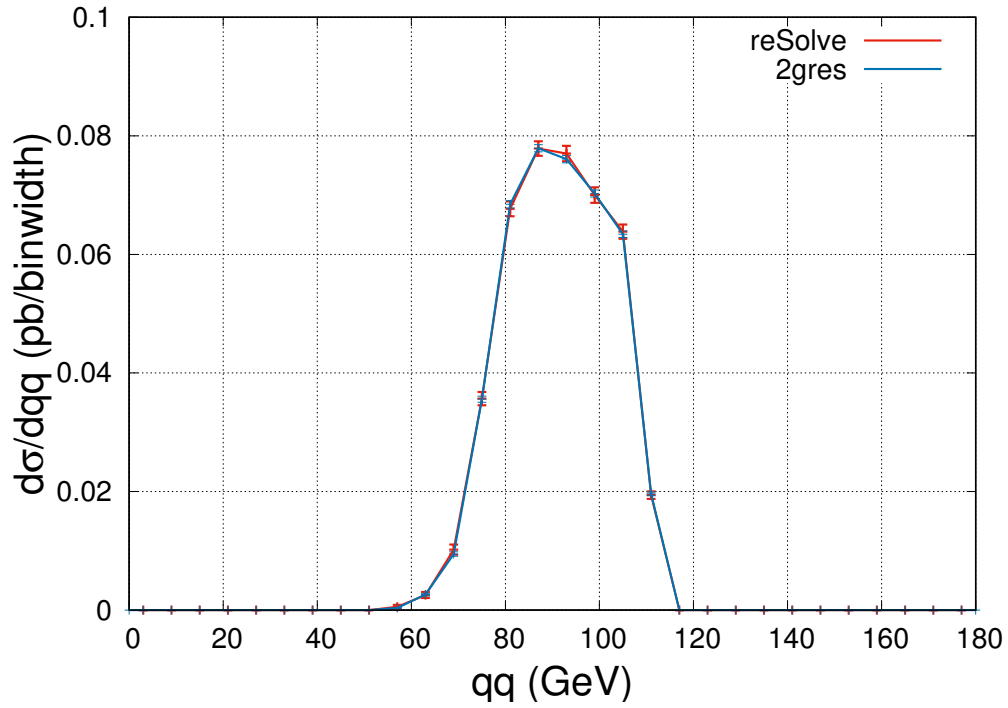


(a) Invariant Mass

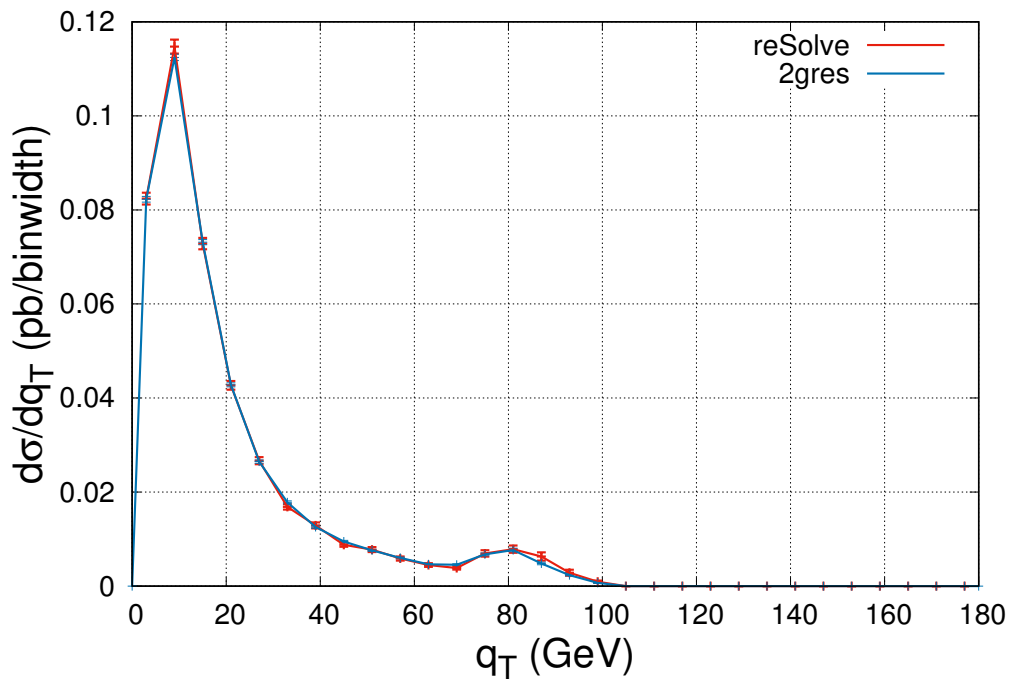


(b) Transverse Momentum

Figure 18: Comparison plots of the diphoton differential cross-sections with invariant mass and transverse momentum for the `test1` inputs as listed in Table 1 for the `reSolve` program against the previous private program `2gres`. These just include the resummed part of the differential cross-section, not the finite piece.

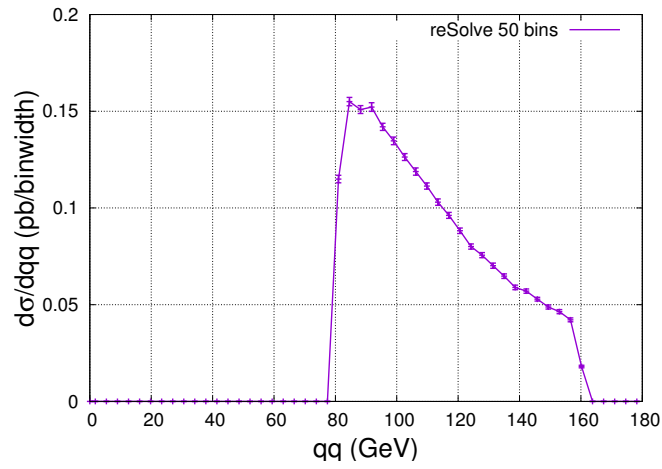


(a) Invariant Mass

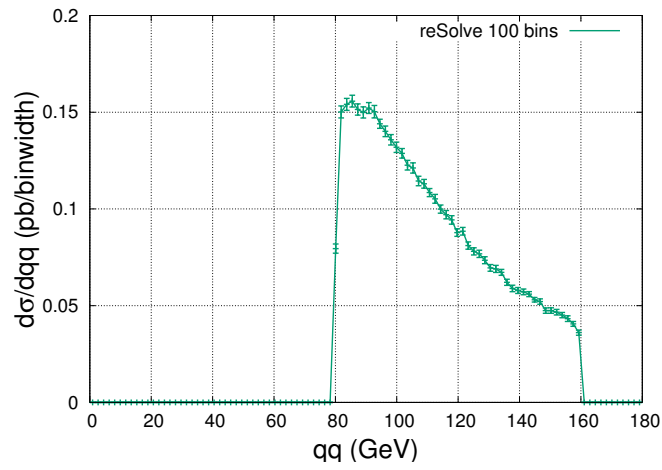


(b) Transverse Momentum

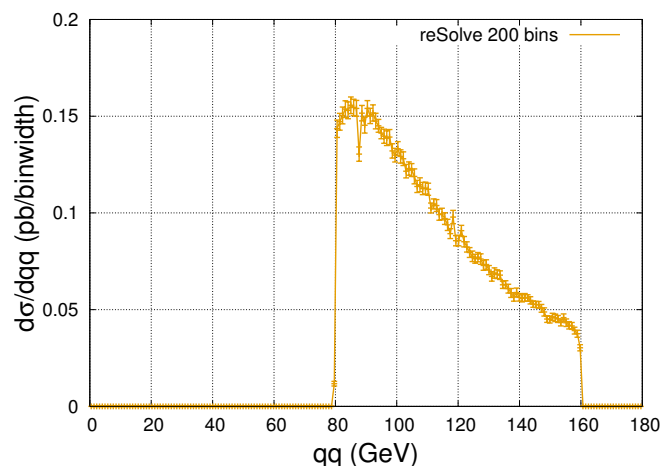
Figure 19: Comparison plots of the diphoton differential cross-sections with invariant mass and transverse momentum for the `test2` inputs as listed in Table 1 for the `reSolve` program against the previous private program `2gres`. These just include the resummed part of the differential cross-section, not the finite piece.



(a) 50 bins



(b) 100 bins



(c) 200 bins

Figure 20: Comparison of the behaviour of the diphoton differential cross-section with invariant mass histograms for the test1 input file with different binnings; with 50, 100 and 200 bins respectively spread across the invariant mass range 0GeV to 180GeV.

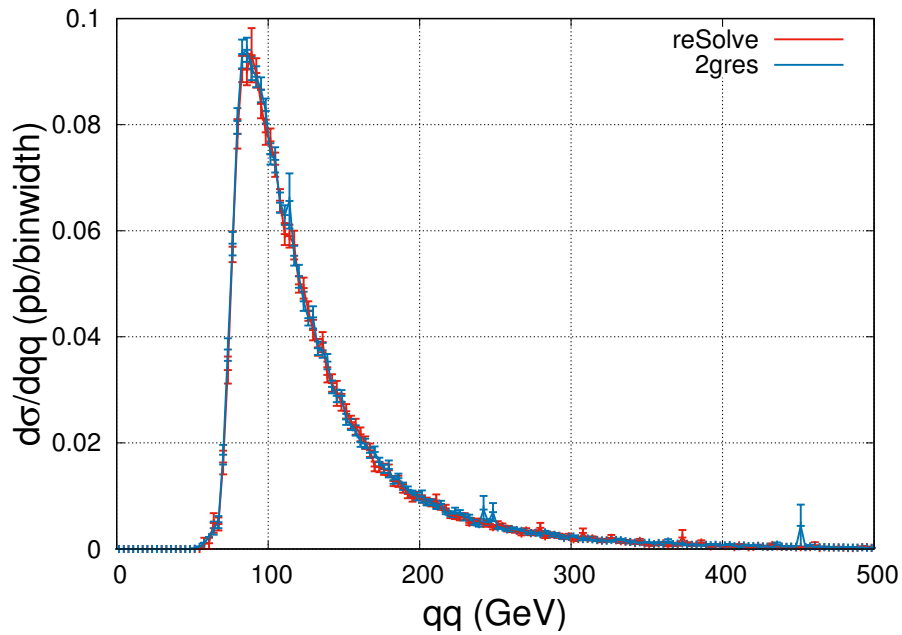


Figure 21: Invariant mass spectrum for diphoton production for the `Diphoton.Atlas.A.dat` input file provided with the program, and whose inputs are also listed in the text, as produced by the `reSolve` program and the previous private program `2gres`, which was used in the work in [1] and [2]. Only the resummed part of the differential cross-section is shown.

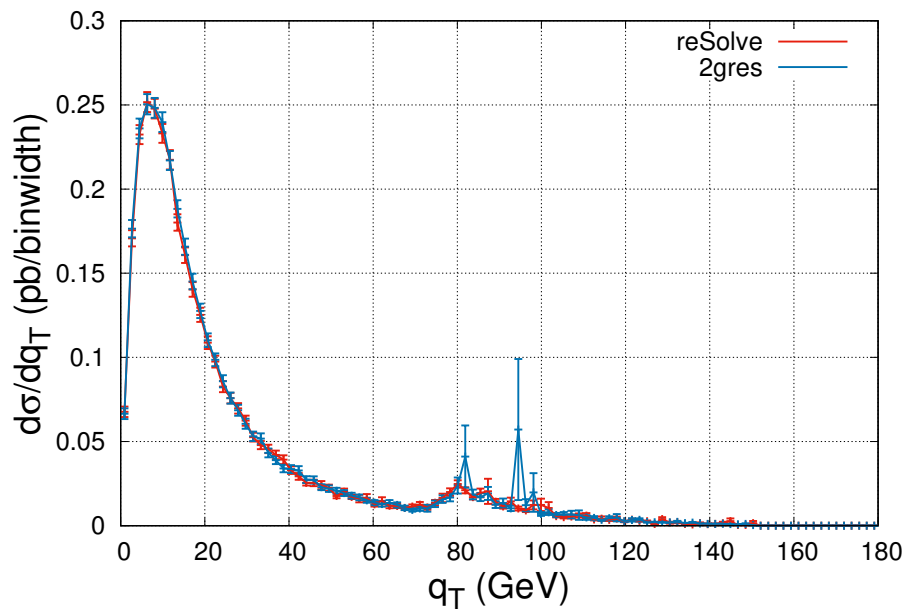


Figure 22: Transverse momentum spectrum for diphoton production for the `Diphoton.Atlas.A.dat` input file provided with the program, and whose inputs are also listed in the text, as produced by the `reSolve` program and the previous private program `2gres`, which was used in the work in [1] and [2]. Only the resummed part of the differential cross-section is shown.

