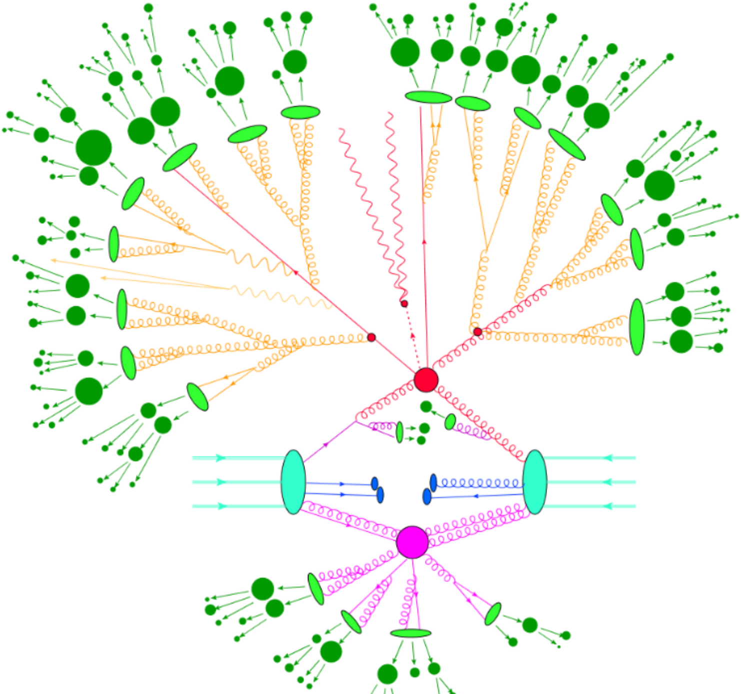
Tropical Feynman integration in the physical region

Felix Tellander

June 15, 2023

Based on [2302.08955] with
M. Borinsky and H. J. Munch

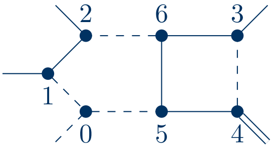


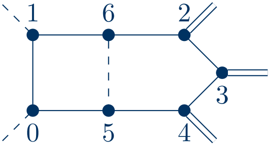
Two-loop box and pentagon integrals

indep. kinem. scales	massive/ off-shell legs	internal masses	process	full σ
$2 \to 2$				
2	0	0	$\gamma\gamma$	2011
2	0	0	$j\,j$ (lc)	2017
2	0	0	$\gamma + j$	2017
3	2	1	$t\bar{t}$	2013
3	2	0	VV	2014
4	2	0	VV'	2015
3	1	0	$V + j$	2015
3	1	0	$H + j$ (HTL)	2015
4	2	1	HH	2016
4	1	1	$H + j$	2018
3	0	1	$gg \to \gamma\gamma$	2019
4	2	1	$gg \to ZZ$	2020
4	2	1	$gg \to WW$	2020
5	2	1	$gg \to ZH$	2021
4	2	1	QCD-EW DY	2022
$2 \to 3$				
4	0	0	3γ	2019
4	0	0	$\gamma\gamma j$	2021
4	0	0	$3\,j$	2021
5	1	0	$Wb\bar{b}$	2022

Analytic results: one off-shell leg [2005.04195,2107.14180]

Today:





Why not `NIntegrate`?

Why not `NIntegrate`?

$$\int_0^1 \int_0^1 \frac{1}{x+y} dx dy = 2 \log 2 \approx 1.3863$$

Why not `NIntegrate`?

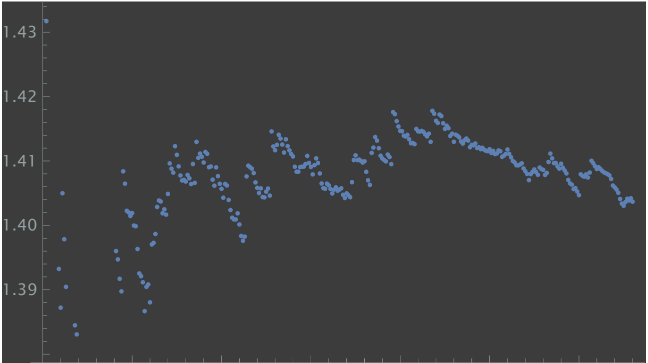
$$\int_0^1 \int_0^1 \frac{1}{x+y} dxdy = 2\log 2 \approx 1.3863$$

```
In[]  NIntegrate[1/(x+y), {x,0,1}, {y,0,1},
      Method->{"MonteCarlo","RandomSeed"->19950309}]
Out[] 0.998259
```

Why not `NIntegrate`?

$$\int_0^1 \int_0^1 \frac{1}{x+y} dxdy = 2\log 2 \approx 1.3863$$

```
In[]  NIntegrate[1/(x+y), {x,0,1}, {y,0,1},
      Method->{"MonteCarlo","RandomSeed"->19950309}]
Out[] 0.998259
```



Reason?

