

REDUZE 2 - DISTRIBUTED FEYNMAN INTEGRAL REDUCTION

C. Studerus

The logo of the University of Bielefeld, consisting of a green square with a white horizontal bar across its middle. The text 'Universität Bielefeld' is written in white on the bar.

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 ANALYSIS

4 DISTRIBUTED REDUCTIONS

OUTLINE

1 INTRODUCTION

2 JOB SYSTEM (LOAD BALANCING)

3 TOPOLOGICAL ANALYSIS

4 DISTRIBUTED REDUCTIONS

OVERVIEW

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

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

- 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 $\{1/D_1, \dots, 1/D_N\}$ such that: all scalar products are linear combinations of D_i and kinematic invariants
- counting propagator exponents **indexes integrals**:

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

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

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

$$\begin{aligned} & I[\text{FAM} \quad \quad \quad \text{T ID} \quad \text{R S} \quad \text{n}_1 \quad \dots \quad \text{n}_9] \\ & I[\text{planarbox} \quad 5 \quad 182 \quad 6 \quad 1 \quad -1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 2 \quad 0] \end{aligned}$$

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

OVERVIEW 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 → reduce them first

need a job system to handle the dependencies

OUTLINE

1 INTRODUCTION

2 **JOB SYSTEM (LOAD BALANCING)**

3 TOPOLOGICAL ANALYSIS

4 DISTRIBUTED REDUCTIONS

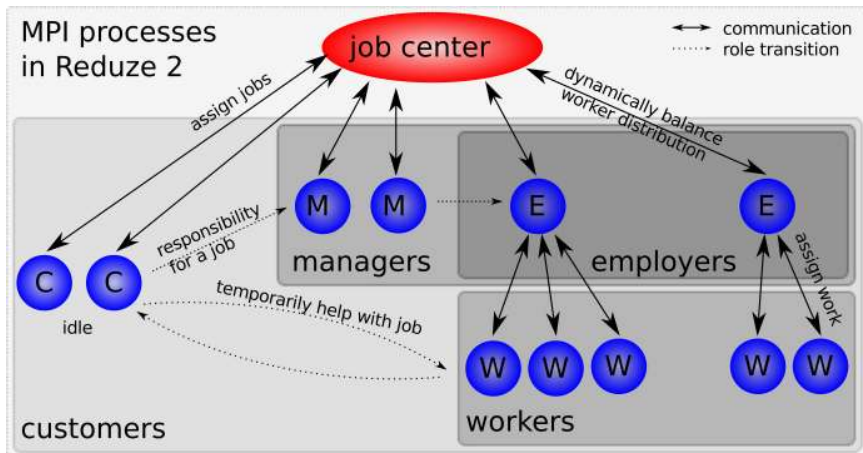
AVAILABLE JOBS IN Reduze 2

```
$ reduce -h jobs
```

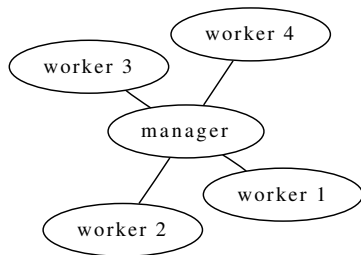
List of available job types:

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

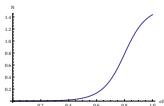
MPI PROCESSES IN Reduze 2



DYNAMICAL LOAD BALANCING



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



OUTLINE

1 INTRODUCTION

2 JOB SYSTEM (LOAD BALANCING)

3 TOPOLOGICAL ANALYSIS

4 DISTRIBUTED REDUCTIONS

GRAPH ISOMORPHISM

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



0	1	0	0
1	0	2	0
0	2	0	1
0	0	1	0

- Two graphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$ are **isomorphic**, $G_1 \sim G_2$, if there is a permutation $\sigma : V \rightarrow V$ 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**)

$$(\{0,1,2,3\}) \rightarrow (\{0,3\}, \{1,2\}) \Rightarrow (\{0\}, \{3\}, \{1\}, \{2\}) \rightarrow$$

0	0	0	1
0	0	1	0
0	1	0	2
1	0	2	0

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:

$$U = x_1 x_2 + x_1 x_5 + x_4 x_5 + x_4 x_2 + x_3 x_1 + x_3 x_2 + x_3 x_5 + x_3 x_4$$

Can also be calculated by given (inverse) propagators $P_i = q^2 - m_i^2$:

$$U = \det(M), \quad k_i M_{ij} k_j = x_j P_i, \quad \text{loop momenta } k_i$$

- 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) \Rightarrow corresponding graph is not unique in general.



