ETH zürich

Machine learning for Monte-Carlo integration

Valentin Hirschi

ACAT

11th January 2019

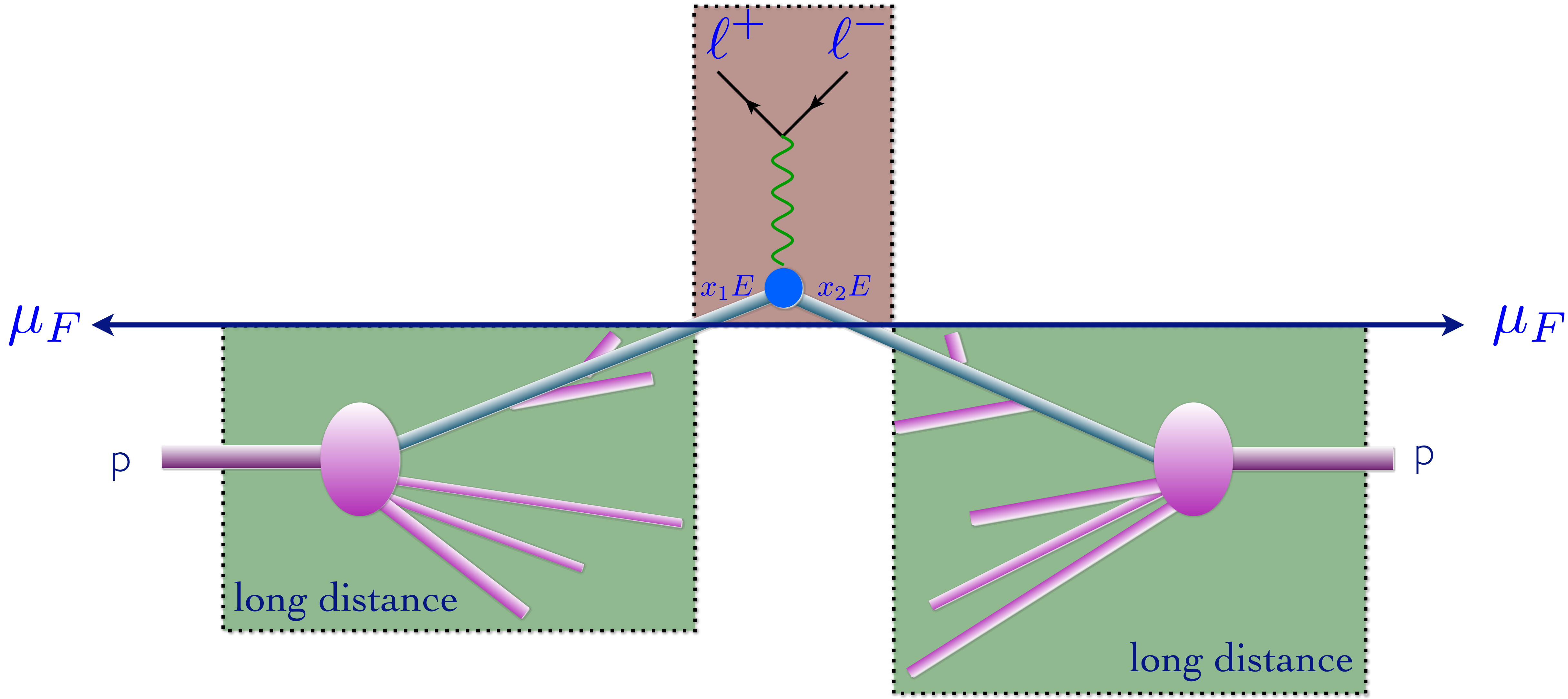
Plan

- Type of integrals appearing in HEP
- Standard phase-space integration techniques
- Current numerical methods for loop integrals
- Machine learning assisted MC-integration
- Prospects

Plan

- Type of integrals appearing in HEP
- Standard phase-space integration techniques
- Current numerical methods for loop integrals
- Machine learning assisted MC-integration
- Prospects

Cross-section computation



$$\sum_{a,b} \int dx_1 dx_2 d\Phi_{\text{FS}}\, f_a(x_1,\mu_F) f_b(x_2,\mu_F)\, \hat{\sigma}_{ab\to X}(\hat{s},\mu_F,\mu_R)$$

Phase-space integral → $Dim[\Phi(n)] \sim 3n$

Parton density functions

Parton-level cross section → Peaked function

Phase-space integrals

$$\int d^4p_1 \dots d^4p_N \, |\mathcal{M}(\{p_i\})|^2 \, \delta p_1^2 \dots \delta p_N^2 \delta^4 \left(p_1 + \dots p_N\right)$$

- Need a parametrisation to solve the deltas, e.g. N=2 :

$$s = (p_1 + p_2)^2 \quad t = (p_1 + p_3)^2 \quad u = (p_1 + p_4)^2 = \sum_i^N M_i^2 - s - t$$

- Neural networks may help to cope with the high PS dim $(d = 3N - 3)$ and peak structure of PDF & matrix elements.

- Timings

	Tree-level MG5aMC [arXiv:1405.0301]		One-loop MadLoop [arXiv:1103.0621]		Two-loop VVamp [arXiv:1503.08835]
$d\bar{d} \to ZZ$	7 µs	x 10^2 →	0.6 ms	x 10^4 →	$\mathcal{O}$(1 min)
$d\bar{d} \to ZZg$	35 µs	x 10^3 →	38 ms		N/A
$d\bar{d} \to ZZgg$	220 µs	x 10^4 →	1200 ms		N/A

Loop integrals

$$\int d^d k_1 \ldots d^d k_{N_l} \; \frac{N(\{k_i\})}{\Pi_1^{N_d} D_i}$$

- Dimensionality fixed by number of loops, not ext. kinematics
- Features mass threshold singularities and IR+UV divergences
- Polynomial integrand, fast evaluation time for all topologies
- Advanced analytic integration techniques developed, but purely numerical methods are now (re-)surfacing.
- Machine learning can help cope with the large variance of the resulting integrands.

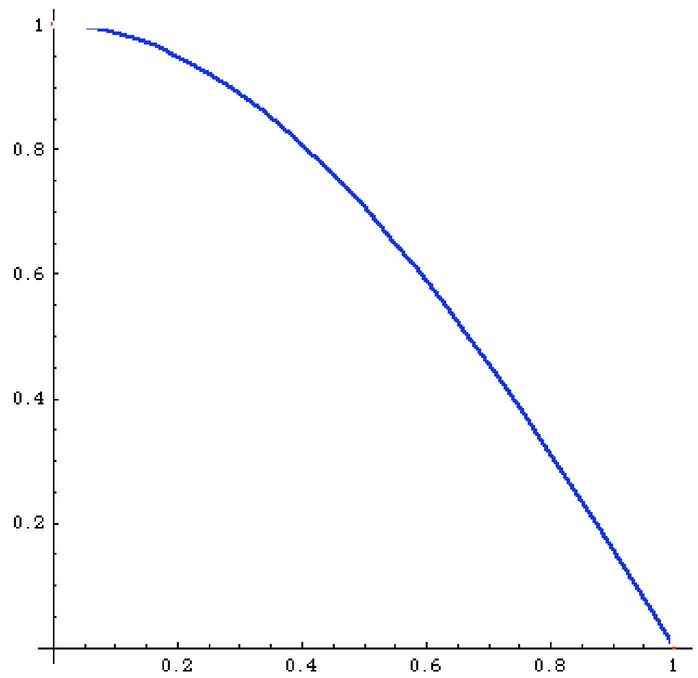
Plan

- Type of integrals appearing in HEP
- Standard phase-space integration techniques
- Current numerical methods for loop integrals
- Machine learning assisted MC-integration
- Prospects

PLAN

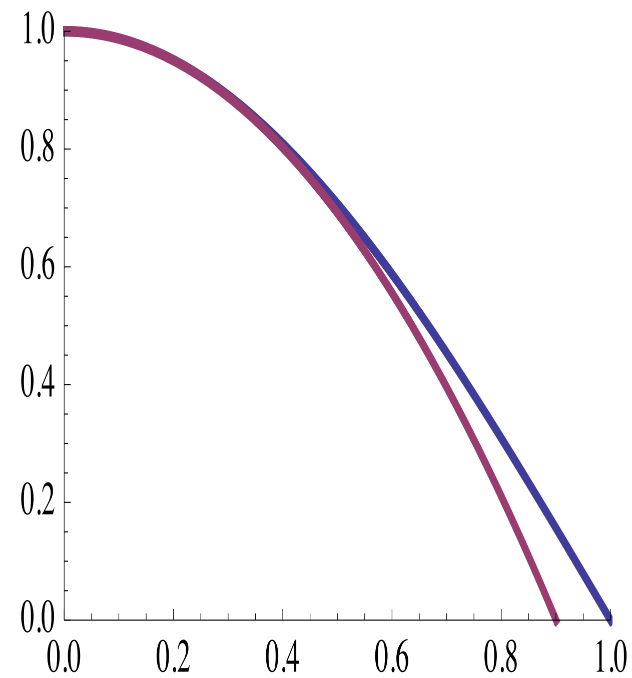
- Type of integrals appearing in HEP
- Standard phase-space integration techniques
- Current numerical methods for loop integrals
- Machine learning assisted MC-integration
- Prospects

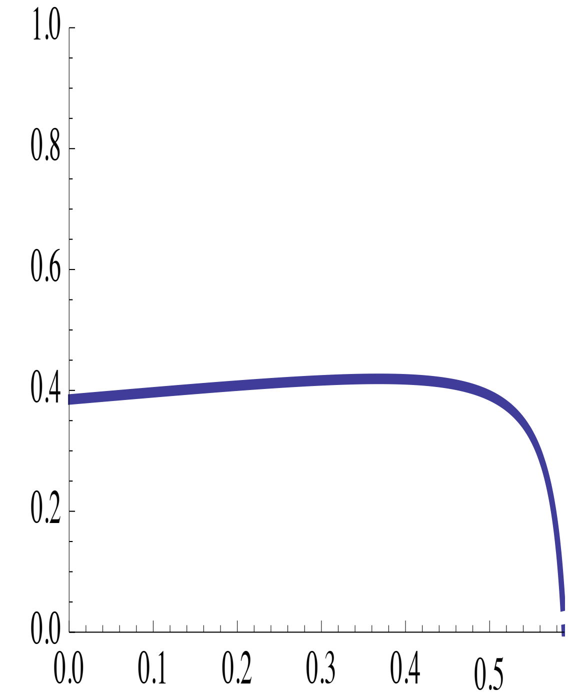
Importance sampling



$$I = \int_0^1 dx \cos \frac{\pi}{2} x$$

$$I_N = 0.637 \pm 0.307/\sqrt{N}$$





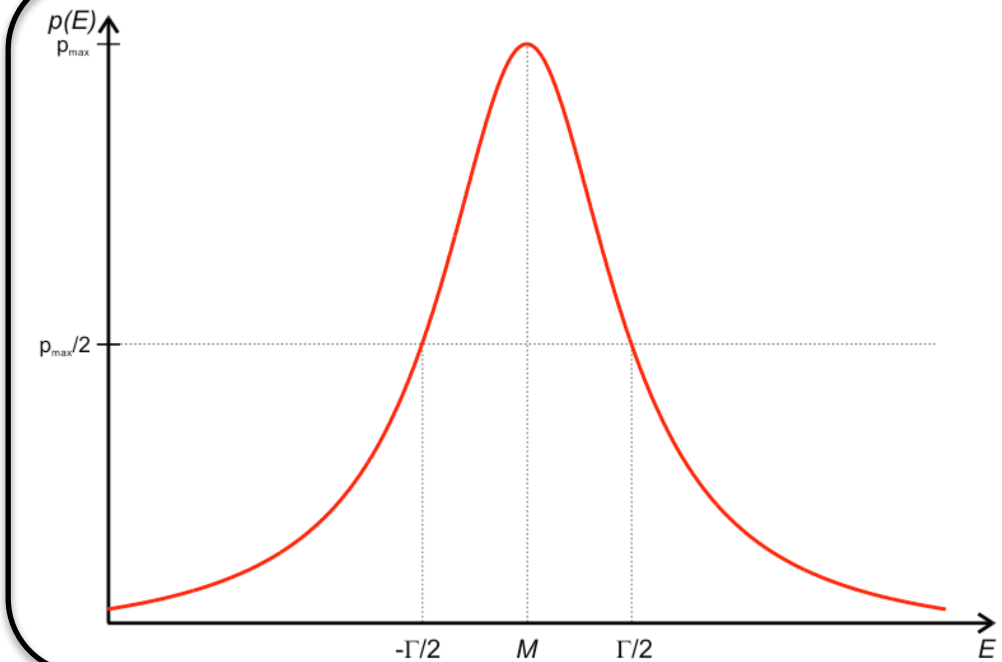
$$I = \int_0^1 dx (1 - cx^2) \frac{\cos\left(\frac{\pi}{2} x\right)}{(1 - cx^2)} = \int_{\xi_1}^{\xi_2} d\xi \frac{\cos \frac{\pi}{2} x[\xi]}{1 - x[\xi]^2 c}$$

$$\simeq 1$$

$$I_N = 0.637 \pm 0.031/\sqrt{N}$$

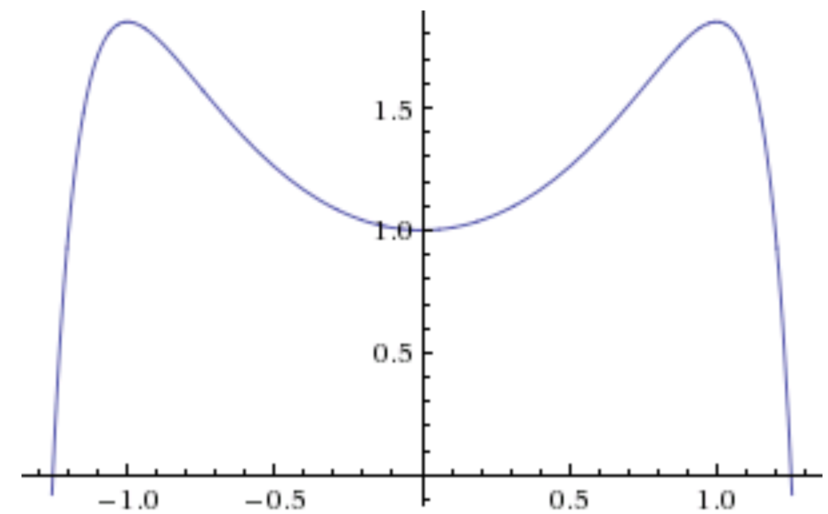
Phase-Space parametrisation is key to tame the variance

