GLAS: General Loop Amplitude System

A Unified Framework for Automated NLO QCD Calculations

M. Houmani

High Energy Physics Group

elmahdi.houmani@rwth-aachen.de

January 26, 2026

Abstract

We present GLAS (General Loop Amplitude System), a comprehensive software framework for automated next-to-leading order (NLO) QCD calculations. The system provides a unified pipeline that seamlessly integrates diagram generation via QGRAF, symbolic amplitude manipulation using FORM, integral reduction through MATHEMATICA with the BLADE package, and UV renormalization in the $\overline{\text{MS}}$ scheme. A key feature is the implementation of finite field techniques using FINITEFLOW for efficient identification of linear relations among master integral coefficients. GLAS implements a run-based workflow with parallel execution capabilities, enabling efficient computation of multi-leg processes relevant for LHC phenomenology. We describe the architecture, command interface, and internal algorithms in detail, with particular emphasis on the topology extraction, IBP reduction, renormalization, and finite field reconstruction stages.

Contents

1 Introduction

Precision predictions for hadron collider observables require the computation of scattering amplitudes at next-to-leading order (NLO) and beyond in the strong coupling expansion. While significant progress has been achieved in automating such calculations [1–3], the workflow remains technically demanding, involving multiple specialized tools that must be orchestrated carefully.

The typical NLO calculation pipeline consists of several stages:

1. **Diagram generation**: Enumeration of all Feynman diagrams contributing at a given loop order.
2. **Amplitude construction**: Application of Feynman rules to obtain algebraic expressions.
3. **Amplitude manipulation**: Dirac algebra, color algebra, and kinematic simplifications.
4. **Integral identification**: Extraction of loop integral topologies from the amplitude.
5. **Integral reduction**: Reduction to master integrals via integration-by-parts (IBP) identities.
6. **Renormalization**: Renormalization of UV and IR terms.
7. **Master integral evaluation**: Numerical or analytical computation of the master integrals.

GLAS addresses stages 1–7 by providing a unified command-line interface that automates the workflow and manages intermediate results. The system is designed with the following principles:

- **Modularity**: Each stage is implemented as a separate module with well-defined interfaces.
- **Parallelism**: Computationally intensive tasks are distributed across multiple cores.
- **Reproducibility**: All intermediate results are stored in structured run directories.
- **Extensibility**: New processes and models can be added through configuration files.

This report is organized as follows. In Section 2 we describe the overall architecture and run-based workflow. Section 3 details the three-phase calculation pipeline. The topology extraction algorithm is presented in Section 4, followed by the IBP reduction procedure in Section 5. Section 7 discusses UV renormalization and counterterm computation. The finite field approach for linear relation identification is detailed in Section 6. We discuss the command interface in Section A and conclude in Section 9.

2 Architecture

2.1 Run-Based Workflow

GLAS organizes all calculations around the concept of *runs*. A run represents a complete calculation for a specific process and is stored in a dedicated directory under `runs/{tag}_{nnnn}/`, where `tag` is derived from the process specification and `nnnn` is a zero-padded sequential counter.

Each run directory contains:

- `meta.json`: Process metadata including particle content, diagram counts, and job configuration.
- `diagrams/`: QGRAF output files organized by loop order (`0l/`, `1l/`).
- `form/`: FORM driver scripts and amplitude files.

- `Mathematica/`: Scripts for topology extraction, IBP reduction, coefficient simplification, and utilities.
- `logs/`: Execution logs organized by command.

The run metadata is stored in JSON format and tracks essential information:

```
{
  "process": "g g > t t~",
  "tag": "ggtT",
  "n0l": 3,
  "n1l": 28,
  "mand_define": "#call mandelstam2x2(p1,p2,p3,p4)",
  "gluon_refs": {"p1": "p3", "p2": "p4"},
  "created_at_utc": "2026-01-23T12:00:00Z"
}
```

2.2 Directory Structure

The GLAS installation has the following structure:

```
glas/
  glas.py                  # Entry point
  glaslib/                 # Core library
    cli.py                 # REPL command interface
    commands/              # Command implementations
    contracts/             # Amplitude contraction modules
    core/                  # Utilities (parallel, paths, refs)
  mathematica/scripts/     # Mathematica scripts
  resources/
    formlib/procedures/    # FORM procedure library
    diagrams/              # QGRAF binary and style files
  runs/                    # Output directories
```

2.3 External Dependencies

GLAS requires the following external tools:

- QGRAF [4]: Feynman diagram generator.
- FORM [5, 6]: Symbolic manipulation system.
- Mathematica: For topology extraction and IBP reduction.
- FeynCalc [21]: Mathematica package for loop integrals.
- Blade [20]: IBP reduction package (a KIRA [24] integration is planned).
- Fermat/Singular [22, 23]: Computer algebra backends.
- MultivariateApart [19]: Package for multivariate partial fraction decomposition.
- FiniteFlow [12]: Library for finite-field reconstruction.

3 The Three-Phase Pipeline

The calculation proceeds through three main phases, each corresponding to a set of GLAS commands.

3.1 Phase 1: Generation

The `generate` command initiates a new calculation:

```
glas> generate g g > t t~ --jobs 8
```

This command:

1. Creates a new run directory with a unique identifier.
2. Invokes QGRAF to generate tree-level ($\ell = 0$) and one-loop ($\ell = 1$) diagrams.
3. Parses the QGRAF output and stores diagram counts in metadata.
4. Prepares the FORM project structure with procedures and include files.

The process specification follows the standard notation: initial-state particles on the left of ">" and final-state particles on the right. Antiparticles are denoted by a tilde (e.g., `t~` for $\bar{t}$).

3.2 Phase 2: Evaluation

The evaluation phase applies Feynman rules and performs algebraic simplifications:

```
glas> evaluate lo --jobs 8           # Tree-level
glas> evaluate nlo --jobs 8 --dirac  # One-loop with Dirac simplification
```

For each diagram, the FORM driver:

1. Includes the diagram expression from QGRAF output.
2. Applies Feynman rules via `#call FeynmanRules`.
3. Substitutes Mandelstam invariants.
4. Optionally performs Dirac algebra simplification.
5. Writes the result to `Files/Amps/amp{0,1}l/d{i}.h`.