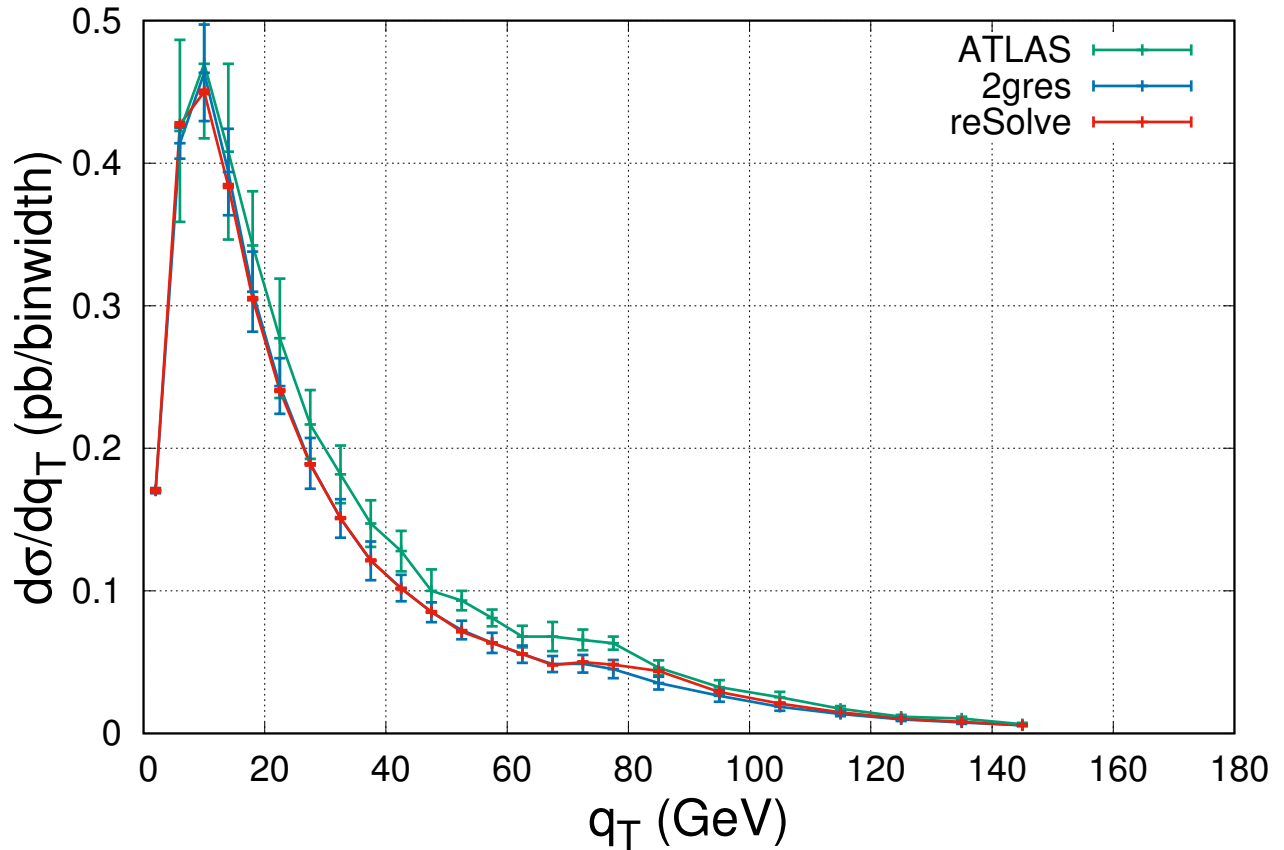


Figure 23: Transverse momentum spectrum, including resummed and finite pieces, for diphoton production for the `Diphoton.Atlas.A.dat` input file provided with the program, and whose inputs are also listed in the text, as produced by the `reSolve` program (with finite pieces from `2gNNLO` [26]) and the previous private program `2gres`, which was used in the work in [1] and [2], also shown are the ATLAS experimental results and corresponding errors. The error bars for `2gres` show the scale variation error, which is the dominant error. This is not calculable in `reSolve` as we do not include the finite part of the cross-section in this first version of the program, therefore the `reSolve` error bars are the Monte Carlo errors from the resummed part only.

6.2. Drell-Yan Results

In order to validate the Drell-Yan process, we perform similar checks as for the diphoton case. First we focus on the Born cross-section, comparing against known results in Section 6.2.1. We then present a comparison against the program DYRes (version 1.0) [16] in various differential variables, including the transverse momentum spectrum and rapidity(η) spectrum; in Section 6.2.4 we provide further differential plots, comparing the rapidity, transverse mass (m_T), and p_T^{\min} and p_T^{\max} distributions against those in [30]. Here further plots of the invariant mass spectrum for the Drell-Yan pair and additional differential distributions are also provided in order to confirm the qualitative behaviour of the results is as expected.

6.2.1. Born Validation

In order to add a new process (in this case all the Drell-Yan processes) to the reSolve program, as described in Section 4.3, we needed only to alter the non-resummation part of the code, providing a new Born-level cross-section for the added process. For this reason, the key test to perform to validate the added process is to confirm it produces the Born cross-section correctly. In order to confirm this, reSolve was run for three different setups, setup 1 is for Z/γ^* at the Tevatron, setup 2 is for Z On-Shell at the 14TeV LHC, and setup 3 is for W^\pm at the Tevatron; the full invariant mass ranges, rapidity ranges, scale and cut setups are listed in Table 2. These benchmarks were also used for NLO+NNLL and NNLO+NNLL comparisons and plots in later sections (where the `resum_flag` must be changed to 1 and the `order` changed accordingly), therefore the transverse momentum ranges are given, however note that there is no transverse momentum (q_T) at Born-level so the q_T range set is unimportant for Born comparisons. These benchmarks were chosen as they reflect the full range of Drell-Yan processes added and there are results quoted in [30] to compare against, in addition results were also obtained from the private DYRes (version 1.0) program [16] and from the MCFM (version 8.1) (“Monte Carlo for FeMtobarn processes”) program [31, 32, 33, 34] for comparison.

Test file	setup 1	setup 2	setup 3
Process	2	2	2
resum_flag	0	0	0
DYProcess	5	4	3
DYnarrowwidthapprox	0	1	0
Order	0	0	0
pdf_flag	80	80	80
CM_energy (GeV)	1960	14000	1960
ih1	1	1	1
ih2	-1	1	-1
μ_S, μ_R, μ_F (GeV)	All $m_Z = 91.187$	All $m_Z = 91.187$	All $m_W = 80.398$
QQ_Min, QQ_Max (GeV)	70, 110	70, 110	0, 200
QT_Min, QT_Max (GeV)	0, 200	0, 200	0, 200
η _Min, η _Max	-3, 3	-10, 10	-3, 3
crack1, crack2	1.37, 1.37	1.37, 1.37	1.37, 1.37
pT1cut, pT2cut (GeV)	20, 20	0, 0	N.A., N.A.
eta1cut = eta2cut	2	10	N.A.
pTcut (GeV)	N.A.	N.A.	20
pTmisscut (GeV)	N.A.	N.A.	25
etaecut	N.A.	N.A.	2
tmasscut (GeV)	N.A.	N.A.	0

Table 2: The three test files used for validation of the Born cross-section for the Drell-Yan added processes in reSolve against the results in [30] and results from the program MCFM [31, 32, 33, 34], as well as against those of DYRes [16]. The pdf sets used are the MSTW2008 LO PDFs, therefore `pdf_flag` = 80. The files are the `yZ_Born.Tevatron.dat`, `Z_OnShell_Born.LHC.dat` and `Wpm_Born.Tevatron.dat` provided with the program. Similar inputs were used for the NLO and NNLO tests, these files are also provided with the reSolve program.

With these inputs as listed in Table 2, the reSolve program obtains the following Born cross-sections: for the Z/γ^* Tevatron setup 1, reSolve calculates $\sigma_{LO} = 103.37 \pm 0.06 pb$, compare this with the results in [30] where $\sigma_{LO} = 103.37 \pm 0.04 pb$, whilst MCFM [31, 32, 33, 34] obtains $\sigma_{LO} = 103.34 \pm 0.04 pb$; for the on-shell Z LHC 14TeV setup 2, reSolve calculates $\sigma_{LO} = 1758.9 \pm 1.1 pb$, for comparison with $\sigma_{LO} = 1761 \pm 1 pb$ in [30] and $\sigma_{LO}(pp \rightarrow Z \rightarrow l^+l^-) = \sigma_{LO}(pp \rightarrow Z) \times BR(Z \rightarrow l^+l^-) = 52.3197 \pm 0.0049 nb \times 0.03366 = 1761.1 \pm 0.1 pb$; finally for the W^\pm Tevatron setup 3, reSolve calculates $\sigma_{LO} = 1160.4 \pm 0.7 pb$ for comparison with $\sigma_{LO} = 1161 \pm 1 pb$ in [30], whilst MCFM obtains $\sigma_{LO}(ppb \rightarrow W^\pm) = 1187.87 \pm 0.43 pb$. Therefore there is good agreement between the reSolve program for these Born cross-sections with known calculations for all three setups.

Given this incorporates the majority of the process dependence of the formalism used for the reSolve program, this indicates the new Drell-Yan processes are working correctly. Nonetheless, we demonstrate many further results and validations over the next few sections.

6.2.2. Total Cross-Sections at LO, NLO and NNLO

Before any of the differential cross-sections are analysed, first we check that the total cross-sections are sensible for each order; leading order (LO) Born cross-section without resummation, next-to-leading order (NLO) cross-section with next-to-leading logarithm (NLL) resummation, and next-to-next-to leading order (NNLO) with next-to-next-to-logarithm (NNLL) resummation. The input files used are those for the 3 benchmark setups of Table 2 at each of the 3 orders; they are provided with the reSolve program and are `yZ_Born_Tevatron.dat`, `yZ_NLO_Tevatron.dat`, `yZ_NNLO_Tevatron.dat`, `Z_OnShell_Born_LHC.dat`, `Z_OnShell_NLO_LHC.dat`, `Z_OnShell_NNLO_LHC.dat`, `Wpm_Born_Tevatron.dat`, `Wpm_NLO_Tevatron.dat` and `Wpm_NNLO_Tevatron.dat`. The results obtained from reSolve are compared with known results calculated in [30], the difference being that, as-of-yet, reSolve does not include the finite part of the cross-section, just the resummed part, therefore we expect our beyond LO results to be lower than in [30] but showing the same trend with NNLO > NLO > LO. The results are summarised in Table 3, those for LO were also given in the previous section. The agreement shown at leading-order is good for all three benchmark setups. Meanwhile the NLO and NNLO results behave as expected, with each successive order increasing the total cross-section as more contributions are added, whilst still being smaller than the known results which include the additional finite contributions. Moreover, the increases in the total cross-section are significant in going from LO to NLO(+NLL) but much smaller upon going to NNLO(+NNLL), exactly as seen in the known results.

		reSolve /pb	Known Result /pb
Z/γ^* setup 1	LO	103.37 ± 0.06	103.37 ± 0.04
	NLO(+NLL)	130.37 ± 0.10	140.43 ± 0.07
	NNLO(+NNLL)	130.40 ± 0.10	143.86 ± 0.12
Z On-shell setup 2	LO	1758.9 ± 1.1	1761 ± 1
	NLO(+NLL)	2009.1 ± 0.5	2030 ± 1
	NNLO(+NNLL)	2056.2 ± 3.0	2089 ± 3
W^\pm setup 3	LO	1160.4 ± 0.7	1161 ± 1
	NLO(+NLL)	1438.6 ± 1.2	1550 ± 1
	NNLO(+NNLL)	1465.4 ± 1.3	1586 ± 1

Table 3: Summary of the total cross-sections as calculated by reSolve compared with known results [30] for the three setups described in Table 2. The agreement is good between reSolve and the known results at LO and the behaviour beyond LO is expected, reSolve results beyond leading order are smaller as reSolve only includes the resummed part of the total cross-section, not the finite part which is important at larger transverse momentum q_T . The errors indicated in the table are Monte Carlo errors only and so are largely determined by the length of the run used and not indicative of the precision of the program. The reSolve input files used are included with the program.

6.2.3. Validation of reSolve against DYRes code

In order to validate the new Drell-Yan process added to reSolve, we can compare with the program DYRes, which is from the same series of programs as the 2gres program against which we compared the diphoton, and allows also for the calculation of the resummed contribution only.

First we consider the Z/γ^* Tevatron case of setup 1 (given previously in table 2). DYRes produces data for the transverse momentum spectrum and rapidity distribution, amongst others, so we may compare these directly with these distributions as produced by reSolve, the comparison plots are shown below in figures 24 and 25 respectively, both figures show the NNLO(+NNLL) spectra and demonstrate very good agreement with the DYRes program, thereby validating reSolve for this process. The total cross-section also agrees well, reSolve obtains $130.1 \pm 0.4 pb$ as detailed in Table 3, whilst DYRes (for the resummed piece only) obtains $130.3 \pm 1.2 pb$.