Importance sampling

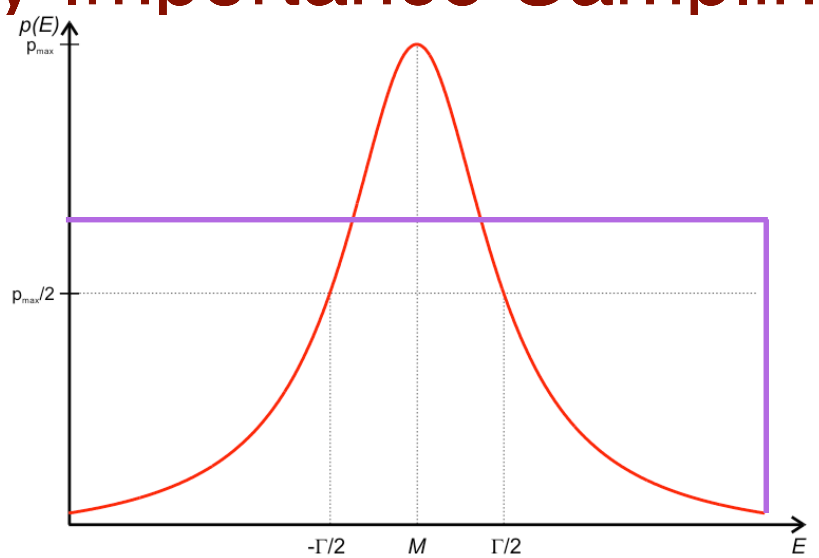


$$\int \frac{dq^2}{(q^2 - M^2 + iM\Gamma)^2}$$

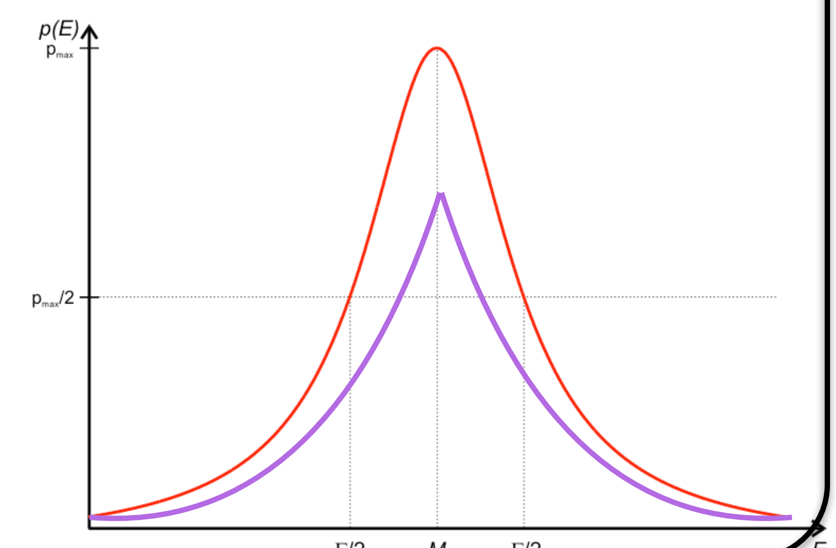
$$\xi = arctan\left(\frac{q^2 - M^2}{\Gamma\, M}\right)$$



Why Importance Sampling?



Probability of using that point p(x)

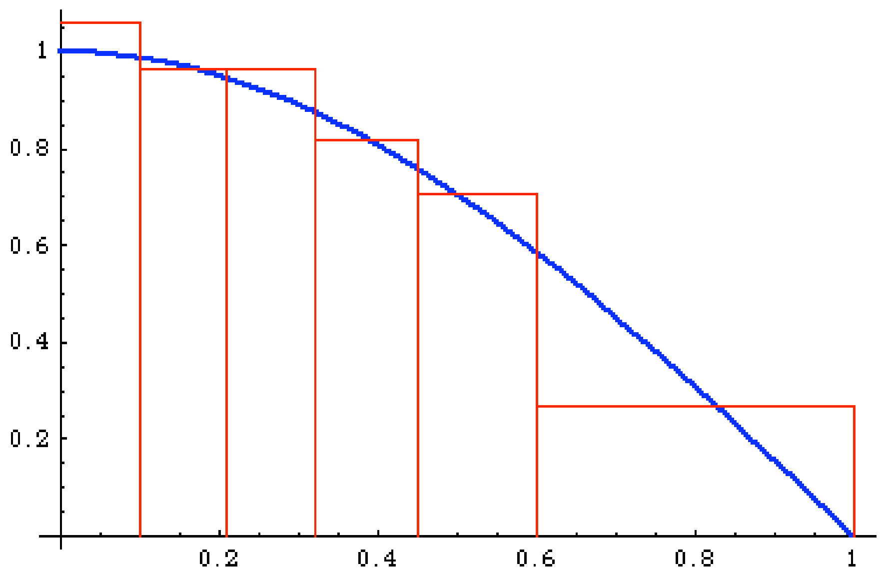


The change of variable ensures that the evaluation of the integrand is done where it is largest

Importance sampling

Adaptative Monte-Carlo

- Create a piece-wise approximation of the function on the fly



Essence of the implementation

1. Setup bins such that each of them contain the same contribution.

➡ Many bins where integrand is large

2. Use the resulting approximation for importance sampling.

VEGAS

More than one Dimension

- VEGAS works only with 1(few) dimension
  - ➡ Otherwise number of “bins” scales like N^d :(

VEGAS

More than one Dimension

- VEGAS works only with 1(few) dimension
  - ➡ Otherwise number of “bins” scales like N^d :(

Solution

- Factorisation ansatz: projection on the axis

$$\vec{p}(x) = p(x) \cdot p(y) \cdot p(z) \ldots$$

VEGAS

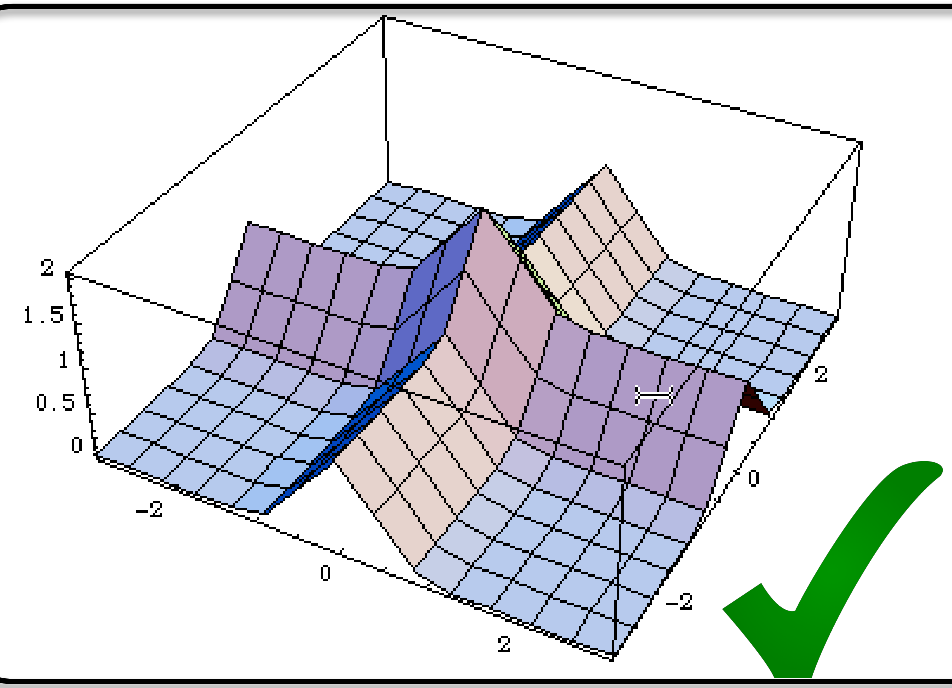
More than one Dimension

- VEGAS works only with 1(few) dimension
  - ➡ Otherwise number of “bins” scales like N^d :(

Solution

- Factorisation ansatz: projection on the axis

$$\vec{p}(x) = p(x) \cdot p(y) \cdot p(z) \ldots$$



VEGAS

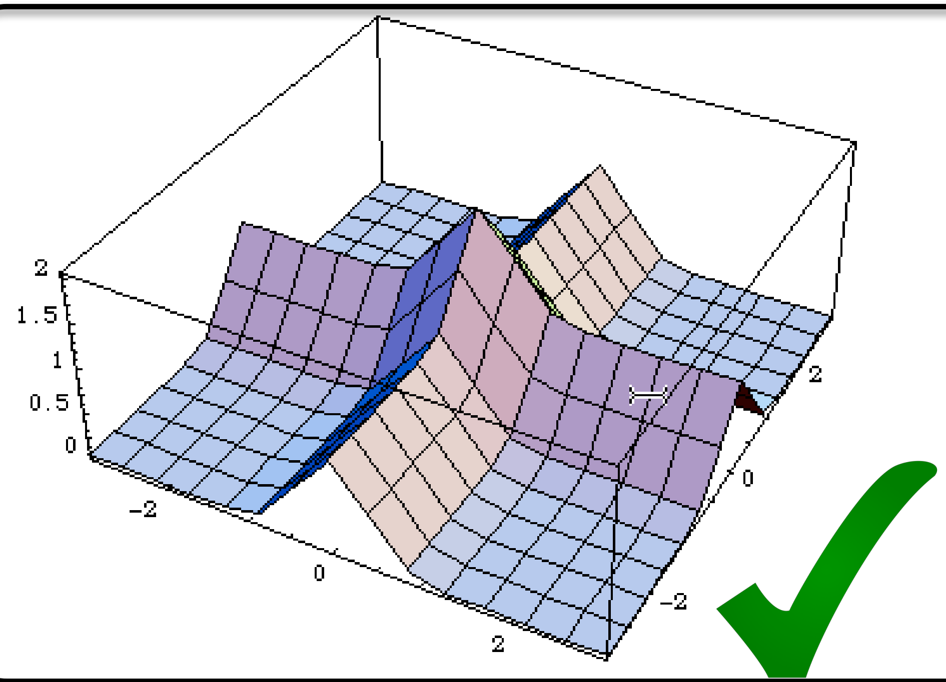
More than one Dimension

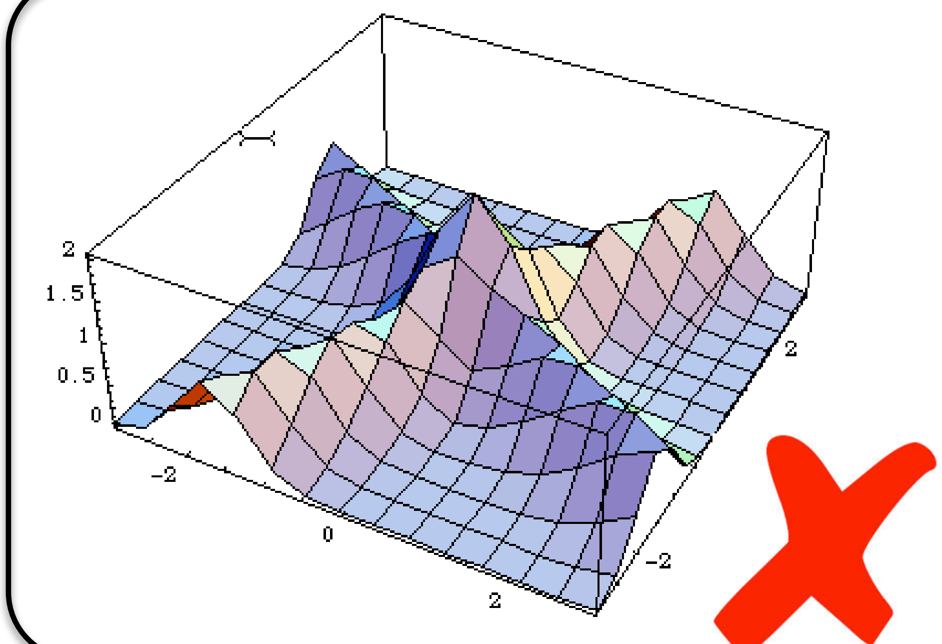
- VEGAS works only with 1(few) dimension
  - ➡ Otherwise number of "bins" scales like N^d :(

