Feynman integral evaluation at supercomputers

A.V. Smirnov

Research computing center, MSU

September 18, 2018

There's Plenty of Room at the Bottom

Talk by Richard Feynman on December 29, 1959

I don't know how to do this on a small scale in a practical way, but I do know that computing machines are very large; they fill rooms. Why can't we make them very small, make them of little wires, little elements – and by little, I mean little. For instance, the wires should be 10 or 100 atoms in diameter, and the circuits should be a few thousand angstroms across. Everybody who has analyzed the logical theory of computers has come to the conclusion that the possibilities of computers are very interesting – if they could be made to be more complicated by several orders of magnitude...

There's Plenty of Room at the Bottom

Talk by Richard Feynman on December 29, 1959

...If they had millions of times as many elements, they could make judgments. They would have time to calculate what is the best way to make the calculation that they are about to make. They could select the method of analysis which, from their experience, is better than the one that we would give to them. And in many other ways, they would have new qualitative features.
...But there is plenty of room to make them smaller. There is nothing that I can see in the physical laws that says the computer elements cannot be made enormously smaller than they are now.

Moore's law

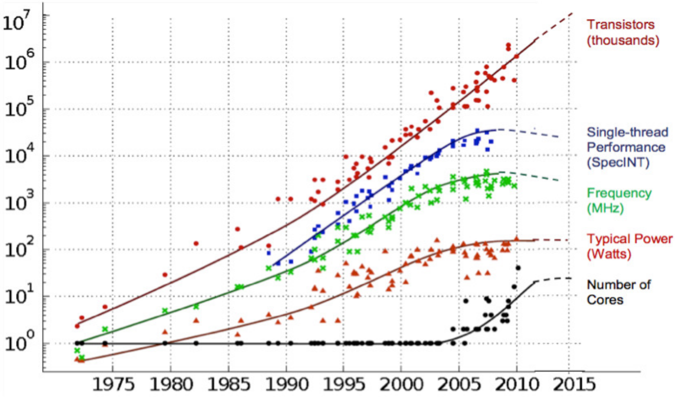
- **Moore's law** is the observation that the number of transistors in a dense integrated circuit doubles about every two years (1965)

Moore's law

- **Moore's law** is the observation that the number of transistors in a dense integrated circuit doubles about every two years (1965)
- We also used to apply it to CPU speed.

Moore's law

- **Moore's law** is the observation that the number of transistors in a dense integrated circuit doubles about every two years (1965)
- We also used to apply it to CPU speed.
- More or less valid till 2005. And till 2010 counting CPU cores. What now?


10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
1975 1980 1985 1990 1995 2000 2005 2010 2015
Transistors (thousands)
Single-thread Performance (SpecINT)
Frequency (MHz)
Typical Power (Watts)
Number of Cores

Feynman integral evaluation - complexity keeps growing

Feynman integral evaluation - complexity keeps growing

- Example: QCD massless form factors

Feynman integral evaluation - complexity keeps growing

- Example: QCD massless form factors
- Two-loop results G. Kramer and B. Lampe'87 T. Matsuura and W. L. van Neerven'88 T. Matsuura, S. C. van der Marck, and W. L. van Neerven'89

Feynman integral evaluation - complexity keeps growing

- Example: QCD massless form factors
- Two-loop results G. Kramer and B. Lampe'87 T. Matsuura and W. L. van Neerven'88 T. Matsuura, S. C. van der Marck, and W. L. van Neerven'89
- Three-loop results P. A. Baikov, K. G. Chetyrkin, A. V. Smirnov, V. A. Smirnov and M. Steinhauser'09, T. Gehrmann, E. W. N. Glover, T. Huber, N. Ikizlerli, and C. Studerus'10, R. N. Lee and V. A. Smirnov'10

Feynman integral evaluation - complexity keeps growing

- Example: QCD massless form factors
- Two-loop results G. Kramer and B. Lampe'87 T. Matsuura and W. L. van Neerven'88 T. Matsuura, S. C. van der Marck, and W. L. van Neerven'89
- Three-loop results P. A. Baikov, K. G. Chetyrkin, A. V. Smirnov, V. A. Smirnov and M. Steinhauser'09, T. Gehrmann, E. W. N. Glover, T. Huber, N. Ikizlerli, and C. Studerus'10, R. N. Lee and V. A. Smirnov'10
- Four-loop results J. M. Henn, A. V. Smirnov, V. A. Smirnov and M. Steinhauser'16 J. Henn, A. V. Smirnov, V. A. Smirnov, M. Steinhauser and R. N. Lee'17 R. N. Lee, A. V. Smirnov, V. A. Smirnov and M. Steinhauser'17 A. von Manteuffel and R. M. Schabinger'17

Parallelization

The key to moving further is **parallelization**.