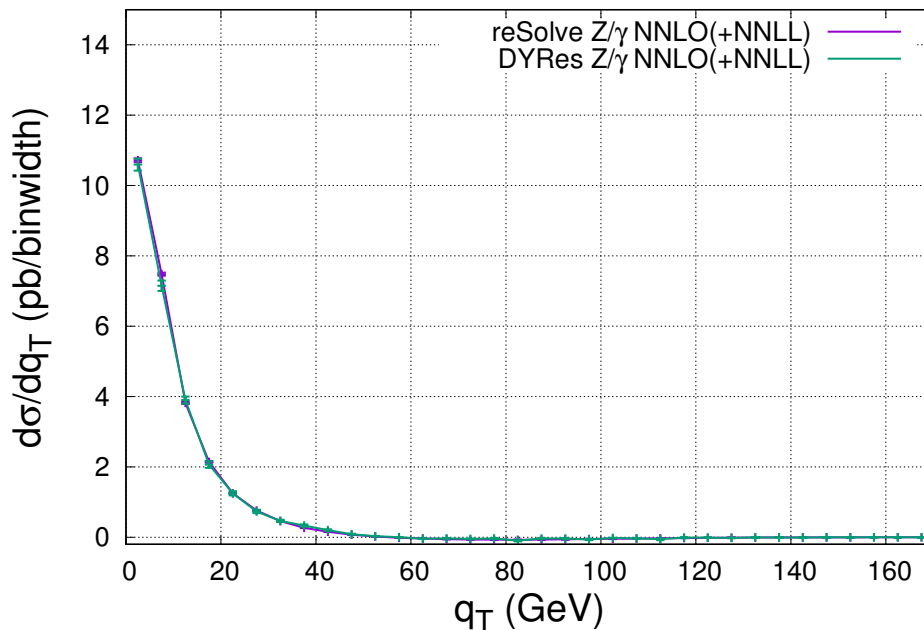


Figure 24: Transverse momentum spectrum, including only the resummed piece, for Drell-Yan production via neutral current Z or γ^* at NNLO (with NNLL resummation) for the setup 1 benchmark, as given in table 2. The agreement between the two programs is excellent, validating reSolve. The error bars are the Monte Carlo errors from the resummed part only and are largely a reflection on the length of the runs performed.

Similar comparisons can be performed for the on-shell Z LHC $14TeV$ case of setup 2 (given previously in table 2). Figure 26 demonstrates the agreement in the NLO(+NLL) and NNLO(+NNLL) transverse momentum spectra between reSolve and DYRes. The total cross-sections are also in agreement between reSolve and DYRes with the programs obtaining $2056.2 \pm 3.0 pb$ and $2050.5 \pm 2.1 pb$ respectively for the resummed pieces only. The transverse momentum spectra in both figures 24 and 26 do not show the expected behaviour of the differential cross-section tending to 0 at zero transverse momentum solely because of the binning (set to match DYRes), figure 31 demonstrates this expected behaviour in reSolve for the W^\pm case of setup 3, where finer binning has been used for the low q_T end of the spectrum to demonstrate this behaviour.

Finally, similar comparisons may be undertaken for the W^\pm channel, but are not included here. The overall resummed piece only NNLO results for reSolve and DYRes are $1465.4 \pm 1.3 pb$ and $1487 \pm 10 pb$, this agrees at the 2% level.

In general, in producing the theoretical predictions beyond leading-order there are a multitude of choices and methodology-linked effects which affect the precise output values of the two programs. In order to produce these comparisons of the reSolve and DYRes results we have sought to minimise these differences and thereby demonstrate the level of agreement of the codes. In general, results may show larger differences down to the exact choices of the running method for α_s , the nature of how higher transverse momenta are dealt with (reSolve essentially uses a step function by allowing the user to specify a q_T range, whereas DYRes gradually reduces the effects of higher q_T s via an arbitrarily-defined “switch”), the precise generation of phase-space points, how the η range is limited at its extremities, the precise nature of the pdf-fitting function (here we refer to not the pdfs - which of course are another

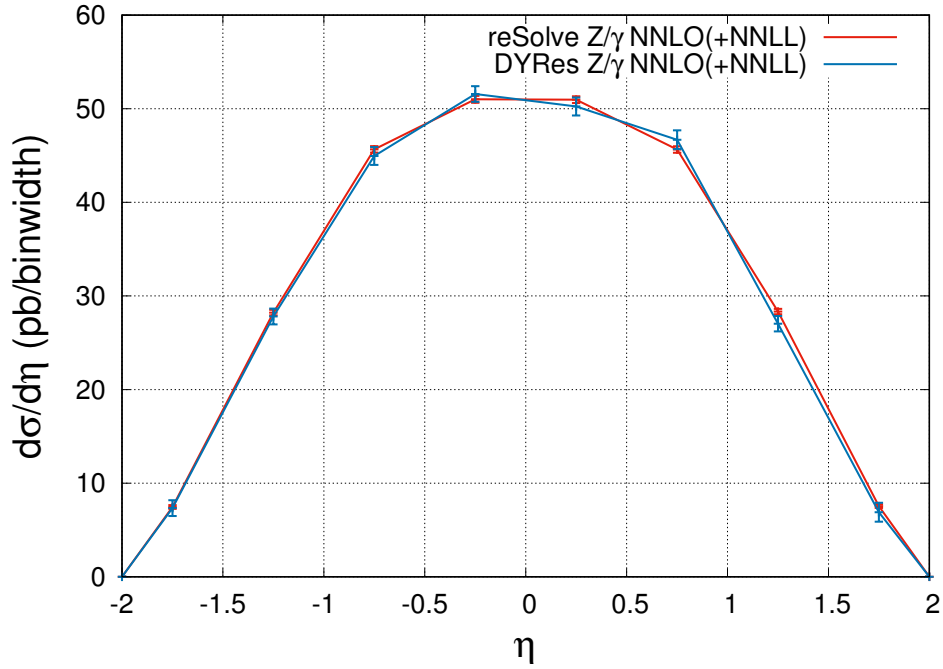


Figure 25: Rapidity spectrum, including only the resummed piece, for Drell-Yan production via neutral current Z or γ^* at NNLO (with NNLL resummation) again for the setup 1 benchmark, as given in table 2. The agreement between the two programs is good, further validating `reSolve`. The error bars are the Monte Carlo errors from the resummed part only and are largely a reflection on the length of the runs performed.

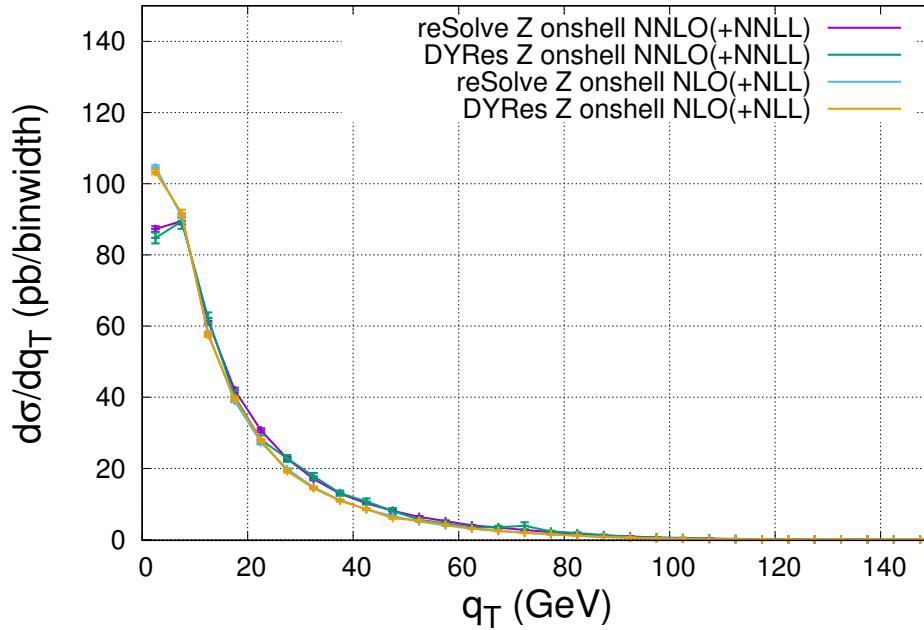


Figure 26: Transverse momentum spectrum, including only the resummed piece, for Drell-Yan production via neutral current Z onshell for the setup 2 benchmark, as given in table 2. The agreement between the two programs is good at both NLO(+NLL) and NNLO(+NNLL), further validating `reSolve`. The error bars are the Monte Carlo errors from the resummed part only and are largely a reflection on the length of the runs performed.

source of variation in theoretical predictions - but to the form of the fitting function itself chosen) and many others, which are chosen differently in the two programs. These choice differences were eliminated as much as possible in the comparisons presented here, nonetheless these effects tend to result in differences of order 5% and so this should be considered the accuracy of the predictions for a generic input².

6.2.4. Further Differential Spectra

Finally, we provide several spectra obtained for each of the three setups to demonstrate the agreement with the corresponding figures provided in [30] and to illustrate that the qualitative behaviour of the reSolve program is as expected.

First again consider the case of Z/γ^* in our benchmark setup 1, the p_T^{min} and p_T^{max} distributions for this setup were given in figure 2 of [30]. The corresponding spectrum, as produced by the new reSolve program, is provided below in figure 27, we provide only the NLO(+NLL) and NNLO(+NNLL) spectra as the transverse momentum of the system at LO is zero and so is meaningless within the formalism. The qualitative agreement between the reSolve spectra and the previous spectra is good, meanwhile the behaviour of the spectra is exactly as expected with the NNLO spectrum having a larger peak. Both the p_T^{min} and p_T^{max} distributions cut-off at 20GeV due to the applied p_T cut, the p_T^{min} spectrum peaks just below $\frac{m_Z}{2}$ and the p_T^{max} spectrum at just above $\frac{m_Z}{2}$. The p_T^{min} spectrum also cuts-off at 55GeV as the q_T range had an upper limit of 110GeV whilst the p_T^{max} spectrum continues above 55GeV , all this behaviour is exactly as expected. Small differences in the figures occur as our calculations include the resummed piece of the cross-section only.

For the on-shell Z case of benchmark setup 2, figure 1 of [30] provides the rapidity distribution, the corresponding distribution calculated by reSolve is given below in our figure 29, the agreement between the two is excellent, with the effects of the resummation beyond LO significantly increasing the cross-section between LO and NLO(+NLL), with NNLO(+NNLL) only offering a small additional correction.

Finally, for the case of W^\pm in our benchmark setup 3 the differential distribution provided in [30] is the transverse mass distribution in their figure 3, compare this with the same transverse mass distribution produced by reSolve in figure 28. Again, there is good agreement between the results, the leading order m_T distribution turns on at 50GeV because the W^\pm is produced at zero net transverse momentum, so without any additional radiation we require $p_T^{lepton} = p_T^{miss}$ therefore the p_T^{miss} cut of 25GeV sets the lower limit of the LO m_T distribution to 50GeV . Of course, this limit is not a hard limit beyond LO as additional radiation can carry away transverse momentum. For LO, NLO(+NLL) and NNLO(+NNLL) the m_T distribution peaks just below the W mass at around 80GeV as expected.

²In particular, the effects of the α_s running method and the q_T switch themselves are the largest differences seen and may cause differences themselves of up to 5%, the choices in DYRes raise the predictions by around this amount relative to the default choices in reSolve.

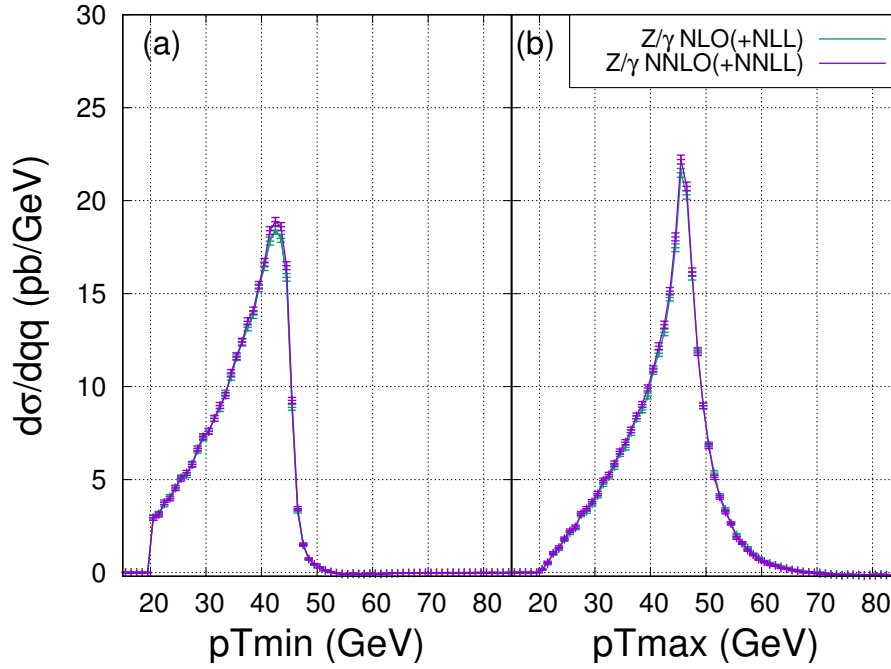


Figure 27: Minimum and maximum transverse momenta spectrum produced by reSolve for the two outgoing leptons produced, including only the resummed piece, for Drell-Yan production via neutral current Z or γ^* for the setup 1 benchmark, as given in table 2. This figure should be compared with figure 2 of [30], the agreement between the two results is good, with slight differences arising due to the implementation of the formalism, the results are consistent within uncertainties. The error bars are the Monte Carlo errors from the resummed part only and are largely a reflection on the length of the runs performed.