Solution

- Factorisation ansatz: projection on the axis

$$\vec{p}(x) = p(x) \cdot p(y) \cdot p(z) \ldots$$



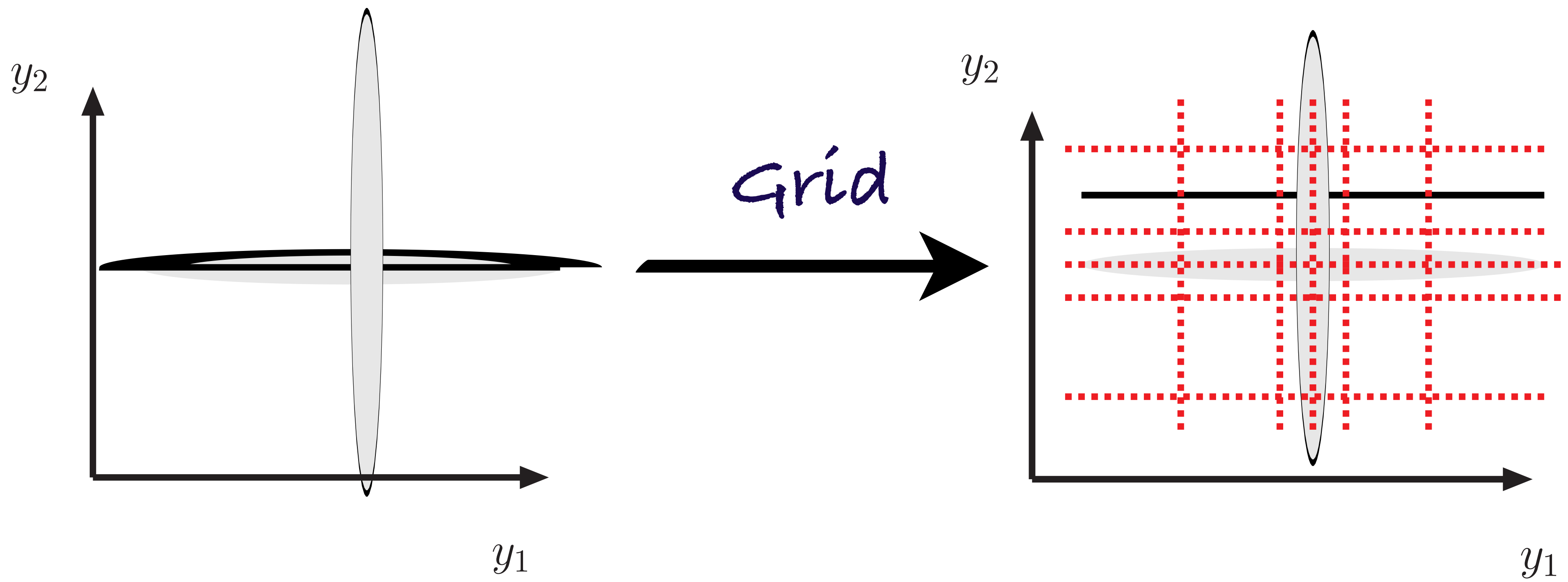


- Factorisation breaks down

➡ Additional change of variable needed

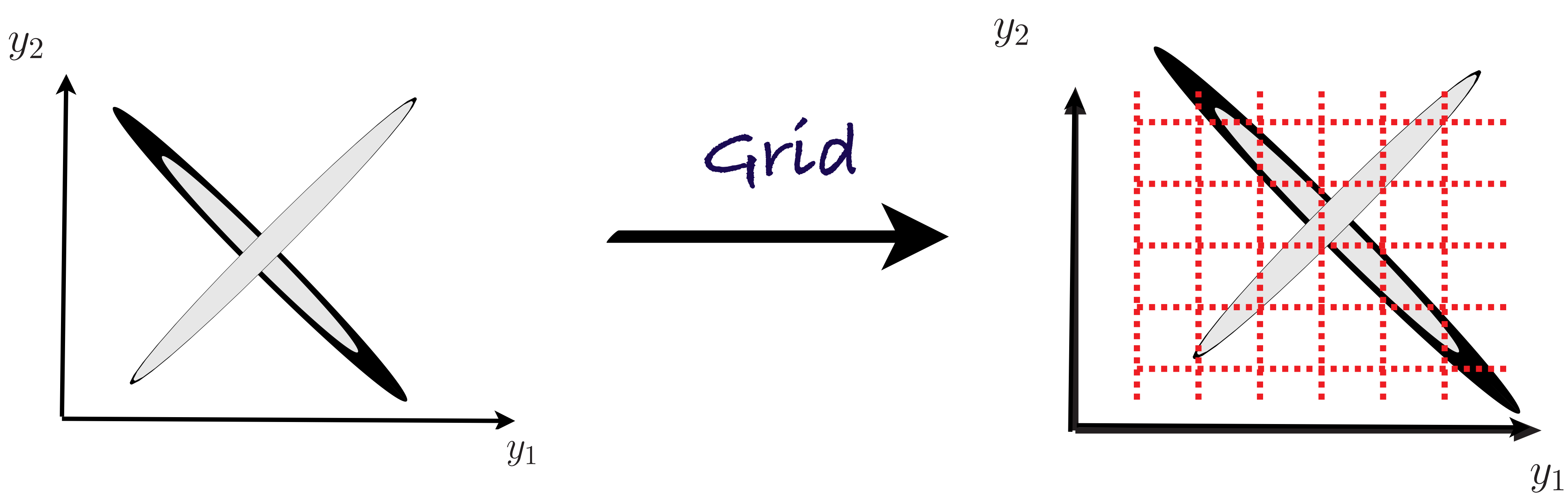
VEGAS

- The choice of the parameterisation has a strong impact on the efficiency



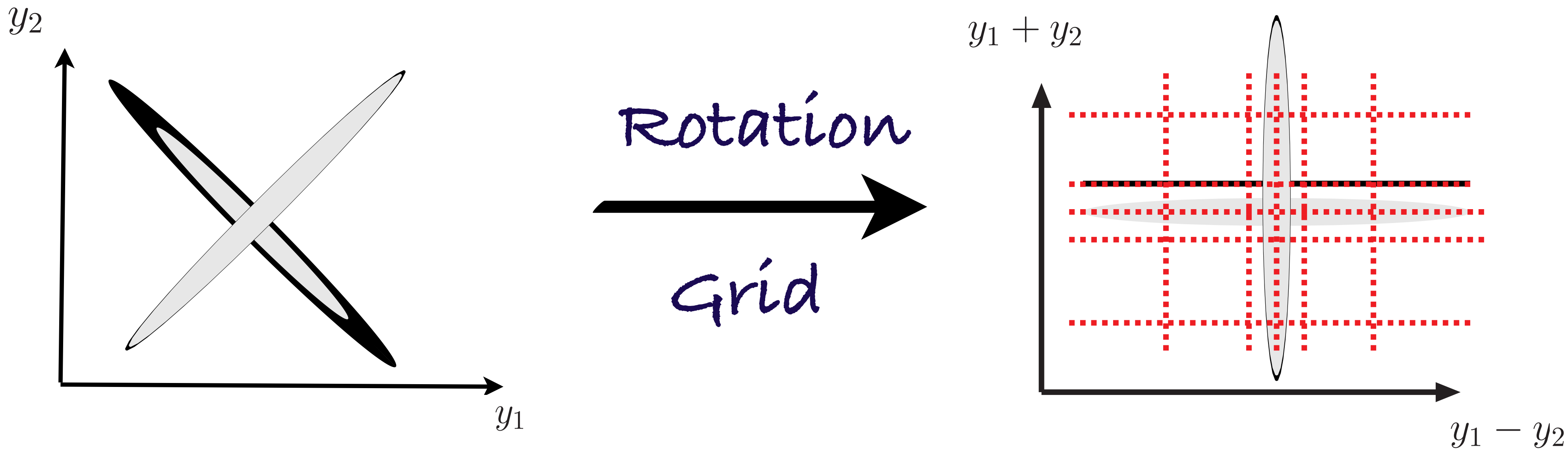
VEGAS

- The choice of the parameterisation has a strong impact on the efficiency

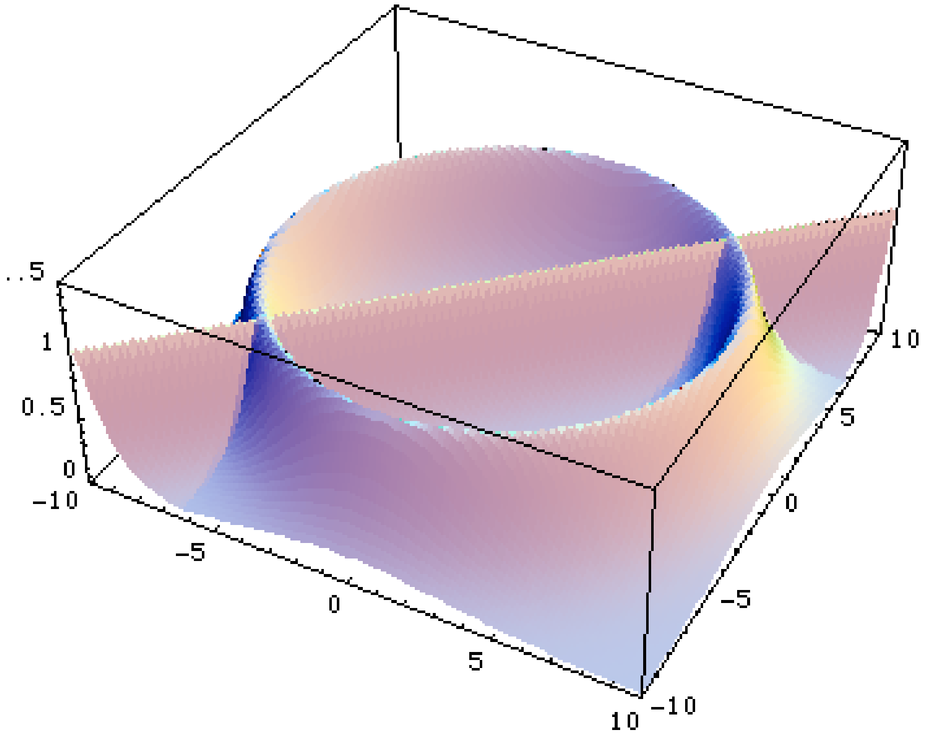


VEGAS

- The choice of the parameterisation has a strong impact on the efficiency

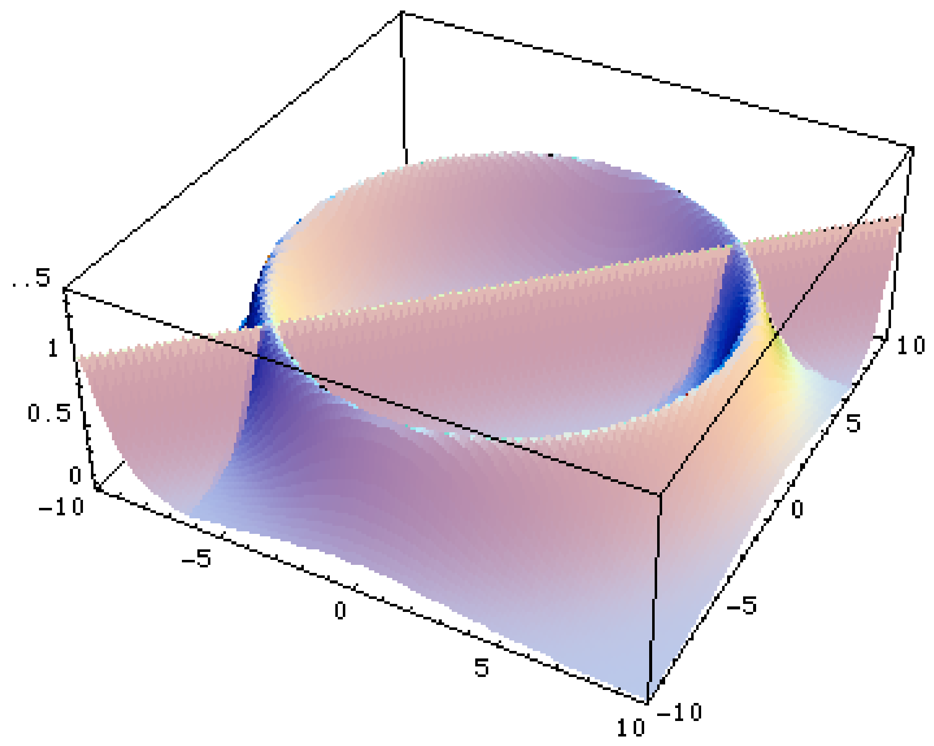


MULTI-CHANNEL



It is often the case that no transformation can align all integrand peaks to the chosen axes.

MULTI-CHANNEL



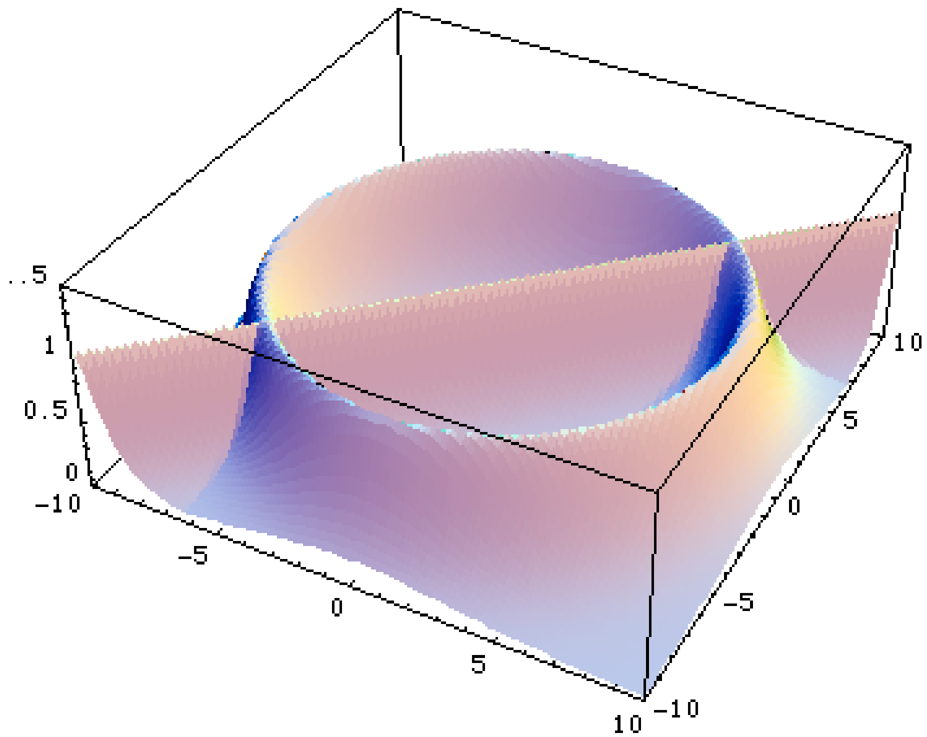
It is often the case that no transformation can align all integrand peaks to the chosen axes.

Solution: combine different transformations = channels

$$p(x) = \sum_{i=1}^{n} \alpha_i p_i(x) \qquad \text{with} \qquad \sum_{i=1}^{n} \alpha_i = 1$$

with each $p_i(x)$ taking care of one "peak" at the time

MULTI-CHANNEL

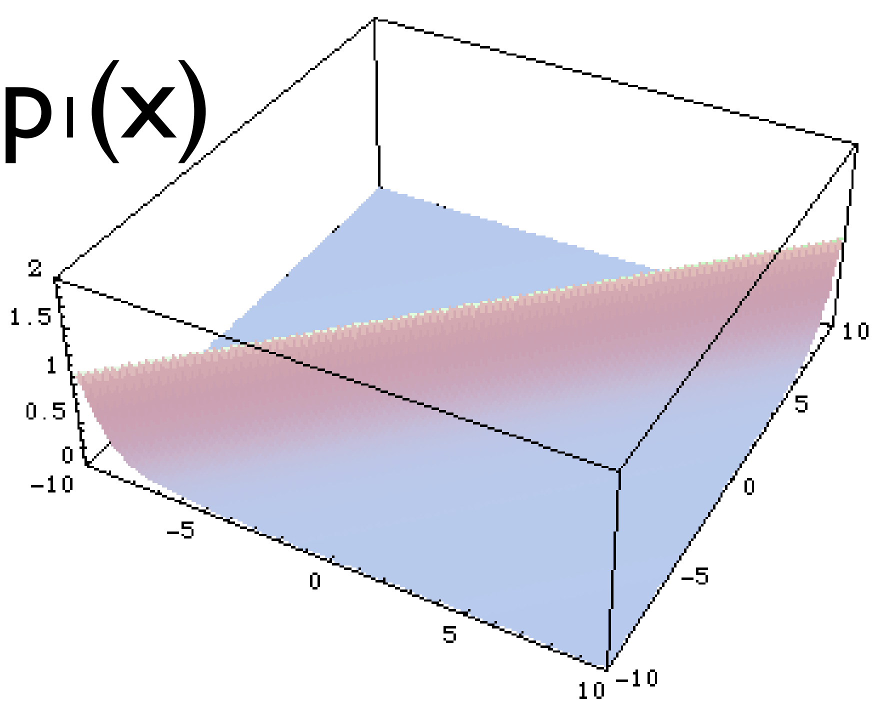


$$p(x) = \sum_{i=1}^{n} \alpha_i p_i(x)$$

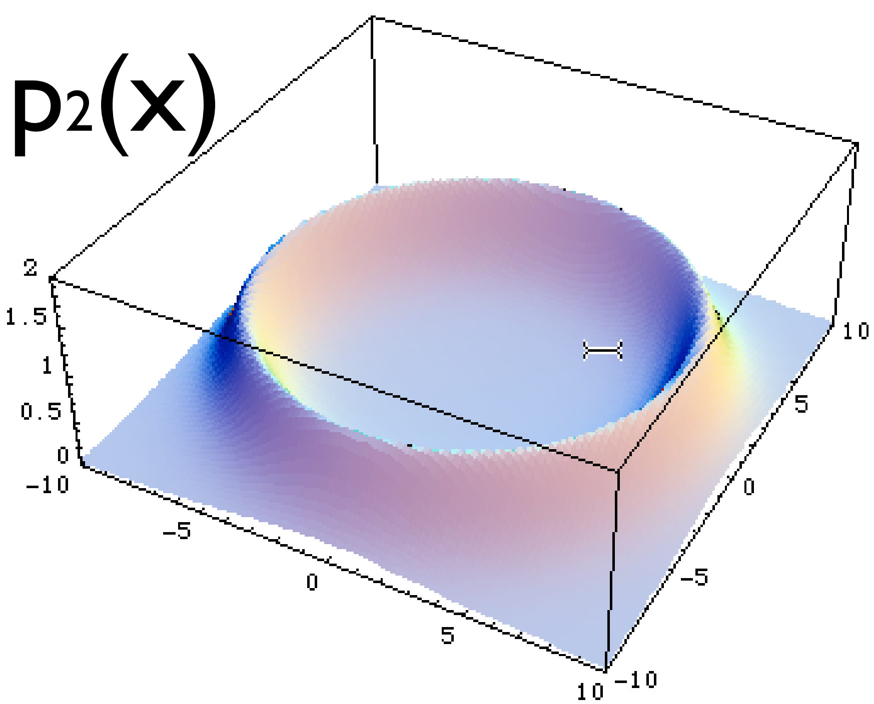
with

$$\sum_{i=1}^{n} \alpha_i = 1$$

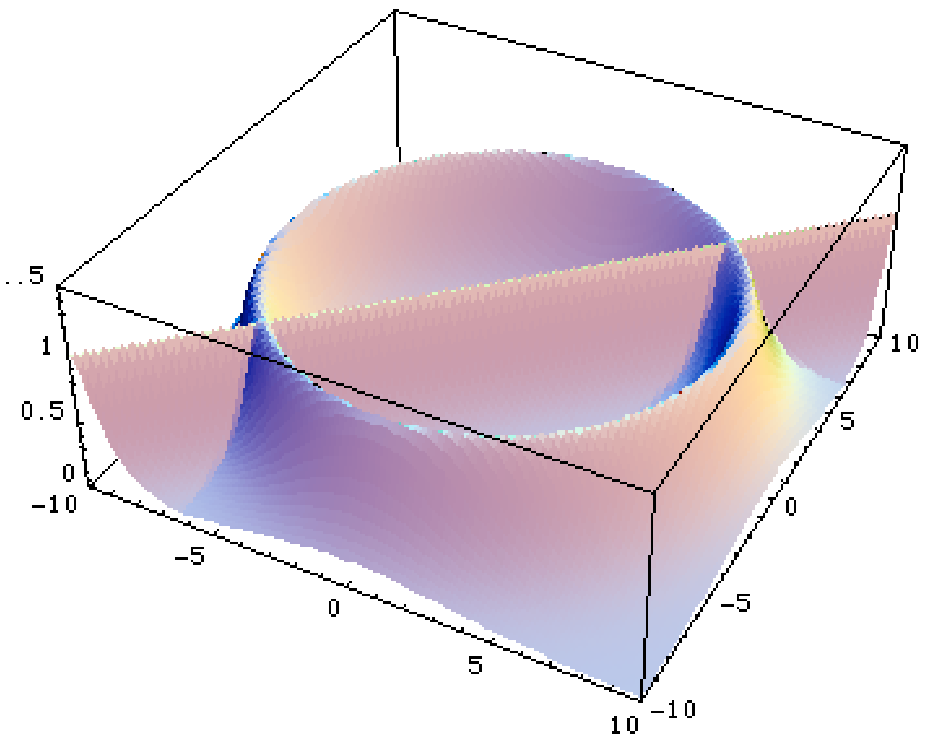
p1(x)



p2(x)