Even though

$$\int_0^1 \int_0^1 \frac{1}{x+y} dxdy = 2\log 2 \approx 1.3863$$

is convergent.

Reason?

Even though

$$\int_0^1 \int_0^1 \frac{1}{x+y} dxdy = 2\log 2 \approx 1.3863$$

is convergent. The integral

$$\int_0^1 \int_0^1 \frac{1}{(x+y)^2} dxdy = \infty$$

is divergent, so

$$\mathrm{Var}\left(\frac{1}{x+y} dx_{[0,1]} dy_{[0,1]} \right) = \infty$$

Reason?

Even though

$$\int_0^1 \int_0^1 \frac{1}{x+y} dxdy = 2\log 2 \approx 1.3863$$

is convergent. The integral

$$\int_0^1 \int_0^1 \frac{1}{(x+y)^2} dxdy = \infty$$

is divergent, so

$$\mathrm{Var}\left(\frac{1}{x+y} dx_{[0,1]} dy_{[0,1]}\right) = \infty$$

The Monte Carlo estimate is

$$\frac{1}{N}\sum_{i=1}^{N} \frac{1}{x_i + y_i} + \mathcal{O}\left(\sqrt{\frac{\infty}{N}}\right)$$

Reason?

Even though

$$\int_0^1 \int_0^1 \frac{1}{x+y} dxdy = 2\log 2 \approx 1.3863$$

is convergent. The integral

$$\int_0^1 \int_0^1 \frac{1}{(x+y)^2} dxdy = \infty$$

is divergent, so

$$\mathrm{Var}\left(\frac{1}{x+y} dx_{[0,1]} dy_{[0,1]}\right) = \infty$$

The Monte Carlo estimate is

$$\frac{1}{N}\sum_{i=1}^{N} \frac{1}{x_i+y_i} + \mathcal{O}\left(\sqrt{\frac{\infty}{N}}\right)$$

This type of integrable boundary singularities are ubiquitous in Feynman integrals.

Feynman integration software

- `pySecDec` [Borowka et al.]
- `FIESTA` [Smirnov]
- `DiffExp` [Hidding]
- `AMFlow` [Liu, Ma]
- `SeaSyde` [Armadillo et al.]
- `HyperInt` [Panzer]

Feynman integration software

- `pySecDec` [Borowka et al.]
- `FIESTA` [Smirnov]
- `DiffExp` [Hidding]
- `AMFlow` [Liu, Ma]
- `SeaSyde` [Armadillo et al.]
- `HyperInt` [Panzer]

- `feyntrop` [Borinsky, Munch and FT]

  Uses *tropical Monte Carlo integration* and can be applied to Euclidean as well as Minkowski kinematics.

  True power: On your laptop you can evaluate high-loop multi-scale integrals in *minutes* to *reasonable* error.

The Feynman Integral

$$\mathcal{I} = \lim_{\varepsilon \to 0^+} \Gamma(\omega) \int_{\mathbb{R}_+^E} \prod_{e \in E} \left(\frac{x^{\nu_e} dx_e}{\Gamma(\nu_e) x_e} \right) \mathcal{U}^{-D/2} \frac{\delta(1 - x_1 - \cdots - x_E)}{\left(\mathcal{V} - i\varepsilon \sum_{e \in E} x_e\right)^{\omega}}$$

with the *superficial degree of divergence*

$$\omega := \sum_{e \in E} \nu_e - LD/2$$

where ν_e are propagator powers and $\mathcal{V} = \mathcal{F}/\mathcal{U}$ with homogeneous graph/Symanzik polynomials

$$\mathcal{U} = \sum_{\substack{T \text{ a spanning} \\ \text{tree of } G}} \prod_{e \notin T} x_e, \qquad \deg(\mathcal{U}) = L$$

$$\mathcal{F} = \mathcal{F}_m + \mathcal{F}_0 = \mathcal{U} \sum_{e \in E} m_e^2 x_e - \sum_{\substack{F \text{ a spanning} \\ 2-\text{forest of } G}} p(F)^2 \prod_{e \notin F} x_e, \qquad \deg(\mathcal{F}) = L + 1$$

Contour Deformation

Why $i\varepsilon$?

