Expansion by regions with pySecDec

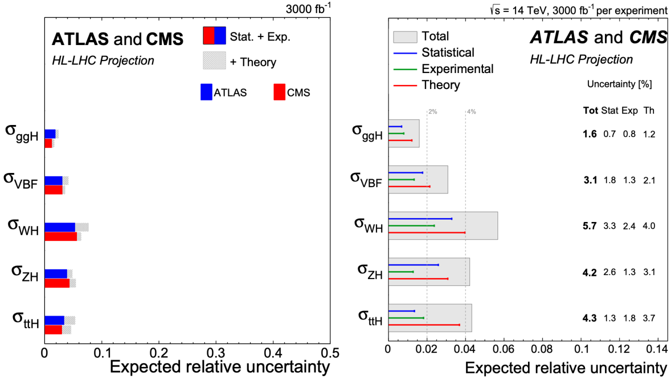
Emilio Villa

RADCOR + LoopFest, 17-21 May 2021
Tallahassee, FL, USA

Karlsruher Institut für Technologie

Introduction

Why caring about loop integrals?



Observables are dominated by theoretical uncertainties[1]

[1]image from *CERN HL-HE Yellow Report 2019*

Introduction

There are **many techniques** to evaluate loop integrals:

- Mellin-Barnes representation
- Differential Equations
- Dimensional Recurrence
- Sector Decomposition
- Asymptotic expansions (e.g. Expansion by regions)

In the following $\rightarrow$ *Expansion by regions*

Expansion by regions

Expansion by regions[2]

First: motivation

$$G = \int \prod_{l=1}^{L} d^D\kappa_l \frac{1}{\prod_{j=1}^{N} P_j^{\nu_j}\left(\{k\},\{p\},m_j^2\right)}, \quad d^D\kappa \equiv \mu^{4-D}\frac{d^D k}{i\pi^{D/2}}$$

G is a complicated function of masses m_j and kinematics invariants $p_i \cdot p_j$

Idea: Exploit parameter hierarchies to expand **integrand** in small parameter, e.g. m^2/p^2 → resulting integrals might be easier to evaluate

Caveat: one cannot just Taylor expand → magnitude of k_l

[2] The method was pioneered in arXiv:hep-ph/9711391 by M. Beneke and V.A. Smirnov

Expansion by regions

Example $\rightarrow$ limit $|p^2| \gg m^2$ of

$$G = \int d^D\kappa \frac{1}{(k+p)^2(k^2-m^2)^2} \equiv \int d^D\kappa\ \mathcal{I}$$

hard region: $|k^2| \gg m^2$

soft region: $|k^2|, |2k \cdot p| \ll p^2$

$$\mathcal{I}_{(h)} \sim \frac{1}{(k+p)^2(k^2)^2}\left(1 + 2\frac{m^2}{k^2}\right)$$

$$\mathcal{I}_{(s)} \sim \frac{1}{p^2(k^2-m^2)^2}\left(1 - \frac{k^2+2p\cdot k}{p^2}\right)$$

Expansion by regions

Next $\rightarrow$ *Integrate over* ***whole*** *domain*

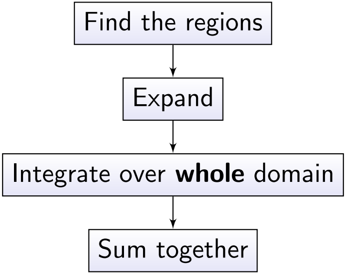
$$G = \int d^D\kappa\, \mathcal{I}_{(h)} + \int d^D\kappa\, \mathcal{I}_{(s)} - \underbrace{\int d^D\kappa\, \mathcal{I}_{(hs)}}_{\text{scaleless} \to 0 \text{ in } DR}$$

$$= \frac{1}{p^2}\left[-\frac{1}{\epsilon} + \ln\left(\frac{-p^2}{\mu^2}\right)\right] + \frac{1}{p^2}\left[\frac{1}{\epsilon} - \ln\left(\frac{m^2}{\mu^2}\right)\right] + o\left(\epsilon, \frac{m^2}{p^2}\right)$$

