Reduze 2 - Distributed Feynman Integral Reduction

C. Studerus

Universität Bielefeld

in collaboration with A. v. Manteuffel (Mainz U.)

PSI: Particle Theory Seminar
March 1, 2012

Outline

1. Introduction

2. Job system (load balancing)

3. Topological analyis

4. Distributed reductions

Outline

1. Introduction
2. Job system (load balancing)
3. Topological analyis
4. Distributed reductions

Overwiew

Reduze 2

- computer program written in C++ to perform reductions of scalar Feynman integrals to master integrals
- arXiv:1201.4330, A.v.Manteuffel, CSt
- successor and major rewrite of `Reduze 1` by CSt
- dependencies
  - requires: `GiNaC` by Bauer, Frink, Kreckel
  - optional: `Open MPI`
  - optional: `Berkeley DB`
  - optional: `Fermat CAS` by Lewis (closed source, non-free)

Main features:

- topological analysis of **graphs** of integrals
- fully **parallelized** reductions
- resume aborted reductions
- computation of QCD diagram interferences up to masters
- generation of differential equations for masters
- ...
- `QGRAF` input and `FORM`, `Mathematica`, `Maple` output

Typical steps in calculations in pertubative Quantum Field Theories

- generate Feynman diagrams: e.g. QGRAF by Nogueira, FeynArts by Hahn
- apply Feynman rules
- build scalar interference terms: multiply diagrams or use projectors
- scalar Feynman integrals: loop/external momenta k_i/p_j

$$\int \mathrm{d}^d k_1 \ldots \int \mathrm{d}^d k_L \, \frac{(q_i q_j)^{\alpha_{ij}}}{D_1^{r_1} \ldots D_t^{r_t}}, \qquad D_i = q_{comb_i}^2 - m_i^2, \qquad q_n \in \{k_i, p_j\}$$

- use integration-by-parts (IBP) identities to reduce the integrals to master integrals: e.g. `AIR` by Anastasiou, `FIRE` by Smirnov, `Reduze`
- calculate the master integrals
- ...

need: standardized representation of integrals

- define **integral family** ("auxiliary topology"): set of propagators $\{1/D_1, \ldots, 1/D_N\}$ such that: all scalar products are linear combinations of D_i and kinematic invariants
- counting propagator exponents **indexes integrals**:

$$\text{Feynman integrals of some topologies} \quad \rightarrow \quad \mathbb{Z}^N$$

$$\int d^d k_1 \cdots d^d k_L \frac{1}{D_1^{n_1} \cdots D_N^{n_N}} \mapsto \{n_1, \ldots, n_N\} \quad \text{with } n_i \in \mathbb{Z}$$

- integrals belong to a **sector** of an integral family

```
I[FAM       T ID   R S  n_1     ...      n_9]
I[planarbox 5 182  6 1  -1 1 1 0 1 1 0 2 0]
```

- define an ordering for integrals (e.g. fewer denominators means simpler)

Overwiev of the main jobs in Reduze 2

input: a list of user-defined integral families

job: setup_sector_mappings

- construct graphs for the sectors (identify physical sectors)
- find zero sectors
- identify isomorphic graphs
- derive shifts to relate isomorphic sectors (sector relations)
- find shifts from sector to itself (sector symmetries)

job: reduce_sectors

- reduce integrals of a collection of sectors to master integrals

need other jobs:

- generate indexed integrals (seed integrals)
- generate IBP identities from the seed integrals
- use reduction results from sub-sectors $\rightarrow$ reduce them first

