FIRE5: a C++ implementation of Feynman Integral REduction

A.V. Smirnov

Scientific Research Center, MSU

July 24, 2015


Fire

The goal of this talk is to present the new version FIRE.

- FIRE - Feynman Integral REduction

FIRE can be downloaded from `http://git.sander.su/fire`

Feynman integrals

- Feynman integrals over loop momenta:

$$\mathcal{F}(a_1, \ldots, a_n) = \int \cdots \int \frac{\mathrm{d}^d k_1 \ldots \mathrm{d}^d k_h}{E_1^{a_1} \ldots E_n^{a_n}} .$$

- Currently one needs to evaluate millions of Feynman integrals with diffeent indices a_i coresponding to a particular diagram, so evaluating each of them analytically turns into an unreal task.

Evaluation of Feynman integrals

Evaluation of Feynman integrals can be divided into two parts:

Evaluation of Feynman integrals

Evaluation of Feynman integrals can be divided into two parts:

- reduction — representing all required integrals as linear combinations of so-called *master integrals*;

Evaluation of Feynman integrals

Evaluation of Feynman integrals can be divided into two parts:

- reduction — representing all required integrals as linear combinations of so-called *master integrals*;
- evaluation of master integrals.

Evaluation of Feynman integrals

Evaluation of Feynman integrals can be divided into two parts:

- reduction — representing all required integrals as linear combinations of so-called *master integrals* $\to$ FIRE;
- evaluation of master integrals

Relation types

Most commonly used relations: IBP relations (Chetyrkin, Tkachev).

$$\int \ldots \int \mathrm{d}^d k_1 \ldots \mathrm{d}^d k_n \frac{\partial}{\partial k_i} \left(p_j \frac{1}{E_1^{a_1} \ldots E_n^{a_n}} \right) = 0 \; ,$$

Relation types

Most commonly used relations: IBP relations (Chetyrkin, Tkachev).

$$\int \ldots \int \mathrm{d}^d k_1 \ldots \mathrm{d}^d k_n \frac{\partial}{\partial k_i} \left(p_j \frac{1}{E_1^{a_1} \ldots E_n^{a_n}} \right) = 0 \, ,$$

$\rightarrow$

$$\sum \alpha_i F(a_1 + b_{i,1}, \ldots, a_n + b_{i,n}) = 0 \, ,$$

Relation types

Symmetry relations, e.g.,

$$F(a_1,\ldots,a_n) = (-1)^{d_1 a_1 + \ldots d_n a_n} F(a_{\sigma(1)},\ldots,a_{\sigma(n)}),$$

Relation types

Symmetry relations, e.g.,

$$F(a_1, \ldots, a_n) = (-1)^{d_1 a_1 + \ldots d_n a_n} F(a_{\sigma(1)}, \ldots, a_{\sigma(n)}),$$

Boundary conditions:

$$F(a_1, a_2, \ldots, a_n) = 0 \text{ when } a_{i_1} \leq 0, \ldots a_{i_k} \leq 0$$

for some subsets of indices i_j;

Multiple programs for Feynman integral reduction

- AIR
- FIRE
- Reduze
- LiteRed

Multiple programs for Feynman integral reduction

- AIR
- FIRE
- Reduze
- LiteRed
- different private implementations

Multiple programs for Feynman integral reduction

- AIR
- FIRE
- Reduze
- LiteRed
- different private implementations
- more public algorithms going to appear?