The `-dirac` flag enables simultaneous Dirac simplification, which includes:

- Gamma matrix algebra using the Chisholm identity.
- Trace evaluation for closed fermion loops.
- Spinor orthogonality conditions for external fermions.

3.3 Phase 3: Contraction

The contraction phase squares amplitudes and sums over helicities:

```
glas> contract lo --jobs 4   # |M_0|^2
glas> contract nlo --jobs 4  # Re(M_0^* M_1)
```

For LO contractions, we compute $|\mathcal{M}_0|^2$ summed over colors and helicities. For NLO, we compute the interference term $\Re(\mathcal{M}_0^* \mathcal{M}_1)$.

The contraction procedure includes:

1. Complex conjugation of amplitudes.
2. Polarization sum insertion for external gluons.
3. Color algebra evaluation.
4. Combination of diagram pairs.

4 Topology Extraction

After contraction, the NLO amplitude contains scalar Feynman integrals that must be identified and reduced. The topology extraction proceeds in four stages.

4.1 Stage 1: Incomplete Topology Identification

The Mathematica script `extract_topologies_stage1.m` loads the contracted amplitudes and identifies the propagator structures. Each loop integral is characterized by its set of propagators:

$$I[\{q_1^2 - m_1^2, q_2^2 - m_2^2, \ldots\}] = \int \frac{\mathrm{d}^D \ell}{(2\pi)^D} \frac{1}{(q_1^2 - m_1^2)(q_2^2 - m_2^2)\cdots} \tag{1}$$

where the q_i are linear combinations of the loop momentum ℓ and external momenta. This step is done in `FeynCalc` to extract and identify topologies.

The output is a list of "incomplete" topologies—propagator sets that may not span the full space needed for IBP reduction.

4.2 Stage 2: Topology Completion

The Python script `extend.py` completes each topology by adding auxiliary propagators. For a one-loop amplitude with n external legs, a complete topology requires $n + 1$ propagators (one more than the number of independent scalar products).

The completion algorithm uses breadth-first search (BFS) on an integer lattice where each node represents a shift vector for the propagator momentum:

$$q = \ell + \sum_{i=1}^{n-1} a_i p_i, \quad a_i \in \mathbb{Z} \tag{2}$$

The algorithm prioritizes:

1. Filling gaps between existing propagators if the path includes missing momentum directions.
2. Adding propagators from endpoints in directions not yet covered.

This ensures that the completed topology has propagators depending on all external momentum directions, which is essential for a valid IBP reduction.

4.3 Stage 2b: Topology Mapping

The script `extract_topologies_stage2.m` maps the amplitude integrals onto the completed topologies using FeynCalc's `FCLoopFindTopologies` and related functions. The output includes:

- `Files/integrals.m`: Integral definitions in Mathematica format.
- `form/Files/intrule.h`: FORM-formatted substitution rules.

4.4 Stage 3: Parallel FORM Processing

The final topology extraction stage processes the mapped integrals in parallel using FORM. The `ToTopos_J{k}of{N}.frm` drivers:

1. Load the integral rules from `intrule.h`.

2. Apply substitutions to express all integrals in terms of standard topology notation.
3. Write output to `Files/MOM1top/d{i}x{j}.h`.

This stage is executed by the command:

```
glas> extract topologies
```

5 IBP Reduction

The IBP reduction stage reduces all loop integrals to a minimal set of master integrals using integration-by-parts identities [8, 9].

5.1 Mandelstam Preparation

The script `mandIBP.m` computes the kinematic replacements needed for the reduction, storing results in `Files/mands.m`. For 5 particle processes the Mandelstam variables are the set:

$$\{s_{12}, s_{23}, s_{34}, s_{45}, s_{15}\}, \tag{3}$$

and for 4 particle processes the Mandelstam variables are the set:

$$\{s_{12}, s_{13}, s_{14}\} \quad \text{where} \tag{4}$$

$$s_{14} = \sum_{i}^{4} m_i^2 - s_{12} - s_{13}, \tag{5}$$

and the mandelstam variables are defined as:

$$s_{ij} = (p_i + p_j) \quad \text{if both momenta are incoming/outgoing} \tag{6}$$

$$s_{ij} = (p_i - p_j) \quad \text{if one of momenta is incoming and the other one is outgoing} \tag{7}$$

with on-shell external momenta $p_i \cdot p_i = m_i^2$.

5.2 IBP Reduction with Blade

The main reduction is performed by `IBP.m` using the Blade package. For each topology T_i, the reduction produces:

$$I_{T_i}[\{n_1, n_2, \ldots\}] = \sum_j c_j(\epsilon, s_{ij}, m^2)\, M_j \tag{8}$$

where M_j are master integrals and c_j are rational functions of the kinematic invariants and the dimensional regulator $\epsilon = (4 - D)/2$.

The output includes:

- `Files/IBP/IBP{i}.m`: Mathematica-format reduction rules.
- `form/Files/IBP/IBP{i}.h`: FORM-format reduction rules.

This workflow targets massive five-point amplitudes. IBP reduction can be expensive, so the top-quark mass is set to one to accelerate finite-field reconstruction of the IBP relations. After the reduction, the mass is restored by dimensional analysis (see the `RestoreMass` function in `glas/mathematica/scripts/IBP.m`). The result is then partially fractioned using `MultivariateApart`; `MultivariatePassToSingular` computes a Groebner basis in `Singular` and performs the reduction in `FORM`.

5.3 Symmetry Relations and PaVe Conversion

The script `SymmetryRelations.m` identifies relations between master integrals from different topologies and converts to the standard Passarino-Veltman (PaVe) basis [10]. The output includes:

- `Files/SymmetryRelations.m`: Master integral list and PaVe rules.
- `form/Files/SymmetryRelations.h`: FORM substitution rules.
- `form/Files/MastersToSym.h`: Symbolic master integral identifiers.