MULTI-CHANNEL



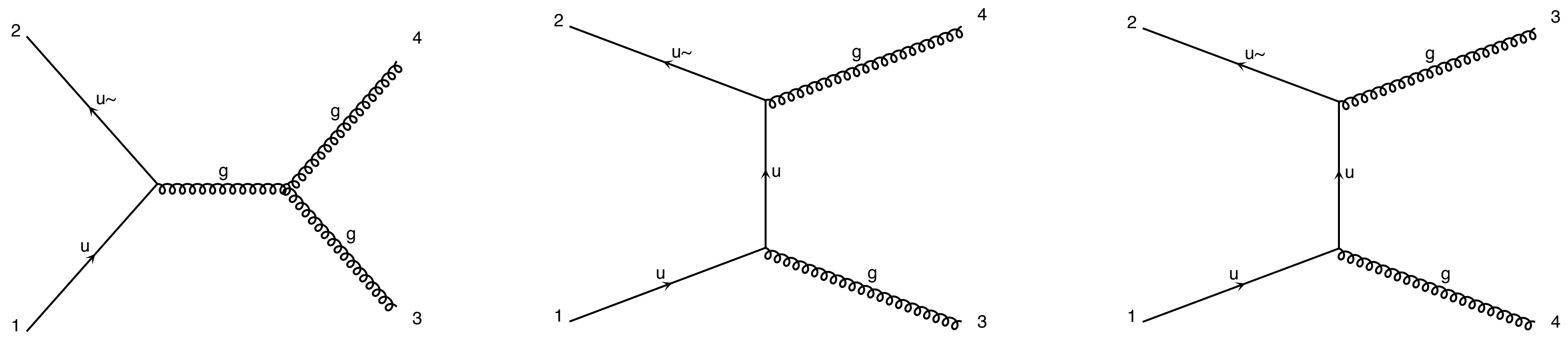
$$p(x) = \sum_{i=1}^{n} \alpha_i p_i(x)$$

with $$\sum_{i=1}^{n} \alpha_i = 1$$

Then,

$$I = \int f(x)dx = \int \frac{f(x)}{p(x)} dx = \sum_{i=1}^{n} \alpha_i \int \underbrace{\boxed{\frac{f(x)}{p(x)}}}_{\approx 1} p_i(x) dx$$

Example on a 2>2 process



$$\propto \frac{1}{\hat{s}} = \frac{1}{(p_1 + p_2)^2} \qquad \propto \frac{1}{\hat{t}} = \frac{1}{(p_1 - p_3)^2} \qquad \propto \frac{1}{\hat{u}} = \frac{1}{(p_1 - p_4)^2}$$

Three very different pole structures contributing to the same matrix element

Diagram-based multi-channels

[Madevent hep-ph/0208156]

$$\int |M_{tot}|^2 = \int \frac{\sum_i |M_i|^2}{\sum_j |M_j|^2} |M_{tot}|^2 = \sum_i \int \frac{|M_i|^2}{\boxed{\sum_j |M_j|^2} } \boxed{|M_{tot}|^2 \approx 1}$$

Key Idea

- Any single diagram is "easy" to integrate (pole structures/ suitable integration variables known from the propagators)
- Divide integration into pieces, based on diagrams
- All other peaks taken care of by denominator sum
- Similar idea possible using jacobians of parametrisation.

[RACOON hep-ph/9912261]

N Integral

- Errors add in quadrature so no extra cost
- "Weight" functions already calculated during $|M|^2$ calculation
- Parallel in nature

Diagram-based multi-channels

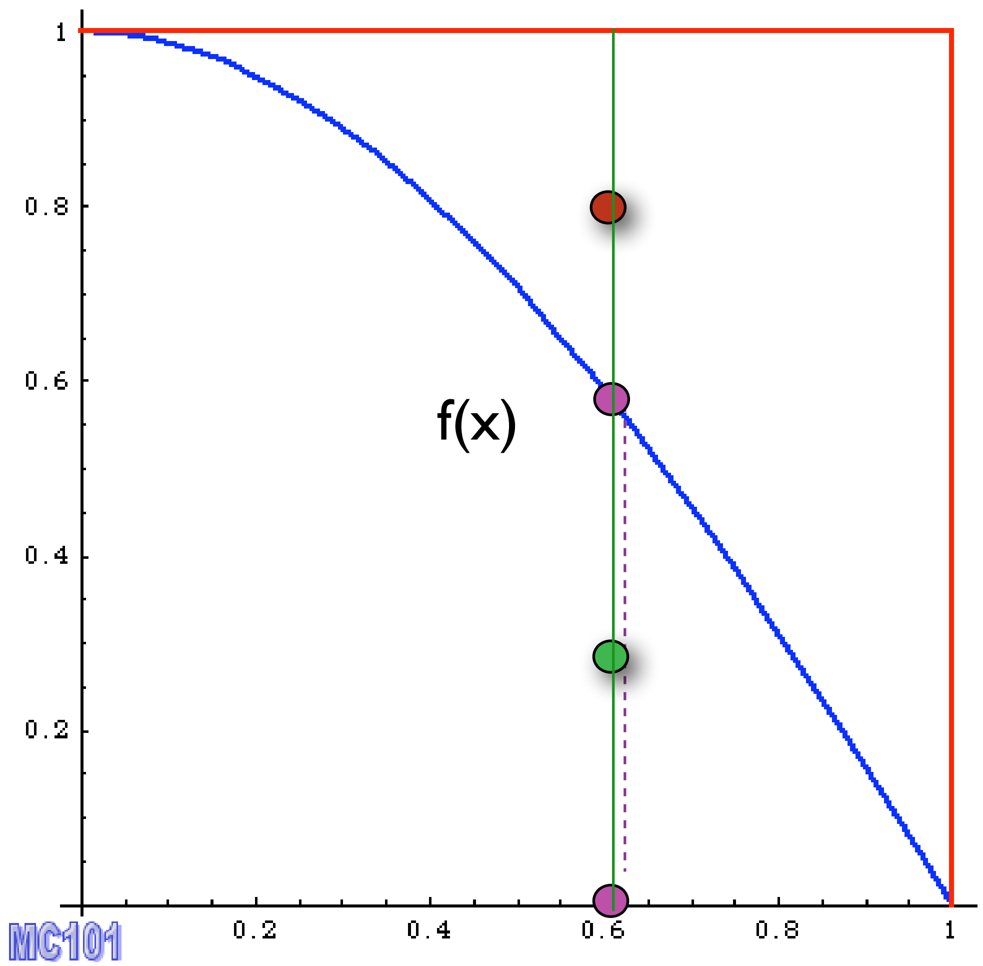
[Madevent hep-ph/0208156]

$$\int |M_{tot}|^2 = \int \frac{\sum_i |M_i|^2}{\sum_j |M_j|^2} |M_{tot}|^2 = \sum_i \int \frac{|M_i|^2}{\boxed{\sum_j |M_j|^2}} \boxed{|M_{tot}|^2} \approx 1$$

Drawbacks

- Cannot account for the **cuts imposed** by the observable.
- For higher-order computations, it cannot reproduce the intricate shape of **IR-subtracted real-emission ME**.
- Fails badly when computing **interferences** only.
- Necessitates **access** to the single squared **topologies** $|M_j|^2$
- Multi-channel+VEGAS **variance reduction saturates** quickly

Event generation

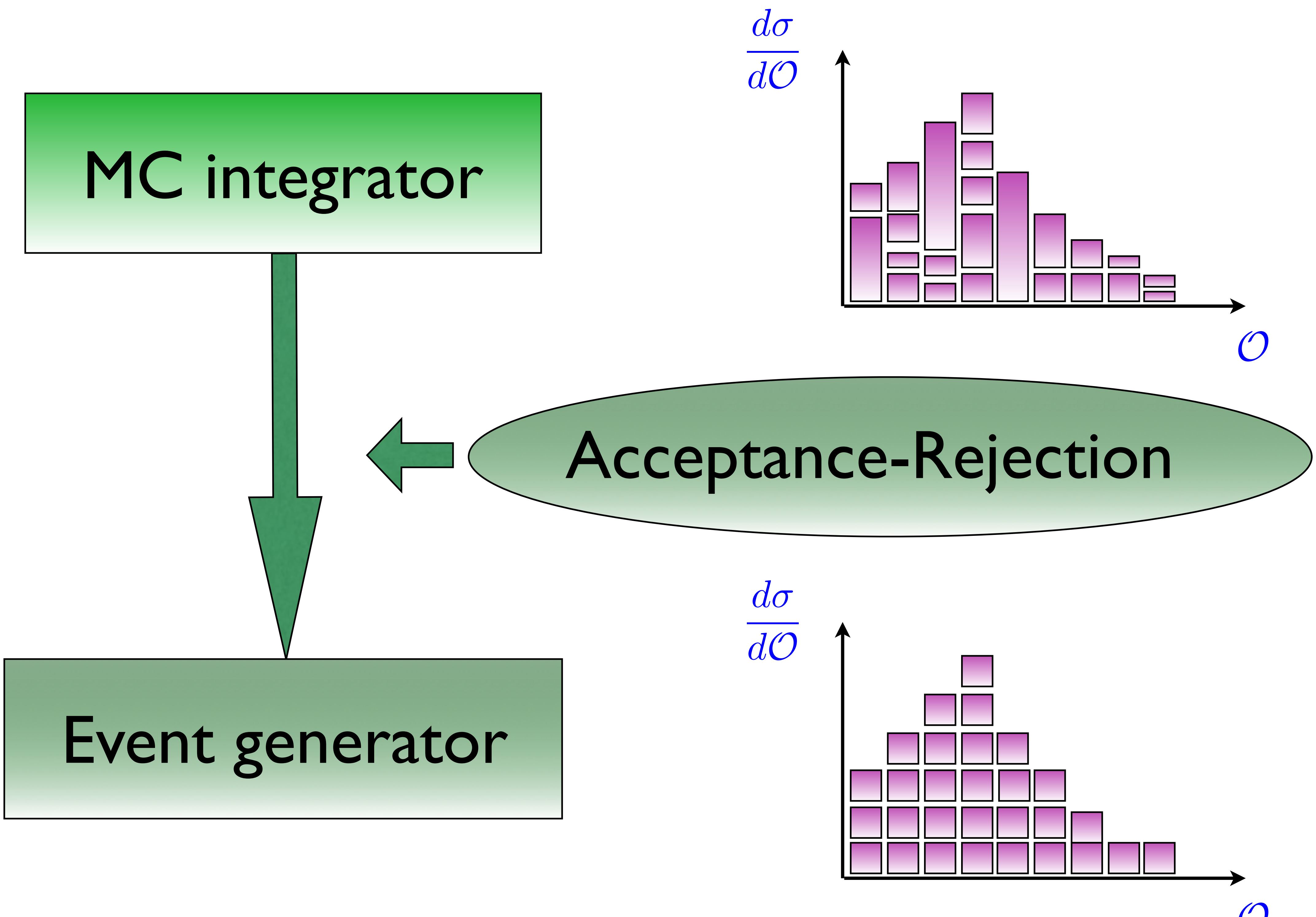


1. pick x
2. calculate f(x)
3. pick 0<y<fmax
4. Compare:
   if f(x)>y accept event,

   else reject it.

$$I = \frac{\text{accepted}}{\text{total tries}} = \text{efficiency}$$

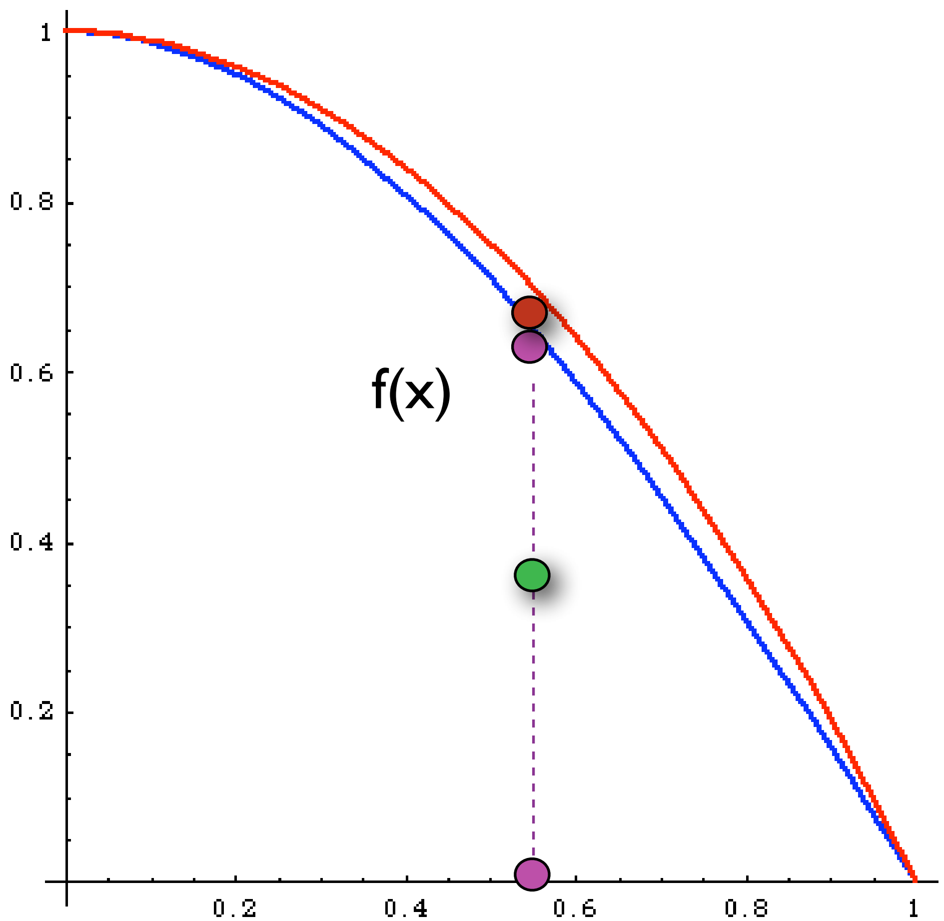
From integration to generation



This is possible only if f(x)<∞ and has definite sign

Improving unweighting efficiency

By combining it with importance sampling:



1. pick x distributed as p(x)
2. calculate f(x) and p(x)
3. pick 0<y<1
4. Compare:
   if f(x)>y p(x) accept event,
   else reject it.