- Given a Feynman integral (sector) \Rightarrow 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 ANALYSIS

4 DISTRIBUTED REDUCTIONS

INTEGRATION BY PARTS (IBP) IDENTITIES

k_i/p_j loop/(independent)external momenta

$q_n \in \{k_i, p_j\}$

$\mathbf{I}'(p_1, \dots, p_N, k_1, \dots, k_L)$ is integrand of a Feynman integral

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

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

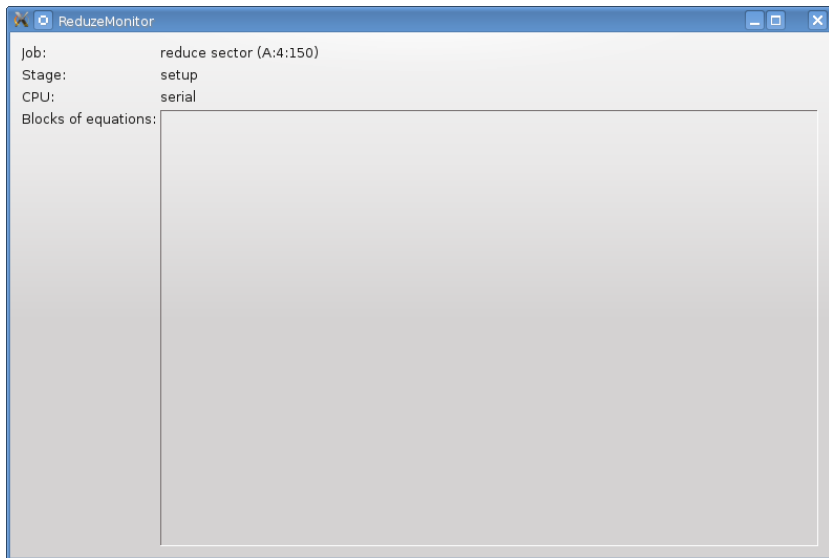
PARALLELIZATION OF LAPORTA-ALGORITHM

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

$l_5 + c_{14}l_4 + c_{13}l_3$	$= 0$
$l_5 + c_{24}l_4$	$+ c_{22}l_2 = 0$
l_5	$+ c_{33}l_3 + c_{32}l_2 = 0$
$l_3 + c_{42}l_2$	$= 0$
l_3	$+ c_{51}l_1 = 0$
$l_2 + c_{61}l_1$	$= 0$

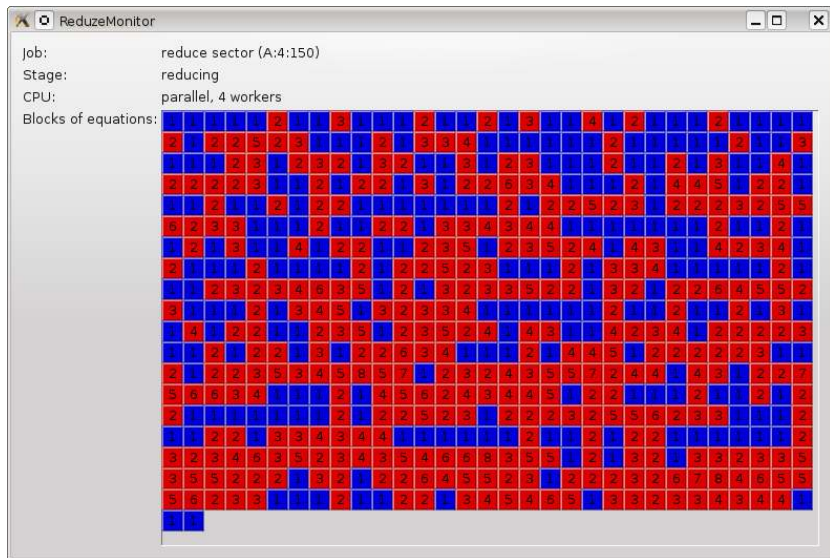
EXAMPLE: DISTRIBUTED REDUCTION OF ONE SECTOR