Chooses the causal branch and ensures the convergence.

Contour Deformation

Why $i\varepsilon$?

Chooses the causal branch and ensures the convergence.

Why <u>not</u> $i\varepsilon$?

- Modifies the analytic structure by displacing branch points and introducing spurious branch cuts.
- Numerics is hard, as $\varepsilon \to 0$ poles can get arbitrarily close to the integration contour.

Contour Deformation

Why $i\varepsilon$?
Chooses the causal branch and ensures the convergence.

Why <u>not</u> $i\varepsilon$?

- Modifies the analytic structure by displacing branch points and introducing spurious branch cuts.
- Numerics is hard, as $\varepsilon \to 0$ poles can get arbitrarily close to the integration contour.

Instead: Change of variables

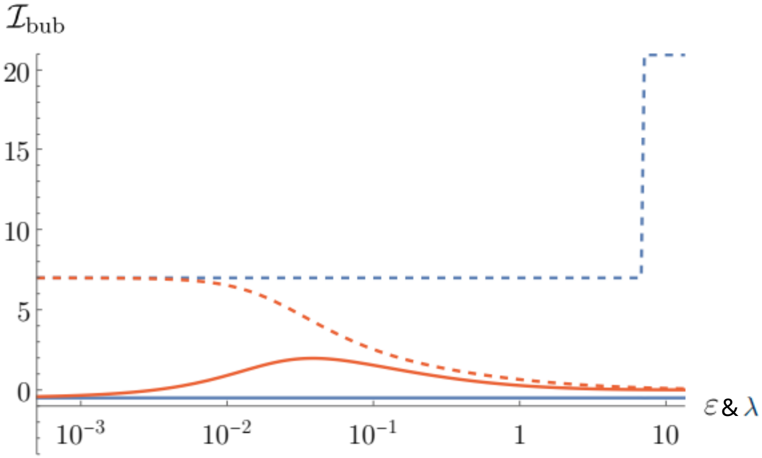
$$X_e = x_e \exp\left(-i\lambda \frac{\partial \mathcal{V}}{\partial x_e}\right)$$

Picks the same causal branch as $i\varepsilon$ as long as λ is *sufficiently small* and

$$x_e \frac{\partial \mathcal{V}}{\partial x_e} \neq 0 \quad \forall e \in E$$

i.e. the *Landau equations* have no solutions.

Comparison with direct numerics on the Feynman parameterization with $i\varepsilon$ and with deformation:



[Hannesdottir,Mizera]

For too large λ we get a jump.

Tropical Monte Carlo

The *tropical approximation* of a polynomial $p(\boldsymbol{x}) = \sum_{\alpha \in \text{supp}(p)} c_\alpha \boldsymbol{x}^\alpha$:

$$p^{\text{tr}}(\boldsymbol{x}) = \max_{\alpha \in \text{supp}(p)} \{\boldsymbol{x}^\alpha\}.$$

Tropical Monte Carlo

The *tropical approximation* of a polynomial $p(\boldsymbol{x}) = \sum_{\alpha \in \mathrm{supp}(p)} c_\alpha \boldsymbol{x}^\alpha$:

$$p^{\mathrm{tr}}(\boldsymbol{x}) = \max_{\alpha \in \mathrm{supp}(p)} \{\boldsymbol{x}^\alpha\}.$$

Theorem

For a homogeneous polynomial $p \in \mathbb{C}[x_1, \ldots, x_n]$ that is completely non-vanishing in $\mathbb{P}^n_+$ there exists constants $C_1, C_2 > 0$ s.t.

$$C_1 \leq \frac{|p(\boldsymbol{x})|}{p^{\mathrm{tr}}(\boldsymbol{x})} \leq C_2 \quad \textit{for all } \boldsymbol{x} \in \mathbb{P}^n_+$$

Tropical Monte Carlo

The *tropical approximation* of a polynomial $p(\boldsymbol{x}) = \sum_{\alpha \in \text{supp}(p)} c_\alpha \boldsymbol{x}^\alpha$:

$$p^{\text{tr}}(\boldsymbol{x}) = \max_{\alpha \in \text{supp}(p)} \{\boldsymbol{x}^\alpha\}.$$