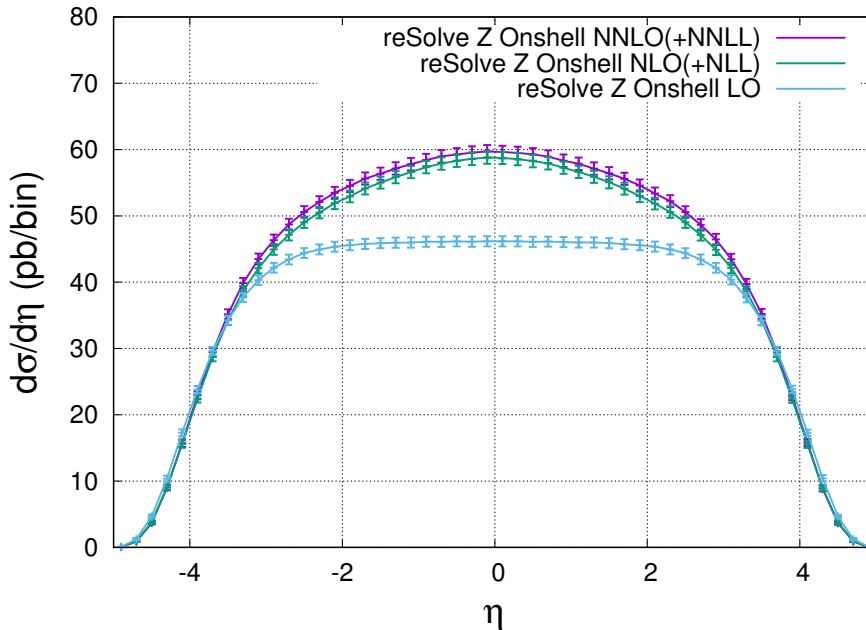


Figure 28: Rapidity distribution for the two outgoing leptons produced by the on-shell Z boson, including only the resummed piece, for Drell-Yan production via neutral current Z on-shell for the setup 2 benchmark, as given in table 2. This figure should be compared with figure 1 of [30], the agreement between the two results is excellent. The LO includes no resummation, whilst for beyond LO resummation is included, so NLO includes NLL resummation and NNLO includes NNLL resummation of logarithms. The error bars are the Monte Carlo errors from the resummed part only and are largely a reflection on the length of the runs performed.

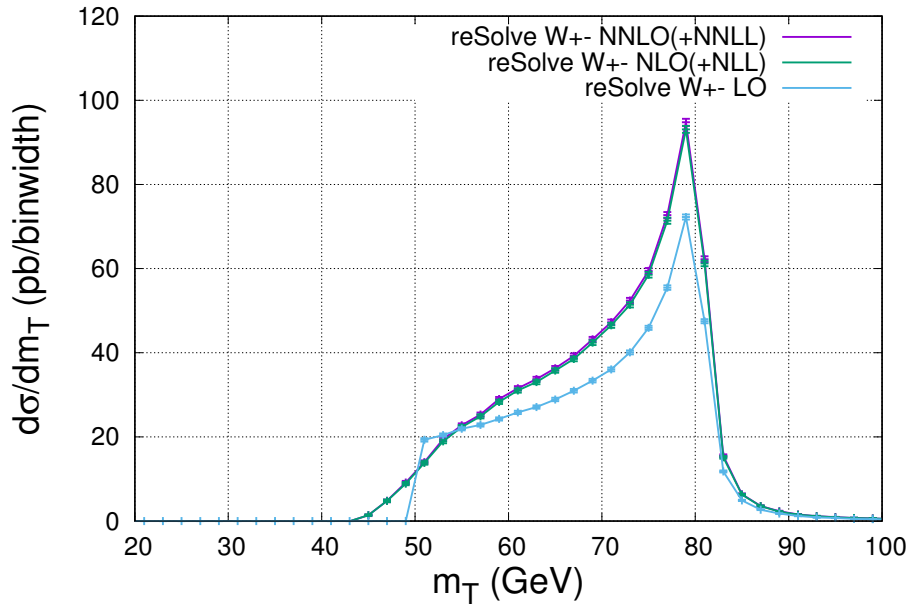


Figure 29: Transverse mass distribution for the W^\pm case of the setup 3 benchmark, including only the resummed piece, as given in table 2. This figure should be compared with figure 3 of [30], the agreement between the two results is good. The LO includes no resummation, whilst for beyond LO resummation is included, so NLO includes NLL resummation and NNLO includes NNLL resummation of logarithms. The error bars are the Monte Carlo errors from the resummed part only and are largely a reflection on the length of the runs performed.

In addition to these comparisons against the figures in the literature, many plots of the various differential distributions produced by reSolve for all three benchmark setups of table 2, and others, were extensively checked for any obvious problems during our validation of the program, figures 30-33 are a small subset of these, again all illustrate the exact expected behaviour.

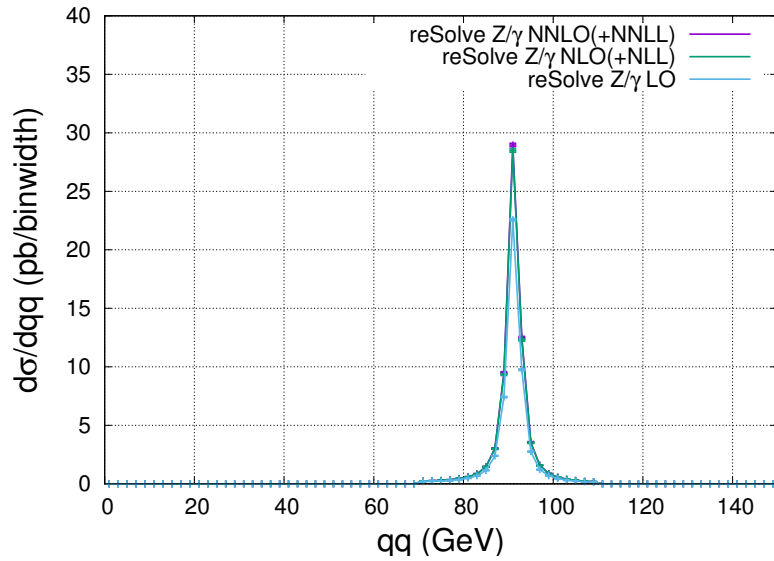


Figure 30: Invariant mass distribution for the neutral current Z/γ^* case of the setup 1 benchmark, including only the resummed piece. This figure illustrates the expected peaking of the invariant mass distribution around m_Z even in the case of Drell-Yan production which isn't specifically onshell, due to the enhancement around $q^2 = m_Z^2$ from the Z propagator. As previously, the NLO and NNLO include resummation to their corresponding orders (NLL and NNLL respectively). The error bars are the Monte Carlo errors from the resummed part only and are largely a reflection on the length of the runs performed.

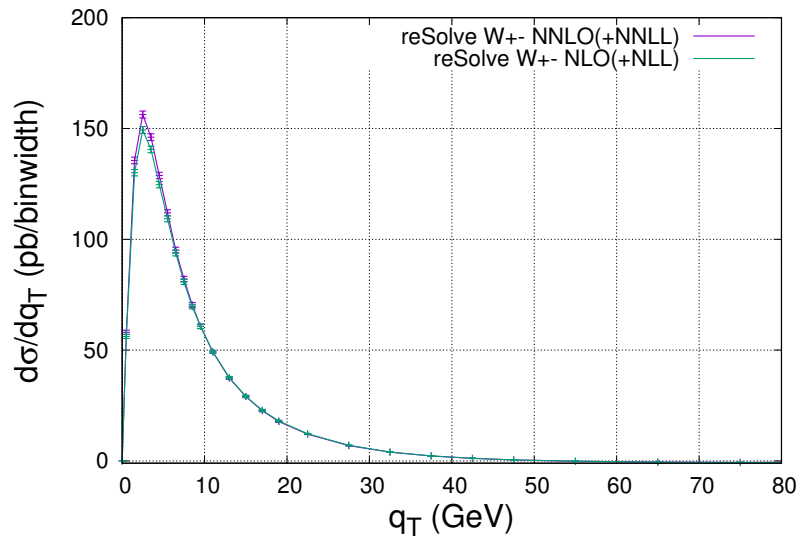


Figure 31: Transverse momentum distribution for the charged current W^\pm case of the setup 3 benchmark, including only the resummed piece. This figure illustrates the expected peaking of the transverse mass distribution near 0 and the fact that, despite this, the differential cross-section is still 0 at zero transverse momentum. As previously, the NLO and NNLO include resummation to their corresponding orders (NLL and NNLL respectively). The error bars are the Monte Carlo errors from the resummed part only and are largely a reflection on the length of the runs performed.

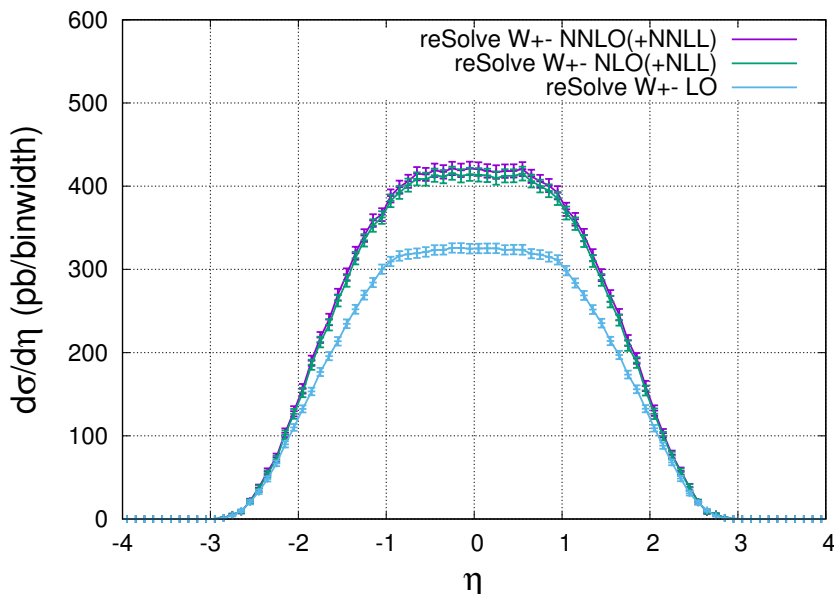


Figure 32: Rapidity distribution for the charged current W^\pm case of the setup 3 benchmark, including only the resummed piece. The distribution is correctly zero outside the range $-3 < \eta < 3$ as a result of the η range set in the input file. The addition of resummation and the additional virtual corrections in going from leading order to next-to-leading order (+next-to-leading-logarithm resummation) significantly increases the amplitude of the rapidity distribution and the total cross-section, contrastingly however the increase between NLO(+NLL) and NNLO(+NNLL) is insignificant. The error bars are the Monte Carlo errors from the resummed part only and are largely a reflection on the length of the runs performed.

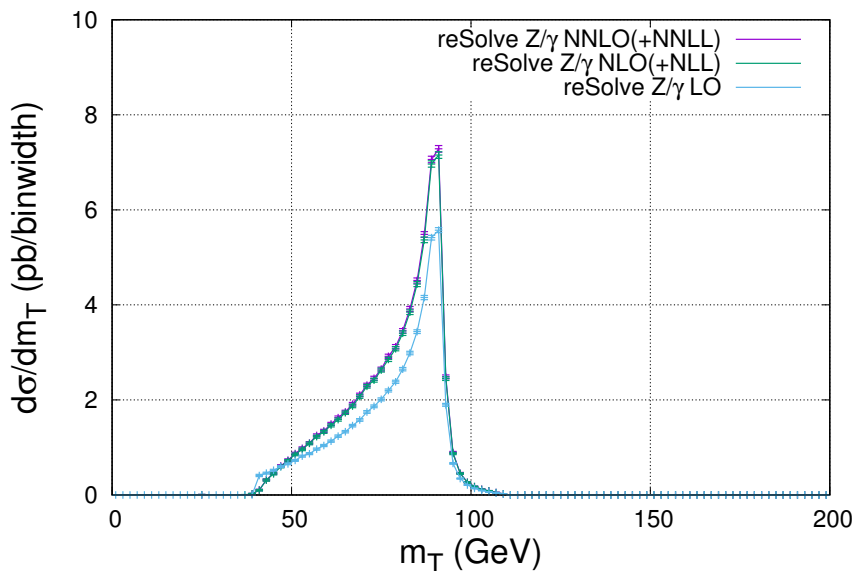


Figure 33: Transverse mass distribution for the neutral current Z/γ^* case of the setup 1 benchmark, including only the resummed piece. The distribution has a kinematic boundary at leading order at 40GeV as each of the outgoing leptons has a $p_T > 20\text{GeV}$ due to the cuts and at LO there are no other particles to carry away any transverse momentum, therefore conservation of transverse momentum imposes this kinematic constraint upon the m_T distribution. As in many other cases shown, the addition of resummation and the additional virtual corrections in going from leading order to next-to-leading order (+next-to-leading-logarithm resummation) significantly increases the amplitude of the distribution and the total cross-section, however the increase between NLO(+NLL) and NNLO(+NNLL) is not significant. The error bars are the Monte Carlo errors from the resummed part only and are largely a reflection on the length of the runs performed.