This pipeline is executed with:

```
glas> ibp
```

5.4 Master Integral Coefficient Extraction

The `reduce` command applies the IBP reduction rules via:

```
glas> reduce --jobs 4
```

The `reduce` command applies the IBP relations to the contracted amplitude and saves the results in:

- `Mathematica/Files/M0M1Reduced/`

The master-integral coefficients are then extracted with:

```
glas> micoef --jobs 4 [--combine]
```

This writes the coefficients of the master integrals as:

- `form/Files/MasterCoefs/c{i}.h`: Coefficient of master integral i.
- `Mathematica/Files/MasterCoefficients/mi{i}/MasterCoefficient{i}.m`: Per-master coefficients and independent-basis rules.
- `Mathematica/Files/MasterCoefficients.m`: Combined coefficient file if the `-combine` flag is used.

6 Linear Relations via Finite Fields

After IBP reduction, the amplitude coefficients are expressed as rational functions of kinematic invariants and ϵ. These expressions can be extremely large, making direct manipulation computationally prohibitive. GLAS employs finite field techniques via FiniteFlow [12] to identify linear relations and simplify the final result.

6.1 Finite Field Reconstruction

FiniteFlow treats each coefficient as a black-box rational function and reconstructs it from numerical evaluations over finite fields $\mathbb{F}_p$. The workflow is:

1. **Sampling**: Evaluate the function at many points in $\mathbb{F}_p^n$ by running the dataflow graph of the amplitude.
2. **Reconstruction**: Use functional reconstruction algorithms to recover a rational function in the kinematic variables from the samples.

3. **Lifting**: Repeat over multiple primes and apply rational reconstruction to lift finite-field results to $\mathbb{Q}$.

4. **Validation**: Check the reconstructed expression at additional random points and/or primes.

Because all intermediate operations stay in finite fields, the method avoids expression swell and large intermediate terms, and it scales much better than purely symbolic manipulation for multi-scale amplitudes.

6.2 Simplifying Rational Functions

After the master coefficients are identified, and optionally combined, a full partial-fractioning step may be impractical for large numbers of denominators and high polynomial degrees with current tools. GLAS addresses this by further simplifying the rational-function basis: it combines terms with shared denominators and eliminates redundant partial-fraction pieces. In practice, each master coefficient is written as a sum of rational functions,

$$c_i = \sum_\ell R_\ell, \tag{9}$$

The rational pieces R_ℓ typically share recurring denominator structures. The simplification proceeds in two stages:

1. **Denominator grouping**: combine terms with identical denominators to reduce the number of distinct rational pieces.

2. **Linear reduction**: use finite-field linear algebra to find relations among the grouped rational functions and keep only an independent basis.

A concrete example of denominator grouping is

$$R_\ell = \frac{n_1}{d_1 d_2} + \frac{n_2}{d_1} + \frac{n_3}{d_1 d_2^2} + \frac{n_4}{d_3} \tag{10}$$

$$= \frac{\frac{d_2 n_1 + n_3}{d_2}}{d_1 d_2} + \frac{n_2}{d_1} + \frac{n_4}{d_3} = \frac{q_1}{d_1 d_2} + \frac{q_2}{d_1} + \frac{q_3}{d_3}, \tag{11}$$

after which linear relations among the q_i can further reduce the basis, e.g. $q_1 = q_2 + q_3 - 2q_4$ (not related to example above).

The denominator grouping step is implemented in the FORM procedure `rationals.prc`.

6.3 Finding Linear Relations

The `linrels` command uses FiniteFlow to identify linear dependencies among the master integral coefficients:

```
glas> linrels [--combine]
```

The algorithm proceeds as follows:

1. **Setup**: Load the n_{mis} master integral coefficients $c_1(\epsilon, s), \ldots, c_{n_{\text{mis}}}(\epsilon, s)$.

2. **Sampling**: Evaluate all coefficients at random points in $\mathbb{F}_p$ for kinematic variables, keeping ϵ symbolic or also numerical.

3. **Linear algebra**: Construct the coefficient matrix and compute its rank over $\mathbb{F}_p$.

4. **Nullspace**: Find the nullspace vectors, which correspond to linear relations:

$$\sum_{i=1}^{n_{\text{mis}}} r_i \, c_i = 0 \tag{12}$$

5. **Reconstruction**: Reconstruct the rational coefficients r_i using multivariate interpolation.

6. **Verification**: Confirm the relation over $\mathbb{Q}$ using additional prime evaluations.

The optional `-combine` flag runs `CombineLinearRelations.m` to merge relations across master integrals and write a combined coefficient file to `Mathematica/Files/MasterCoefficients.m`.

After this step is done, we denote the extracted rational functions as $f\,[i, j]$, where i labels the master integral and j enumerates the rational-function entries. Finally, after finding linear relations in each master coefficient independently, we call:

```
glas> ratcombine
```

This runs `CombineRationalFunctions.m`, which finds linear relations among the independent functions extracted from all master coefficients. It significantly reduces analytic complexity; in five-point processes we have observed reductions from $\approx$ 40000 rational functions to $\approx$ 4000, and a disk-size reduction from roughly 2 GB to a few tens of MB. After the relations are found, the reconstructed independent functions are partially fractioned and collected as coefficients of a denominator basis.

7 Renormalization

The one-loop amplitude contains ultraviolet (UV) and infrared (IR) divergences that must be cancelled by counterterms derived from the renormalization of QCD parameters. GLAS implements the $\overline{\text{MS}}$ renormalization scheme with the following counterterm structure.