Multiple programs for Feynman integral reduction

- AIR
- FIRE
- Reduze
- LiteRed
- different private implementations
- more public algorithms going to appear? WATER, EARTH

Multiple programs for Feynman integral reduction

- AIR
- FIRE
- Reduze
- LiteRed
- different private implementations
- more public algorithms going to appear? WATER, EARTH and 5th element

Versions of FIRE

- FIRE1, FIRE2 — mythical versions that existed only in private, based mostly on Gröbner bases

Versions of FIRE

- FIRE1, FIRE2 — mythical versions that existed only in private, based mostly on Gröbner bases
- FIRE3 — first public version in Wolfram Mathematica

Versions of FIRE

- FIRE1, FIRE2 — mythical versions that existed only in private, based mostly on Gröbner bases
- FIRE3 — first public version in Wolfram Mathematica
- FIRE4 — upgraded Mathematica version, master integral identification (tsort by Alexey Pak), usage of reduction rules (LiteRed)

Versions of FIRE

- FIRE1, FIRE2 — mythical versions that existed only in private, based mostly on Gröbner bases
- FIRE3 — first public version in Wolfram Mathematica
- FIRE4 — upgraded Mathematica version, master integral identification (tsort by Alexey Pak), usage of reduction rules (LiteRed)
- FIRE5 — Mathematica **and c++**

n	1	2	3	4
Mathematica	640	1453	2061	3958
C++, 1 thread, disk mode	134	168	255	552
C++, 4 threads, disk mode	76	100	163	323
C++, 1 thread, RAM mode	107	136	210	473
C++, 4 threads, RAM mode	49	74	108	237

Table : Timings for the on-shell massless double box example, n is the total power of irreducible numerators

Installation of FIRE:

- Either download a binary package from `http://git.sander.su/fire/downloads` and hope that it works for your system

Installation of FIRE:

- Either download a binary package from `http://git.sander.su/fire/downloads` and hope that it works for your system
- Or download the source with git and build it yourself (this also gives you access to dev versions of FIRE)

Building from sources requires KyotoCabinet and Snappy libraries, but they are shipped with FIRE.

Installation from sources:

- git clone https://bitbucket.org/feynmanIntegrals/fire.git && cd fire/FIRE5

Installation from sources:

- git clone https://bitbucket.org/feynmanIntegrals/fire.git && cd fire/FIRE5
- make dep (to build the dependencies)

Installation from sources:

- git clone https://bitbucket.org/feynmanIntegrals/fire.git && cd fire/FIRE5
- make dep (to build the dependencies)
- make

Installation from sources:

- git clone https://bitbucket.org/feynmanIntegrals/fire.git && cd fire/FIRE5
- make dep (to build the dependencies)
- make
- To download the latest version: git pull

Installation from sources:

- git clone https://bitbucket.org/feynmanIntegrals/fire.git && cd fire/FIRE5
- make dep (to build the dependencies)
- make
- To download the latest version: git pull
- Switch to dev version: git checkout dev

Structure of FIRE

- The Mathematica part FIRE5.m — used as the main frontend to provide input and to analyze output

Structure of FIRE

- The Mathematica part FIRE5.m — used as the main frontend to provide input and to analyze output
- The c++ binary bin/FIRE5 — efficiently performs reduction without Mathematica

Structure of FIRE

- The Mathematica part FIRE5.m — used as the main frontend to provide input and to analyze output
- The c++ binary bin/FIRE5 — efficiently performs reduction without Mathematica
- extra/ferl64 (extra/ferm64) — the fermat program by Robert Lewis performing algebraic simplifications

Structure of FIRE

- The Mathematica part FIRE5.m — used as the main frontend to provide input and to analyze output
- The c++ binary bin/FIRE5 — efficiently performs reduction without Mathematica
- extra/ferl64 (extra/ferm64) — the fermat program by Robert Lewis performing algebraic simplifications
- LiteRed package by Roman Lee — can be used to provide additional reduction rules or symmetries

Structure of FIRE

- The Mathematica part FIRE5.m — used as the main frontend to provide input and to analyze output
- The c++ binary bin/FIRE5 — efficiently performs reduction without Mathematica
- extra/ferl64 (extra/ferm64) — the fermat program by Robert Lewis performing algebraic simplifications
- LiteRed package by Roman Lee — can be used to provide additional reduction rules or symmetries
- bin/KLink — auxilirary binary used to access Kyotocabinet databases from Mathematica