Plan

- Type of integrals appearing in HEP
- Standard phase-space integration techniques
- Current numerical methods for loop integrals
- Machine learning assisted MC-integration
- Prospects

Plan

- Type of integrals appearing in HEP
- Standard phase-space integration techniques
- Current numerical methods for loop integrals
- Machine learning assisted MC-integration
- Prospects

Numerical loop integration

$$\int d^d k_1 \ldots d^d k_{N_l} \frac{N(\{k_i\})}{\Pi_1^{N_d}(q_i^2 - m_i^2 + i\delta)}$$

Using Feynman parameters and sector decomposition

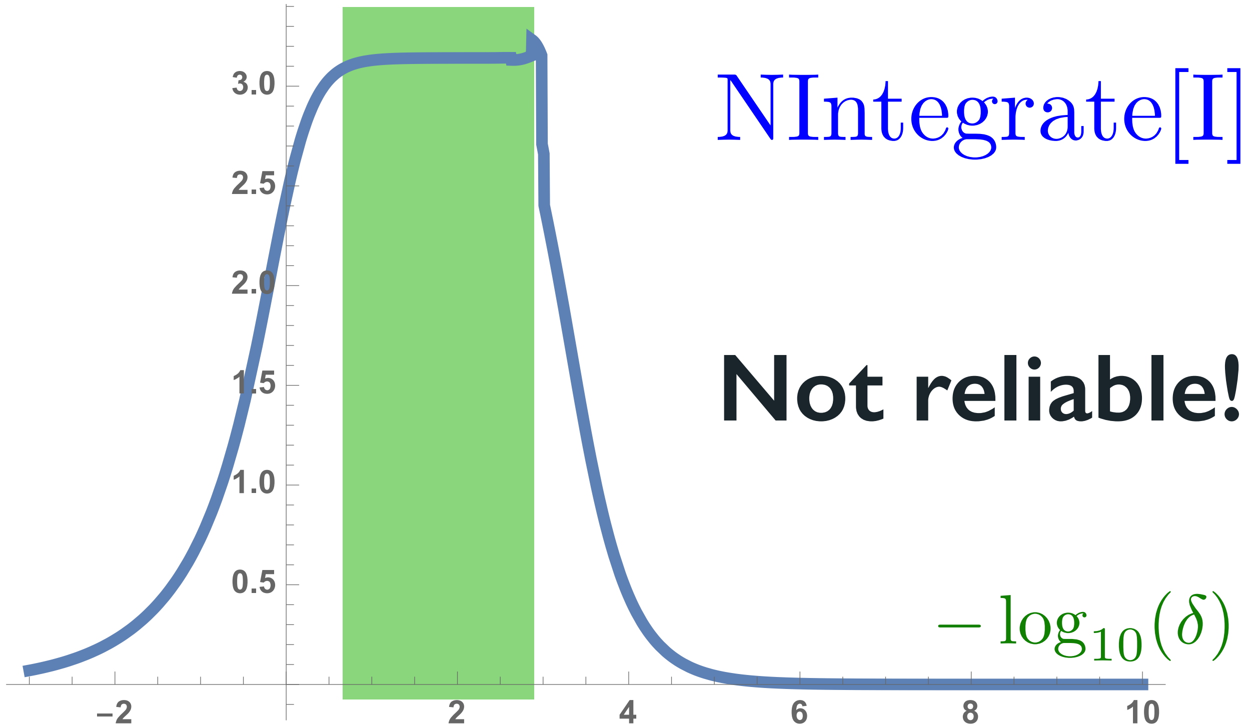
[PySecDec arXiv:1703.09692] [FIESTA arXiv:1511.03614] (not discussed here)

Direct integration in momentum space:

Naive approach inapplicable even for finite integrals in d=4:

$$\int_{-\infty}^{\infty} dx \frac{i}{x^2 - m^2 + i\delta} = \pi$$

Setting $m^2 = 1$ and integrating using a finite δ yields:



Numerical loop integration

$$\int d^d k_1 \ldots d^d k_{N_l} \frac{N(\{k_i\})}{\Pi_1^{N_d}(q_i^2 - m_i^2 + i\delta)}$$

Using Feynman parameters and sector decomposition

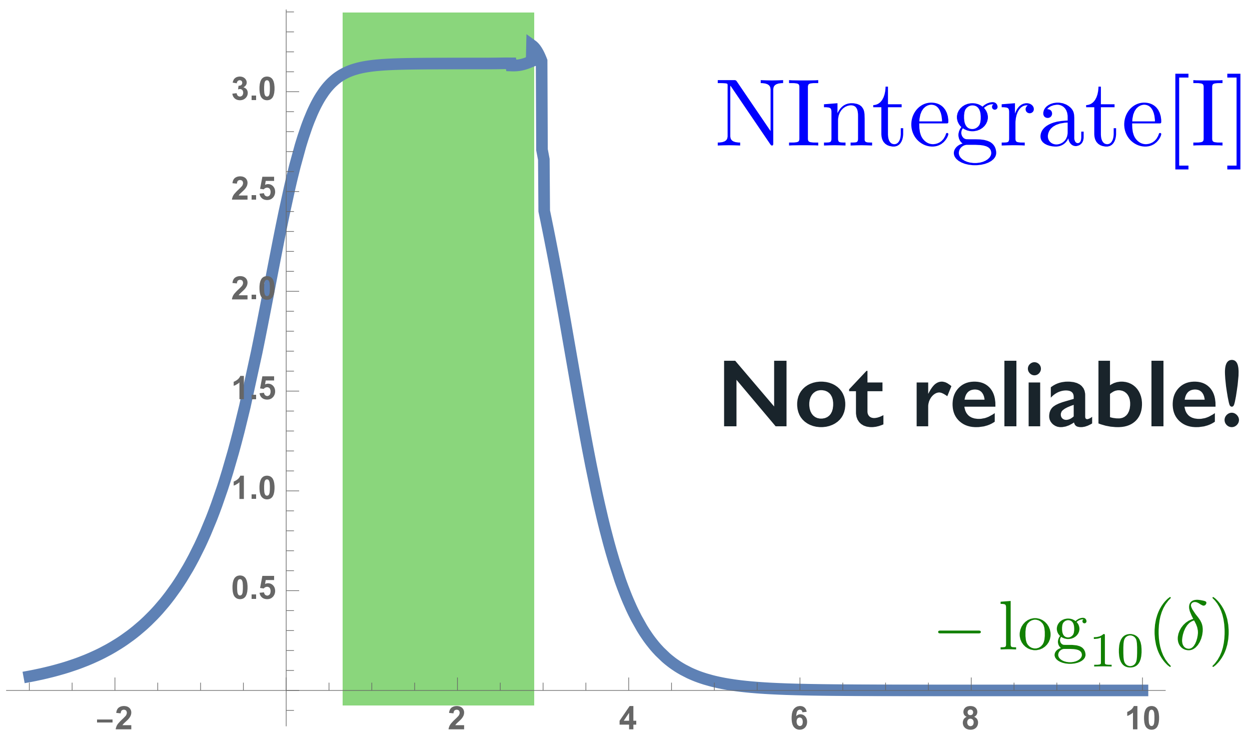
[PySecDec arXiv:1703.09692] [FIESTA arXiv:1511.03614] (not discussed here)

Direct integration in momentum space:

Naive approach inapplicable even for finite integrals in d=4:

$$\int_{-\infty}^{\infty} dx \frac{i}{x^2 - m^2 + i\delta} = \pi$$

Setting $m^2 = 1$ and integrating using a finite δ yields:



NIntegrate[I]

Not reliable!

Numerical loop integration in momentum space

An **exact** numerical representation can be obtained using analytical continuation and integral along a complex contour