7.1 Mass Counterterms

For processes involving massive quarks, the mass counterterm contributes at NLO. For carrying out mass renormalization for a quark line with momentum k, mass m, and color indices i and j, one uses the mass insertion:

$$\mathcal{G}_{ij}^{\text{UV}\delta m}(k) = \frac{i\delta_{ik}}{\not{k} - m}(-i\delta m)\frac{i\delta_{kj}}{\not{k} - m} \tag{13}$$

with the mass counterterm:

$$\delta m = g_s^2 C_F N_\epsilon \left(\frac{\mu_R^2}{m^2}\right)^{\epsilon} \left(4 + \frac{3}{\epsilon}\right) m \tag{14}$$

The `evaluate mct` command processes mass counterterm diagrams:

```
glas> evaluate mct --jobs 4
glas> contract mct --jobs 4
```

These diagrams are tree-level insertions of the mass counterterm vertex and contribute to the finite part of the renormalized amplitude.

7.2 UV Counterterm Computation

After obtaining the tree-level squared amplitude and the mass counterterm from `evaluate/contract lo/mct`, one can call `uvct`, which computes the UV counterterms required to render the virtual amplitude finite:

```
glas> uvct
```

For a Born cross section of order α_s^b, the complete UV counterterm structure is given by the following contributions:

Strong Coupling Renormalization (Vas)

The contribution due to strong-coupling renormalization reads:

$$V_{\alpha_s}^{\mathrm{UV}} = b \left|\mathcal{A}^{(n,0)}\right|^2 g_s^2 N_\epsilon \left[\frac{4}{3\epsilon} T_F n_{lf} - \frac{11}{3\epsilon} C_A + \frac{4}{3\epsilon} T_F \sum_{\{n_{hf}\}} \left(\frac{\mu_R^2}{m_{hf}^2}\right)^\epsilon\right] \tag{15}$$

where n_{lf} is the number of massless (light) quark flavors, and the sum runs over all heavy flavors $\{n_{hf}\}$ circulating in the loops.

Yukawa Coupling Renormalization (Vyuk)

The contribution due to renormalization of Yukawa couplings reads:

$$V_{\mathrm{yuk}}^{\mathrm{UV}} = -\left|\mathcal{A}^{(n,0)}\right|^2 g_s^2 N_\epsilon 2 C_F \left[\frac{3}{\epsilon} n_{\mathrm{yuk},l} + \left(4 + \frac{3}{\epsilon}\right) \sum_{\{n_{\mathrm{yuk},h}\}} \left(\frac{\mu_R^2}{m_{\mathrm{yuk},h}^2}\right)^\epsilon\right] \tag{16}$$

with $n_{\mathrm{yuk},l}$ and $n_{\mathrm{yuk},h}$ the number of Yukawa vertices with massless and massive particles respectively. Color singlets and massless color triplets do not require any wave-function renormalization.

Gluon Wave-Function Renormalization (Vg)

The gluon wave function is renormalized only if there are massive color triplets fermions running in the loop. Denoting by n_g the number of external gluons at Born level, the contribution reads:

$$V_{\mathrm{gwf}}^{\mathrm{UV}} = -n_g \left|\mathcal{A}^{(n,0)}\right|^2 g_s^2 N_\epsilon T_F \frac{4}{3\epsilon} \sum_{\{n_{hf}\}} \left(\frac{\mu_R^2}{m_{hf}^2}\right)^\epsilon \tag{17}$$

External Massive Quark Wave-Function Renormalization (Vzt)

The wave-function renormalization of the external massive quarks (denoted ext_{hf}) gives:

$$V_{\mathrm{ext}_{hf}}^{\mathrm{UV}} = -\left|\mathcal{A}^{(n,0)}\right|^2 g_s^2 N_\epsilon C_F \left(4 + \frac{3}{\epsilon}\right) \sum_{\{\mathrm{ext}_{hf}\}} \left(\frac{\mu_R^2}{m_{\mathrm{ext}_{hf}}^2}\right)^\epsilon \tag{18}$$

Here $N_\epsilon = \frac{(4\pi)^\epsilon}{16\pi^2}\Gamma(1+\epsilon)$ is the standard loop normalization factor.

The counterterm contributions are stored in `Mathematica/UVCT/{Vas,Vzt,Vg,Vm}.m` for subsequent combination with the virtual amplitude.

7.3 Infrared Subtraction

While UV divergences are removed by renormalization, the virtual amplitude still contains infrared (IR) divergences from soft and collinear gluon exchange. These divergences cancel against corresponding singularities in the real emission contributions when computing physical observables. The Catani-Seymour dipole subtraction formalism [17] provides a systematic framework for this cancellation.

The IR-divergent structure of the one-loop amplitude is captured by the $\mathbf{I}$ operator:

$$\mathcal{M}_1^{\text{ren}} = \mathbf{I}(\epsilon) \cdot \mathcal{M}_0 + \mathcal{M}_1^{\text{fin}} + \mathcal{O}(\epsilon) \tag{19}$$

where $\mathcal{M}_1^{\text{fin}}$ is the finite remainder and $\mathbf{I}(\epsilon)$ contains the universal IR poles.

The matrix element of the $\mathbf{I}$ operator for the interference with the tree-level amplitude reads [18]:

$$\begin{aligned} 2\,\Re\langle \mathcal{M}_{n+1}^{(0)} | \mathbf{I} | \mathcal{M}_{n+1}^{(0)} \rangle = \frac{\alpha_s}{4\pi} \frac{1}{\epsilon} \Bigg[& \left(-\frac{2}{\epsilon} \sum_{i_0} C_{i_0} + \sum_i \gamma_0^i \right) |\mathcal{M}_{n+1}^{(0)}|^2 \\ & + 2 \sum_{(i_0,j_0)} \ln \left| \frac{\mu_R^2}{s_{i_0 j_0}} \right| \langle \mathcal{M}_{n+1}^{(0)} | \mathbf{T}_{i_0} \cdot \mathbf{T}_{j_0} | \mathcal{M}_{n+1}^{(0)} \rangle \\ & - \sum_{(I,J)} \frac{1}{v_{IJ}} \ln \left(\frac{1+v_{IJ}}{1-v_{IJ}} \right) \langle \mathcal{M}_{n+1}^{(0)} | \mathbf{T}_I \cdot \mathbf{T}_J | \mathcal{M}_{n+1}^{(0)} \rangle \\ & + 4 \sum_{I,j_0} \ln \left| \frac{m_I \mu_R}{s_{I j_0}} \right| \langle \mathcal{M}_{n+1}^{(0)} | \mathbf{T}_I \cdot \mathbf{T}_{j_0} | \mathcal{M}_{n+1}^{(0)} \rangle \Bigg] \end{aligned} \tag{20}$$

where:

- The sum $\sum_{i_0}$ runs over massless partons, while $\sum_i$ runs over all partons.
- The sum $\sum_{(i_0,j_0)}$ runs over distinct pairs of massless partons.
- The sum $\sum_{(I,J)}$ runs over distinct pairs of massive partons.
- The sum $\sum_{I,j_0}$ runs over mixed massive-massless pairs.

Kinematic Invariants

The Mandelstam-like kinematic invariants are defined as:

$$p_I^2 = m_I^2, \qquad v_I = p_I / m_I, \qquad v_{IJ} = \sqrt{1 - \frac{m_I^2 m_J^2}{(p_I \cdot p_J)^2}} \tag{21}$$

where $I, J, K, \ldots$ denote indices for massive partons, while $i_0, j_0, k_0, \ldots$ denote massless partons.

The invariant s_{ij} is defined with an imaginary prescription:

$$s_{ij} = 2\sigma_{ij}\, p_i \cdot p_j + i0^+ \tag{22}$$

where:

$$\sigma_{ij} = \begin{cases} +1 & \text{if } p_i \text{ and } p_j \text{ are both incoming or both outgoing} \\ -1 & \text{otherwise} \end{cases} \tag{23}$$

Color Charge Operators

The color charge operators $\mathbf{T}_i$ act on the color space of parton i:

$$\langle c_1, \ldots, c_i, \ldots, c_n, c|\mathbf{T}_i|b_1, \ldots, b_i, \ldots, b_n\rangle = \langle c_1, \ldots, c_i, \ldots, c_n|T_i^c|b_1, \ldots, b_i, \ldots, b_n\rangle = \delta_{c_1 b_1} \cdots T^c_{c_i b_i} \cdots \delta_{c_n b_n} \tag{24}$$

The generators $T^c_{c_1 c_2}$ depend on the parton type:

$$T^c_{c_1 c_2} = i f^{c_1 c c_2} \qquad \text{(emitter is a gluon)} \tag{25}$$
$$T^c_{c_1 c_2} = t^c_{c_1 c_2} = -t^c_{c_2 c_1} \qquad \text{(emitter is an outgoing quark/anti-quark)} \tag{26}$$
$$T^c_{c_1 c_2} = -t^c_{c_2 c_1} = t^c_{c_1 c_2} \qquad \text{(emitter is an incoming quark/anti-quark)} \tag{27}$$

The color operators satisfy important identities:

$$\sum_i \mathbf{T}_i|\mathcal{M}_n\rangle = 0, \qquad T_i^c T_j^c = \mathbf{T}_i \cdot \mathbf{T}_j = \mathbf{T}_j \cdot \mathbf{T}_i, \qquad \mathbf{T}_i \cdot \mathbf{T}_i = \mathbf{T}_i^2 = C_i = C_{a_i} \tag{28}$$

where the Casimir operators are:

$$C_g = C_A, \qquad C_q = C_{\bar{q}} = C_F \tag{29}$$

and the trace normalization is:

$$\mathrm{Tr}[t^a t^b] = T_F \delta^{ab} = \frac{1}{2}\delta^{ab} \tag{30}$$

Anomalous Dimensions

The leading-order anomalous dimensions appearing in the IR structure are:

$$\gamma_0^q = -3C_F \qquad \text{(massless quarks/anti-quarks)} \tag{31}$$
$$\gamma_0^Q = -2C_F \qquad \text{(massive quarks/anti-quarks)} \tag{32}$$
$$\gamma_0^g = -\beta_0 = -\frac{11}{3}C_A + \frac{4}{3}T_F n_l \qquad \text{(gluons)} \tag{33}$$

The `ioperator` command computes the $\mathbf{I}$ operator contribution:

```
glas> ioperator
```

This command:

1. Generates FORM drivers for each pair of external partons (i, j).
2. Computes the color-correlated Born amplitudes $\langle\mathcal{M}_0|\mathbf{T}_i \cdot \mathbf{T}_j|\mathcal{M}_0\rangle$.
3. Combines with the kinematic factors $(-s_{ij})^{-\epsilon}$.
4. Produces the integrated dipole contribution $\mathbf{I} \cdot |\mathcal{M}_0|^2$.

The output files are:

- `form/Files/Ioperator/I{i}x{j}.h`: Color-correlated contributions for parton pair (i, j).
- `form/Files/Ioperator/Ioperator_master.h`: Combined $\mathbf{I}$ operator result.
- `Mathematica/Files/Ioperator.m`: Mathematica-format output.

7.4 Renormalized Virtual Amplitude

The complete renormalized one-loop amplitude is:

$$2\,\Re(\mathcal{M}_0^*\mathcal{M}_1^{\text{fin}}) = 2\,\Re(\mathcal{M}_0^*\mathcal{M}_1^{\text{bare}}) - 2 * \Re\left[\langle\mathcal{M}_0|\mathbf{I}|\mathcal{M}_0\rangle\right] + V_{a_s}^{\text{UV}} + V_{\text{ext}_{hf}}^{\text{UV}} + V_{\text{m}}^{\text{UV}} + V_{\text{yuk}}^{\text{UV}} \tag{34}$$

which is free of both UV and IR poles and ready for numerical integration.

The combination is performed automatically once all counterterm contributions have been computed, yielding the finite remainder of the amplitude. Since we work in CDR, both UV and IR counterterms are expanded up to order ϵ^0.