need a job system to handle the dependencies

Outline

1 Introduction

2 Job system (load balancing)

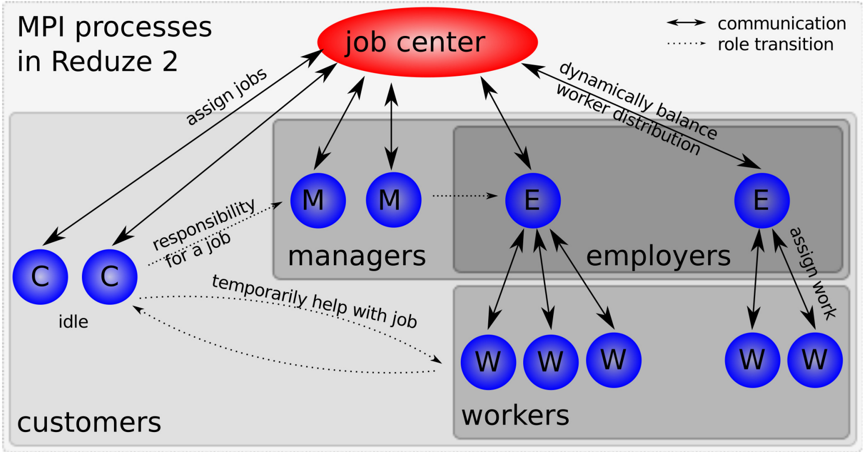
3 Topological analyis

4 Distributed reductions

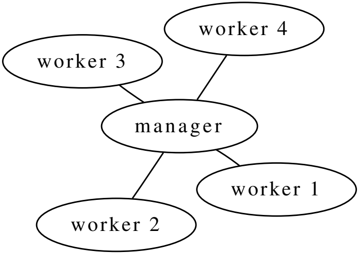
Available jobs in Reduze 2

```
$ reduze -h jobs

List of available job types:

apply_crossings:                  Generates reduction results for crossed sectors.
cat_files:                        Concatenates files.
collect_integrals:                Collects all integrals appearing in the input file.
compute_diagram_interferences:    Computes interferences of diagrams.
compute_differential_equations:   Computes derivatives of integrals wrt invariants.
export:                           Exports to FORM, Mathematica or Maple format.
extract_database_contents:        Extracts intermediate results from aborted reduction.
find_diagram_shifts:              Matches diagrams to sectors via graphs.
find_diagram_shifts_alt:          Matches diagrams to sectors via combinatorics.
generate_identities:              Generates identities like IBPs for given seeds.
generate_seeds:                   Generates integrals from a sector.
insert_reductions:                Inserts reductions in expressions.
normalize:                        Simplifies linear combinations and equations.
print_reduction_info_file:        Analyzes reductions in a file.
print_reduction_info_sectors:     Analyzes reductions available for sectors.
print_sector_info:                Prints diagrams and other information for sectors.
reduce_files:                     Reduces identities in given files.
reduce_sectors:                   Reduces integrals from a selection of sectors.
run_reduction:                    Low-level job to run a reduction.
select_reductions:                Selects reductions for integrals.
setup_sector_mappings:            Finds shifts between sectors via graphs.
setup_sector_mappings_alt:        Finds shifts between sectors via combinatorics.
sum_terms:                        Sums terms.
test:                             Performs some tests.
verify_same_terms:                Verifies two files contain the same terms.
```

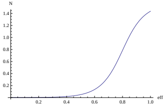
MPI processes in Reduze 2



Dynamical load balancing



- high efficiency: manager idle, workers busy
- dynamically reassing workers to managers
  - minimal worker requirement
  - worker distribution



Outline

1. Introduction

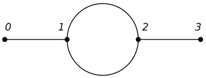
2. Job system (load balancing)

3. Topological analyis

4. Distributed reductions

Graph Isomorphism

- **Graph** $G = (V, E)$, vertices V and edges E (pairs of vertices)
- Representation of graphs: eg. **adjacency matrix**



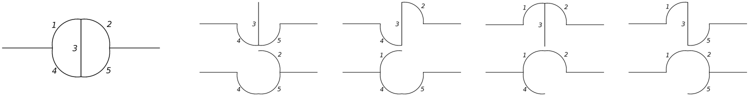
$$\begin{matrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 2 & 0 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{matrix}$$

- Two graphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$ are **isomorphic**, $G_1 \sim G_2$, if there is a permutation $\sigma : V \to V$ such that $\sigma(E_1) = E_2$
  eg. if (v_1, v_2) has k edges then also $(\sigma(v_1), \sigma(v_2))$
- **Canonical labeling** (unique representation):
  Permute the n vertices such that the adjacency matrix is minimal w.r.t. lexicographic ordering: $n!$ cases to check (if allowed, if minimal)
- **Refinement procedure**: Brendan D. McKay, Practical Graph Isomorphism, 1981
  define a degree for vertices (#adjacent edges) and partition vertices into a sequence of subsets with equal degree **(equitable partition)**

$$(\{0,1,2,3\}) \to (\{0,3\},\ \{1,2\}) \quad \Rightarrow \quad (\{0\},\{3\},\{1\},\{2\}) \to \begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 \\ 1 & 0 & 2 & 0 \end{matrix}$$

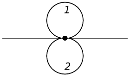
Graphs of a sector

- **Spanning tree** of a graph are all connected tree graphs which contain all vertices of the original graph (Obtained by deleting #loop edges).