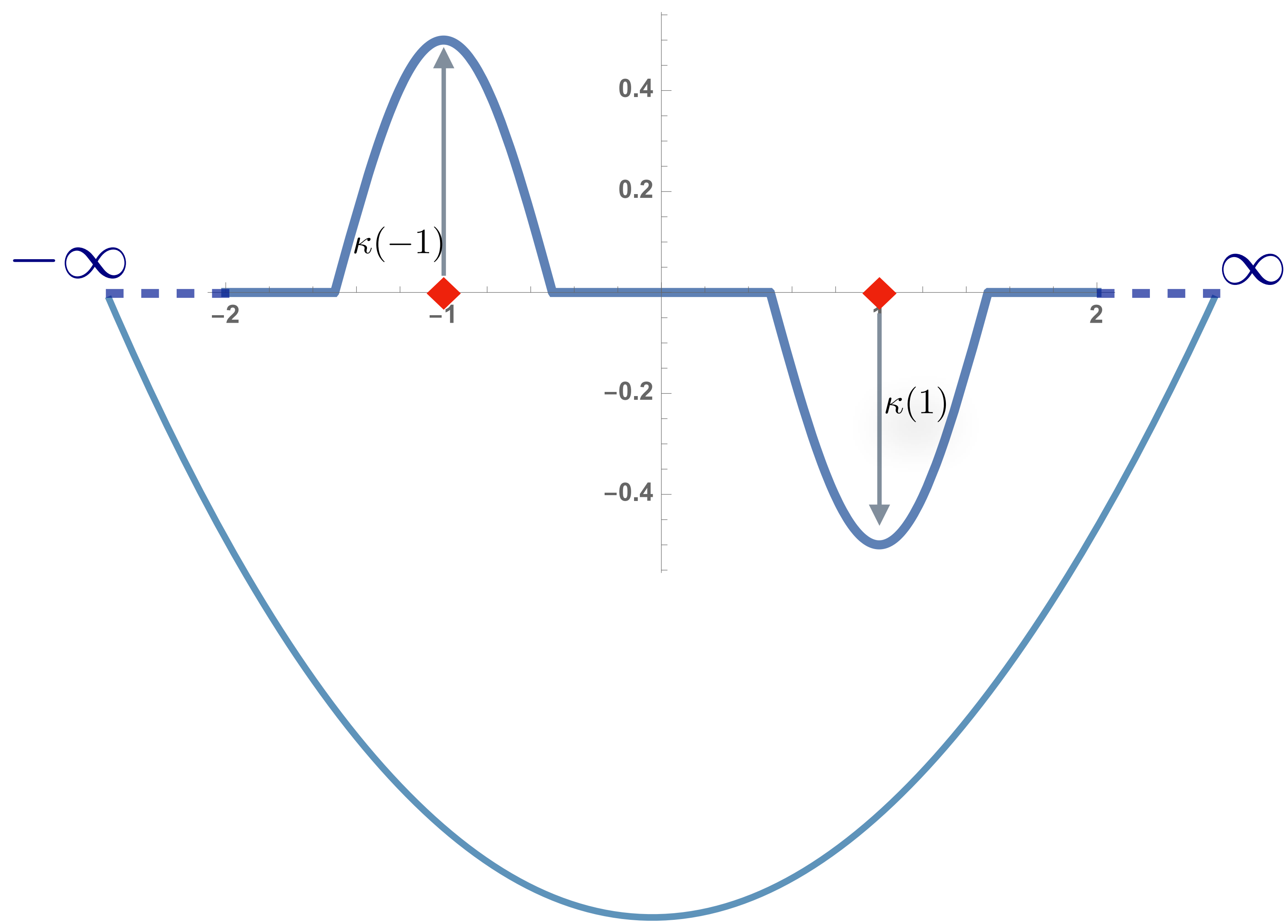
$$\int_{-\infty}^{\infty} dx I[x]$$

$$\downarrow$$

$$\bar{x} = x + i\kappa(x)$$

$$\downarrow$$

$$\oint_{-\infty}^{\infty} dx \frac{d\bar{x}}{dx} I[\bar{x}(x)]$$

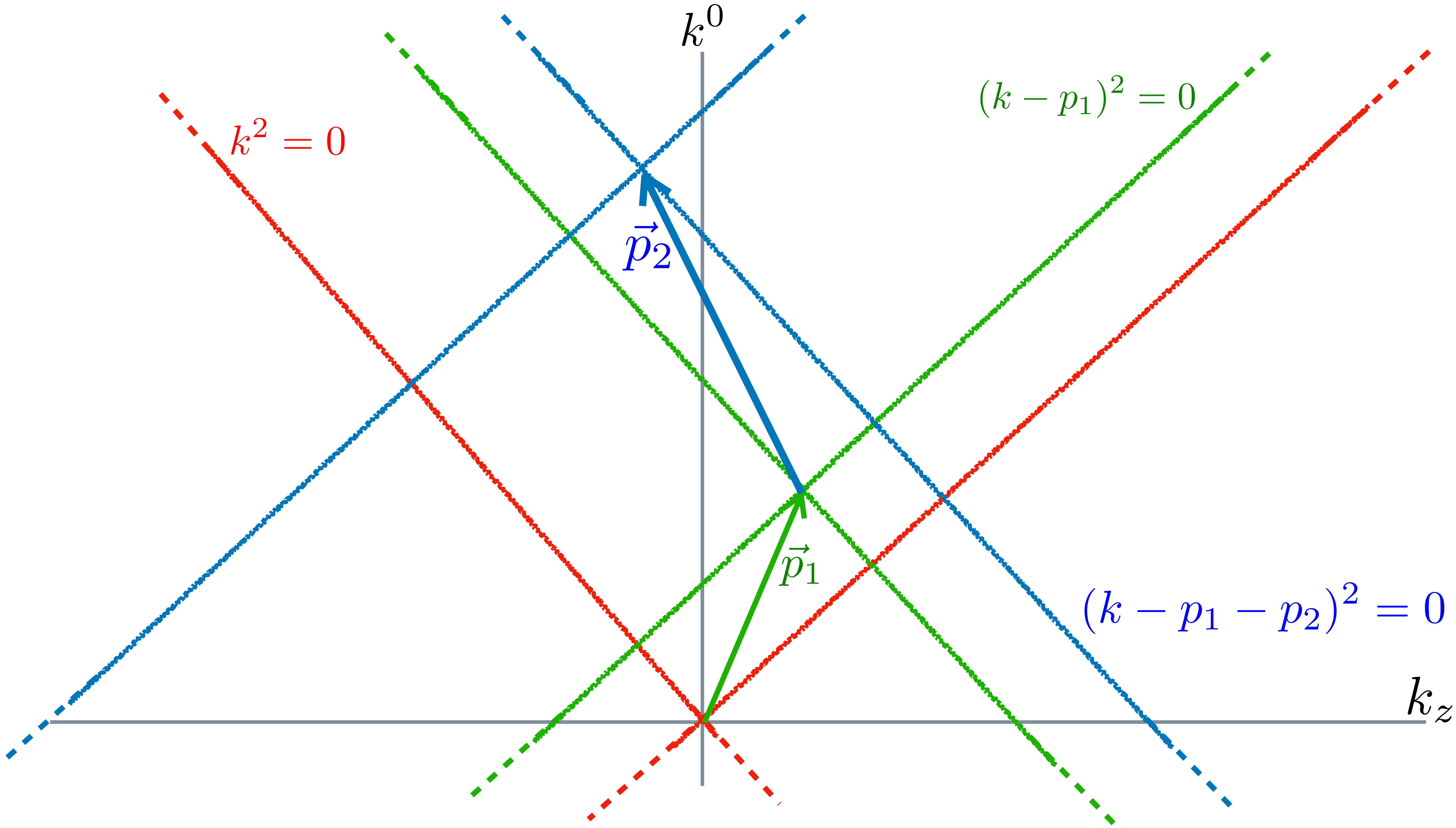


Complex loop deformation

Quickly becomes very intricate

$$\boxed{\bar{k}^\mu(k) \to k^\mu + i\kappa^\mu(k)}$$

$$\int d^4k \frac{1}{k^2 (k-p_1)^2 (k-p_1-p_2)^2}$$

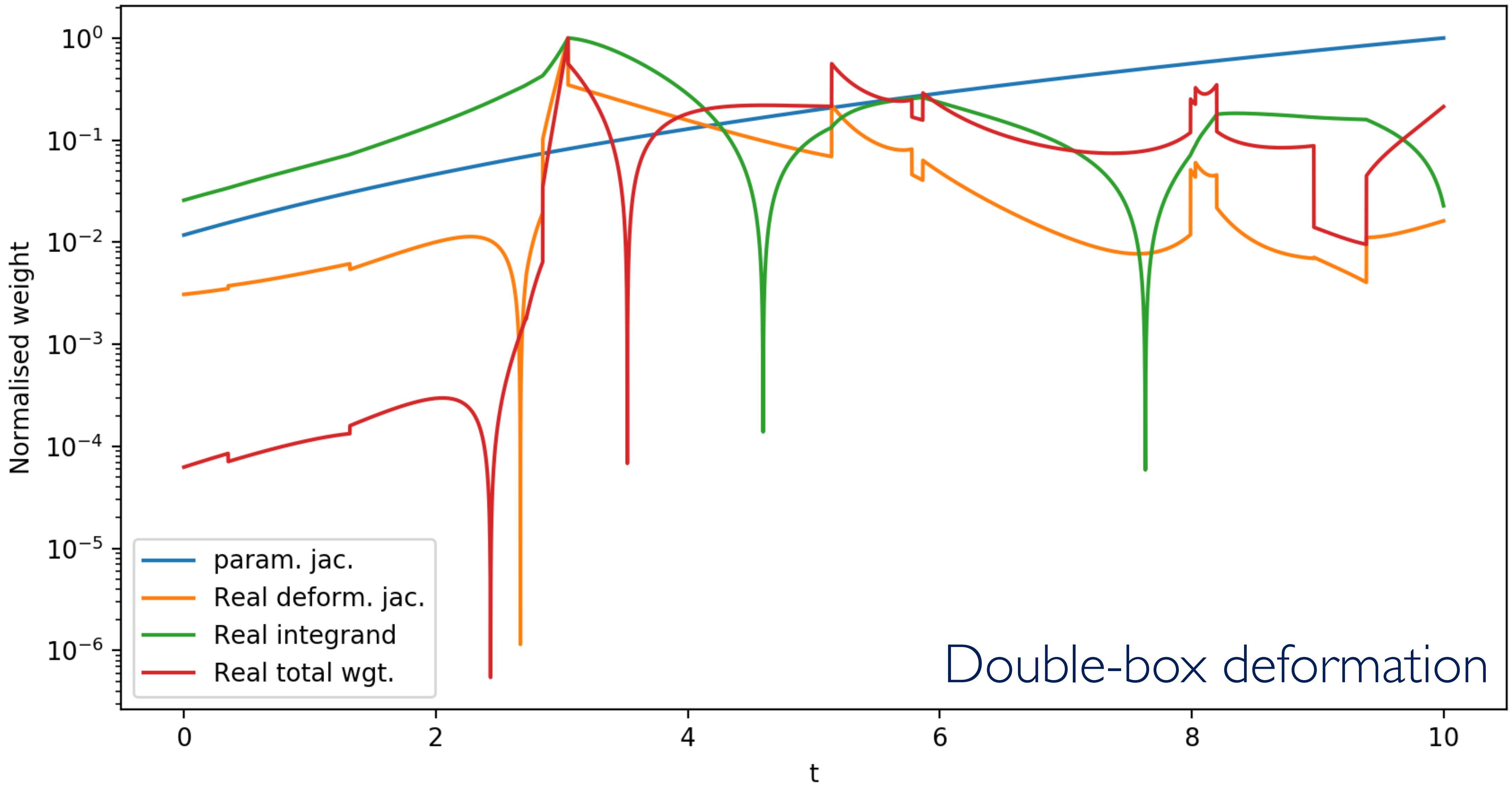


Deformation required along a specific direction for each coloured surface.

Complex loop deformation

Valid deformation in $4N_{loop}$ - dimensional Minkowski space:

[Weinzierl & al., arxiv:1211.0509] , applicable to arbitrary loop count

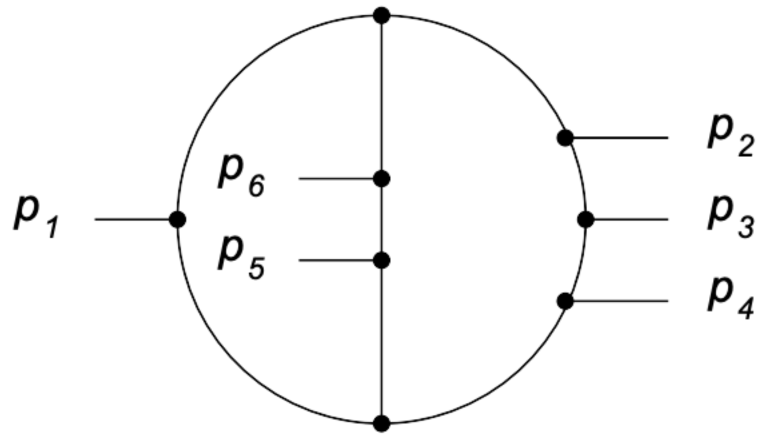


Complicated (non-C1) deformation yields an **integrand with large variance**

Still, typically fast integrands, $\mathcal{O}(50\mu s)$.

Complex loop deformation

- Rather imprecise results (slow convergence) but for impressive topologies!
  [Weinzierl & al., arxiv:1211.0509]



(e)

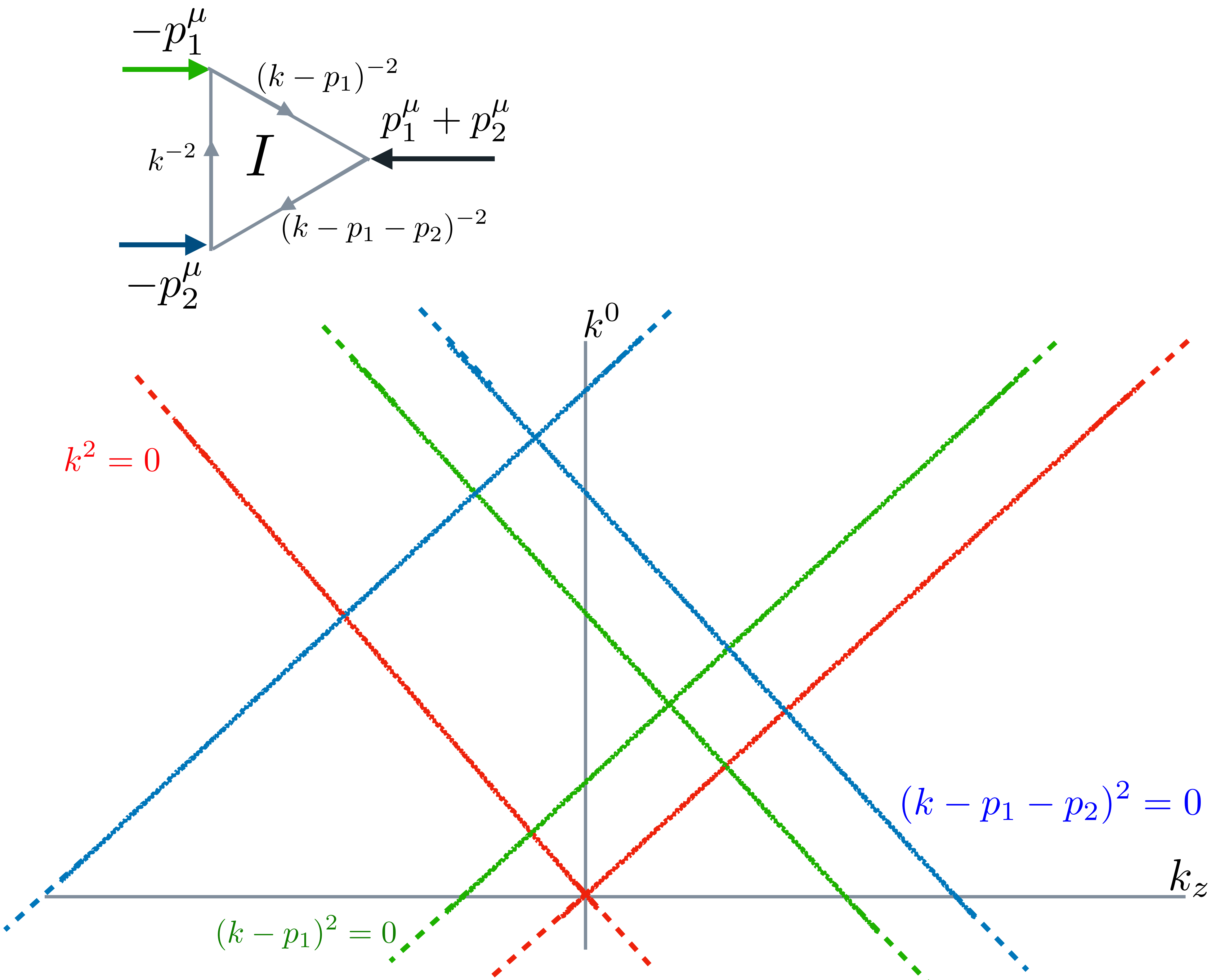
$$\text{(e) } [\text{GeV}^{-10}] \;\Big|\; (-4.64 \pm 0.08) \cdot 10^{-19}$$

- Note: Integrand is a product, so multi-channeling methods are rather inefficient.
- **Integrand-level counterterms** can be devised for **divergent integrals**:
  [Weinzierl & al., arxiv:1112.3521] [Anastasiou & al., arxiv:1812.03753]
- Recently, the focus shifted to an alternative formulation of direct integration in momentum space: **Loop-Tree duality**.
- It consists in analytically **integrating over the loop momenta energies** using **Cauchy's theorem**, resulting in an integration in a cartesian product of **3D euclidian spaces**, with a **simplified deformation**.

Loop-Tree Duality (LTD)

Analytically integrate over the loop energies using Cauchy's theorem.

[Catani & al., arxiv:0804.3170]

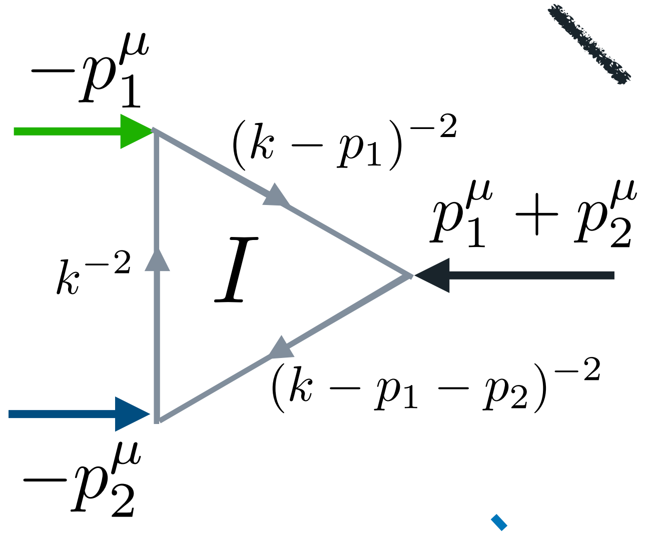


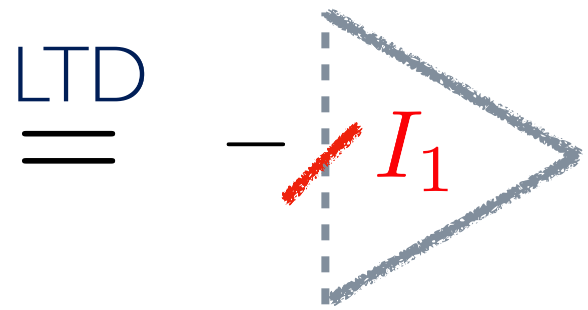
Loop-Tree Duality (LTD)

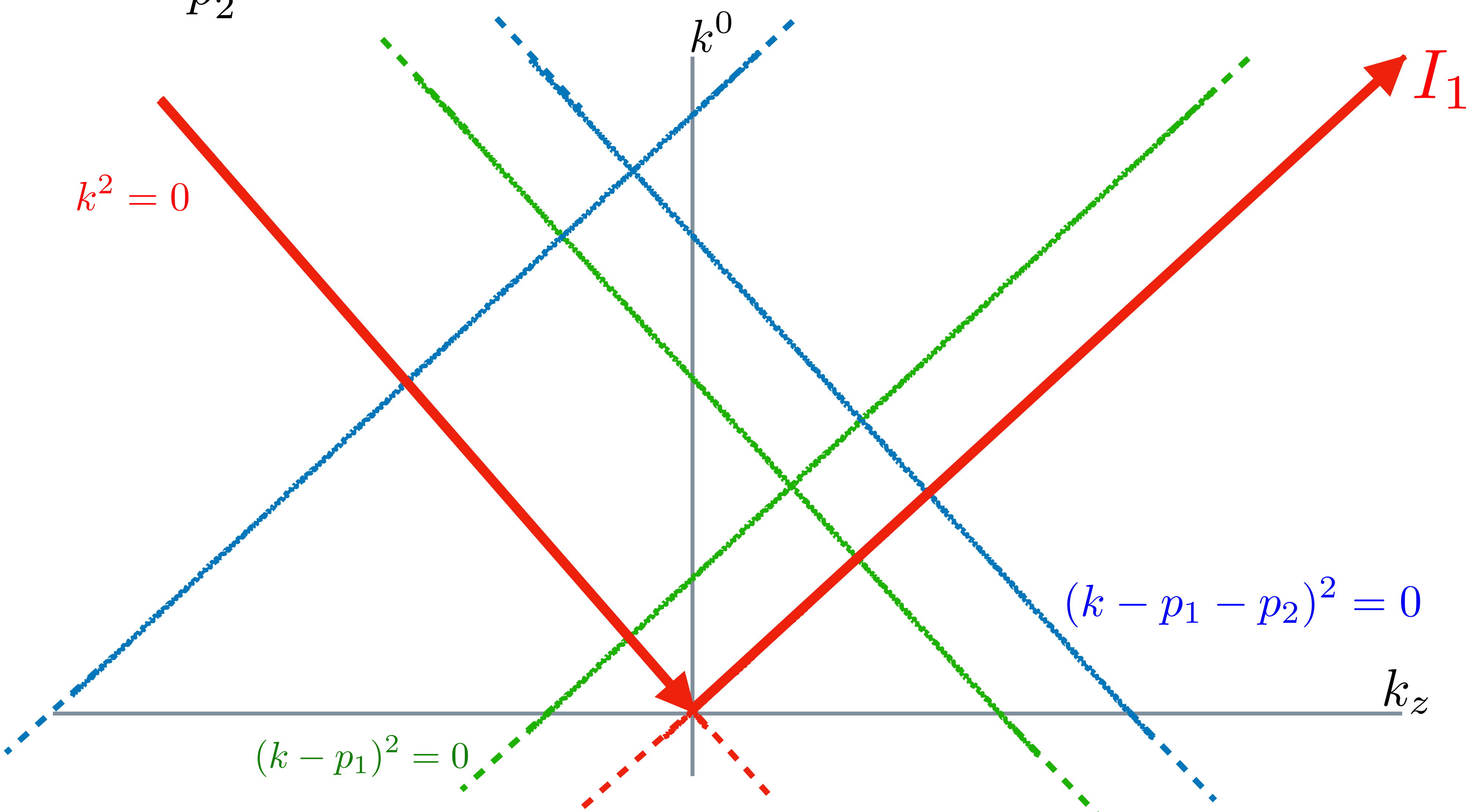
Analytically integrate over the loop energies using Cauchy's theorem.