7.5 Complete NLO Virtual Result

The full workflow for obtaining the finite virtual contribution for the process $q\bar{q} \to t\bar{t}$, for example, is:

```
glas> generate q q~ > t t~ --jobs 4
glas> evaluate nlo --jobs 4 --dirac
glas> contract nlo --jobs 4
glas> evaluate mct --jobs 4
glas> contract mct --jobs 4
glas> uvct
glas> ioperator
glas> extract topologies
glas> ibp
glas> reduce --jobs 4
glas> micoef --jobs 4 [--combine]
glas> linrels [--combine]
glas> ratcombine
```

After these steps, the finite virtual matrix element squared is available for combination with real emission contributions computed using standard dipole subtraction.

8 FORM Procedure Library

GLAS includes a library of FORM procedures in `resources/formlib/procedures/`. The key procedures are:

Physics Rules and Algebra

- `FeynmanRules.prc`: Applies QCD Feynman rules, converting QGRAF fields to spinors and polarization vectors, inserting vertices and propagators, and normalizing Lorentz and color structures.
- `MassCT.prc`: Implements mass counterterm insertions for massive fermions, including the explicit δm term and bookkeeping for the coupling power.
- `DiracSimplify.prc`: Performs gamma-matrix algebra, contracts Lorentz indices, applies on-shell relations, and reduces spinor chains to trace form.
- `Conjugate.prc`: Complex-conjugates amplitudes, reverses spinor chains, and swaps polarization vectors and color structures consistently.
- `PolarizationSums.prc`: Executes polarization sums and traces for external states, producing contracted chains ready for scalar integral extraction.
- `color.prc`: Reduces SU(N) color structures, expands traces, contracts adjoint indices, and substitutes SU(3) constants.

Kinematic Substitutions

- `mandelstam2x2.prc`: Rewrites dot products for $2 \to 2$ kinematics in terms of Mandelstam variables and masses.
- `mandelstam2x3.prc`: Rewrites dot products for $2 \to 3$ kinematics in terms of Mandelstam variables and masses.

Rational Function Handling

- `SymToRat.prc`: Converts kinematic symbols and denominators into rational-function placeholders for downstream collection.
- `RationalFunction.prc`: Simplifies `den(x)*x = 1` if x is a combination of terms (e.g. $s_{12} * den(s_{12} - m_t^2) - m_t^2 * den(s_{12} - m_t^2) = 1$). And this is done for each denominator alone.
- `PolyRat.prc`: Promotes selected symbols to rational form and expands `rat` objects into `den`-based expressions.
- `Together.prc`: Takes common denominator and keeps the output in terms of the `FORM PolyRatFun rat(num,den)`.
- `rationals.prc`: Groups identical denominators, builds polynomial rational functions, and normalizes `Rat` structures before applying `linrels`.
- `toden.prc`: Rewrites inverse kinematic invariants into explicit denominator factors.

9 Conclusions

We have presented GLAS, a unified framework for automated NLO QCD calculations. The system integrates diagram generation, symbolic manipulation, integral reduction, renormalization, and coefficient simplification into a coherent pipeline with parallel execution capabilities.

Key features include:

- Run-based workflow with complete reproducibility.
- Parallel execution at the diagram level.
- Automated topology completion and IBP reduction.
- $\overline{\text{MS}}$ renormalization with UV counterterm computation.
- Finite field techniques for linear relation identification via FiniteFlow.
- Verbose streaming mode for debugging and monitoring.
- Modular architecture enabling easy extension.

The finite field approach implemented in the `linrels` command provides significant computational advantages for complex processes, enabling the simplification of master integral coefficients that would be intractable with purely symbolic methods.

Future developments will include support for NNLO calculations, extended renormalization schemes, and integration with Monte Carlo event generators for phenomenological applications.

Acknowledgments

We thank the authors of QGRAF, FORM, FeynCalc, Blade, and FiniteFlow for making their software publicly available. This work was supported in part by [funding agency].

References

[1] A. Signer and D. Stöckinger, "Using dimensional reduction for hadronic collisions," Nucl. Phys. B **808** (2009) 88.

[2] S. Frixione, P. Nason and C. Oleari, "Matching NLO QCD computations with Parton Shower simulations: the POWHEG method," JHEP **11** (2007) 070.

[3] J. Alwall *et al.*, "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations," JHEP **07** (2014) 079.

[4] P. Nogueira, "Automatic Feynman graph generation," J. Comput. Phys. **105** (1993) 279.

[5] J. A. M. Vermaseren, "New features of FORM," arXiv:math-ph/0010025.

[6] J. Kuipers, T. Ueda, J. A. M. Vermaseren and J. Vollinga, "FORM version 4.0," Comput. Phys. Commun. **184** (2013) 1453.

[7] V. Shtabovenko, R. Mertig and F. Orellana, "FeynCalc 9.3: New features and improvements," Comput. Phys. Commun. **256** (2020) 107478.

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

[9] F. V. Tkachov, "A Theorem on Analytical Calculability of Four Loop Renormalization Group Functions," Phys. Lett. B **100** (1981) 65.

[10] G. Passarino and M. J. G. Veltman, "One Loop Corrections for e+ e- Annihilation Into mu+ mu- in the Weinberg Model," Nucl. Phys. B **160** (1979) 151.

[11] M. Czakon, "Tops from Light Quarks: Full Mass Dependence at Two-Loops in QCD," Phys. Lett. B **664** (2008) 307.

[12] T. Peraro, "FiniteFlow: multivariate functional reconstruction using finite fields and dataflow graphs," JHEP **07** (2019) 031.

[13] A. von Manteuffel and R. M. Schabinger, "A novel approach to integration by parts reduction," Phys. Lett. B **744** (2015) 101.

[14] T. Peraro, "Scattering amplitudes over finite fields and multivariate functional reconstruction," JHEP **12** (2016) 030.

[15] J. Klappert and F. Lange, "Reconstructing rational functions with FireFly," Comput. Phys. Commun. **247** (2020) 106951.