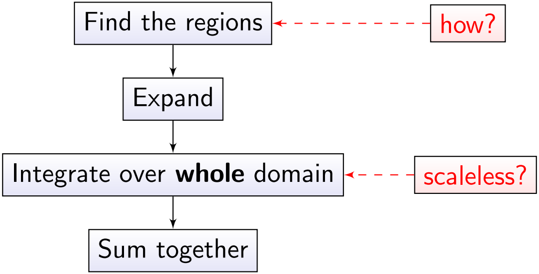
$$= \frac{1}{p^2}\ln\left(\frac{-p^2}{m^2}\right) + o\left(\epsilon, \frac{m^2}{p^2}\right)$$

Expansion by regions

Workflow



Workflow


Find the regions
how?
Expand
Integrate over **whole** domain
scaleless?
Sum together

Expansion by regions

Let's move to Feynman parameters → *Lee-Pomeransky parametrisation*[3]

$$G \propto \int_0^\infty \prod_j dx_j \; x_j^{\nu_j - 1} \; P^{-D/2}$$

where $P = F + U$ is a **polynomial**

→ to find the regions we use the *Geometric Approach*[4]

[3] arXiv:1308.6676 by R. Lee and A. Pomeranksy
[4] arXiv:1011.4863 by A. Pak and A. Smirnov

Expansion by regions

Consider

$$P(\mathbf{x}, t) = \sum_{i=0}^{m} c_i x_1^{p_{i,1}} \dots x_n^{p_{i,n}} t^{p_{i,n+1}}$$

with

- $c_i \rightarrow$ non-negative coefficients
- $x_i \rightarrow$ integration variables
- $\mathbf{p}_i = (p_{i,1}, \dots, p_{i,n+1}) \in \mathbb{N}^{n+1} \rightarrow$ exponent vectors
- $t \rightarrow$ small parameter

Expansion by regions

We define $\mathbf{u}$ such that $x_i = t^{u_i}$ (note: $t = t \to u_{n+1} = 1$) and write

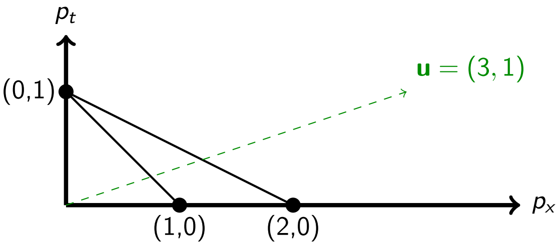
$$P(\mathbf{u}, t) = \sum_{i=0}^{m} c_i t^{\mathbf{p}_i \cdot \mathbf{u}}$$

The **largest term** of the polynomial is the one with the smallest value of $\mathbf{p}_i \cdot \mathbf{u} \to$ let's visualise this with the **Newton polytope** $\equiv$ **convHull** $(\mathbf{p}_1, \mathbf{p}_2, \dots)$

$$\text{convHull}\,(\mathbf{p}_1, \mathbf{p}_2, \dots) = \left\{ a_1\mathbf{p}_1 + \cdots + a_n\mathbf{p}_n \mid a_i > 0 \;\; \forall i, \sum_{i=1}^{n} a_i = 1 \right\}$$

Expansion by regions

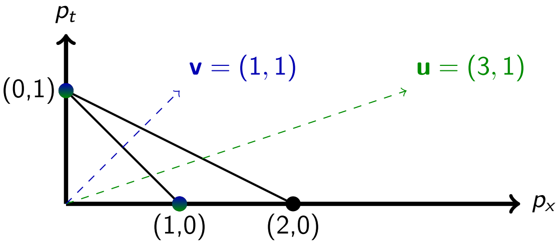
Newton polytope for $P(x) = x + x^2 + t$, along with an example vector $\mathbf{u}$



$\mathbf{p}_0 = (1, 0), \mathbf{p}_1 = (2, 0), \mathbf{p}_2 = (0, 1) \rightarrow P(t) = t^3 + t^6 + t$

Expansion by regions

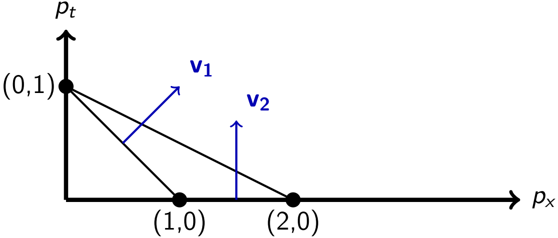
When expanding according to **v** gives a convergent expansion at **u**?