Structure of FIRE

- The Mathematica part FIRE5.m — used as the main frontend to provide input and to analyze output
- The c++ binary bin/FIRE5 — efficiently performs reduction without Mathematica
- extra/ferl64 (extra/ferm64) — the fermat program by Robert Lewis performing algebraic simplifications
- LiteRed package by Roman Lee — can be used to provide additional reduction rules or symmetries
- bin/KLink — auxilirary binary used to access Kyotocabinet databases from Mathematica
- bin/FLink — auxilirary binary used to run fermat from Mathematica

0. Loading FIRE in Mathematica

```
SetDirectory[<path to the folder with FIRE>];
Get["FIRE5.m"];
```

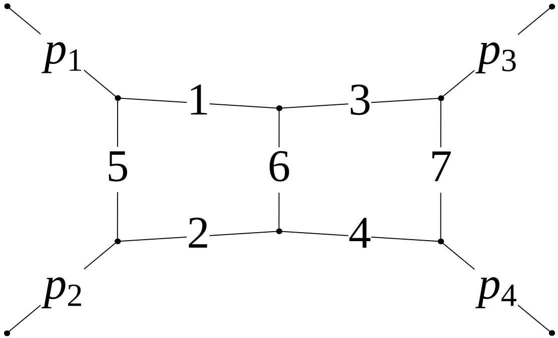
or

0. Loading FIRE in Mathematica

```
SetDirectory[<path to the folder with FIRE>];
Get["FIRE5.m"];
```

or

```
FIREPath = <path to the folder with FIRE>;
Get[FIREPath<>"FIRE5.m"];
```


p1
p2
p3
p4
1
2
3
4
5
6
7

1. Creating a start file in Mathematica

- `Internal = {k1, k2};`

1. Creating a start file in Mathematica

- `Internal = {k1, k2};`
- `External = {p1, p2, p3};`

1. Creating a start file in Mathematica

- `Internal = {k1, k2};`
- `External = {p1, p2, p3};`
- `Propagators = {-k1^2, -(k1 + p1 + p2)^2, -k2^2, -(k2 + p1 + p2)^2, -(k1 + p1)^2, -(k1 - k2)^2, -(k2 - p3)^2, -(k2 + p1)^2, -(k1 - p3)^2};`

1. Creating a start file in Mathematica

- Internal = {k1, k2};
- External = {p1, p2, p3};
- Propagators = {-k1^2, -(k1 + p1 + p2)^2, -k2^2, -(k2 + p1 + p2)^2, -(k1 + p1)^2, -(k1 - k2)^2, -(k2 - p3)^2, -(k2 + p1)^2, -(k1 - p3)^2};
- Replacements = {p1^2 -> 0, p2^2 -> 0, p3^2 -> 0,
  p1 p2 -> s/2,
  p1 p3 -> t/2, p2 p3 -> -1/2 (s + t)};

1. Creating a start file in Mathematica

- `Internal = {k1, k2};`
- `External = {p1, p2, p3};`
- `Propagators` = {-k1^2, -(k1 + p1 + p2)^2, -k2^2, -(k2 + p1 + p2)^2, -(k1 + p1)^2, -(k1 - k2)^2, -(k2 - p3)^2, -(k2 + p1)^2, -(k1 - p3)^2};
- `Replacements` = {p1^2 -> 0, p2^2 -> 0, p3^2 -> 0, p1 p2 -> s/2, p1 p3 -> t/2, p2 p3 -> -1/2 (s + t)};
- `PrepareIBP[]; Prepare[AutoDetectRestrictions -> True]; SaveStart["doublebox"];`

1. Reduction in Mathematica

- `LoadStart["doublebox",1]; Burn[];`

1. Reduction in Mathematica

- `LoadStart["doublebox",1]; Burn[];`
  You can load multiple start files a time!
- `F[1, {1, 1, 1, 1, 1, 1, 1, -1, -1}]`