[Catani & al., arxiv:0804.3170]

$$\equiv \delta(q_i^2)\Theta(q_i^0)$$





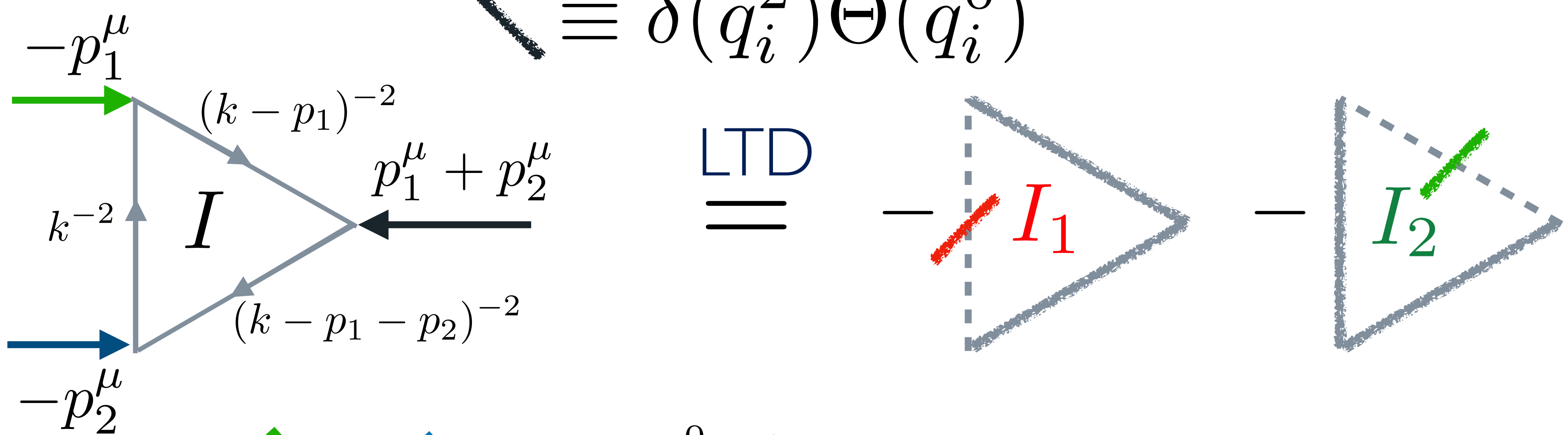


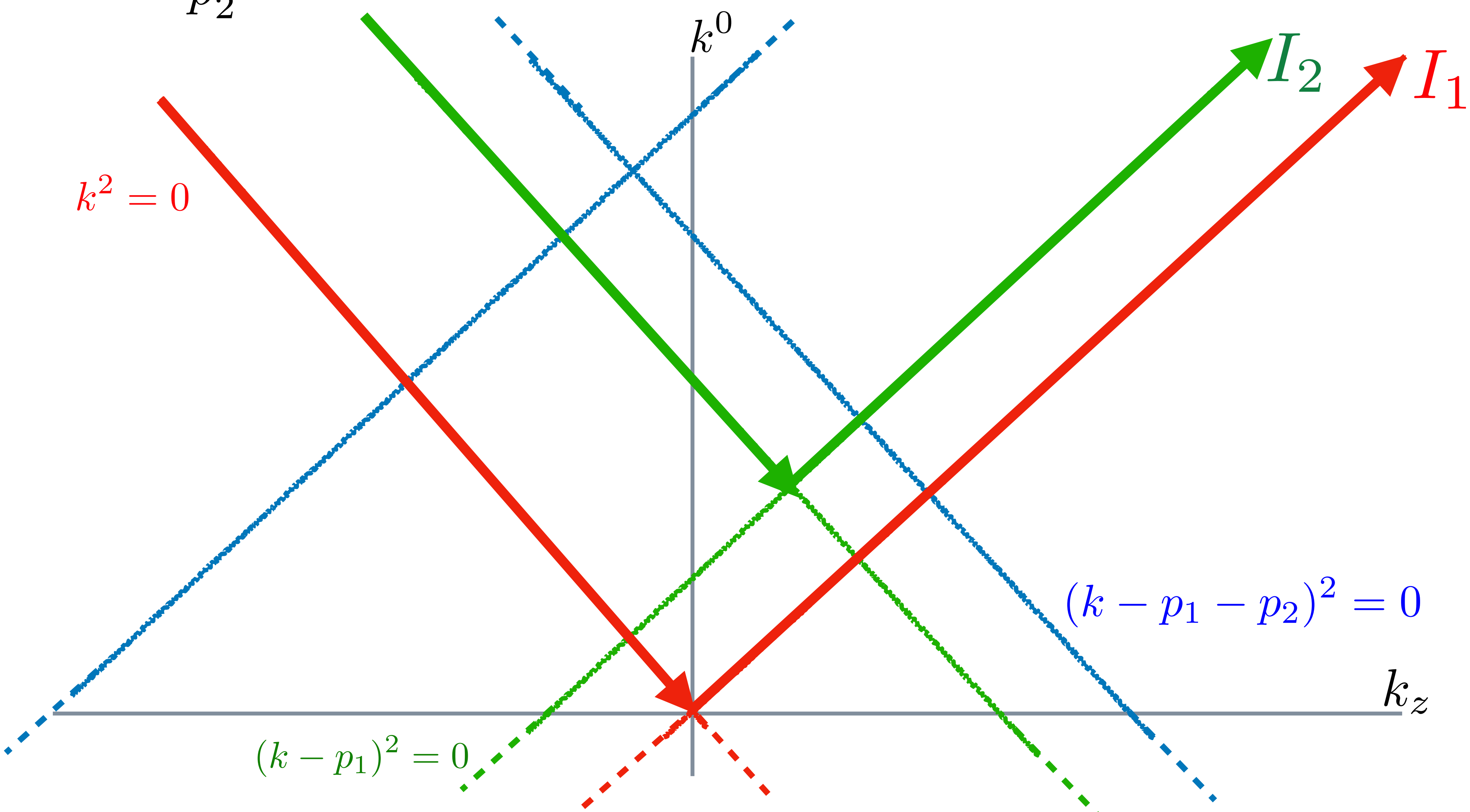
Loop-Tree Duality (LTD)

Analytically integrate over the loop energies using Cauchy's theorem.

[Catani & al., arxiv:0804.3170]

$$\equiv \delta(q_i^2)\Theta(q_i^0)$$

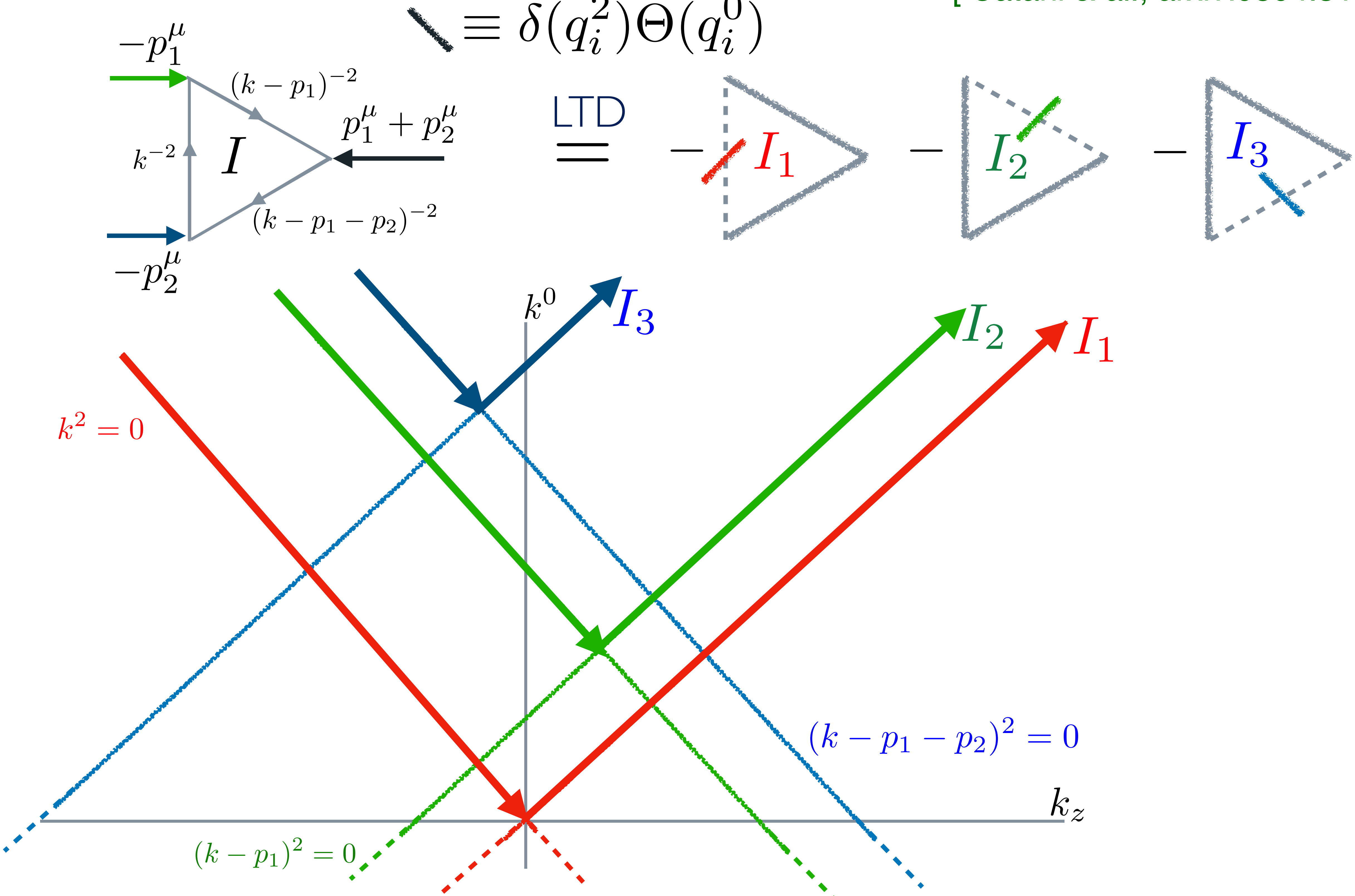




Loop-Tree Duality (LTD)

Analytically integrate over the loop energies using Cauchy's theorem.

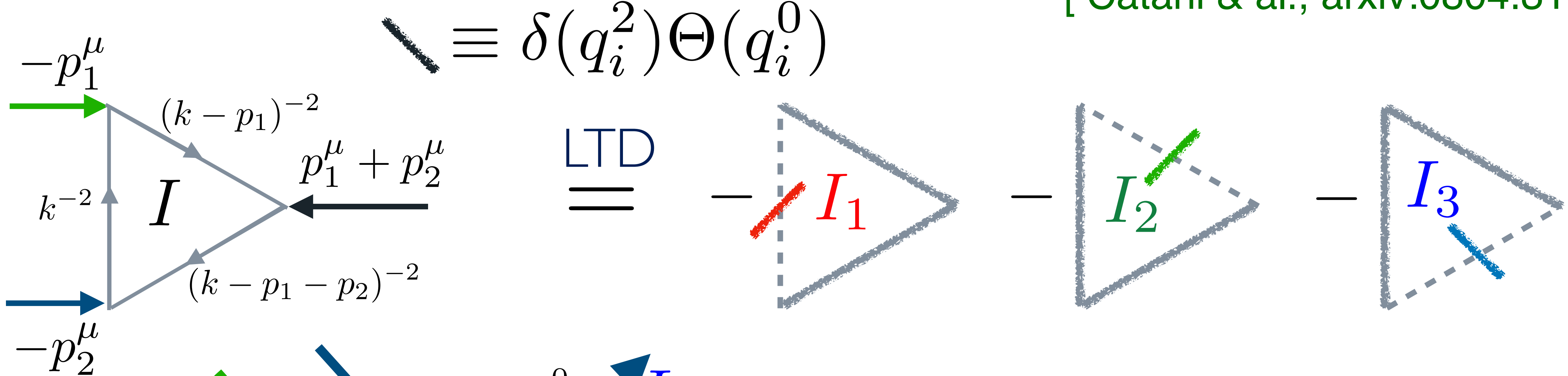
[Catani & al., arxiv:0804.3170]