- **U-polynomial**
  Sum over product of edges not in the spanning tree:
  $U = x_1x_2 + x_1x_5 + x_4x_5 + x_4x_2 + x_3x_1 + x_3x_2 + x_3x_5 + x_3x_4$
  Can also be calculated by given (inverse) propagators $P_i = q^2 - m_i^2$:
  $U = \det(M), \quad k_i M_{ij} k_j = x_l P_l, \quad$ loop momenta k_i

- Select propagators of one term, eg. x_1, x_2, attach them to a node, split vertex and try to insert the rest of the propagators (momentum conservation)



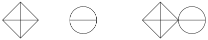
- Given a Feynman integral (sector) ⇒ corresponding graph is not unique in general.

Matroids

- Given a Feynman integral (sector) ⇒ corresponding graph is not unique in general.

- Both graphs have the same (cyclic) **matroid**
  - matroids introduced as a generalization of the concept of "linear independence"
  - graphs with the same cyclic matroid have the same U-polynomial

- **Theorem**: Hassler Whitney, 2-isomorphic graphs, AJM 55:245-254, 1933

  Two cyclic matroids are isomorphic if their graphs can be transformed into each other by a sequence of the operations:
  - vertex cleaving and identification: 
  - twisting: 

Outline

1. Introduction
2. Job system (load balancing)
3. Topological analyis
4. Distributed reductions

k_i/p_j loop/(independent)external momenta
$q_n \in \{k_i, p_j\}$
$\mathbf{I}'(p_1, \ldots, p_N, k_1, \ldots, k_L)$ is integrand of a Feynman integral

$$\int \mathrm{d}^d k_i \, \frac{\partial}{\partial k_i^\mu} \left[q^\mu \, \mathbf{I}'(p_1, \ldots, p_N, k_1, \ldots, k_L) \right] = 0$$

sum over μ (no sum over i)
$L(N+L)$ equations per seed integral

Laporta algorithm:

- define **ordering** for integrals
- generate IBPs: **sparse system** of equations
- **solve linear system** of equations

- generate the system of equations
- sort equations in blocks with the **same leading integral**
- send blocks to workers

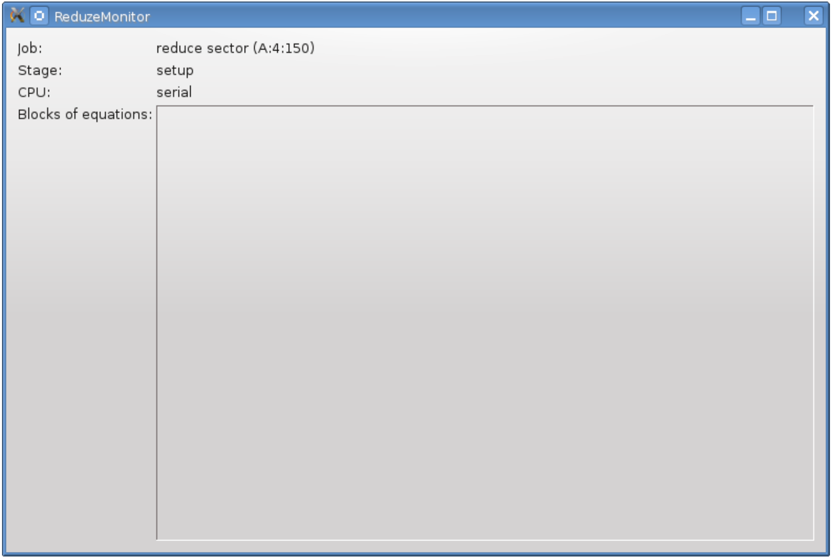
$$\begin{aligned}
\mathbf{I_5} + c_{14}\mathbf{I_4} + c_{13}\mathbf{I_3} &= 0 \\
\mathbf{I_5} + c_{24}\mathbf{I_4} + c_{22}\mathbf{I_2} &= 0 \\
\mathbf{I_5} + c_{33}\mathbf{I_3} + c_{32}\mathbf{I_2} &= 0
\end{aligned}$$