6.3. Multiple PDF fits validation

In order to validate the use of multiple PDF fits, we used the `test1` input parameters for the diphoton process, detailed in table 1, and carried out the same resummation calculation of the cross-section with 1 PDF fit at 113.14GeV ; 2 PDF fits at 97.98GeV and 138.56GeV (corresponding to `en_sec_multiplier = 1.5`); and 4 PDF fits at 87.64GeV , 105.16GeV , 126.20GeV and 148.72GeV (corresponding to `en_sec_multiplier = 1.2`). Given the invariant mass range for the diphoton here was narrow, from 80GeV to 160GeV , we expect no particular benefit from using multiple PDF fits, therefore we use this as a check that all 3 runs produce a consistent differential cross-section. Figure 34 and figure 35 provide the invariant mass spectrum and transverse momentum spectrum comparison respectively between the 1, 2 and 4 PDF fit results. The input files are again provided with the `reSolve` program and are `Diphoton_NNLO_test1.dat`, `Diphoton_NNLO_test1_twopdffits.dat` and `Diphoton_NNLO_test1_fourpdffits.dat`. The results show very good agreement, within the errors, thereby validating the program for use with multiple PDF fits. The total cross-sections evaluated for 1, 2 and 4 PDF fits are given in table 5 in section 6.4 as part of the discussion of the speed of the program and are also in agreement. Meanwhile, the invariant mass and transverse momentum spectrum produced for the diphoton ATLAS validation in section 6.1.3 in figures 21-23 for the `reSolve` data used 5 PDF fits across the large invariant mass (qq) range, thereby also providing validation of the multiple PDF fit running of `reSolve` against experimental data.

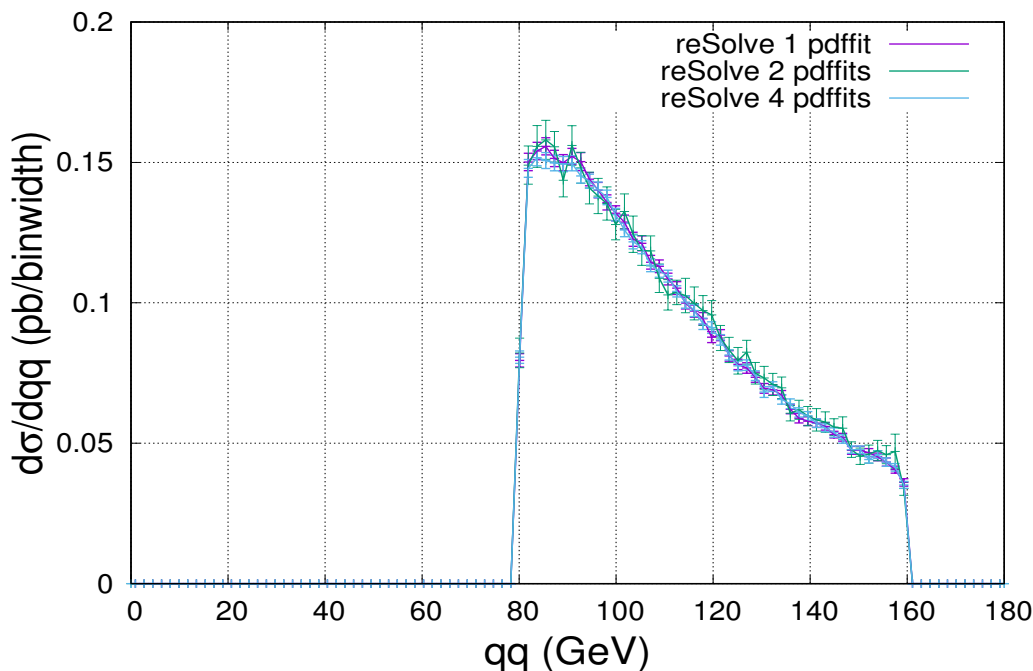


Figure 34: The invariant mass spectrum for diphoton production for the `test1` input parameters provided in table 1 for 1, 2 and 4 pdffits made. The input files are again provided with the `reSolve` program and are `Diphoton_NNLO_test1.dat`, `Diphoton_NNLO_test1_twopdffits.dat` and `Diphoton_NNLO_test1_fourpdffits.dat`. The results show excellent agreement, validating the program for use with multiple PDF fits.

6.4. Performance

Given the number of Monte Carlo iterations it is necessary to perform for the phase space integral to produce the desired accuracy, and the fact that for each phase space point an inverse Fourier transform and double inverse Mellin transform is required - each of which require around 20, 50 and 50 points respectively, speed can be very important for transverse momentum resummation programs in this formalism. As a demonstration, should one require 1,000,000 phase space points, one can expect parts of the resummation code (parts of `inverse_mellinresummed.cc` and `hardfns.cc`) to be called $1,000,000 \times 20 \times 50 \times 50 = 5 \times 10^{10}$ times. Therefore particular care has been taken, even within this first main version of the code, to ensure it runs quickly. Whilst the run time varies significantly depending

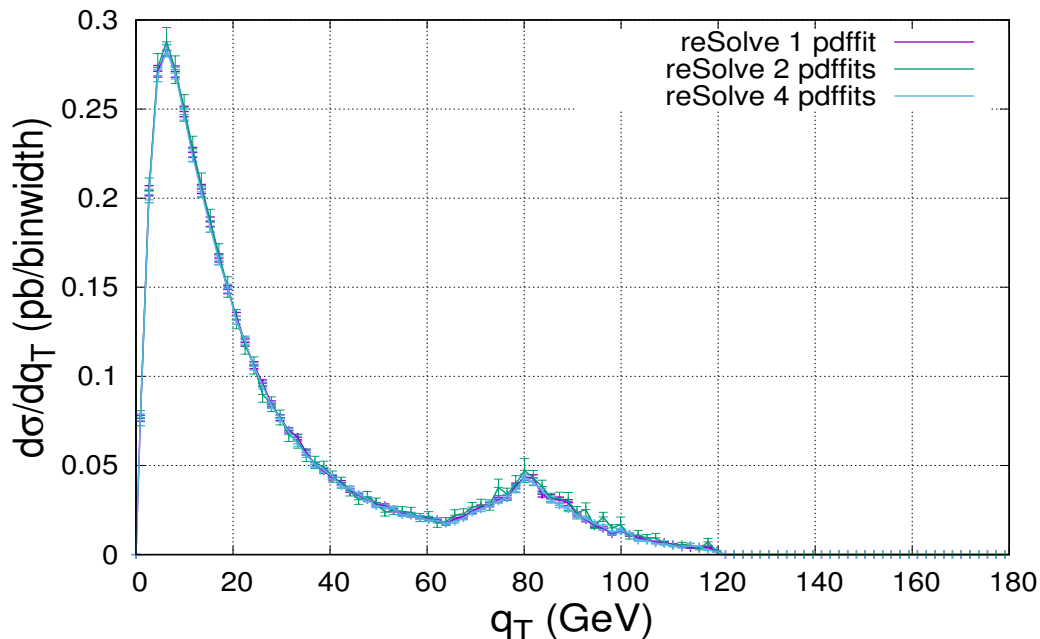


Figure 35: The transverse momentum spectrum for diphoton production for the `test1` input parameters provided in table 1 for 1, 2 and 4 pdf fits made. The input files are again provided with the `reSolve` program and are `Diphoton_NNLO_test1.dat`, `Diphoton_NNLO_test1_twopdffits.dat` and `Diphoton_NNLO_test1_fourpdffits.dat`. The results show excellent agreement, validating the program for use with multiple PDF fits.

upon the input file, as certain points may require more b points to be evaluated in the inverse Fourier transform, large η values have more points on each Mellin contour, and different processes require different numbers of non-zero contributions to the cross-section to be summed; we seek here to give a guide as to the speed of this first main version of the `reSolve` program. In particular, in table 4 we compare it against the private code `2gres` used also for the validations in Section 6.1.2, again for the diphoton `test1` and `test2` inputs at NNLO(+NNLL) listed in table 1 with 550,000 phase space points. As demonstrated, the `reSolve` program is consistently almost twice as quick as the previous `2gres` program, completing the same run in 53% of the time. This speed up is important, allowing a greater number of evaluations to be performed and thereby attaining a greater accuracy with the same computer resources.

Program	Diphoton_NNLO_test_1		Diphoton_NNLO_test_2	
	σ (pb)	time(min:s)	σ (pb)	time(min:s)
<code>reSolve</code>	7.68 ± 0.03	1678m:22s	2.536 ± 0.009	1370m:11s
<code>2gres</code>	7.68 ± 0.03	3178m:35s	2.556 ± 0.008	2763m:46s

Table 4: Comparison of the time taken to evaluate 550,000 phase space points in the new public code `reSolve` and the old private code `2gres`. The times listed are total core times, summing those across all cores. The total cross-sections are also given, demonstrating good consistency between the programs. Note `reSolve` here used one PDF fit, as that is all that was available in the previous `2gres` program, and the integrator `Cuba` was used by both programs to allow a fair comparison. The test files are the `Diphoton_NNLO_test_1` and `Diphoton_NNLO_test_2` files used previously and listed in table 1.

Running with multiple PDF fits will also slow down the running of the `reSolve` program; multiple PDF fits therefore should only be used for a wide invariant mass range, where the adoption of multiple PDF fits at different scales through the invariant mass range may offer increased precision. In order to demonstrate this slow-down, table 5 provides the run-times of the `reSolve` program using 1, 2 and 4 PDF fit files respectively, once

Number of PDF fits	Diphoton_NNLO_test_1	
	$\sigma(\text{pb})$	time(min:s)
1	7.68 ± 0.03	1678m:22s
2	7.67 ± 0.03	3636m:35s
4	7.63 ± 0.03	3617m:52s

Table 5: Comparison of the time taken to evaluate 550,000 phase space points in the reSolve program for different numbers of PDF fits. Again the times listed are total core times, summing those across all cores. The total cross-sections are also given, demonstrating consistency between the fits. As expected multiple PDF fits take much longer, however 4 PDF fits took no longer than 2 PDF fits. The test files used were the Diphoton_NNLO_test_1.dat, Diphoton_NNLO_test1_twopdffits.dat and Diphoton_NNLO_test1_fourpdffits.dat provided with the reSolve program. The general inputs for this setup were listed in table 1 in the case of 1 PDF fit.

Time elapsed(min:s)					
One Core		4 cores (one machine) using Cuba		12 cores across 3 machines using k_vegas	
Core time	Physical time	Core time	Physical time	Core time	Physical time
~1678m:22s	~1678m:22s	1678m:22s	474m:34s	2756m:12s	229m:41s

Table 6: The time taken to evaluate over 500,000 phase space points in the reSolve program with different degrees of parallelisation. ‘‘One Core’’ indicates either k_vegas or Cuba used with one core only, the second column shows Cuba used parallelising across the 4 cores of the machine used as standard, finally the time taken parallelising across 12 cores across 3 machines using the k_vegas parallelisation routine multi_machine_parallel_local provided with reSolve is given. Note the time given here for one core is approximate as this test was not run, it is an indication based on extrapolation from 4 cores on the same machine. This comparison was performed using the Diphoton_NNLO_test_1 inputs given in table 1.

more for the Diphoton_NNLO_test_1 inputs listed previously. Again the specific input files are available pre-setup for 1, 2 and 4 PDF fits as Diphoton_NNLO_test_1.dat, Diphoton_NNLO_test1_twopdffits.dat and Diphoton_NNLO_test1_fourpdffits.dat. The invariant mass and transverse momentum spectra for these runs were given previously in section 6.3 in figures 34 and 35. Note the time listed does not include that to produce the fits, the PDF fit files were provided here. The comparison demonstrates that, as one might expect, the runtime is significantly longer for multiple PDF fits, indeed it is more than twice as long, however between 2 and 4 PDF fits the runtime does not increase, demonstrating that the main difference comes when one starts to use multiple PDF fits. Note however that even with multiple PDF fits, the reSolve program is similar in speed to the old private 2gres program, which only used one PDF fit.

Many of these issues of speed and time taken to evaluate the phase space points required are further ameliorated by the ability of the reSolve program to allow parallelisation across many cores in many machines. A detailed description of how to perform the parallelisation and how the program works when parallelised using k_vegas was given in Section 3.4. A comparison of the physical time elapsed when running more than 500,000 phase space points on one core, on 4 cores with Cuba, and on many cores on many machines with k_vegas is provided in table 6. The run-times clearly demonstrate how much physical time may be saved using the parallelisation option in reSolve. The results obtained are for the Diphoton_NNLO_test_1 inputs used previously and were also consistent with those provided in table 4, with the 12 cores parallelisation across 3 machines obtaining 7.67 ± 0.08 pb. Note the error is however larger here as the same number of total iterations were performed in smaller batches. To obtain an error of similar magnitude will require larger batches for each core, and therefore a longer runtime than indicated here, however the combination of batches from different cores and machines still means the total runtime required for a given error will also of course be significantly smaller than when unparallelised. Also it should be noted that, as the reSolve program when parallelised using k_vegas waits for all cores to be complete at a given iteration, the speed of the program will be governed by the slowest core - this is why the core time in the parallelised setup is longer than for the unparallelised or Cuba parallelised over one machine implementations. This is necessary so that after each iteration the overall grid of points and weights for the Monte Carlo integration is updated and used by all cores for the next iteration. We therefore recommend parallelising across cores and machines of similar speeds.

