GLAS: General Loop Amplitude System

A Unified Framework for Automated NLO QCD Calculations

M. Houmani

High Energy Physics Group

`mahdi.houmani@email.com`

January 24, 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**: UV divergence cancellation through counterterm insertion.
7. **Coefficient simplification**: Identification of linear relations using finite field techniques.
8. **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 paper 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 6 discusses UV renormalization and counterterm computation. The finite field approach for linear relation identification is detailed in Section 7. We discuss the command interface in Section 8 and conclude in Section 12.

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 and IBP reduction.
- `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 [7]: Mathematica package for loop integrals.
- Blade: IBP reduction package.
- Fermat/Singular: Computer algebra backends.

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 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 with 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  # 2 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 $2\,\mathrm{Re}(\mathcal{M}_0^*\mathcal{M}_1)$.

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.

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`.

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`.

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:

$$I_{T_i}[\{n_1, n_2, \ldots\}] = \sum_j c_j(\epsilon, s_{ij}, m^2)\, M_j \tag{3}$$

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.

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.

6 Renormalization

The one-loop amplitude contains ultraviolet (UV) 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.

6.1 Counterterm Lagrangian

The QCD counterterm Lagrangian in the $\overline{\text{MS}}$ scheme reads:

$$\mathcal{L}_{\text{CT}} = -\delta Z_t\, m_t \bar{\psi}_t \psi_t - \delta Z_g\, g_s \sum_q \bar{\psi}_q \gamma^\mu T^a \psi_q A^a_\mu + \ldots \tag{4}$$

where the renormalization constants are expanded as:

$$\delta Z_t = \frac{\alpha_s}{4\pi}\left(\frac{1}{\epsilon} - \gamma_E + \ln 4\pi\right)\delta Z_t^{(1)} + \mathcal{O}(\alpha_s^2), \tag{5}$$

$$\delta Z_g = \frac{\alpha_s}{4\pi}\left(\frac{1}{\epsilon} - \gamma_E + \ln 4\pi\right)\delta Z_g^{(1)} + \mathcal{O}(\alpha_s^2). \tag{6}$$

6.2 UV Counterterm Computation

The `uvct` command 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:

$$V_{\alpha_s}^{\mathrm{UV}} = b \left|\mathcal{A}^{(n,0)}\right|^2 g_s^2 N_\epsilon \left[\frac{4}{3\epsilon} T_F n_{lf} - \frac{11}{3\epsilon} C_A + \frac{4}{3\epsilon} T_F \sum_{\{n_{hf}\}} \left(\frac{\mu_R^2}{m_{hf}^2}\right)^\epsilon\right] \tag{7}$$

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:

$$V_{\mathrm{yuk}}^{\mathrm{UV}} = -\left|\mathcal{A}^{(n,0)}\right|^2 g_s^2 N_\epsilon 2 C_F \left[\frac{3}{\epsilon} n_{\mathrm{yuk},l} + \left(4 + \frac{3}{\epsilon}\right) \sum_{\{n_{\mathrm{yuk},h}\}} \left(\frac{\mu_R^2}{m_{\mathrm{yuk},h}^2}\right)^\epsilon\right] \tag{8}$$

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:

$$V_{\mathrm{gwf}}^{\mathrm{UV}} = -n_g \left|\mathcal{A}^{(n,0)}\right|^2 g_s^2 N_\epsilon T_F \frac{4}{3\epsilon} \sum_{\{n_{hf}\}} \left(\frac{\mu_R^2}{m_{hf}^2}\right)^\epsilon \tag{9}$$

External Massive Quark Wave-Function Renormalization (Vzt)

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

$$V_{\mathrm{ext}_{hf}}^{\mathrm{UV}} = -\left|\mathcal{A}^{(n,0)}\right|^2 g_s^2 N_\epsilon C_F \left(4 + \frac{3}{\epsilon}\right) \sum_{\{\mathrm{ext}_{hf}\}} \left(\frac{\mu_R^2}{m_{\mathrm{ext}_{hf}}^2}\right)^\epsilon \tag{10}$$

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.