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



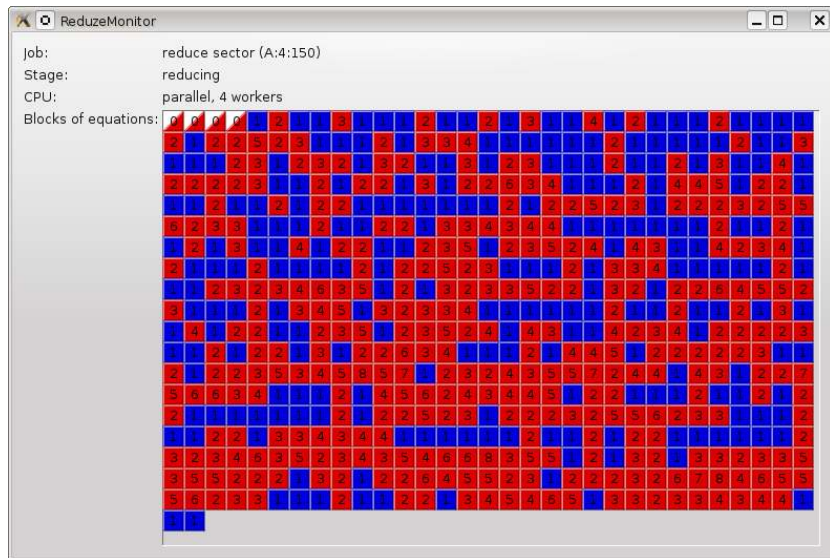
EXAMPLE: DISTRIBUTED REDUCTION OF ONE SECTOR

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



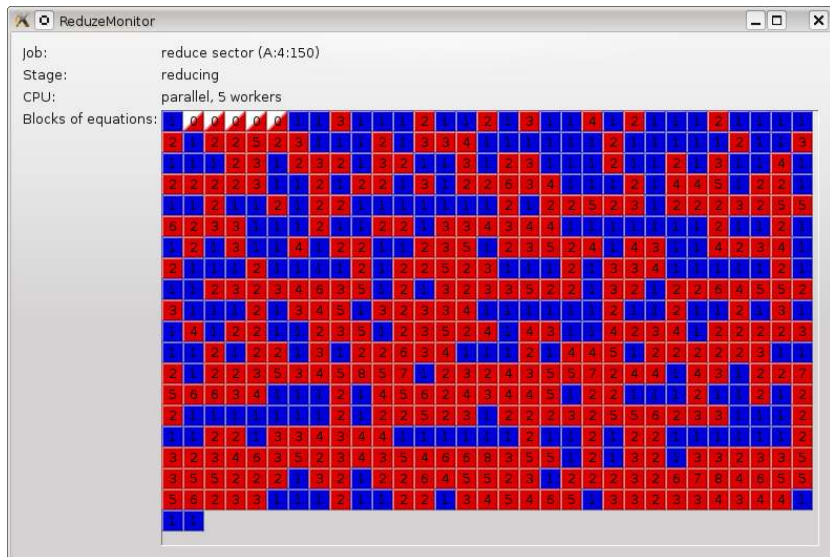
EXAMPLE: DISTRIBUTED REDUCTION OF ONE SECTOR

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):



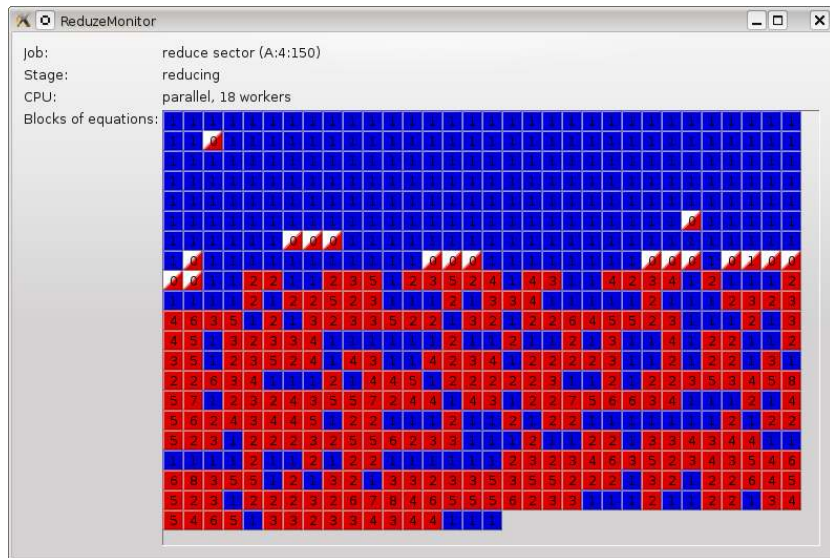
EXAMPLE: DISTRIBUTED REDUCTION OF ONE SECTOR