7. Future Developments

This version of the reSolve program is the first main version of many, therefore we intend to optimise and extend this version further, undertaking an ongoing development program. There are many areas for improvement in the program, a few of those we consider the most important are listed here:

- Add the finite parts for the diphoton and Drell-Yan spectra. Given the program currently only includes the resummed (i.e. low q_T) part of the transverse momentum differential cross-section, the obvious extension is to add the finite piece which is dominant at high q_T . This will require matching of the low and high q_T pieces at intermediate q_T using the matching procedure outlined in [21]. This extension to the program will be performed in the near future and will enable the production of the transverse momentum spectrum across the whole q_T range.
- Extend to additional processes, Higgs production foremost amongst them, perhaps also Beyond Standard Model contributions in cases such as Z' .
- Inclusion of the gg hard factors to higher orders, this will enable Higgs production to be incorporated. With this extension the program will include both signal and background (diphoton) for the Higgs, therefore signal-background interference could be examined [2].
- Currently the PDFs are fitted, and used at given scales, we would like to examine the possibility of interpolating and using them directly at the scale desired.
- Currently the only PDF sets available to use are the MSTW PDFs [19], in future versions we will broaden to allow any PDF set to be used. In order to do so we will allow Les Houches Accord PDF formats [35] to be read in.
- There is substantial scope to further speed up the program significantly, the speed of the program is currently hampered by memory considerations which slows the program down by a further 20%, this could be reduced substantially.
- The formalism could be relatively easily adapted to allow the implementation of a jet veto, which may be of interest in certain applications.
- The program can be relatively easily extended to QED resummation, an area itself of increasing interest.

Beyond these shorter-term objectives, the universality of the formalism applied within reSolve, along with the program's clearly-designed modularity, allow the potential to interface the code with existing more general packages in order to allow their extension to higher accuracy. This could incorporate interfacing with existing Matrix Element generators for automatic generation of resummed spectra for a much wider class of processes, for example NLO Matrix Element generators could be interfaced to allow semi-automatic production of differential spectra at next-to-next-to-leading order with next-to-next-to-leading logarithm resummation. In any case, whichever the precise longer-term direction taken, the properties of the reSolve transverse momentum resummation program mean it can form a key part of current and future tools for precise theoretical predictions for collider processes.

8. Conclusions

We have presented a new transverse resummation program, reSolve, specifically designed to allow generalisation to many processes. We described in detail the structure of the program and its modularisation, which allows different processes of interest to be added straightforwardly, and outlined how to do this. Currently the processes included are diphoton + jet and Drell-Yan + jet, both up to NNLO + NNLL, these processes have been extensively validated both internally and externally against known results and previous programs where available, as well as against experimental results. The code itself is publicly available with this paper and also on Github - the latter is where the most up-to-date version is guaranteed to be found. This paper provides details of the theoretical explanation of the b-space Mellin-space formalism applied, whilst also acting as the manual for users, we therefore hope it will prove useful. This version

of reSolve represents the first main incarnation of the development of this program and what we hope will become a “go-to” public code in this area for transverse momentum resummation for any appropriate process. reSolve has several advantages over previous programs for transverse momentum resummation; being simple to use, transparently documented and modularised in a way to allow many different processes to be considered within the same program structure. Given the increasing interest in precision physics, both in order to increase understanding of the standard model and to search for hints of new physics, such tools are of vital importance to the ongoing program at the Large Hadron Collider and beyond. We therefore expect the reSolve program to be key to many future phenomenological applications.

Acknowledgements

This research has been partially supported by STFC consolidated grant ST/L000385/1 and by STFC consolidated grant ST/P000681/1, and also in part by the National Science Foundation under Grant No. NSF PHY-1748958 along with the Gordon and Betty Moore Foundation. We wish also to thank both the Cambridge SUSY working group and Leandro Cieri for helpful discussions. TC wishes to thank The Kavli Institute for Theoretical Physics, University of California Santa Barbara for their hospitality, supporting him for a KITP Graduate Fellowship, during which part of this work was completed; additionally he wishes to thank the Cambridge Philosophical Society and the Department of Applied Mathematics and Theoretical Physics at the University of Cambridge for their support during this work.

Appendix A. Summary of reSolve Usage

A detailed description of how to use the reSolve program is given in the main text of the paper in Section 3, however the details are summarised and condensed here for ease of reference.

Appendix A.1. Where to obtain the reSolve program

The reSolve program is available online with this paper, the latest version will also always be found on Github at <https://github.com/fkhorad/reSolve>.

Appendix A.2. Basic Usage

- Upon downloading and extracting the program to your desired directory, reSolve should work “straight out of the box” with the built-in `k_vegas` integrator, nonetheless if the user wishes to use Cuba [18] they must download this separately and alter the reSolve makefile to detail where this is found on their PC.
- First make the program, then the reSolve program can be used with a variety of built-in input files provided with the download - these are listed in Tables A.7 and A.8. These may be setup to run with Cuba, if so either download Cuba or change “`integrator_flag`” in the input file from 2 to 1.
- To run an input file type in the terminal (in the reSolve main working directory):

```
./reSolve.out input/{name of input file}
```

For example:

```
./reSolve.out input/Diphoton.Born.LHC.dat.
```

- The reSolve program will then run and generate events for this setup, with the overall total cross-section output into the working directory output folder (“`workdir/`”) specified in the input file as `reSolve_main_out.dat` (or `reSolve_main_out_END_ITER.dat` for `k_vegas` parallelised cases). Meanwhile the events are also included in this specified output folder, along with any specified histogram data files that were required for the various differential cross-sections, in particular for the q_T spectrum³.
- There are a plethora of input flags which can be used to alter the phase space region, the process or subprocess, the pdf-fitting, the various scales associated with the theoretical calculation, the cuts, the number of evaluations, the working directory, the seeds used for the random generation of the Monte Carlo, the histograms generated, and many other aspects; these are all described in Section 3.

³Note histograms are only generated for those variables and binnings specified in the input file in the “Histograms” section, nonetheless additional histograms can be calculated for the events without rerunning the whole program by using it in `hist_only` mode - see Section 3.6.

Appendix A.3. Sample Input Files included with reSolve

Diphoton Input Files Included	Description
Diphoton_Born_LHC.dat	Leading-order diphoton production at the LHC at $14TeV$
Diphoton_Born_LHC_parallel.dat	Leading-order diphoton production at the LHC at $14TeV$ setup for <code>k_vegas</code> parallelisation
Diphoton_NNLO_test_1.dat	NNLO diphoton production at the LHC at $14TeV$
Diphoton_NNLO_test_1_parallel.dat	NNLO diphoton production at the LHC at $14TeV$ setup for <code>k_vegas</code> parallelisation on one machine
Diphoton_NNLO_test_1_parallel_multi.dat	NNLO diphoton production at the LHC at $14TeV$ setup for <code>k_vegas</code> parallelisation across many machines
Diphoton_NNLO_test1_twopdffits.dat	NNLO diphoton production at the LHC at $14TeV$ using two PDF fits at different scales across the invariant mass range
Diphoton_NNLO_test1_fourpdffits.dat	NNLO diphoton production at the LHC at $14TeV$ using four PDF fits at different scales across the invariant mass range
Diphoton_NNLO_test_2.dat	NNLO diphoton production at the LHC at $8TeV$
Diphoton_Atlas_A.dat	NNLO diphoton production at the LHC at $8TeV$ setup for experimental comparison

Table A.7: The sample input files which are included with the reSolve program download for the Diphoton process, and which have been used throughout this paper in validation and results generation, see Section 6 for more information on the input files and for the corresponding results and histograms produced. Note the only difference between the files `Diphoton_NNLO_test_1_parallel.dat` and `Diphoton_NNLO_test_1_parallel_multi.dat` is in the working directories to which they output, in order to allow the use of the `single_machine_parallel` and `multi_machine_parallel_local` scripts for the Diphoton NNLO test 1 setup case without the need to delete events between runs.

Drell-Yan Input Files Included	Description
Wpm_Born_Tevatron.dat	Leading-order W^\pm production at the Tevatron
Wpm_NLO_Tevatron.dat	NLO W^\pm production at the Tevatron
Wpm_NNLO_Tevatron.dat	NNLO W^\pm production at the Tevatron
yZ_Born_Tevatron.dat	Leading-order Z/γ^* production at the Tevatron
yZ_Born_Tevatron_parallel.dat	Leading-order Z/γ^* production at the Tevatron setup for <code>k_vegas</code> parallelisation
yZ_NLO_Tevatron.dat	NLO Z/γ^* production at the Tevatron
yZ_NNLO_Tevatron.dat	NNLO Z/γ^* production at the Tevatron
yZ_NNLO_Tevatron_parallel.dat	NNLO Z/γ^* production at the Tevatron setup for <code>k_vegas</code> parallelisation on one machine
yZ_NNLO_Tevatron_parallel_multi.dat	NNLO Z/γ^* production at the Tevatron setup for <code>k_vegas</code> parallelisation across many machines
Z_OnShell_Born_LHC.dat	Leading-order on-shell Z production at the LHC
Z_OnShell_NLO_LHC.dat	NLO on-shell Z production at the LHC
Z_OnShell_NNLO_LHC.dat	NNLO on-shell Z production at the LHC

Table A.8: The sample input files which are included with the reSolve program download for the Drell-Yan processes, and which have been used throughout this paper in validation and results generation, see Section 6 for more information on the input files and for the corresponding results and histograms produced. Again the only difference between the files `yZ_NNLO_Tevatron_parallel.dat` and `yZ_NNLO_Tevatron_parallel_multi.dat` is in the working directories to which they output so single and multiple machine parallelisations can be run separately without the need to delete events between running them.

Appendix B. Resummation coefficients

Here we report a collection of the various coefficients used in the master resummation formula (4), along with the references they were extracted from. We start from process-independent coefficients $A_c^{(n)}$, B_c^n , $C_{ab}^{(n)}$ and $G_{ab}^{(n)}$, commenting on the H_c^F for some specific processes in Appendix B.1.

A useful note regarding flavour dependence: all flavour indices in eq. (4) could in principle assume 13 different values (from \bar{t} to t plus g). However, since quark mass effects are not included in this formalism (that is, we work in a *fixed flavour* scheme), the coefficients are related by flavour symmetry. In practice, this means that in coefficients $A_c^{(n)}$ and B_c^n , the subscripts can only take 3 different values: $c = q, \bar{q}$ or g . In the doubly-indexed objects C_{ab} and G_{ab} , in general there is additionally the need to distinguish *diagonal* and *non-diagonal* flavour contributions, leading to the possible combinations $C_{qq'}^F$, $C_{\bar{q}\bar{q}'}^F$, $C_{q\bar{q}'}^F$ and $C_{\bar{q}q'}^F$ for the indices, with obvious meaning of the symbols. Furthermore, due to C symmetry, all coefficients are invariant under barring of all indices: this implies $A_q = A_{\bar{q}}$, $B_q^F = B_{\bar{q}}^F$ to all orders, and that there are only *seven* independent combinations for the C_{ab} and G_{ab} coefficients: C_{gg}^F , C_{qq}^F , C_{qg}^F , C_{gq}^F , $C_{\bar{q}\bar{q}}^F$, $C_{q\bar{q}}^F$ and $C_{\bar{q}q}^F$, and similarly for G_{ab} .