Theorem

For a homogeneous polynomial $p \in \mathbb{C}[x_1, \ldots, x_n]$ that is completely non-vanishing in $\mathbb{P}^n_+$ there exists constants $C_1, C_2 > 0$ s.t.

$$C_1 \leq \frac{|p(\boldsymbol{x})|}{p^{\text{tr}}(\boldsymbol{x})} \leq C_2 \quad \textit{for all } \boldsymbol{x} \in \mathbb{P}^n_+$$

Key assumption: You can find bounds on a deformed polynomial with the un-deformed one.
I.e. there are λ dependent constants $C_1(\lambda), C_2(\lambda) > 0$ s.t.

$$C_1(\lambda) \leq \left| \left(\frac{\mathcal{U}^{\text{tr}}(\boldsymbol{x})}{\mathcal{U}(\boldsymbol{X})}\right)^{D_0/2} \left(\frac{\mathcal{V}^{\text{tr}}(\boldsymbol{x})}{\mathcal{V}(\boldsymbol{X})}\right)^{\omega_0} \right| \leq C_2(\lambda) \quad \text{for all} \quad \boldsymbol{x} \in \mathbb{P}^E_+$$

where the denominators are the deformed polynomials.

Expanding in ϵ:

Assuming that the only potential divergence comes from $\Gamma(\omega)$ we have:

$$\mathcal{I} = \Gamma(\omega_0 + \epsilon L) \sum_{k=0}^{\infty} \frac{\epsilon^k}{k!} \int_{\mathbb{P}_+^E} \left(\prod_{e \in E} \frac{X_e^{\nu_e}}{\Gamma(\nu_e)} \right) \frac{\det \mathcal{J}_\lambda(\boldsymbol{x})}{\mathcal{U}(\boldsymbol{x})^{D_0/2} \cdot \mathcal{V}(\boldsymbol{x})^{\omega_0}} \log^k \left(\frac{\mathcal{U}(\boldsymbol{X})}{\mathcal{V}(\boldsymbol{X})^L} \right) \Omega$$

where $\omega_0 = \sum_{e \in E} \nu_e - D_0 L/2$.

Expanding in ϵ:

Assuming that the only potential divergence comes from $\Gamma(\omega)$ we have:

$$\mathcal{I} = \Gamma(\omega_0 + \epsilon L) \sum_{k=0}^{\infty} \frac{\epsilon^k}{k!} \int_{\mathbb{P}_+^E} \left(\prod_{e \in E} \frac{X_e^{\nu_e}}{\Gamma(\nu_e)} \right) \frac{\det \mathcal{J}_\lambda(\boldsymbol{x})}{\mathcal{U}\left(\boldsymbol{X}\right)^{D_0/2} \cdot \mathcal{V}\left(\boldsymbol{X}\right)^{\omega_0}} \log^k \left(\frac{\mathcal{U}(\boldsymbol{X})}{\mathcal{V}(\boldsymbol{X})^L} \right) \Omega$$

where $\omega_0 = \sum_{e \in E} \nu_e - D_0 L/2$.

Writing the integral with these fractions in the integrand:

$$\mathcal{I} = \frac{\Gamma(\omega_0 + \epsilon L)}{\prod_{e \in E} \Gamma(\nu_e)} \sum_{k=0}^{\infty} \frac{\epsilon^k}{k!} \mathcal{I}_k$$

with

$$\mathcal{I}_k = I^{\text{tr}} \int_{\mathbb{P}_+^E} \frac{\left(\prod_{e \in E} (X_e/x_e)^{\nu_e}\right) \det \mathcal{J}_\lambda(\boldsymbol{x})}{\left(\mathcal{U}\left(\boldsymbol{X}\right) / \mathcal{U}^{\text{tr}}\left(\boldsymbol{x}\right)\right)^{D_0/2} \cdot \left(\mathcal{V}\left(\boldsymbol{X}\right) / \mathcal{V}^{\text{tr}}\left(\boldsymbol{x}\right)\right)^{\omega_0}} \log^k \left(\frac{\mathcal{U}(\boldsymbol{X})}{\mathcal{V}(\boldsymbol{X})^L} \right) \mu^{\text{tr}}$$

and

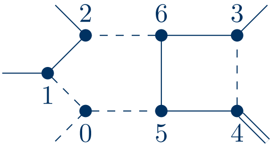
$$\mu^{\text{tr}} = \frac{1}{I^{\text{tr}}} \frac{\prod_{e \in E} x_e^{\nu_e}}{\mathcal{U}^{\text{tr}}(\boldsymbol{x})^{D_0/2} \mathcal{V}^{\text{tr}}(\boldsymbol{x})^{\omega_0}} \Omega, \quad \int_{\mathbb{P}_+^E} \mu^{\text{tr}} = 1.$$