- Multicore systems. Shared memory, easier to parallelize

Parallelization

The key to moving further is **parallelization**.

- Multicore systems. Shared memory, easier to parallelize
- GPU, FPGU and so on. Special approach sutable for some problems.

Parallelization

The key to moving further is **parallelization**.

- Multicore systems. Shared memory, easier to parallelize
- GPU, FPGU and so on. Special approach sutable for some problems.
- Supercomputers

Specifics of supercomputers

- Smaller nodes (compared to top nodes on our clusters)

Specifics of supercomputers

- Smaller nodes (compared to top nodes on our clusters)
- MANY nodes!!!!

Specifics of supercomputers

- Smaller nodes (compared to top nodes on our clusters)
- MANY nodes!!!!
- Time limits for jobs and high chance of failure due to hardware

Specifics of supercomputers

- Smaller nodes (compared to top nodes on our clusters)
- MANY nodes!!!!
- Time limits for jobs and high chance of failure due to hardware
- No magic button to make your code work at a supercomputer. There is no shared memory!

Specifics of supercomputers

- Smaller nodes (compared to top nodes on our clusters)
- MANY nodes!!!!
- Time limits for jobs and high chance of failure due to hardware
- No magic button to make your code work at a supercomputer. There is no shared memory!
- One needs a special code structure and special resource for parallelization.

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.

α-representation

Feynman parametric representation:

$$\mathcal{F}(a_1 \ldots, a_L; d) = \frac{\mathrm{i}^{a+h(1-d/2)} \pi^{hd/2}}{\prod_l \Gamma(a_l)}$$
$$\times \int_0^\infty \ldots \int_0^\infty \prod_l \alpha_l^{a_l-1} U^{-d/2} \mathrm{e}^{\mathrm{i}F/U - \mathrm{i}\sum m_l^2 \alpha_l} \mathrm{d}\alpha_1 \ldots \mathrm{d}\alpha_L \ .$$

where U и F are polynomials α that can be algorithmically determined by the initial diagram.

Sector decomposition

$$\int_0^1 \int_0^1 \frac{1}{(x+y)^{2-\varepsilon}} dydx$$

Sector decomposition

$$\int_0^1\int_0^1 \frac{1}{(x+y)^{2-\varepsilon}}dydx = 2\int_0^1\int_0^x \frac{1}{(x+y)^{2-\varepsilon}}dydx$$

Sector decomposition

$$\int_0^1 \int_0^1 \frac{1}{(x+y)^{2-\varepsilon}} dydx = 2 \int_0^1 \int_0^x \frac{1}{(x+y)^{2-\varepsilon}} dydx =$$

$$2 \int_0^1 \int_0^1 \frac{x}{(x+xz)^{2-\varepsilon}} dzdx$$

Sector decomposition

$$\int_0^1 \int_0^1 \frac{1}{(x+y)^{2-\varepsilon}} dydx = 2 \int_0^1 \int_0^x \frac{1}{(x+y)^{2-\varepsilon}} dydx =$$

$$2 \int_0^1 \int_0^1 \frac{x}{(x+xz)^{2-\varepsilon}} dzdx = 2 \int_0^1 \int_0^1 x^{-1+\varepsilon} \frac{1}{(1+z)^{2-\varepsilon}} dzdx$$

FIESTA

- `SDEvaluate[{U,F,l},indices,order]`

FIESTA

- `SDEvaluate[{U,F,l},indices,order]`
- `SDEvaluate[UF[loop_momenta,propagators, subst], indices,order]`