First of all let us define the auxiliary constants:

$$T_R : \text{tr}(t^a t^b) = T_R \delta_{ab} \quad \rightarrow \text{fundamental colour matrices normalisation} \quad (\text{B.1})$$

$$C_A \equiv C_g = N_c \quad \rightarrow \text{SU}(N_c) \text{ adjoint representation Casimir} \quad (\text{B.2})$$

$$C_F \equiv C_q \equiv C_{\bar{q}} = T_R \frac{N_c^2 - 1}{N_c} \quad \rightarrow \text{SU}(N_c) \text{ fundamental representation Casimir} \quad (\text{B.3})$$

$$N_f \quad \rightarrow \text{Number of } \textit{active} \text{ (effectively massless) flavours} \quad (\text{B.4})$$

$$\zeta_n \quad \rightarrow \text{Value of Riemann Zeta function on point } n \quad (\text{B.5})$$

Notice that the expansion of these and the other perturbative coefficients in the existing literature sees different normalisation choices. For the sake of the generality, we use in this section the definition:

$$Z(\alpha_s) = \sum_n \left(\frac{\alpha_s}{k\pi} \right)^n Z^{(n)}, \quad (\text{B.6})$$

where k is an arbitrary numerical constant and Z a generic perturbative function of α_s . Choices actually used in the literature include at least $k = 1, 2$ and 4 . For completeness, we report the β function (see eq. (8)) coefficients (taken from ref. [21] eq. (28) and ref. [36] eq. (B5)):

$$\beta_0 = \frac{k}{12} (11C_A - 4T_R N_f) \quad (\text{B.7})$$

$$\beta_1 = \frac{k^2}{24} (17C_A^2 - 10C_A T_R N_f - 6C_F T_R N_f) \quad (\text{B.8})$$

$$\begin{aligned} \beta_2 = \frac{k^3}{64} & \left(\frac{2857}{54} C_A^3 - \frac{1415}{27} C_A^2 T_R N_f - \frac{205}{9} C_A C_F T_R N_f \right. \\ & \left. + 2C_F^2 T_R N_f + \frac{158}{27} C_A T_R^2 N_f^2 + \frac{44}{9} C_F T_R^2 N_f^2 \right) \quad (\text{B.9}) \end{aligned}$$

Now we list the actual resummation coefficients, A_c , B_c , C_{ab} and G_{ab} . All coefficients are defined in *hard scheme* (see Section 5.3) unless otherwise specified. The $A_c^{(n)}$ coefficients (which are actually resummation scheme independent) are given by ($A_c^{(1,2)}$ from ref. [21] eq. (47) and $A_c^{(3)}$ from ref. [36] eqs. (51), (74) and (B3) – see also the

comment under eq. (74):

$$A_c^{(1)} = k C_c \quad (\text{B.10})$$

$$A_c^{(2)} = \frac{k^2}{2} C_c \left[\left(\frac{67}{18} - \frac{\pi^2}{6} \right) C_A - \frac{10}{9} T_R N_f \right] \quad (\text{B.11})$$

$$A_g^{(3)} = C_A \frac{k^3}{16} \left[C_A^2 \left(\frac{245}{6} - \frac{134}{27} \pi^2 + \frac{11}{45} \pi^4 + \frac{22}{3} \zeta_3 \right) \right. \\ \left. + C_A T_R N_f \left(-\frac{418}{27} + \frac{40}{27} \pi^2 - \frac{56}{3} \zeta_3 \right) + C_F T_R N_f \left(-\frac{55}{3} + 16 \zeta_3 \right) + \frac{16}{27} T_R^2 N_f^2 \right] \quad (\text{B.12})$$

$$A_q^{(3)} = A_{\bar{q}}^{(3)} = \frac{C_F}{C_A} A_g^{(3)} + 2\beta_0 \frac{k^2}{16} C_F \left[C_A \left(\frac{808}{27} - 28 \zeta_3 \right) - \frac{224}{27} T_R N_f \right] \quad (\text{B.13})$$

The $B_c^{(n)}$ coefficients, from ref. [21] eq. (49) for $B^{(1)}$ and ref. [15] eq. (34)-(36) for $B^{(2)}$:

$$B_g^{(1)} = -\frac{k}{6} (11 C_A - 4 T_R N_f) \quad (\text{B.14})$$

$$B_q^{(1)} = B_{\bar{q}}^{(1)} = -k \frac{3}{2} C_F \quad (\text{B.15})$$

$$B_c^{(2)} = k \left(\frac{\gamma_c^{(1)}}{16} + \pi \beta_0 C_c \zeta_2 \right), \quad \text{with} \quad (\text{B.16})$$

$$\gamma_q^{(1)} = \gamma_{\bar{q}}^{(1)} = k \left[C_F^2 (-3 + 24 \zeta_2 - 48 \zeta_3) + C_A C_F \left(-\frac{17}{3} - \frac{88}{3} \zeta_2 + 24 \zeta_3 \right) \right. \\ \left. + C_F T_R N_f \left(\frac{2}{3} + \frac{16}{3} \zeta_2 \right) \right] \quad (\text{B.17})$$

$$\gamma_g^{(1)} = k \left[C_A^2 \left(-\frac{64}{3} - 24 \zeta_3 \right) + \frac{16}{3} C_A T_R N_f + 4 C_F T_R N_f \right];$$

the $\gamma_c^{(1)}$ coefficients are the $\delta(1-z)$ parts of the first order Altarell-Parisi splitting kernels.

Now we list the $C_{ab}^{(n)}$ and $G_{ab}^{(n)}$ coefficients. We start with the first order coefficients; the explicit expressions for these are taken from ref. [15], eqs. (29)-(33):

$$C_{qq}^{(1)}(z) = \frac{k}{2} C_F (1-z) \quad (\text{B.18})$$

$$C_{gq}^{(1)}(z) = \frac{k}{2} C_F z \quad (\text{B.19})$$

$$C_{qg}^{(1)}(z) = \frac{k}{2} z (1-z) \quad (\text{B.20})$$

$$C_{gg}^{(1)}(z) = C_{q\bar{q}}^{(1)}(z) = C_{q'q'}^{(1)}(z) = C_{q\bar{q}'}^{(1)}(z) = 0 \quad (\text{B.21})$$

$$G_{gq}^{(1)}(z) = \frac{k}{2} C_F \frac{1-z}{z} \quad (\text{B.22})$$

$$G_{gg}^{(1)}(z) = \frac{k}{2} C_A \frac{1-z}{z}. \quad (\text{B.23})$$

Notice that 3 out of 5 of the non-diagonal C_{ab} are actually *vanishing* at first order. Finally, let us give references for the $C_{ab}^{(2)}$ coefficients; since these 2nd order coefficients have very long and involved expressions, we will not report them in a completely explicit form, but give them in terms of more fundamental objects whose definitions can be found in the listed references. Also, since they are not needed for the only process currently implemented in reSolve (diphoton production), we will not report the $G_{ga}^{(2)}$ coefficients. All the following definitions are extracted from [15],

eqs. (37)-(40) and nearby text.

$$C_{q\bar{q}}^{(2)}(z) = \frac{k^2}{2} \left\{ \mathcal{H}_{q\bar{q} \leftarrow q\bar{q}}^{DY(2)}(z)|_{\text{no } \delta(1-z)} - \frac{C_F^2}{4} \left[(2\pi^2 - 18)(1-z) - (1+z) \log z \right] \right\}, \quad (\text{B.24})$$

with $\mathcal{H}_{q\bar{q} \leftarrow q\bar{q}}^{DY(2)}(z)$ defined in eq. (23) of [4];

$$C_{qg}^{(2)}(z) = k^2 \left\{ \mathcal{H}_{q\bar{q} \leftarrow qg}^{DY(2)}(z) - \frac{C_F}{4} \left[z \log z + \frac{1}{2}(1-z^2) + (\pi^2 - 8)z(1-z) \right] \right\}, \quad (\text{B.25})$$

with $\mathcal{H}_{q\bar{q} \leftarrow qg}^{DY(2)}(z)$ defined in eq. (32) of [4];

$$C_{q\bar{q}}^{(2)}(z) = k^2 \mathcal{H}_{q\bar{q} \leftarrow q\bar{q}}^{DY(2)}(z), \text{ with } \mathcal{H}_{q\bar{q} \leftarrow q\bar{q}}^{DY(2)}(z)|_{\text{no } \delta(1-z)} \text{ defined in eq. (24) of [4];} \quad (\text{B.26})$$

$$C_{q\bar{q}'}^{(2)}(z) = k^2 \mathcal{H}_{q\bar{q} \leftarrow q\bar{q}'}^{DY(2)}(z), \text{ with } \mathcal{H}_{q\bar{q} \leftarrow q\bar{q}'}^{DY(2)}(z)|_{\text{no } \delta(1-z)} \text{ defined in eq. (25) of [4];} \quad (\text{B.27})$$

$$C_{q\bar{q}'}^{(2)}(z) = C_{q\bar{q}'}^{(2)}(z), \text{ by eqs. (24), (26) and (35) of [4];} \quad (\text{B.28})$$

$$C_{g\bar{g}}^{(2)}(z) = \frac{k^2}{2} \left\{ \mathcal{H}_{g\bar{g} \leftarrow g\bar{g}}^{H(2)}(z)|_{\text{no } \delta(1-z)} + C_A^2 \left[\frac{1+z}{z} \log z + 2 \frac{1-z}{z} \right] \right\}, \quad (\text{B.29})$$

with $\mathcal{H}_{g\bar{g} \leftarrow g\bar{g}}^{H(2)}(z)$ defined in eq. (24) of [37];

$$C_{gq}^{(2)}(z) = k^2 \left\{ \mathcal{H}_{g\bar{g} \leftarrow gq}^{H(2)}(z) + C_F^2 \frac{3z}{4} - \frac{C_A C_F}{z} \left[(1+z) \log z + 2(1-z) - \frac{5+\pi^2}{4} z^2 \right] \right\}, \quad (\text{B.30})$$

with $\mathcal{H}_{g\bar{g} \leftarrow gq}^{H(2)}(z)$ defined in eq. (23) of [37].

All C_{ab} coefficients given here are defined in z -space. In reSolve, as commented in section 5.2, the corresponding Mellin space expressions are actually used.

Appendix B.1. Examples of process-dependent coefficients

Here we will list the process-dependent $H_c^{F(n)}$ coefficients for selected processes, again in *hard scheme*. The processes we list them for are Drell-Yan, Higgs and diphoton production, with, remember, only the first and last of these actually implemented in the current version of reSolve. The main ref. here is [15]: eqs. (82-83) for DY and eqs. (85), (87) for Higgs (obtained in the large m_t limit):

$$H_q^{DY(1)} = k C_F \left(\frac{\pi^2}{2} - 4 \right) \quad (\text{B.31})$$

$$\begin{aligned} H_q^{DY(2)} = k^2 & \left[C_F C_A \left(\frac{59}{18} \zeta_3 - \frac{1535}{192} + \frac{215}{216} \pi^2 - \frac{\pi^4}{240} \right) \right. \\ & + \frac{1}{4} C_F^2 \left(-15 \zeta_3 + \frac{511}{16} - \frac{67}{12} \pi^2 + \frac{17}{45} \pi^4 \right) \\ & \left. + \frac{1}{864} C_F T_R N_f (192 \zeta_3 + 1143 - 152 \pi^2) \right] \end{aligned} \quad (\text{B.32})$$

$$H_g^{H(1)} = k \left[C_A \frac{\pi^2}{2} + \frac{5C_A - 3C_F}{2} \right] \quad (\text{B.33})$$

$$H_g^{H(2)} = k^2 \left[C_A^2 \left(\frac{3187}{288} + \frac{7}{8} L_Q + \frac{157}{72} \pi^2 + \frac{13}{144} \pi^4 - \frac{55}{18} \zeta_3 \right) \right. \quad (\text{B.34})$$

$$\left. + C_A C_F \left(-\frac{145}{24} - \frac{11}{8} L_Q - \frac{3}{4} \pi^2 \right) + \frac{9}{4} C_F^2 - \frac{5}{96} C_A - \frac{1}{12} C_F \right. \quad (\text{B.35})$$

$$\left. - C_A T_R N_f \left(\frac{287}{144} + \frac{5}{36} \pi^2 + \frac{4}{9} \zeta_3 \right) + C_F T_R N_f \left(-\frac{41}{24} + \frac{1}{2} L_Q + \zeta_3 \right) \right], \quad (\text{B.36})$$

where $L_Q = \log(m_H^2/m_t^2)$. For Higgs at NLO, also the contribution for finite top mass is known: see for instance [38], eqs. (21)-(30).

Finally, for the diphoton ($\gamma\gamma$) process, from [15] eqs. (88-91):

$$H_q^{\gamma\gamma(1)} = k \frac{C_F}{2} \left[\pi^2 - 7 + \frac{1}{(1-v)^2 + v^2} \left[((1-v)^2 + 1) \log^2(1-v) + v(v+2) \log(1-v) + (v^2 + 1) \log^2 v + (1-v)(3-v) \log v \right] \right] \quad (\text{B.37})$$

$$H_q^{\gamma\gamma(2)} = \frac{k^2}{4\mathcal{A}_{LO}} [\mathcal{F}_{inite,q\bar{q}\gamma\gamma;s}^{0\times 2} + \mathcal{F}_{inite,q\bar{q}\gamma\gamma;s}^{1\times 1}] + 3\zeta_2 C_F H_q^{\gamma\gamma(1)} - \frac{45}{4} C_F^2 + C_F C_A \left(\frac{607}{324} + \frac{1181}{144} \zeta_2 - \frac{187}{144} \zeta_3 - \frac{105}{32} \zeta_4 \right) + C_F T_R N_f \left(-\frac{41}{162} - \frac{97}{72} \zeta_2 + \frac{17}{72} \zeta_3 \right), \quad (\text{B.38})$$

where $v = -u/s$ (partonic variables), \mathcal{A}_{LO} is just proportional to the LO matrix element $u/t + t/u$ (and to KinFac in our code),

$$\mathcal{A}_{LO} = -8N_c \left(\frac{u}{t} + \frac{t}{u} \right) = 8N_c \frac{1 - 2v + 2v^2}{v(1-v)}, \quad (\text{B.39})$$

and $\mathcal{F}_{inite,q\bar{q}\gamma\gamma;s}^{0\times 2}$, $\mathcal{F}_{inite,q\bar{q}\gamma\gamma;s}^{1\times 1}$ are defined in [39], eqs. (4.6) and (5.3), see also appendices A and B in the same reference. A subtlety which appears in the $\gamma\gamma$ process is that the $\mathcal{F}_{inite,q\bar{q}\gamma\gamma;s}^{0\times 2}$ factor, and thus $H_q^{\gamma\gamma(2)}$, is actually dependent on the *electric charge* (squared) of the involved quark. So there are actually two different $H_q^{\gamma\gamma(2)}$ coefficients, namely $H_u^{\gamma\gamma(2)}$ and $H_d^{\gamma\gamma(2)}$. This is of no particular consequence for our expressions, but it implies that a modicum of care must be used in setting the resummation scheme for the resummation coefficients: if one was for instance to define a $\gamma\gamma$ scheme along the lines of the *DY-H scheme* defined in section 5.3, the charge dependence would creep in the definition of the $C_{ab}^{(2)}$ coefficients, which would be impractical. In the $\gamma\gamma$ case, in principle also the gg channel contributes, with the LO term given by the $gg \rightarrow \gamma\gamma$ box, so that $H_{gg}^{\gamma\gamma} \neq 0$. However, the $gg \rightarrow \gamma\gamma$ partonic process is suppressed by α_s^2 with respect to the $q\bar{q}$ -initiated one (which is $\mathcal{O}(\alpha_s^0)$, a purely electromagnetic process at LO), so a non-trivial $H_{gg}^{\gamma\gamma}$ is only needed for N³LL onwards, and so is not included in reSolve.