6.3 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:

$$\mathcal{G}_{ij}^{\mathrm{UV}\delta m}(k) = \frac{i\delta_{ik}}{\not{k} - m}(-i\delta m)\frac{i\delta_{kj}}{\not{k} - m} \tag{11}$$

with the mass counterterm:

$$\delta m = g_s^2 C_F N_\epsilon \left(\frac{\mu_R^2}{m^2}\right)^\epsilon \left(4 + \frac{3}{\epsilon}\right) m \tag{12}$$

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.

6.4 Infrared Subtraction: The I Operator

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 **I** operator:

$$\mathcal{M}_1^{\mathrm{ren}} = \mathbf{I}(\epsilon) \cdot \mathcal{M}_0 + \mathcal{M}_1^{\mathrm{fin}} + \mathcal{O}(\epsilon) \tag{13}$$

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

The matrix element of the **I** operator for the interference with the tree-level amplitude reads:

$$\begin{aligned}
2\,\mathrm{Re}\langle \mathcal{M}_{n+1}^{(0)}|\mathbf{I}|\mathcal{M}_{n+1}^{(0)}\rangle = \frac{\alpha_s}{4\pi}\frac{1}{\epsilon}\Bigg[&\left(-\frac{2}{\epsilon}\sum_{i_0} C_{i_0} + \sum_i \gamma_0^i\right)|\mathcal{M}_{n+1}^{(0)}|^2 \\
&+ 2\sum_{(i_0,j_0)} \ln\left|\frac{\mu_R^2}{s_{i_0 j_0}}\right| \langle \mathcal{M}_{n+1}^{(0)}|\mathbf{T}_{i_0}\cdot\mathbf{T}_{j_0}|\mathcal{M}_{n+1}^{(0)}\rangle \\
&- \sum_{(I,J)} \frac{1}{v_{IJ}} \ln\left(\frac{1+v_{IJ}}{1-v_{IJ}}\right) \langle \mathcal{M}_{n+1}^{(0)}|\mathbf{T}_I\cdot\mathbf{T}_J|\mathcal{M}_{n+1}^{(0)}\rangle \\
&+ 4\sum_{I,j_0} \ln\left|\frac{m_I \mu_R}{s_{I j_0}}\right| \langle \mathcal{M}_{n+1}^{(0)}|\mathbf{T}_I\cdot\mathbf{T}_{j_0}|\mathcal{M}_{n+1}^{(0)}\rangle\Bigg]
\end{aligned} \tag{14}$$

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:

$$p_I^2 = m_I^2, \qquad v_I = p_I/m_I, \qquad v_{IJ} = \sqrt{1 - \frac{m_I^2 m_J^2}{(p_I \cdot p_J)^2}} \tag{15}$$

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:

$$s_{ij} = 2\sigma_{ij}\, p_i \cdot p_j + i0^+ \tag{16}$$

where:

$$\sigma_{ij} = \begin{cases} +1 & \text{if } p_i \text{ and } p_j \text{ are both incoming or both outgoing} \\ -1 & \text{otherwise} \end{cases} \tag{17}$$

Color Charge Operators

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

$$\langle c_1, \ldots, c_i, \ldots, c_n, c|\mathbf{T}_i|b_1, \ldots, b_i, \ldots, b_n\rangle = \langle c_1, \ldots, c_i, \ldots, c_n|T_i^c|b_1, \ldots, b_i, \ldots, b_n\rangle = \delta_{c_1 b_1} \cdots T^c_{c_i b_i} \cdots \delta_{c_n b_n} \tag{18}$$

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

$$T^c_{c_1c_2} = if^{c_1cc_2} \qquad \text{(emitter is a gluon)} \tag{19}$$
$$T^c_{c_1c_2} = t^c_{c_1c_2} = -t^c_{c_2c_1} \qquad \text{(emitter is an outgoing quark/anti-quark)} \tag{20}$$
$$T^c_{c_1c_2} = -t^c_{c_2c_1} = t^c_{c_1c_2} \qquad \text{(emitter is an incoming quark/anti-quark)} \tag{21}$$