FIESTA

- `SDEvaluate[{U,F,l},indices,order]`
- `SDEvaluate[UF[loop_momenta,propagators, subst], indices,order]`
- *Example*:
  $\mathtt{SDEvaluate[UF[\{k\},\{-k^2,-(k+p_1)^2,-(k+p_1+p_2)^2,}$
  $\mathtt{-(k+p_1+p_2+p_4)^2\}, \{p_1^2 \to 0, p_2^2 \to 0, p_4^2 \to 0,}$
  $\mathtt{p_1\ p_2 \to -S/2, p_2\ p_4 \to -T/2, p_1\ p_4 \to (S+T)/2,}$
  $\mathtt{S \to 3, T \to 1\}], \{1,1,1,1\},0]}$

FIESTA

- SDEvaluate[{U,F,l},indices,order]
- SDEvaluate[UF[loop_momenta,propagators, subst], indices,order]
- *Example*: SDEvaluate[UF[{k},{-k^2,-(k+p_1)^2,-(k+p_1+p_2)^2, -(k+p_1+p_2+p_4)^2}, {p_1^2 $\rightarrow$0,p_2^2 $\rightarrow$0,p_4^2 $\rightarrow$0, p_1 p_2 $\rightarrow$-S/2,p_2 p_4 $\rightarrow$-T/2,p_1 p_4 $\rightarrow$(S+T)/2, S$\rightarrow$3,T$\rightarrow$ 1}], {1,1,1,1},0]
- *Answer*: -4.38658 + 1.3333/ep^2 - 0.732466/ep + 0.001 pm9

How can supercomputers help in evaluation?

- We need a huge parallel resource

How can supercomputers help in evaluation?

- We need a huge parallel resource
- 1) The number of master integrals to be evaluated

How can supercomputers help in evaluation?

- We need a huge parallel resource
- 1) The number of master integrals to be evaluated
- 2) The number of sectors in the sector-decomposition approach.

Classical usage of FIESTA

- Integrands are prepared in Mathematica and saved in a database;

Classical usage of FIESTA

- Integrands are prepared in Mathematica and saved in a database;
- Integration performed by a c++ program (called from Mathematica);

Classical usage of FIESTA

- Integrands are prepared in Mathematica and saved in a database;
- Integration performed by a c++ program (called from Mathematica);
- Mathematica gathers results from the database.

Classical usage of FIESTA

- Integrands are prepared in Mathematica and saved in a database;
- Integration performed by a c++ program (called from Mathematica);
- Mathematica gathers results from the database.

Parallization resource

- Number of masters (100+)

Parallization resource

- Number of masters (100+)
- Times the number of sectors (1000+ or more)

Parallization resource

- Number of masters (100+)
- Times the number of sectors (1000+ or more)
- -> more than a million integrations!

Parallization resource

- Number of masters (100+)
- Times the number of sectors (1000+ or more)
- -> more than a million integrations!
- And each integration leads to millions of function evaluations!

Parallization resource

- Number of masters (100+)
- Times the number of sectors (1000+ or more)
- -> more than a million integrations!
- And each integration leads to millions of function evaluations!

Usage of FIESTA at supercomputers

Usage of FIESTA at supercomputers

- PrepareDatabase=True; Store the integrands in a database, upload it to a supercomputer

Usage of FIESTA at supercomputers

- PrepareDatabase=True; Store the integrands in a database, upload it to a supercomputer
- Run the integrarion separately (this part does not require Mathematica!) with the use of MPI

Usage of FIESTA at supercomputers

- PrepareDatabase=True; Store the integrands in a database, upload it to a supercomputer
- Run the integrarion separately (this part does not require Mathematica!) with the use of MPI
- Worker tasks work with independent integrations

Usage of FIESTA at supercomputers

- PrepareDatabase=True; Store the integrands in a database, upload it to a supercomputer
- Run the integrarion separately (this part does not require Mathematica!) with the use of MPI
- Worker tasks work with independent integrations
- Also use the GPU acceleration if GPU nodes are available at the cluster.

Usage of FIESTA at supercomputers

- PrepareDatabase=True; Store the integrands in a database, upload it to a supercomputer
- Run the integrarion separately (this part does not require Mathematica!) with the use of MPI
- Worker tasks work with independent integrations
- Also use the GPU acceleration if GPU nodes are available at the cluster.
- Analyze the results with Mathematica

Some results obtained on supercomputers evaluating master integrals with FIESTA.