The program `feyntrop`

Available at https://github.com/michibo/feyntrop

A `C++` program with `Python` interface:

Example:



Dashed lines: massless, the solid lines: mass m and double $p_4^2 \neq 0$.

```
edges = [((0,1), 1, '0'), ((1,2), 1, 'mm'), ((2,6), 1, '0'),
         ((6,3), 1, 'mm'), ((3,4), 1, '0'), ((4,5),1, 'mm'),
         ((5,0), 1, '0'), ((5,6), 1, 'mm')]
```

Phase space point

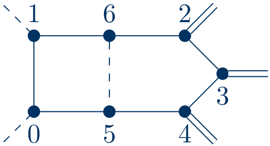
$$p_0^2 = 0\,, \quad p_1^2 = p_2^2 = p_3^2 = m^2 = 1/2\,, \quad s_{01} = 2.2\,, \quad s_{02} = 2.3\,,$$
$$s_{03} = 2.4\,, \quad s_{12} = 2.5\,, \quad s_{13} = 2.6\,, \quad s_{23} = 2.7\,,$$

where $s_{ij} = (p_i + p_j)^2$. With $\lambda = 0.28\,,\ N = 10^8$, we obtain:

```
Prefactor: gamma(2*eps + 2).
(Effective) kinematic regime: Minkowski (exceptional).
Finished in 8.20 seconds.
-- eps^0: [0.06480 +/- 0.00078]  +  i * [-0.08150 +/- 0.00098]
-- eps^1: [0.4036  +/- 0.0045 ]  +  i * [ 0.3257  +/- 0.0035 ]
-- eps^2: [-0.7889 +/- 0.0060 ]  +  i * [ 0.957   +/-  0.016 ]
-- eps^3: [-1.373  +/-  0.030 ]  +  i * [ -1.181  +/-  0.034 ]
-- eps^4: [ 1.258  +/-  0.088 ]  +  i * [ -1.205  +/-  0.036 ]
```

This is a **two-loop** integral with different mass scales that you can integrate on your **laptop** in **8 seconds**.

Example:




```
edges = [((0,1), 1, 'mm_top'), ((1,6), 1, 'mm_top'),
         ((5,6), 1, '0'), ((6,2), 1, 'mm_top'),
         ((2,3), 1, 'mm_top'), ((3,4), 1, 'mm_top'),
         ((4,5), 1, 'mm_top'), ((5,0), 1, 'mm_top')]
```

With $s_{ij} := (p_i + p_j)^2$, we have the following kinematic setup:

$$p_0^2 = p_1^2 = 0\,, \quad p_2^2 = p_3^2 = p_4^2 = m_H^2\,,$$

$$s_{01} = 5m_H^2 - s_{02} - s_{03} - s_{12} - s_{13} - s_{23}\,.$$

Phase space point:

$$m_t^2 = 1.8995\,, \quad m_H^2 = 1\,,$$

$$s_{02} = -4.4\,, \quad s_{03} = -0.5\,, \quad s_{12} = -0.6\,, \quad s_{13} = -0.7\,, \quad s_{23} = 1.8\,,$$

Setting $\lambda = 0.64$ and $N = 10^8$, we get:

```
Prefactor: gamma(2*eps + 4).
(Effective) kinematic regime: Minkowski (generic).
Finished in 8.12 seconds.
-- eps^0: [-0.0114757 +/- 0.0000082]
           +  i * [0.0035991 +/- 0.0000068]
-- eps^1: [ 0.003250  +/- 0.000031 ]
           +  i * [-0.035808 +/- 0.000041 ]
-- eps^2: [ 0.046575  +/- 0.000098 ]
           +  i * [0.016143  +/- 0.000088 ]
-- eps^3: [ -0.01637  +/-  0.00017 ]
           +  i * [ 0.03969  +/-  0.00016 ]
-- eps^4: [ -0.02831  +/-  0.00023 ]
           +  i * [-0.00823  +/-  0.00024 ]
```

- `feyntrop` 8.12 seconds with relative error $\sim 10^{-3}$
- `pySecDec` 3 hours with relative error $\sim 10^{-2}$

Thank you!