Answer: {vertices closest along **v**} $\subseteq$ {vertices closest along **u**}

Expansion by regions

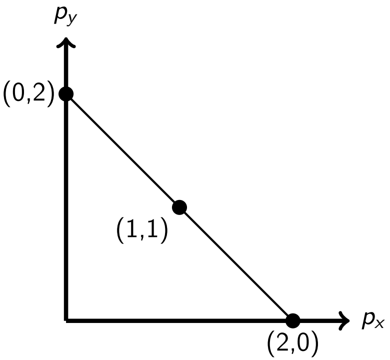
We can find all the regions choosing the $\mathbf{v_i}$



to be the **normal vectors** to the facets **pointing upwards** → *"how?" solved*

Expansion by regions

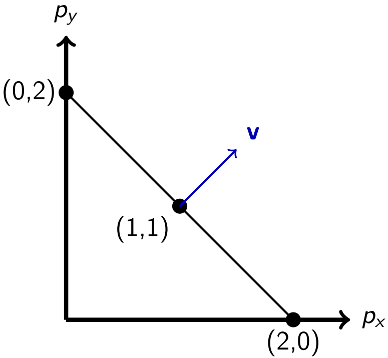
Consider now $P(x, y) = x^2 + y^2 + xy$ and the corresponding polytope



the points lie on the line $p_x + p_y = 2$ orthogonal to the $\mathbf{v} = (1, 1)$ direction.

Expansion by regions

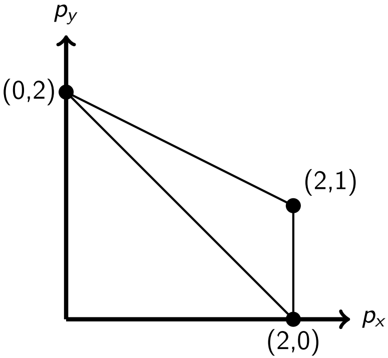
Rescaling with $\mathbf{v} = (1,1)$, i.e. $x \to \rho x$, $y \to \rho y$ gives $P(\rho x, \rho y) = \rho^2 P(x,y)$



Note that the **area** of the Newton polytope $\mathcal{N}_P$ is **0**.

Expansion by regions

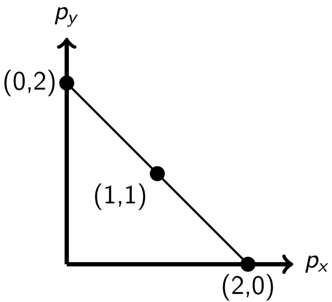
For non homogeneous polynomials $\rightarrow Q(x,y) = x^2 + y^2 + x^2y$

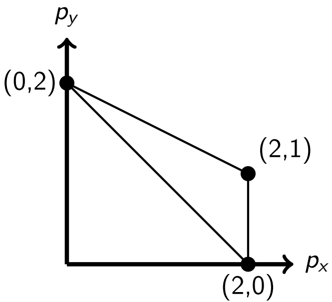


The area of $\mathcal{N}_Q$ is non-zero.

Expansion by regions

Homogeneity[5] $\equiv$ **Scalelessness**





Multiple expansions produce lower dimensional polytope $\rightarrow$ *"scaleless?" solved*

[5] homogeneity w.r.t. a subset of the Feynman parameters.

Expansion by regions

it's actually not that easy ...