- Corrections to the muon anomalous magnetic moment at four-loop order A. Kurz, T. Liu, P. Marquard, A. V. Smirnov, V. A. Smirnov and M. Steinhauser'15
- Quark Mass Relations to Four-Loop Order in Perturbative QCD P. Marquard, A. V. Smirnov, V. A. Smirnov and M. Steinhauser'15 P. Marquard, A. V. Smirnov, V. A. Smirnov, M. Steinhauser and D. Wellmann'17

Multiple programs for Feynman integral reduction

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

Parallel approach to reduction

Reduction is solving a huge sparse matrix with polynomial coefficients
Current diagrams need (A LOT OF RAM) and (A LOT OF TIME)!

- Parallel reduction in sectors of same level

Parallel approach to reduction

Reduction is solving a huge sparse matrix with polynomial coefficients
Current diagrams need (A LOT OF RAM) and (A LOT OF TIME)!

- Parallel reduction in sectors of same level
- Multiple fermat workers (GCD application)

Parallel approach to reduction

Reduction is solving a huge sparse matrix with polynomial coefficients
Current diagrams need (A LOT OF RAM) and (A LOT OF TIME)!

- Parallel reduction in sectors of same level
- Multiple fermat workers (GCD application)
- Prime field approach (Manteuffel, Panzer, Schabinger)

Parallel approach to reduction

Reduction is solving a huge sparse matrix with polynomial coefficients
Current diagrams need (A LOT OF RAM) and (A LOT OF TIME)!

- Parallel reduction in sectors of same level
- Multiple fermat workers (GCD application)
- Prime field approach (Manteuffel, Panzer, Schabinger)
- Separate evaluation of coefficients at different masters (Chawdhry, Lim, Mitov)

Prime field approach

- Substitute different vaues of d and kinematic invariants, now we result in large rational numbers

Prime field approach

- Substitute different vaues of d and kinematic invariants, now we result in large rational numbers
- Take different large prime numbers, move from Z to Zp

Prime field approach

- Substitute different vaues of d and kinematic invariants, now we result in large rational numbers
- Take different large prime numbers, move from Z to Zp
- Run MANY reductions that are much more simple than the original one

Prime field approach

- Substitute different vaues of d and kinematic invariants, now we result in large rational numbers
- Take different large prime numbers, move from Z to Zp
- Run MANY reductions that are much more simple than the original one
- Reconstruct the coefficients

100 values of d, 100 values of x, 20 prime numbers -> 200000 reductions, each of those takes time and use threads -> fits for a super computer.

Rational reconstruction

- An integer is unequely reconstructed by enough of its projections to Zp
- When reconstructing a rational number, we look for smallest possible numerator and denominator
- A few extra prime numbers are for checks

Polynomial reconstruction

- Newton approach

$f(x) = c0 + (x - x_0)(c_1 + (x - x_1)(c_2 + \ldots) \ldots)$
coefficients c_i are algorithmically evaluated from the values $f(x_i)$.

Rational reconstruction

- Thiele approach

$f(x) = c0 + (x - x_0)/(c_1 + (x - x_1)/(c_2 + \ldots) \ldots)$
coefficients c_i are algorithmically evaluated from the values $f(x_i)$.

Rational reconstruction (multiple variables)

Combine two approaches (when coefficients are again functions)

- Newton-Newton (for polynomials)
- Newton-Thiele (when polynomial in one variable)
- Thiele-Newton (something in between)
- Thiele-Thiele (universal but too complex)

Rational reconstruction (multiple variables)

- The ideal case is when denominators of coefficients at master integrals are split into a product of a function of d and a function of x.

Rational reconstruction (multiple variables)

- The ideal case is when denominators of coefficients at master integrals are split into a product of a function of d and a function of x.
- In this case first the x denominators are recovered for a given d.

Rational reconstruction (multiple variables)

- The ideal case is when denominators of coefficients at master integrals are split into a product of a function of d and a function of x.
- In this case first the x denominators are recovered for a given d.
- Then the results are multiplied by the worst denominator and Newton-Thiele is used.

Rational reconstruction (multiple variables)

- The ideal case is when denominators of coefficients at master integrals are split into a product of a function of d and a function of x.
- In this case first the x denominators are recovered for a given d.
- Then the results are multiplied by the worst denominator and Newton-Thiele is used.
- Can we have such a basis with proper coefficients? We beleive that YES!