The color operators satisfy important identities:

$$\sum_i \mathbf{T}_i|\mathcal{M}_n\rangle = 0, \qquad T_i^c T_j^c = \mathbf{T}_i \cdot \mathbf{T}_j = \mathbf{T}_j \cdot \mathbf{T}_i, \qquad \mathbf{T}_i \cdot \mathbf{T}_i = \mathbf{T}_i^2 = C_i = C_{a_i} \tag{22}$$

where the Casimir operators are:

$$C_g = C_A, \qquad C_q = C_{\bar{q}} = C_F \tag{23}$$

and the trace normalization is:

$$\mathrm{Tr}[t^a t^b] = T_F \delta^{ab} = \frac{1}{2}\delta^{ab} \tag{24}$$

Anomalous Dimensions

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

$$\gamma_0^q = -3C_F \qquad \text{(massless quarks/anti-quarks)} \tag{25}$$
$$\gamma_0^Q = -2C_F \qquad \text{(massive quarks/anti-quarks)} \tag{26}$$
$$\gamma_0^g = -\beta_0 = -\frac{11}{3}C_A + \frac{4}{3}T_F n_l \qquad \text{(gluons)} \tag{27}$$

The `ioperator` command computes the **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.

The finite virtual contribution is then:

$$2\,\mathrm{Re}(\mathcal{M}_0^* \mathcal{M}_1^{\mathrm{fin}}) = 2\,\mathrm{Re}(\mathcal{M}_0^* \mathcal{M}_1^{\mathrm{ren}}) - \mathbf{I} \cdot |\mathcal{M}_0|^2 \tag{28}$$

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

6.5 Renormalized Virtual Amplitude

The complete renormalized one-loop amplitude is:

$$\mathcal{M}_1^{\mathrm{ren}} = \mathcal{M}_1^{\mathrm{bare}} + \mathcal{M}_0 \cdot \delta Z_{\mathrm{ext}} + \mathcal{M}_{\mathrm{CT}} \tag{29}$$

where δZ_{ext} includes external leg corrections and $\mathcal{M}_{\mathrm{CT}}$ contains vertex counterterm insertions.

The combination is performed automatically when all counterterm contributions have been computed, resulting in a UV-finite (but IR-divergent) virtual amplitude.

6.6 Complete NLO Virtual Result

The full workflow for obtaining the finite virtual contribution is:

```
glas> evaluate nlo --jobs 4 --dirac
glas> contract nlo --jobs 4
glas> uvct
glas> evaluate mct --jobs 4
glas> contract mct --jobs 4
glas> ioperator
```

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

7 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.

7.1 Finite Field Reconstruction

The key insight is that rational function coefficients can be reconstructed from their numerical evaluations over finite fields $\mathbb{F}_p$ (integers modulo a prime p). For a rational function $f(x_1, \ldots, x_n)$, we:

1. Evaluate f at multiple points $(x_1^{(k)}, \ldots, x_n^{(k)}) \in \mathbb{F}_p^n$.
2. Use Thiele's interpolation for univariate reconstruction.
3. Apply multivariate Newton interpolation for the full result.
4. Verify using additional evaluation points.

This approach reduces memory requirements by orders of magnitude compared to symbolic manipulation.

7.2 Master Integral Coefficient Extraction

The `reduce` command applies the IBP reduction rules and extracts master integral coefficients:

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

The `-micoef` flag triggers the master integral coefficient projection, producing:

- `form/Files/MasterCoefs/c{i}.h`: Coefficient of master integral i.
- `Mathematica/Files/MasterCoefficients.m`: Combined coefficient file.

7.3 Finding Linear Relations

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

```
glas> linrels
```

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:
$$\sum_{i=1}^{n_{\text{mis}}} r_i\, c_i = 0 \tag{30}$$
5. **Reconstruction**: Reconstruct the rational coefficients r_i using multivariate interpolation.
6. **Verification**: Confirm the relation over $\mathbb{Q}$ using additional prime evaluations.