$$\begin{aligned}
\mathbf{I_3} + c_{42}\mathbf{I_2} &= 0 \\
\mathbf{I_3} + c_{51}\mathbf{I_1} &= 0
\end{aligned}$$

$$\mathbf{I_2} + c_{61}\mathbf{I_1} = 0$$

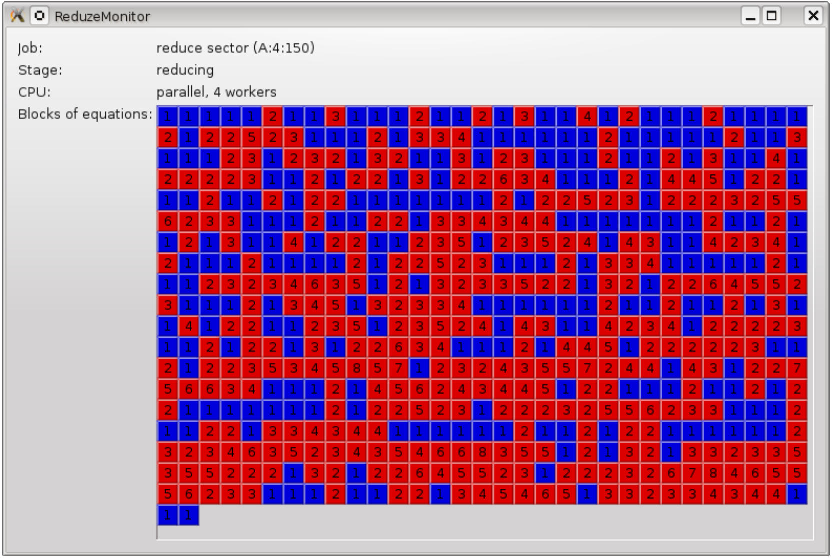
Example: distributed reduction of one sector

visualisation of reduction (subtopology of 2-loop massive double box):



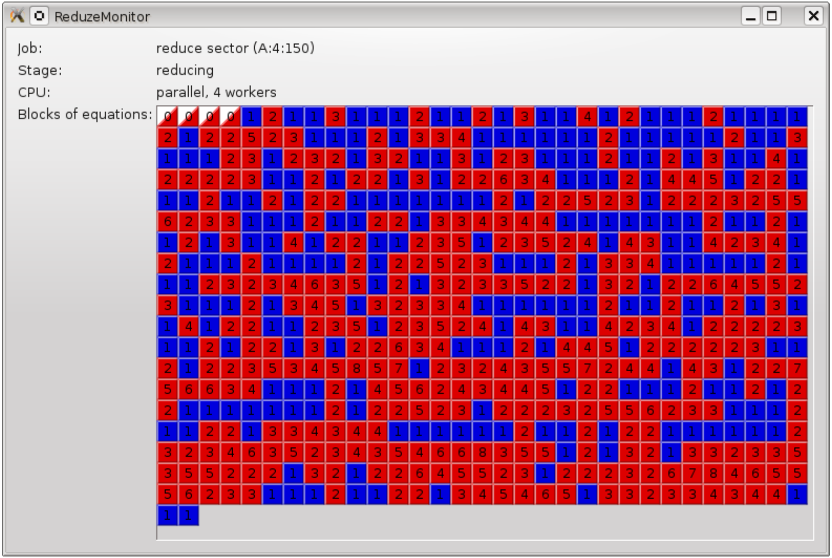
Example: distributed reduction of one sector

visualisation of reduction (subtopology of 2-loop massive double box):