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



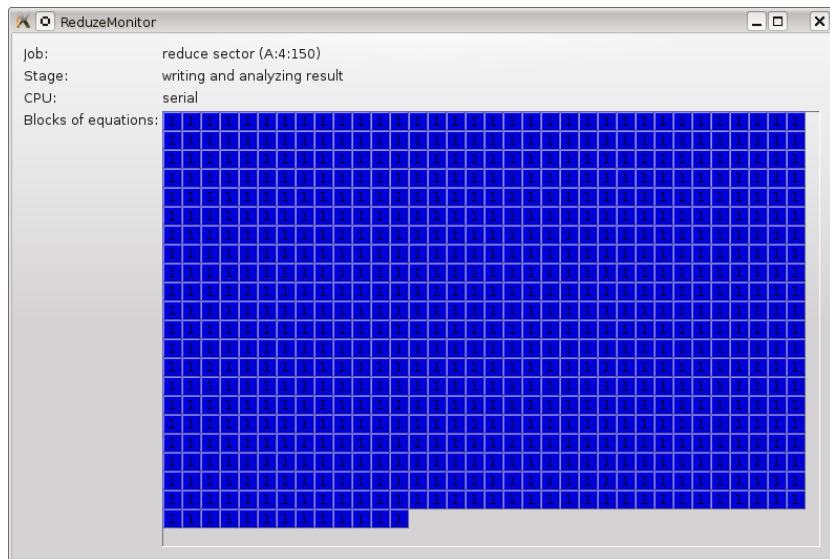
EXAMPLE: DISTRIBUTED REDUCTION OF ONE SECTOR

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

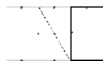
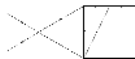


EXAMPLE: DISTRIBUTED REDUCTION OF ONE SECTOR

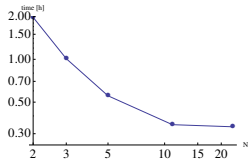
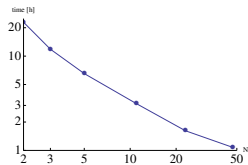
visualisation of reduction (subtopology of 2-loop massive double box):



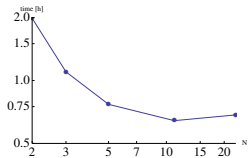
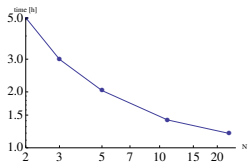
PERFORMANCE: SINGLE SECTORS



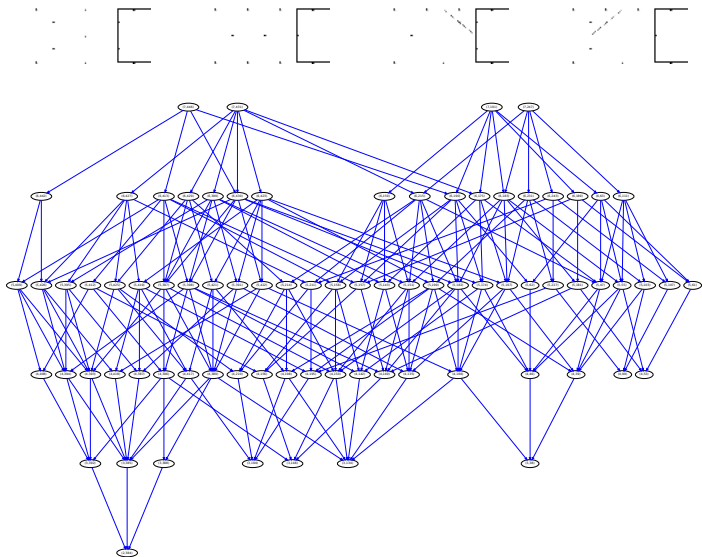
GiNaC



Fermat



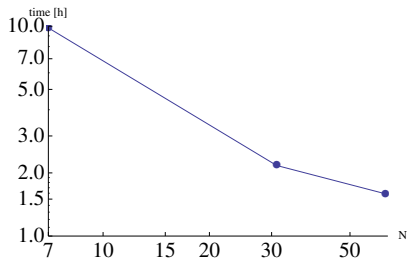
PERFORMANCE: SECTOR TREE



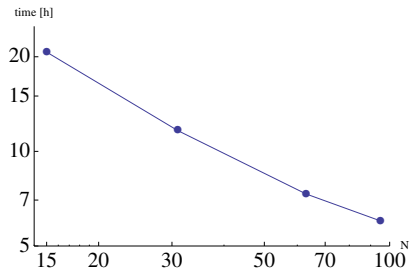
PERFORMANCE: SECTOR TREE



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



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



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
```

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] }
  reducer_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!