[16] A. Denner, "Techniques for calculation of electroweak radiative corrections at the one loop level and results for W physics at LEP-200," Fortsch. Phys. **41** (1993) 307.

[17] S. Catani and M. H. Seymour, "A General algorithm for calculating jet cross-sections in NLO QCD," Nucl. Phys. B **485** (1997) 291.

[18] M. Czakon and D. Heymes, Nucl. Phys. B **890** (2014), 152-227 doi:10.1016/j.nuclphysb.2014.11.006 [arXiv:1408.2500 [hep-ph]].

[19] M. Heller and A. von Manteuffel, Comput. Phys. Commun. **271** (2022), 108174 doi:10.1016/j.cpc.2021.108174 [arXiv:2101.08283 [cs.SC]].

[20] X. Guan, X. Liu, Y. Q. Ma and W. H. Wu, Comput. Phys. Commun. **310** (2025), 109538 doi:10.1016/j.cpc.2025.109538 [arXiv:2405.14621 [hep-ph]].

[21] R. Mertig, M. Bohm and A. Denner, Comput. Phys. Commun. **64** (1991), 345-359 doi:10.1016/0010-4655(91)90130-D

[22] Decker, W.; Greuel, G.-M.; Pfister, G.; Schönemann, H.: SINGULAR 4-4-0 — A computer algebra system for polynomial computations. https://www.singular.uni-kl.de (2024).

[23] Robert H. Lewis. Computer Algebra System Fermat. http://home.bway.net/lewis.

[24] P. Maierhöfer, J. Usovitsch and P. Uwer, Comput. Phys. Commun. **230** (2018), 99-112 doi:10.1016/j.cpc.2018.04.012 [arXiv:1705.05610 [hep-ph]].

A Command Reference

A.1 Core Commands

- `generate` *`process`*: Generate diagrams and initialize a new run.
- `evaluate {lo|nlo|mct}`: Apply Feynman rules and kinematics to diagram amplitudes.
- `contract {lo|nlo|mct}`: Square amplitudes and sum over colors and helicities.
- `uvct`: Compute UV counterterms.
- `extract topologies`: Identify and complete loop topologies.
- `ibp`: Perform IBP reduction (mandIBP, IBP, SymmetryRelations).
- `reduce`: Apply reduction rules to contracted amplitudes.
- `micoef`: Extract master-integral coefficients; use `-combine` to sum across diagrams.
- `linrels`: Find linear relations between coefficients; use `-combine` to merge results.
- `ratcombine`: Combine rational functions across coefficients.
- `ioperator`: Apply IR subtraction operators.

A.2 Utility Commands

- `runs [`*`tag`*`]`: List available runs, optionally filtered by tag.
- `use` *`run`*: Attach to an existing run directory.
- `show`: Display current configuration and active run metadata.
- `setrefs`: Set gluon polarization references for external gluons.
- `verbose [on|off]`: Toggle verbose output streaming.
- `clean`: Delete all run directories.

A.3 Common Flags

All commands support the following flags:

- `-jobs K`: Number of parallel workers (default: 1).
- `-verbose`: Enable live output streaming.
- `-quiet`: Suppress verbose output.

The `evaluate` command additionally supports `-dirac` for simultaneous Dirac simplification.

A.4 Verbose Mode

When verbose mode is enabled (either per-command with `-verbose` or globally with `verbose on`), all subprocess output is streamed live to the terminal with informative prefixes:

```
[form eval lo J1/4] Processing diagram 1...
[mma stage1] Loading FeynCalc...
[py extend] Extending topology 3 of 5...
```

Logs are always written to `logs/{command}/{step}.log` regardless of verbose setting.

B Parallel Execution

GLAS implements parallel execution at the diagram level. When `-jobs K` is specified, the diagram range is partitioned into K chunks, each processed by an independent FORM worker.

The chunking algorithm ensures balanced workload:

$$\text{chunk}(k) = \left[\left\lfloor \frac{(k-1)N}{K} \right\rfloor + 1, \left\lfloor \frac{kN}{K} \right\rfloor \right] \tag{35}$$

where N is the total number of diagrams and $k \in \{1, \ldots, K\}$.

Each worker writes to separate output files, avoiding race conditions. The effective number of jobs is automatically clamped to the diagram count to prevent empty chunks.