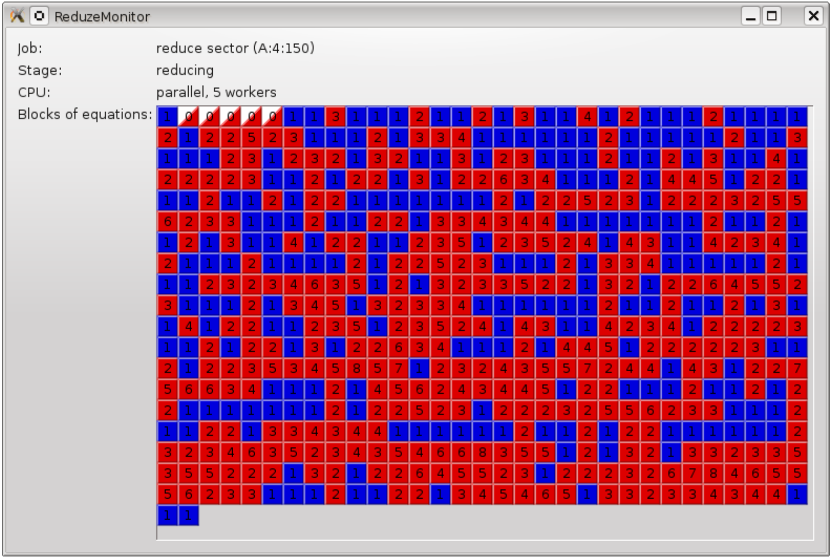
Example: distributed reduction of one sector

visualisation of reduction (subtopology of 2-loop massive double box):



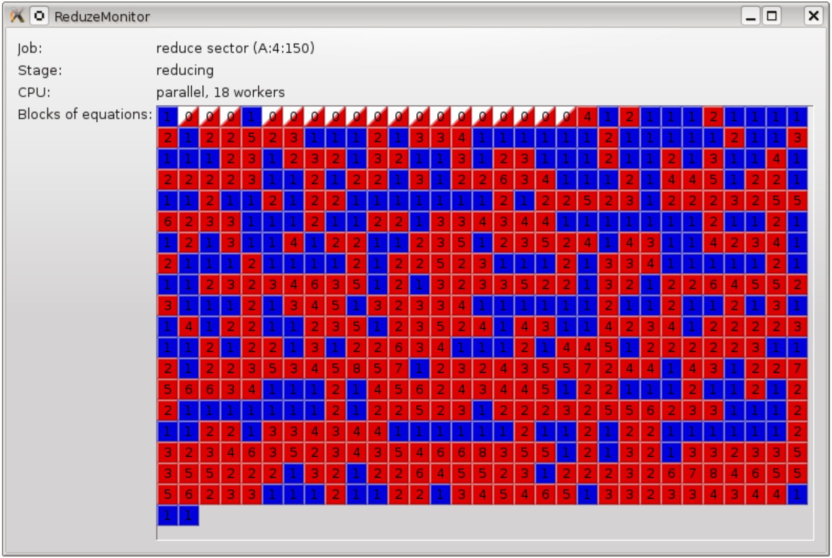
Example: distributed reduction of one sector

visualisation of reduction (subtopology of 2-loop massive double box):



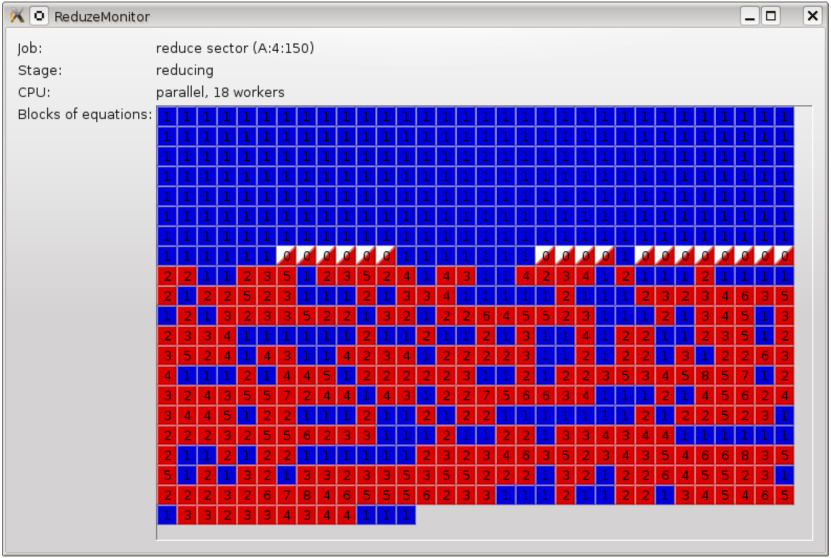
Example: distributed reduction of one sector

visualisation of reduction (subtopology of 2-loop massive double box):