1. Reduction in Mathematica

- `LoadStart["doublebox",1]; Burn[];`
  You can load multiple start files a time!
- `F[1, {1, 1, 1, 1, 1, 1, 1, -1, -1}]`
  or a better way for multiple integrals:
- `EvaluateAndSave[{{1, {1, 1, 1, 1, 1, 1, 1, -1, -1}}, {1, {1, 1, 1, 1, 1, 1, 1, 0, -2}}},"doublebox.tables"]`

1. Reduction in Mathematica

- `LoadStart["doublebox",1]; Burn[];`
  You can load multiple start files a time!
- `F[1, {1, 1, 1, 1, 1, 1, 1, -1, -1}]`
  or a better way for multiple integrals:
- `EvaluateAndSave[{{1, {1, 1, 1, 1, 1, 1, 1, -1, -1}}, {1, {1, 1, 1, 1, 1, 1, 1, 0, -2}}},"doublebox.tables"]`
  Later load a new kernel, load the start file and...
- `LoadTables["doublebox.tables"];`

- FIRE automatilly starts the reduction from the top-level sectors, then lower and lower (already with the knowledge of what integrals are required at this level). Then it performs the backward-substitutions;

- FIRE automatilly starts the reduction from the top-level sectors, then lower and lower (already with the knowledge of what integrals are required at this level). Then it performs the backward-substitutions;
- The same is true for the c++ FIRE, however it can also work in parallel with sectors of the same level (the number of positive indices).

2. Finding equivalents between master integrals

During the main reduction phase FIRE cannot find equivalents between master integrals in different sectors.

`MasterIntegrals[]`

2. Finding equivalents between master integrals

During the main reduction phase FIRE cannot find equivalents between master integrals in different sectors.

```
MasterIntegrals[]
{{1, {0, 0, 0, 0, 1, 1, 1, 0, 0}}, {1, {0, 0, 1,
1, 1, 1, 0, 0, 0}},{1, {0, 0, 1, 1, 1, 1, 1, 0,
0}}, {1, {0, 1, 1, 0, 0, 1, 0, 0, 0}}, {1, {0, 1,
1, 0, 1, 1, 1, 0, 0}}, {1, {1, 0, 0, 1, 0, 1, 0,
0, 0}}, {1, {1, 0, 0, 1, 1, 1, 1, 0, 0}}, {1, {1,
1, 0, 0, 0, 1, 1, 0, 0}}, {1, {1, 1, 0, 0, 1, 1,
1, 0, 0}}, {1, {1, 1, 1, 1, 0, 0, 0, 0, 0}}, {1,
{1, 1, 1, 1, 1, 1, 1, 0, 0}}, {1, {1, 1, 1, 1, 1,
1, 1, -1, 0}}}
```

2. Finding equivalents between master integrals

- Internal = {k1, k2};
- External = {p1, p2, p3};
- Propagators = {$-k1^2$, $-(k1 + p1 + p2)^2$, $-k2^2$, $-(k2 + p1 + p2)^2$, $-(k1 + p1)^2$, $-(k1 - k2)^2$, $-(k2 - p3)^2$, $-(k2 + p1)^2$, $-(k1 - p3)^2$};
- Replacements = {$p1^2$ -> 0, $p2^2$ -> 0, $p3^2$ -> 0,
  p1 p2 -> s/2,
  p1 p3 -> t/2, p2 p3 -> -1/2 (s + t)};

2. Finding equivalents between master integrals

- `Internal = {k1, k2};`
- `External = {p1, p2, p3};`
- `Propagators = {-k1^2, -(k1 + p1 + p2)^2, -k2^2, -(k2 + p1 + p2)^2, -(k1 + p1)^2, -(k1 - k2)^2, -(k2 - p3)^2, -(k2 + p1)^2, -(k1 - p3)^2};`
- `Replacements = {p1^2 -> 0, p2^2 -> 0, p3^2 -> 0, p1 p2 -> s/2, p1 p3 -> t/2, p2 p3 -> -1/2 (s + t)};`
- `FindRules[MasterIntegrals[]]`