- with negative coefficients $\rightarrow$ **new regions** arise, hard to detect
- dimension as regulator not enough $\rightarrow$ **additional regulators** needed
- **overlap contributions** $\neq 0 \rightarrow$ e.g. when not using analytic regulators

For more details $\rightarrow$ *arXiv:1111.2589 by B. Jantzen*

However: Problematic cases can in general be **anticipated** and the validity of the method **assessed**

pySecDec: new release!

What's new?

1. automated **Expansion by regions**

but also

2. automatic reduction of λ_i → **no more sign check error!**
3. $\sum_k c_k I_k$ → automatic adjustment # evaluation points
4. `FORM` settings adjusted automatically[6] → based on detected hardware

→ towards *amplitudes* evaluation

[6] Based on the work of T. Ueda

pySecDec: λ_i reduction

For physical kinematics, contour deformation might be needed:

$$z_i(\mathbf{x}) = x_i - i\lambda_i x_i(1 - x_i)\frac{\partial F}{\partial x_i}(\mathbf{x})$$

In order to preserve Feynman prescription $-i\delta$, λ_i should be small enough.

Before: sign-check error and stop of the integration

Now: **automatic λ_i reduction**

pySecDec: sum of integrals and coefficients

The # of sampling points N_s for each integral is set depending on its contribution to the **error estimate** of the sum and on the **time required** for each integrand evaluation. We set N_s minimising:

$$T = \sum_i t_i + \beta \left(\Delta_S^2 - \sum_i c_i^2 \Delta_i^2 \right)$$

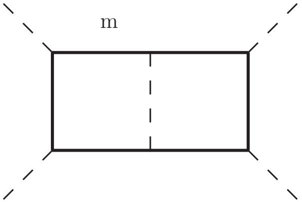
where:

- $t_i \rightarrow$ integration time of I_i
- $\Delta_i \rightarrow$ absolute error of I_i
- $\Delta_S \rightarrow$ absolute error of S (accuracy goal)
- $\beta \rightarrow$ Lagrange multiplier

$\rightarrow$ **global accuracy goal** for the sum reached more **efficiently**.

Examples

pySecDec: a hard example

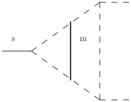
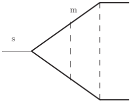
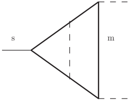


For $s,\ t,\ m^2 = 5.3,\ -1.86,\ 0.1$ and expanding at LO in m^2:

- regions: 13
- integrals: 5866
- time[1] (compile + integrate): 10 [h]
- accuracy: 1.4 %

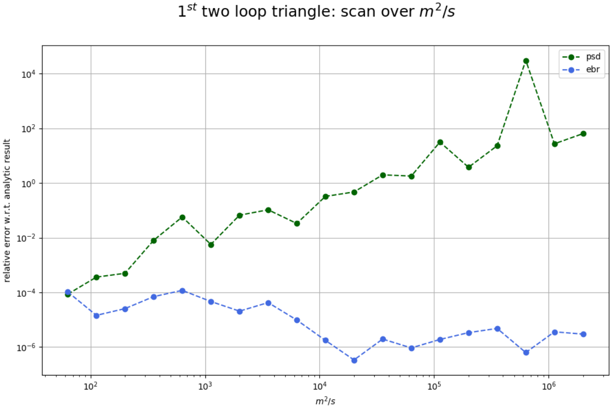
[1] Integration ran on a system with 4 GeForce 1080 Ti GPUs

pySecDec: timings

Diagram	psd (r: 10^1) [min]	psd (r: 10^3) [min]	ebr (r: 10^3) [min]
	5.23	101.94	1.61
	1.52	33.77	8.55
	0.12	0.13	0.09

r $\equiv$ invariants ratio, accuracy: 10^{-2}

pySecDec: scan (ebr vs psd)



ebr is numerically stable over many orders of magnitude as ratio of scales increases

Conclusions

Conclusions

Summary:

- Expansion by regions
- `pySecDec` new features:
  1. automatic λ_i reduction
  2. automatic adjustment # evaluation points
  3. `FORM` settings
- Examples

Thank you for listening!