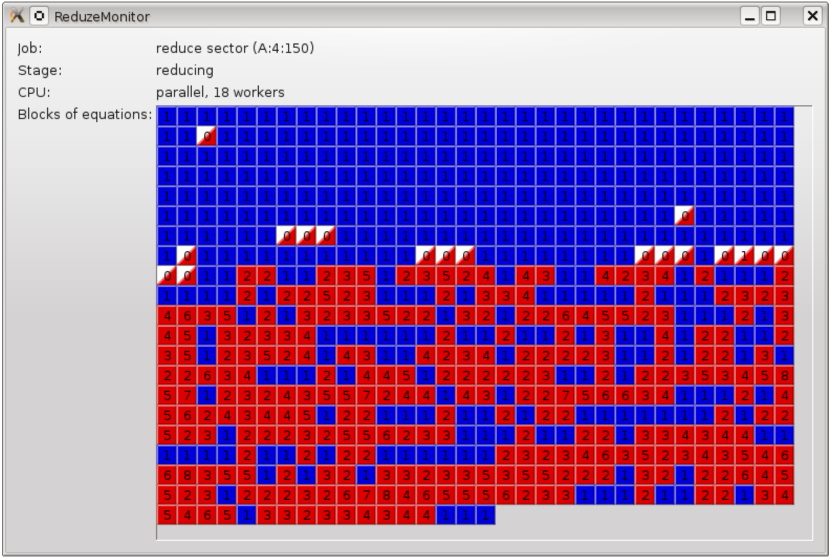
Example: distributed reduction of one sector

visualisation of reduction (subtopology of 2-loop massive double box):



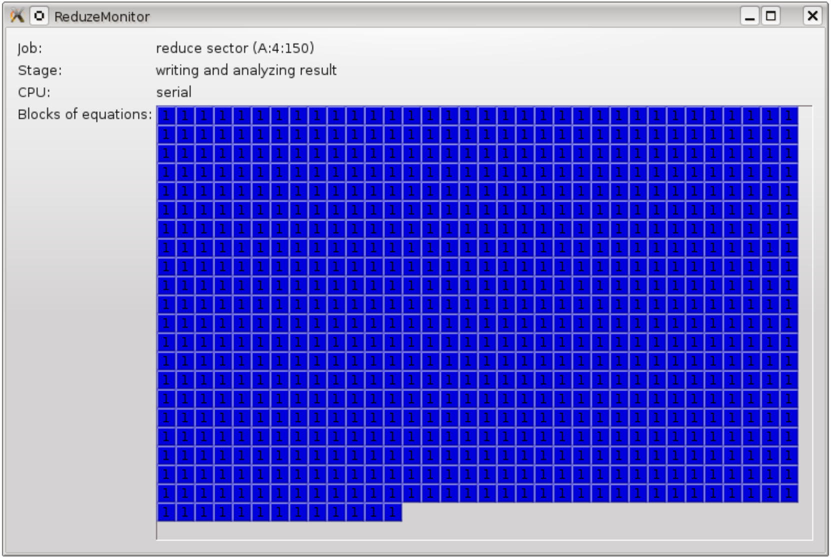
Example: distributed reduction of one sector

visualisation of reduction (subtopology of 2-loop massive double box):



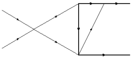
EXAMPLE: DISTRIBUTED REDUCTION OF ONE SECTOR

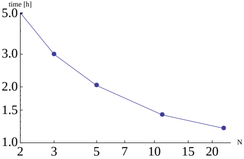
visualisation of reduction (subtopology of 2-loop massive double box):