7.4 Implementation with FiniteFlow

The Mathematica script `LinearRelations.m` implements this procedure using FiniteFlow's functional reconstruction:

```
(* Load FiniteFlow *)
<< FiniteFlow`

(* Define the graph for coefficient evaluation *)
FFNewGraph[graph, in, {s12, s13, s23, mt2}];
FFAlgRatFunEval[graph, coeffs, {in}, coeffList];

(* Find linear relations *)
FFLinearRelations[graph, relations, {coeffs}];

(* Reconstruct rational coefficients *)
FFReconstructFunction[graph, relations, vars]
```

The output is written to `Files/MasterCoefficients.m` containing the simplified coefficient expressions.

7.5 Complexity Reduction

Linear relations arise from several sources:

- **Symmetry**: Bose symmetry of identical external particles.
- **Gauge invariance**: Ward identity constraints.
- **IBP relations**: Hidden dependencies from incomplete reduction.
- **Kinematic identities**: Schouten identity and momentum conservation.

For typical NLO calculations, the number of independent coefficients can be reduced by 20–50% through linear relation identification, significantly simplifying the final result.

7.6 Amplitude Reconstruction

Once the independent coefficients are identified, the full amplitude is reconstructed:

$$2\,\mathrm{Re}(\mathcal{M}_0^*\mathcal{M}_1) = \sum_{i\in\text{indep}} c_i(\epsilon, s_{jk}, m^2)\, M_i \tag{31}$$

where the sum runs only over the independent master integrals. The result is written in a form suitable for numerical evaluation or analytical expansion in ϵ.

8 Command Reference

8.1 Core Commands

8.2 Utility Commands

8.3 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.

Command	**Description**
`generate` *`process`*	Generate diagrams for the specified process
`evaluate {lo|nlo|mct}`	Apply Feynman rules and kinematics
`contract {lo|nlo|mct}`	Square amplitudes and sum helicities
`uvct`	Compute UV counterterms
`extract topologies`	Extract and complete loop topologies
`ibp`	Perform IBP reduction
`reduce`	Apply reduction rules to amplitudes
`linrels`	Find linear relations between coefficients
`ioperator`	Apply IR subtraction operators

Table 1: Core GLAS commands for amplitude calculation.

Command	**Description**
`runs [`*`tag`*`]`	List available runs
`use` *`run`*	Attach to an existing run
`show`	Display current configuration
`setrefs`	Set gluon polarization references
`verbose [on|off]`	Toggle verbose output streaming
`clean`	Delete all run directories

Table 2: Utility commands for run management.

8.4 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.

9 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:

$$\text{chunk}(k) = \left[\left\lfloor \frac{(k-1)N}{K} \right\rfloor + 1, \left\lfloor \frac{kN}{K} \right\rfloor \right] \tag{32}$$

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.

10 FORM Procedure Library

GLAS includes a library of FORM procedures in `resources/formlib/procedures/`:

Procedure	Description
`FeynmanRules.prc`	QCD Feynman rule substitutions
`DiracSimplify.prc`	Dirac algebra simplification
`PolarizationSums.prc`	Gluon polarization sum insertion
`color.prc`	SU(N) color algebra
`Conjugate.prc`	Complex conjugation
`mandelstam2x2.prc`	2→2 Mandelstam invariants
`mandelstam2x3.prc`	2→3 Mandelstam invariants
`Together.prc`	Rational function combination

Table 3: FORM procedures included with GLAS.

11 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 4
[generate] Created run: ggtT_0001
[generate] n0l=3, n1l=28

glas> evaluate lo --jobs 4
[evaluate lo] All jobs finished OK.

glas> evaluate nlo --jobs 4 --dirac
[evaluate nlo] All jobs finished OK.
[dirac nlo] All jobs finished OK.

glas> contract lo --jobs 4
[contract lo] All jobs finished OK.

glas> contract nlo --jobs 4
[contract nlo] All jobs finished OK.

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/ (5 topology files)
[ibp] Running SymmetryRelations.m...
[ibp] SymmetryRelations.m OK -> Files/SymmetryRelations.m
[ibp] Recorded nmis = 12 in meta.json
```

The complete calculation produces:

- 3 tree-level diagrams and 28 one-loop diagrams.

- 5 independent loop topologies after completion.
- 12 master integrals after IBP reduction.

12 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.