C Example: $gg \to t\bar{t}$ at NLO

We illustrate the complete workflow for top quark pair production in gluon fusion:

```
glas> generate g g > t t~ --jobs 3
[generate] Created run: ggtT_0001
[generate] n0l=3, n1l=28

glas> evaluate lo --jobs 3
[evaluate lo] All jobs finished OK.

glas> evaluate nlo --jobs 3 --dirac
[polarization] Gluon reference vectors / orthogonality choices
  process     : g g > t t~
  gluons @    : p1, p2
  default ref : p1 (first massless momentum)
  Tip: enter '2' for p2, or 'p2'. Empty -> default.

Vector orthogonal to gluon p1 [default p1]: p2
Vector orthogonal to gluon p2 [default p1]: p1

[evaluate nlo] All jobs finished OK.
[dirac nlo] All jobs finished OK.

glas> evaluate mct
[evaluate mct] All jobs finished OK.

glas> contract lo --jobs 3
[contract lo] All jobs finished OK.

glas> contract nlo --jobs 3
[contract nlo] All jobs finished OK.

glas> contract mct --jobs 3
[contract mct] All jobs finished OK.

glas> uvct
[start Vas] Vas.frm
[done Vas]
[start Vzt] Vzt.frm
[done Vzt]
[start Vg] Vg.frm
[done Vg]
[uvct] Completed.
```

```
glas> extract topologies
[extract] Running stage1...
[extract] Stage1 OK
[extract] Running extend.py...
[extract] extend.py OK
[extract] Running stage2...
[extract] Stage2 OK
[extract] Jobs (1-3): 3
[extract] Running ToTopos with 3 parallel job(s)...
[extract] ToTopos OK

glas> ibp
[ibp] Running mandIBP.m...
[ibp] mandIBP.m OK -> Files/mands.m
[ibp] Running IBP.m...
[ibp] IBP.m OK -> Files/IBP/ (7 topology files)
[ibp] Running SymmetryRelations.m...
[ibp] SymmetryRelations.m OK -> Files/SymmetryRelations.m
[ibp] Recorded nmis = 17 in meta.json
[ibp] Also generated ../form/Files/MastersToSym.h (master integral substitution rules)


glas> ioperator
.
.
.
[ioperator] Completed.

glas> reduce --jobs 3
[reduce] All reduction jobs finished OK.

glas> micoef --combine --jobs 3
Running MasterCoefficients (3 jobs for 3 tree diagrams)...
[micoef] MasterCoefficients finished OK.
[micoef --combine] Running SumMasterCoefs (3 jobs for 17 master integrals)...
[micoef --combine] SumMasterCoefs finished OK.

glas> linrels --verbose
[linrels] Running LinearRelations.m ...
[mma linrels] ******************
[mma linrels] ******************
[mma linrels] Total Number of Initial rational functions: 782 was reduced to 272 in
    the first step.
[mma linrels] ******************
[mma linrels] ******************
[linrels] LinearRelations.m OK.

glas> ratcombine --verbose
[ratcombine] Copied AmpResult4.m
[ratcombine] Running CombineRationalFunctions.m ...
[mma ratcombine] Performing partial fractioning on the final independent rational
    functions.
[ratcombine] OK -> Files/MasterCoefficients.m
[mma ratcombine] ******************
[mma ratcombine] ******************
[mma ratcombine] Total Number of rational functions: 782 was reduced to 142.
[mma ratcombine] ******************
[mma ratcombine] ******************
```

```
[mma ratcombine]
```

Finally one can run the example by going to `runs/{ggtT}_{0001}/AmpResults4.m` and run the script to obtain:

```
Tree-level Comparison with madgraph: 1- amp/MGFinite = -3.55065*10^-7
Pole Cancellation: Poles[-1]+ Poles[-2]+IR+UV = 0
Finite contribution Comparison with madgraph: 1- amp/MGFinite = 8.77076*10^-14
```

The complete calculation produces:

- 3 tree-level diagrams and 33 one-loop diagrams.
- 7 independent loop topologies after completion.
- 17 independent master integrals after IBP reduction.

D Example: $q\bar{q} \to t\bar{t}$ at NLO

We illustrate the complete workflow for top quark pair production in gluon fusion:

```
glas> generate q q~ > t t~
[generate] Created run: ggtT_0001
[generate] n0l=1, n1l=10

glas> evaluate lo
[evaluate lo] All jobs finished OK.

glas> evaluate nlo
[evaluate nlo] All jobs finished OK.
[dirac nlo] All jobs finished OK.

glas> evaluate mct
[evaluate mct] All jobs finished OK.

glas> contract lo
[contract lo] All jobs finished OK.

glas> contract nlo
[contract nlo] All jobs finished OK.

glas> contract mct
[contract mct] All jobs finished OK.

glas> uvct
[start Vas] Vas.frm
[done Vas]
[start Vzt] Vzt.frm
[done Vzt]
[start Vg] Vg.frm
[done Vg]
[uvct] Completed.

glas> extract topologies
[extract] Running stage1...
[extract] Stage1 OK
[extract] Running extend.py...
[extract] extend.py OK
[extract] Running stage2...
```

```
[extract] Stage2 OK
[extract] Jobs (1-1): 1
[extract] Running ToTopos with 1 parallel job(s)...
[extract] ToTopos OK

glas> ibp
[ibp] Running mandIBP.m...
[ibp] mandIBP.m OK -> Files/mands.m
[ibp] Running IBP.m...
[ibp] IBP.m OK -> Files/IBP/ (3 topology files)
[ibp] Running SymmetryRelations.m...
[ibp] SymmetryRelations.m OK -> Files/SymmetryRelations.m
[ibp] Recorded nmis = 8 in meta.json
[ibp] Also generated ../form/Files/MastersToSym.h (master integral substitution rules)


glas> ioperator
.
.
.
[ioperator] Completed.

glas> reduce
[reduce] All reduction jobs finished OK.

glas> micoef --combine
Running MasterCoefficients (1 jobs for 1 tree diagrams)...
[micoef] MasterCoefficients finished OK.
[micoef --combine] Running SumMasterCoefs (1 jobs for 8 master integrals)...
[micoef --combine] SumMasterCoefs finished OK.

glas> linrels --verbose
[linrels] Running LinearRelations.m ...
[mma linrels] ******************
[mma linrels] ******************
[mma linrels] Total Number of Initial rational functions: 133 was reduced to 57 in the
     first step.
[mma linrels] ******************
[mma linrels] ******************
[linrels] LinearRelations.m OK.

glas> ratcombine
[ratcombine] Copied AmpResult4.m
[ratcombine] Running CombineRationalFunctions.m ...
[mma ratcombine] Performing partial fractioning on the final independent rational
    functions.
[mma ratcombine] ******************
[mma ratcombine] ******************
[mma ratcombine] Total Number of rational functions 133 was reduced to 44.
[mma ratcombine] ******************
[mma ratcombine] ******************

[ratcombine] OK -> Files/MasterCoefficients.m
```

Finally one can run the example by going to `runs/{qQtT}_{0001}/AmpResults4.m` and run the script to obtain:

```
Tree-level Comparison with madgraph: 1- amp/MGFinite = 1.22125*10^-15
Pole Cancellation: Poles[-1]+ Poles[-2]+IR+UV = 0
```

```
Finite contribution Comparison with madgraph: 1- amp/MGFinite = -6.21725*10^-15
```

The complete calculation produces:

- 1 tree-level diagrams and 10 one-loop diagrams.
- 3 independent loop topologies after completion.
- 8 independent master integrals after IBP reduction.