2. Finding equivalents between master integrals

- `Internal = {k1, k2};`
- `External = {p1, p2, p3};`
- `Propagators = {-k1^2, -(k1 + p1 + p2)^2, -k2^2, -(k2 + p1 + p2)^2, -(k1 + p1)^2, -(k1 - k2)^2, -(k2 - p3)^2, -(k2 + p1)^2, -(k1 - p3)^2};`
- `Replacements = {p1^2 -> 0, p2^2 -> 0, p3^2 -> 0, p1 p2 -> s/2, p1 p3 -> t/2, p2 p3 -> -1/2 (s + t)};`
- `FindRules[MasterIntegrals[]]`
  Or save rules to a file with
- `WriteRules[MasterIntegrals[], FIREPath <> "examples/doublebox"];`

2. Finding equivalents between master integrals

- The file contains 4 lines:

```
G[1, {0, 0, 1, 1, 1, 1, 1, 0, 0}] ->
{{1, G[1, {1, 1, 0, 0, 1, 1, 1, 0, 0}]}};
G[1, {1, 0, 0, 1, 1, 1, 1, 0, 0}] ->
{{1, G[1, {0, 1, 1, 0, 1, 1, 1, 0, 0}]}};
G[1, {1, 1, 0, 0, 0, 1, 1, 0, 0}] ->
{{1, G[1, {0, 0, 1, 1, 1, 1, 0, 0, 0}]}};
G[1, {1, 0, 0, 1, 0, 1, 0, 0, 0}] ->
{{1, G[1, {0, 1, 1, 0, 0, 1, 0, 0, 0}]}};
```

2. Finding equivalents between master integrals

- The file contains 4 lines:

```
G[1, {0, 0, 1, 1, 1, 1, 1, 0, 0}] ->
{{1, G[1, {1, 1, 0, 0, 1, 1, 1, 0, 0}]}};
G[1, {1, 0, 0, 1, 1, 1, 1, 0, 0}] ->
{{1, G[1, {0, 1, 1, 0, 1, 1, 1, 0, 0}]}};
G[1, {1, 1, 0, 0, 0, 1, 1, 0, 0}] ->
{{1, G[1, {0, 0, 1, 1, 1, 1, 0, 0, 0}]}};
G[1, {1, 0, 0, 1, 0, 1, 0, 0, 0}] ->
{{1, G[1, {0, 1, 1, 0, 0, 1, 0, 0, 0}]}};
```

- It can be loaded with

```
LoadRules[FIREPath <> "examples/doublebox", 1];
```

3. The c++ reduction

To run the reduction one has to create a configuration file and run the c++ `FIRE` with `bin/FIRE5 -c ConfigFileName`

3. The c++ reduction

To run the reduction one has to create a configuration file and run the c++ FIRE with `bin/FIRE5 -c ConfigFileName`

```
#threads 4
#variables d, s, t
#start
#folder examples/
#problem 1 doublebox.start
#integrals doublebox.m
#output doublebox.tables
```

4. Usage of LiteRed

One can produce LiteRed rules and use them with FIRE. This is demonstrated on one of the examples coming with LiteRed — a vertex 2-loop diagram.

4. Usage of LiteRed

One can produce LiteRed rules and use them with FIRE. This is demonstrated on one of the examples coming with LiteRed — a vertex 2-loop diagram.