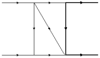


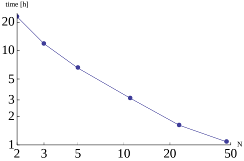
Performance: single sectors

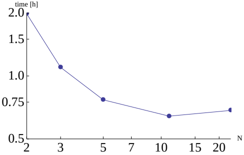
GiNaC | **Fermat**

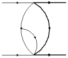


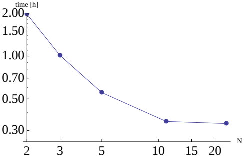




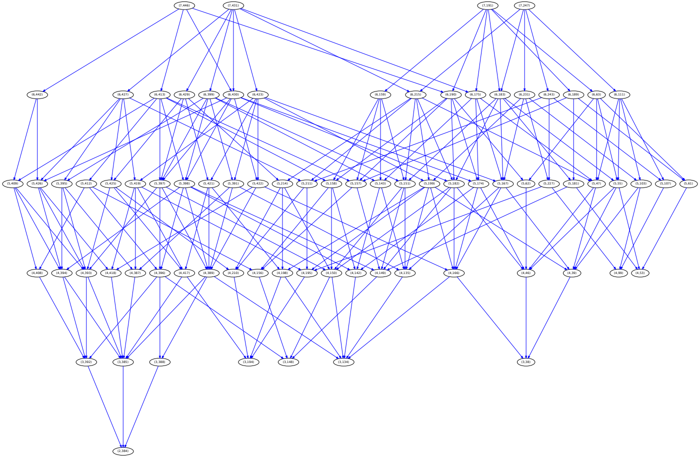


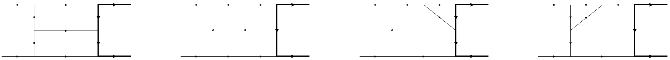




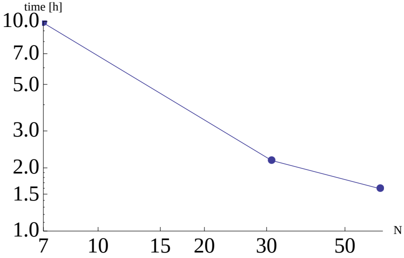


Performance: sector tree

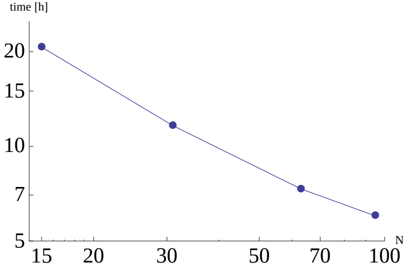




$s \in \{0, 1, 2, 3\}$



$s \in \{0, 1, 2, 3, 4\}$



Usage

Required input files in directory "config"

```
# kinematics.yaml

kinematics:
  incoming_momenta: [ p1, p2 ]
  outgoing_momenta: [ p3, p4 ]
  momentum_conservation: [p4, p1 + p2 - p3]
  kinematic_invariants:
    - [mt, 1]
    - [s,  2]
    - [t,  2]
  scalarproduct_rules:
    - [[p1,p1],  0]
    - [[p2,p2],  0]
    - [[p3,p3],  mt^2]
    - [[p1+p2, p1+p2], s]
    - [[p1-p3, p1-p3], t]
    - [[p2-p3, p2-p3], -s-t+2*mt^2] # == u
  symbol_to_replace_by_one: mt
```

```
# integralfamilies.yaml

integralfamilies:
  - name: planarbox
    loop_momenta: [k1, k2]
    propagators:
      - [ k1, 0 ]
      - [ k2, 0 ]
      - [ k1-k2, 0 ]
      - [ k1-p1, 0 ]
      - [ k2-p1, 0 ]
      - [ k1-p1-p2, 0 ]
      - [ k2-p1-p2, 0 ]
      - [ k1-p3, mt ]
      - [ k2-p3, mt ]
    permutation_symmetries:
      - [ [ 1, 6 ], [ 2, 7 ] ]
      - [ [ 1, 2 ], [ 4, 5 ], [ 6, 7 ], [ 8, 9] ]
```

Optional input files in directory "config"

```
# global.yaml

paths:
  fermat: /path/to/fermat/executable
```

```
# feynmanrules.yaml

feynmanrules: {} # see example 2 of the source package
```

Usage

Job file:

```
# myjobs.yaml

jobs:
  - setup_sector_mappings: {}
  - reduce_sectors:
      sector_selection:
        select_recursively:
          - [planarbox, 182]
      identities:
        ibp:
          - { r: [t, 5], s: [0, 1] }
        sector_symmetries:
          - { r: [t, t], s: [0, 1] }
      reduzer_options:
        use_transactions: true
  - select_reductions:
      input_file: "myintegrals"
      output_file: "myintegrals.sol"
  - export:
      input_file: "myintegrals.sol"
      output_file: "myintegrals.sol.inc"
      output_format: "form"
```

and start the program with:

```
$ reduze myjobs.yaml
```

or in parallel mode:

```
$ mpirun -np 32 reduze myjobs.yaml
```

Output: default directories: `graphs`, `sectormappings`, `reductions`, `log`, `tmp`

Thank you!