Bibliography

- [1] L. Cieri, F. Coradeschi, D. de Florian, Diphoton production at hadron colliders: transverse-momentum resummation at next-to-next-to-leading logarithmic accuracy, *JHEP* 06 (2015) 185. [arXiv:1505.03162](#), [doi:10.1007/JHEP06\(2015\)185](#).
- [2] L. Cieri, F. Coradeschi, D. de Florian, N. Fidanza, Transverse-momentum resummation for the signal-background interference in the H channel at the LHC, *Phys. Rev. D* 96 (5) (2017) 054003. [arXiv:1706.07331](#), [doi:10.1103/PhysRevD.96.054003](#).
- [3] G. Bozzi, S. Catani, G. Ferrera, D. de Florian, M. Grazzini, Production of Drell-Yan lepton pairs in hadron collisions: Transverse-momentum resummation at next-to-next-to-leading logarithmic accuracy, *Phys. Lett. B* 696 (2011) 207–213. [arXiv:1007.2351](#), [doi:10.1016/j.physletb.2010.12.024](#).
- [4] S. Catani, L. Cieri, D. de Florian, G. Ferrera, M. Grazzini, Vector boson production at hadron colliders: hard-collinear coefficients at the NNLO, *Eur. Phys. J. C* 72 (2012) 2195. [arXiv:1209.0158](#), [doi:10.1140/epjc/s10052-012-2195-7](#).
- [5] M. Aaboud, et al., Measurement of the W -boson mass in pp collisions at $\sqrt{s} = 7$ TeV with the ATLAS detector [arXiv:1701.07240](#).
- [6] Y. L. Dokshitzer, D. Diakonov, S. I. Troian, Hard Processes in Quantum Chromodynamics, *Phys. Rept.* 58 (1980) 269–395. [doi:10.1016/0370-1573\(80\)90043-5](#).
- [7] G. Parisi, R. Petronzio, Small Transverse Momentum Distributions in Hard Processes, *Nucl. Phys. B* 154 (1979) 427–440. [doi:10.1016/0550-3213\(79\)90040-3](#).
- [8] G. Curci, M. Greco, Y. Srivastava, QCD Jets From Coherent States, *Nucl. Phys. B* 159 (1979) 451–468. [doi:10.1016/0550-3213\(79\)90345-6](#).
- [9] J. C. Collins, D. E. Soper, Back-To-Back Jets in QCD, *Nucl. Phys. B* 193 (1981) 381, [Erratum: *Nucl. Phys.* B213,545(1983)]. [doi:10.1016/0550-3213\(81\)90339-4](#).
- [10] J. Kodaira, L. Trentadue, Summing Soft Emission in QCD, *Phys. Lett.* 112B (1982) 66. [doi:10.1016/0370-2693\(82\)90907-8](#).
- [11] J. C. Collins, D. E. Soper, G. F. Sterman, Transverse Momentum Distribution in Drell-Yan Pair and W and Z Boson Production, *Nucl. Phys. B* 250 (1985) 199–224. [doi:10.1016/0550-3213\(85\)90479-1](#).
- [12] S. Catani, E. D’Emilio, L. Trentadue, The Gluon Form-factor to Higher Orders: Gluon Gluon Annihilation at Small Q^- transverse, *Phys. Lett. B* 211 (1988) 335–342. [doi:10.1016/0370-2693\(88\)90912-4](#).
- [13] D. de Florian, M. Grazzini, Next-to-next-to-leading logarithmic corrections at small transverse momentum in hadronic collisions, *Phys. Rev. Lett.* 85 (2000) 4678–4681. [arXiv:hep-ph/0008152](#), [doi:10.1103/PhysRevLett.85.4678](#).
- [14] S. Catani, M. Grazzini, QCD transverse-momentum resummation in gluon fusion processes, *Nucl. Phys. B* 845 (2011) 297–323. [arXiv:1011.3918](#), [doi:10.1016/j.nuclphysb.2010.12.007](#).
- [15] S. Catani, L. Cieri, D. de Florian, G. Ferrera, M. Grazzini, Universality of transverse-momentum resummation and hard factors at the NNLO, *Nucl. Phys. B* 881 (2014) 414–443. [arXiv:1311.1654](#), [doi:10.1016/j.nuclphysb.2014.02.011](#).
- [16] S. Catani, D. de Florian, G. Ferrera, M. Grazzini, Vector boson production at hadron colliders: transverse-momentum resummation and leptonic decay, *JHEP* 12 (2015) 047. [arXiv:1507.06937](#), [doi:10.1007/JHEP12\(2015\)047](#).
- [17] D. de Florian, G. Ferrera, M. Grazzini, D. Tommasini, Higgs boson production at the LHC: transverse momentum resummation effects in the $H \rightarrow 2\gamma$, $H \rightarrow WW \rightarrow l\nu l\nu$ and $H \rightarrow ZZ \rightarrow 4l$ decay modes, *JHEP* 06 (2012) 132. [arXiv:1203.6321](#), [doi:10.1007/JHEP06\(2012\)132](#).
- [18] T. Hahn, CUBA: A Library for multidimensional numerical integration, *Comput. Phys. Commun.* 168 (2005) 78–95. [arXiv:hep-ph/0404043](#), [doi:10.1016/j.cpc.2005.01.010](#).
- [19] A. D. Martin, W. J. Stirling, R. S. Thorne, G. Watt, Parton distributions for the LHC, *Eur. Phys. J. C* 63 (2009) 189–285. [arXiv:0901.0002](#), [doi:10.1140/epjc/s10052-009-1072-5](#).
- [20] G. P. Lepage, A new algorithm for adaptive multidimensional integration, *Journal of Computational Physics* 27 (1978) 192–203. [doi:10.1016/0021-9991\(78\)90004-9](#).
- [21] G. Bozzi, S. Catani, D. de Florian, M. Grazzini, Transverse-momentum resummation and the spectrum of the Higgs boson at the LHC, *Nucl. Phys. B* 737 (2006) 73–120. [arXiv:hep-ph/0508068](#), [doi:10.1016/j.nuclphysb.2005.12.022](#).
- [22] T. Oura, M. Mori, The double exponential formula for oscillatory functions over the half infinite interval, *Journal of Computational and Applied Mathematics* 38 (1) (1991) 353 – 360. [doi:https://doi.org/10.1016/0377-0427\(91\)90181-I](#).
URL <http://www.sciencedirect.com/science/article/pii/037704279190181I>
- [23] J. C. Collins, D. E. Soper, Back-To-Back Jets: Fourier Transform from B to K -Transverse, *Nucl. Phys. B* 197 (1982) 446–476. [doi:10.1016/0550-3213\(82\)90453-9](#).
- [24] G. Bozzi, S. Catani, D. de Florian, M. Grazzini, Higgs boson production at the LHC: Transverse-momentum resummation and rapidity dependence, *Nucl. Phys. B* 791 (2008) 1–19. [arXiv:0705.3887](#), [doi:10.1016/j.nuclphysb.2007.09.034](#).
- [25] C. Balazs, E. L. Berger, P. M. Nadolsky, C. P. Yuan, Calculation of prompt diphoton production cross-sections at Tevatron and LHC energies, *Phys. Rev. D* 76 (2007) 013009. [arXiv:0704.0001](#), [doi:10.1103/PhysRevD.76.013009](#).
- [26] S. Catani, L. Cieri, D. de Florian, G. Ferrera, M. Grazzini, Diphoton production at hadron colliders: a fully-differential QCD calculation at NNLO, *Phys. Rev. Lett.* 108 (2012) 072001, [Erratum: *Phys. Rev. Lett.* 117, no.8, 089901(2016)]. [arXiv:1110.2375](#), [doi:10.1103/PhysRevLett.108.072001](#), [doi:10.1103/PhysRevLett.117.089901](#).
- [27] ATLAS COLLABORATION, Atlas detector and physics performance, Technical Design Report, ATLAS-TDR-15, (Vol., 2 CERN/LHCC 99-15).
- [28] CMS COLLABORATION, Cms: The electromagnetic calorimeter, Technical Design Report, CMS-TDR-4, (CERN/LHCC 97-33).
- [29] M. Aaboud, et al., Measurements of integrated and differential cross sections for isolated photon pair production in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector, *Phys. Rev. D* 95 (11) (2017) 112005. [arXiv:1704.03839](#), [doi:10.1103/PhysRevD.95.112005](#).
- [30] S. Catani, L. Cieri, G. Ferrera, D. de Florian, M. Grazzini, Vector boson production at hadron colliders: a fully exclusive QCD calculation at NNLO, *Phys. Rev. Lett.* 103 (2009) 082001. [arXiv:0903.2120](#), [doi:10.1103/PhysRevLett.103.082001](#).
- [31] J. M. Campbell, R. K. Ellis, An Update on vector boson pair production at hadron colliders, *Phys. Rev. D* 60 (1999) 113006. [arXiv:hep-ph/9905386](#), [doi:10.1103/PhysRevD.60.113006](#).

- [32] J. M. Campbell, R. K. Ellis, C. Williams, Vector boson pair production at the LHC, *JHEP* 07 (2011) 018. [arXiv:1105.0020](#), [doi:10.1007/JHEP07\(2011\)018](#).
- [33] J. M. Campbell, R. K. Ellis, W. T. Giele, A Multi-Threaded Version of MCFM, *Eur. Phys. J. C* 75 (6) (2015) 246. [arXiv:1503.06182](#), [doi:10.1140/epjc/s10052-015-3461-2](#).
- [34] R. Boughezal, J. M. Campbell, R. K. Ellis, C. Focke, W. Giele, X. Liu, F. Petriello, C. Williams, Color singlet production at NNLO in MCFM, *Eur. Phys. J. C* 77 (1) (2017) 7. [arXiv:1605.08011](#), [doi:10.1140/epjc/s10052-016-4558-y](#).
- [35] M. R. Whalley, D. Bourilkov, R. C. Group, The Les Houches accord PDFs (LHAPDF) and LHAGLUE, in: *HERA and the LHC: A Workshop on the implications of HERA for LHC physics. Proceedings, Part B, 2005*, pp. 575–581. [arXiv:hep-ph/0508110](#).
- [36] T. Becher, M. Neubert, Drell-Yan Production at Small q_T , Transverse Parton Distributions and the Collinear Anomaly, *Eur. Phys. J. C* 71 (2011) 1665. [arXiv:1007.4005](#), [doi:10.1140/epjc/s10052-011-1665-7](#).
- [37] S. Catani, M. Grazzini, Higgs Boson Production at Hadron Colliders: Hard-Collinear Coefficients at the NNLO, *Eur. Phys. J. C* 72 (2012) 2013, [Erratum: *Eur. Phys. J. C* 72, 2132 (2012)]. [arXiv:1106.4652](#), [doi:10.1140/epjc/s10052-012-2013-2](#), [doi:10.1140/epjc/s10052-012-2132-9](#).
- [38] U. Aglietti, R. Bonciani, G. Degrossi, A. Vicini, Analytic Results for Virtual QCD Corrections to Higgs Production and Decay, *JHEP* 01 (2007) 021. [arXiv:hep-ph/0611266](#), [doi:10.1088/1126-6708/2007/01/021](#).
- [39] C. Anastasiou, E. W. N. Glover, M. E. Tejeda-Yeomans, Two loop QED and QCD corrections to massless fermion boson scattering, *Nucl. Phys. B* 629 (2002) 255–289. [arXiv:hep-ph/0201274](#), [doi:10.1016/S0550-3213\(02\)00140-2](#).