- `FIREPath = <path to the folder with FIRE>;`
- `SetDirectory[FIREPath <> "extra/LiteRed/Setup/"];`
- `Get["LiteRed.m"];`
- `Get[FIREPath <> "FIRE5.m"];`
- `Internal = {l, r};`
- `External = {p, q};`
- `Propagators = (-Power[##, 2]) & /@ {l - r, l, r, p - l, q - r, p - l + r, q - r + l};`
- Replacements = {p^2 -> 0, q^2 -> 0; p q -> -1/2};

4. Usage of LiteRed

- `CreateNewBasis[v2, Directory -> FIREPath <> "temp/v2.dir"];`
- `GenerateIBP[v2];`
- `AnalyzeSectors[v2, {0, __}];`
  (basic operations including zero sector detection)

4. Usage of LiteRed

- `CreateNewBasis[v2, Directory -> FIREPath <> "temp/v2.dir"];`
- `GenerateIBP[v2];`
- `AnalyzeSectors[v2, {0, __}];`
  (basic operations including zero sector detection)
- `FindSymmetries[v2,EMs->True];`
  (symmetries between sectors)

4. Usage of LiteRed

- `CreateNewBasis[v2, Directory -> FIREPath <> "temp/v2.dir"];`
- `GenerateIBP[v2];`
- `AnalyzeSectors[v2, {0, __}];`
  (basic operations including zero sector detection)
- `FindSymmetries[v2,EMs->True];`
  (symmetries between sectors)
- `SolvejSector /@ UniqueSectors[v2];`
  (full solution of IBPs)

4. Usage of LiteRed

- CreateNewBasis[v2, Directory -> FIREPath <> "temp/v2.dir"];
- GenerateIBP[v2];
- AnalyzeSectors[v2, {0, __}];
  (basic operations including zero sector detection)
- FindSymmetries[v2,EMs->True];
  (symmetries between sectors)
- SolvejSector /@ UniqueSectors[v2];
  (full solution of IBPs)
- DiskSave[v2];

The solution stage is not guaranteed to work, but at least symmetries normally help.

4. Usage of LiteRed

LiteRed files can be converted so that they can be used by the `c++` `FIRE`.

- FIREPath = <path to the folder with FIRE>;
- Get[FIREPath <> "FIRE5.m"];
- LoadStart[FIREPath <> "examples/v2"];
- TransformRules[FIREPath <> "temp/v2.dir", FIREPath <> "examples/v2.lbases", 2];
- SaveSBases[FIREPath <> "examples/v2"];

FIRE workflow

FIRE workflow

- `Internal, External, Propagators` → `problem.start` *(initial input)*

FIRE workflow

- `Internal, External, Propagators` → `problem.start` *(initial input)*
- `problem.config, problem.start` or `problem.sbases, problem.m` (list of integrals), `problem.rules` (if exists), `problem.lbases` (if exists) → `problem.tables` *(reduction)*

FIRE workflow

- `Internal, External, Propagators` → `problem.start` *(initial input)*
- `problem.config, problem.start` or `problem.sbases, problem.m` (list of integrals), `problem.rules` (if exists), `problem.lbases` (if exists) → `problem.tables` *(reduction)*
- `Internal, External, Propagators, problem.tables` → `problem.rules` *(detection of equivalent masters)*

FIRE workflow

- `Internal`, `External`, `Propagators` → `problem.start` *(initial input)*
- `problem.config`, `problem.start` or `problem.sbases`, `problem.m` (list of integrals), `problem.rules` (if exists), `problem.lbases` (if exists) → `problem.tables` *(reduction)*
- `Internal`, `External`, `Propagators`, `problem.tables` → `problem.rules` *(detection of equivalent masters)*
- `Internal`, `External`, `Propagators` → folder with `LiteRed` rules and symmetries *(loading LiteRed)*

FIRE workflow

- `Internal`, `External`, `Propagators` → `problem.start` *(initial input)*
- `problem.config`, `problem.start` or `problem.sbases`, `problem.m` (list of integrals), `problem.rules` (if exists), `problem.lbases` (if exists) → `problem.tables` *(reduction)*
- `Internal`, `External`, `Propagators`, `problem.tables` → `problem.rules` *(detection of equivalent masters)*
- `Internal`, `External`, `Propagators` → folder with `LiteRed` rules and symmetries *(loading LiteRed)*
- `problem.start`, folder with `LiteRed` rules and symmetries → `problem.sbases` and `problem.lbases` *(transforming LiteRed rules)*

Optimization hints. Organiztion

- Reduction should be performed by the `c++ FIRE`; provide the complete list of integrals to be reduced;

Optimization hints. Organiztion

- Reduction should be performed by the `c++ FIRE`; provide the complete list of integrals to be reduced;
- Reduction can work in parallel, do not forget the #`threads` setting;

Optimization hints. Organiztion

- Reduction should be performed by the `c++ FIRE`; provide the complete list of integrals to be reduced;
- Reduction can work in parallel, do not forget the #`threads` setting;
- Zero-sectors not covered by restrictions slow the reduction. If they are not detected properly automatically, provide them manually or use LiteRed;

Optimization hints. Organiztion

- Reduction should be performed by the `c++ FIRE`; provide the complete list of integrals to be reduced;
- Reduction can work in parallel, do not forget the #`threads` setting;
- Zero-sectors not covered by restrictions slow the reduction. If they are not detected properly automatically, provide them manually or use LiteRed;
- If the diagram has global symmetries, specify them;

Optimization hints. Organiztion

- Reduction should be performed by the `c++ FIRE`; provide the complete list of integrals to be reduced;
- Reduction can work in parallel, do not forget the #`threads` setting;
- Zero-sectors not covered by restrictions slow the reduction. If they are not detected properly automatically, provide them manually or use LiteRed;
- If the diagram has global symmetries, specify them;
- Non-global symmetry rules produced by LiteRed can improve performance;

Optimization hints. Organiztion

- Reduction should be performed by the `c++ FIRE`; provide the complete list of integrals to be reduced;
- Reduction can work in parallel, do not forget the #`threads` setting;
- Zero-sectors not covered by restrictions slow the reduction. If they are not detected properly automatically, provide them manually or use LiteRed;
- If the diagram has global symmetries, specify them;
- Non-global symmetry rules produced by LiteRed can improve performance;
- Equivalent master integrals slow the reduction a lot. One can first make a test run only detecting masters, then produce rules for equivalents, then do the final run.

Optimization hints. Hardware

- FIRE has two different modes — disk mode and RAM mode (depending on the way coefficients and other things are stored while FIRE works)

Optimization hints. Hardware

1. Disk mode. All databases for all sectors are stored on disk. It is essential to use a fast local hard disk in this case. It is also important to have caching set up in your operating system like an intermidiate buffer before disk access. While FIRE works with a number of sectors in parallel, those databases are open (plus one global database).

Optimization hints. Hardware

1. Disk mode. All databases for all sectors are stored on disk. It is essential to use a fast local hard disk in this case. It is also important to have caching set up in your operating system like an intermidiate buffer before disk access. While FIRE works with a number of sectors in parallel, those databases are open (plus one global database).
2. RAM mode. The open databases are in-memory databases. However after work is over in a sector, it is dumped to disk. This mode does not need caching and requires more RAM than in the disk mode, but it does not rely that much on the disk speed.

Optimization hints. Hardware

- Choose an appropriate mode for you. In both cases one should have enough RAM to prevent swapping and enough disk space to avoid crashes with the “can’t write” message.

Optimization hints. Hardware

- Choose an appropriate mode for you. In both cases one should have enough RAM to prevent swapping and enough disk space to avoid crashes with the “can't write” message.
- The more threads are in use, the more RAM FIRE needs. Sometimes one has to use less threads (the `#threads` setting) than the number of processor cores globally or only at the substitution stage (the `#sthreads` setting). In this case the number of fermat processes can be increased with the `#fhreads` setting.

- There is a nice program FIRE that you can use for integral reduction
- :)


FIRE