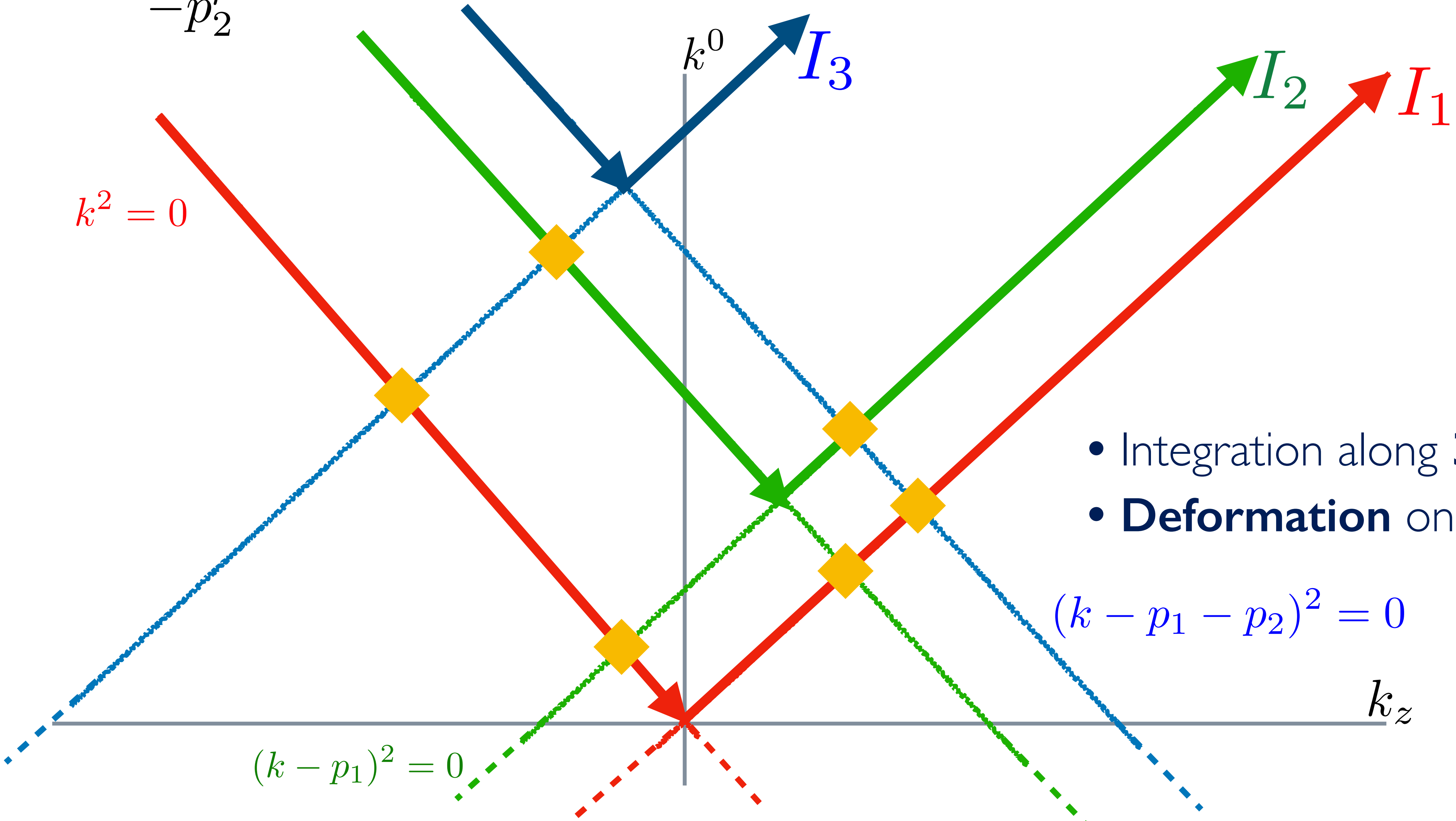


Loop-Tree Duality (LTD)

Analytically integrate over the loop energies using Cauchy's theorem.

[Catani & al., arxiv:0804.3170]



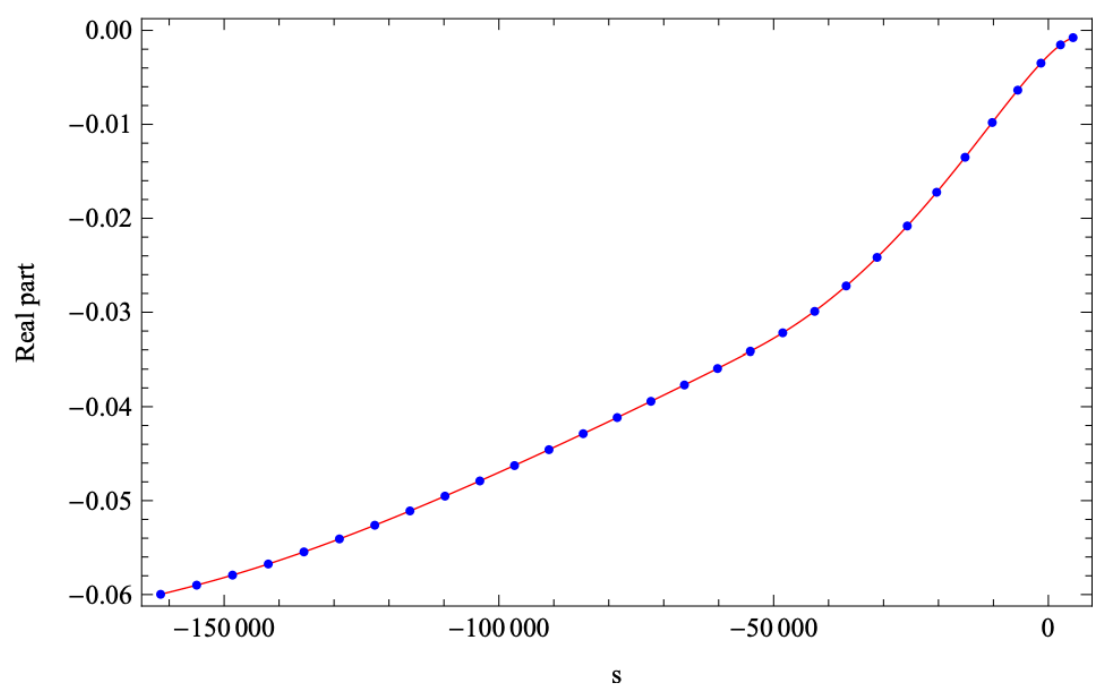


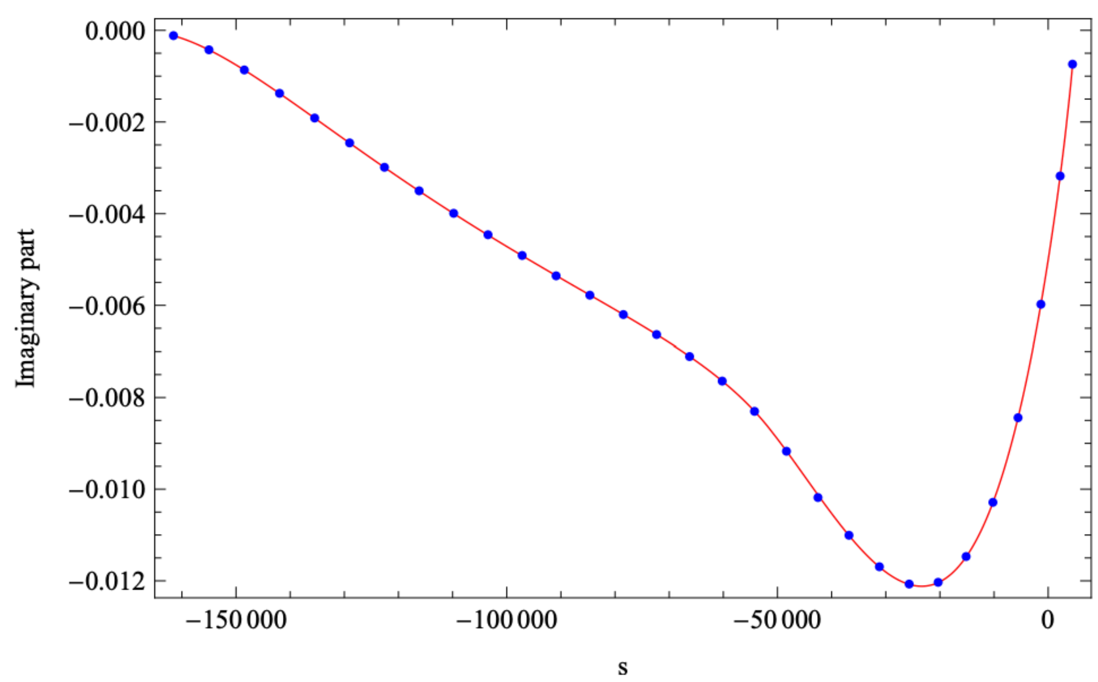
- Integration along **3D euclidian** thick lines
- **Deformation** only needed **on** ◆ !

Loop-Tree Duality (LTD)

Promising results for **one-loop finite integrals**: [Rodrigo & al., arxiv:1510.00187]

	Rank	Tensor Pentagon	Real Part	Imaginary Part	Time [s]
P19	3	LoopTools	-8.34718×10^{-2}	$+i\ 1.10217 \times 10^{-2}$	
		SecDec	$-8.33284(829) \times 10^{-2}$	$+i\ 1.10232(107) \times 10^{-2}$	1501
		LTD	$-8.34829(757) \times 10^{-2}$	$+i\ 1.10119(757) \times 10^{-2}$	38





- Formal LTD developments beyond one loop:

  [Bierenbaum & al., arxiv:1007.0213] [Weinzierl & al., arxiv:1902.02135]

- First two-loop numerical result for a full 3-point amplitude (no deformation though)

  $H \rightarrow \gamma\gamma$ @ $2-$loops [Rodrigo & al., arxiv:1901.09853]

Summary : integrals in HEP

Phase space integrals:

- Relative **high dimensionality** (10-20)
- Very **variable** integrand **runtime speed** (10 μs - 1 min)
- Prior knowledge of integrand
  - ➡ Possible multi-channeling and **parametrisation optimisation**
- **Differential** unweighted predictions are desirable (**cuts**).

Loop integrals:

- **Lower dimensionality** (< 10)
- **Fast** integrands (~100 μs)
- **Large variance** and little prior integrand knowledge
- Only **inclusive** result is of interest

PLAN

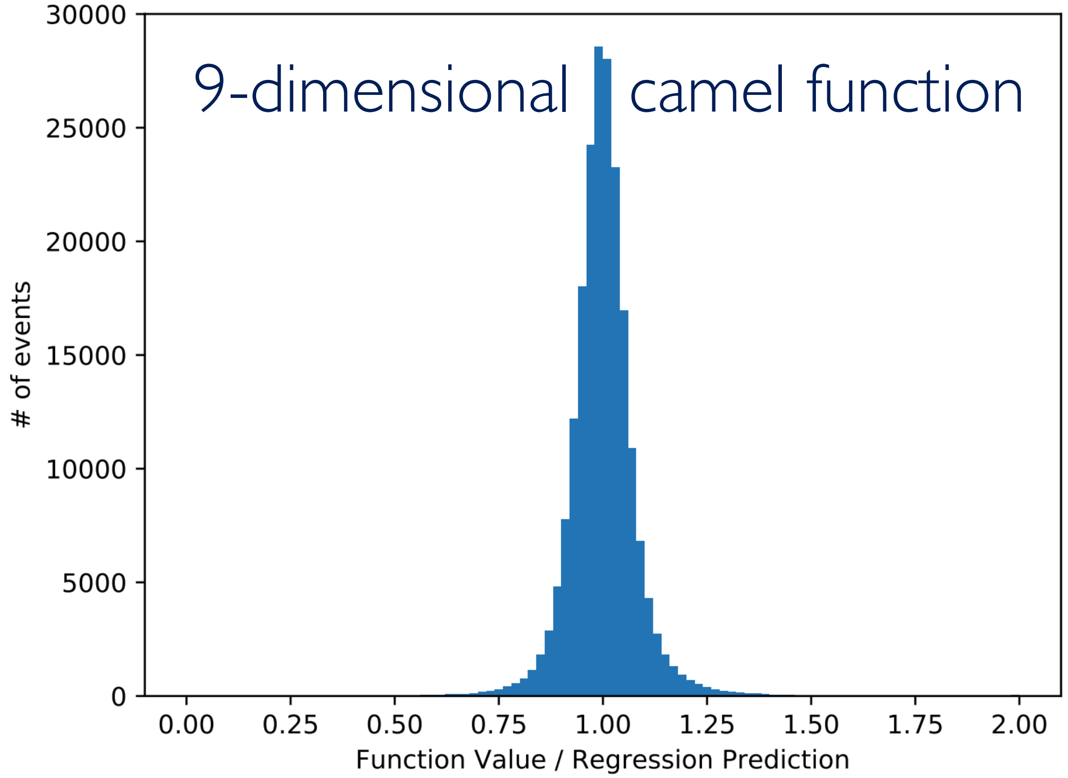
- Type of integrals appearing in HEP
- Standard phase-space integration techniques
- Current numerical methods for loop integrals
- Machine learning assisted MC-integration
- Prospects

Plan

- Type of integrals appearing in HEP
- Standard phase-space integration techniques
- Current numerical methods for loop integrals
- Machine learning assisted MC-integration
- Prospects

NN regression of integrand

First simple application, use NN to create a reliable integrand fit:

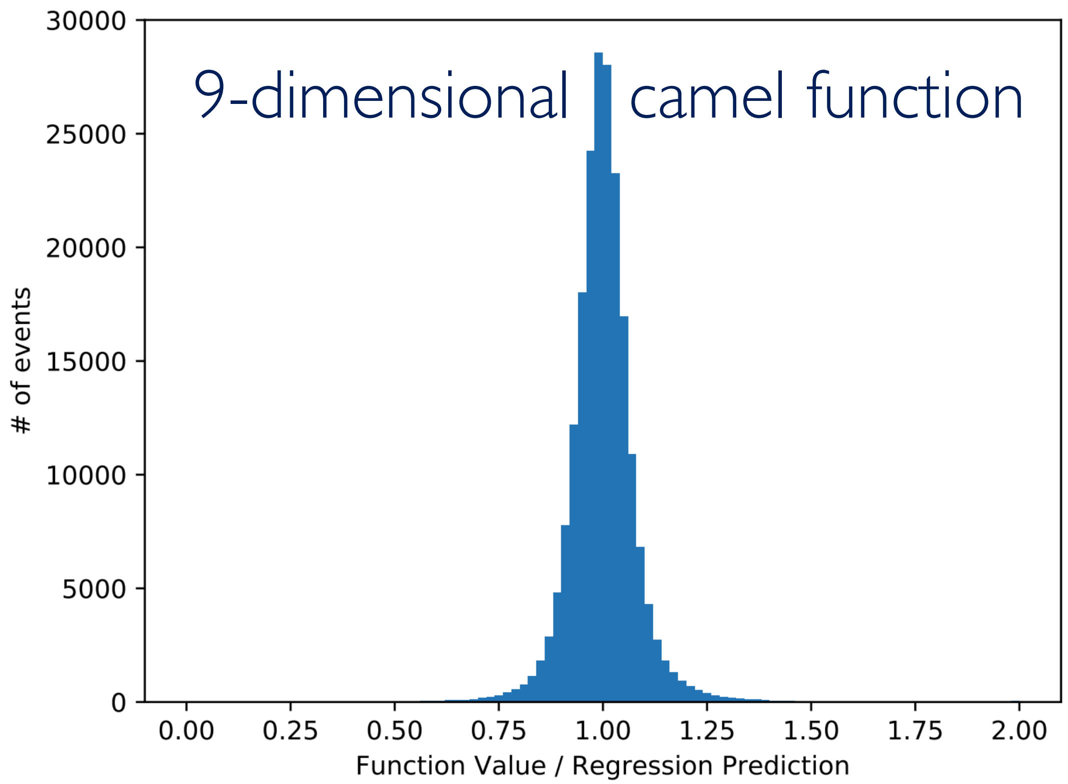


[Bendavid arxiv:1707.00028]

- Must be **checked** on **real-life MEs**.
- Observable **cuts** (theta functions) must be **smoothened**
- Many **hyperparameters** still need to be **tuned** for the particular case at hand.

NN REGRESSION OF INTEGRAND

First simple application, use NN to create a reliable integrand fit:



[Bendavid arxiv:1707.00028]

- Must be **checked** on **real-life MEs**.
- Observable **cuts** (theta functions) must be **smoothened**
- Many **hyperparameters** still need to be **tuned** for the particular case at hand.

If **NN inference much faster than integrand**, then we are basically done:

[Eq.2.102 arxiv:1405.0301]

$$\int d\phi_n \boxed{V} = \int d\phi_n \left[\boxed{\widetilde{V}_k} + \left(\boxed{V} - \boxed{\widetilde{V}_k} \right) \right]$$

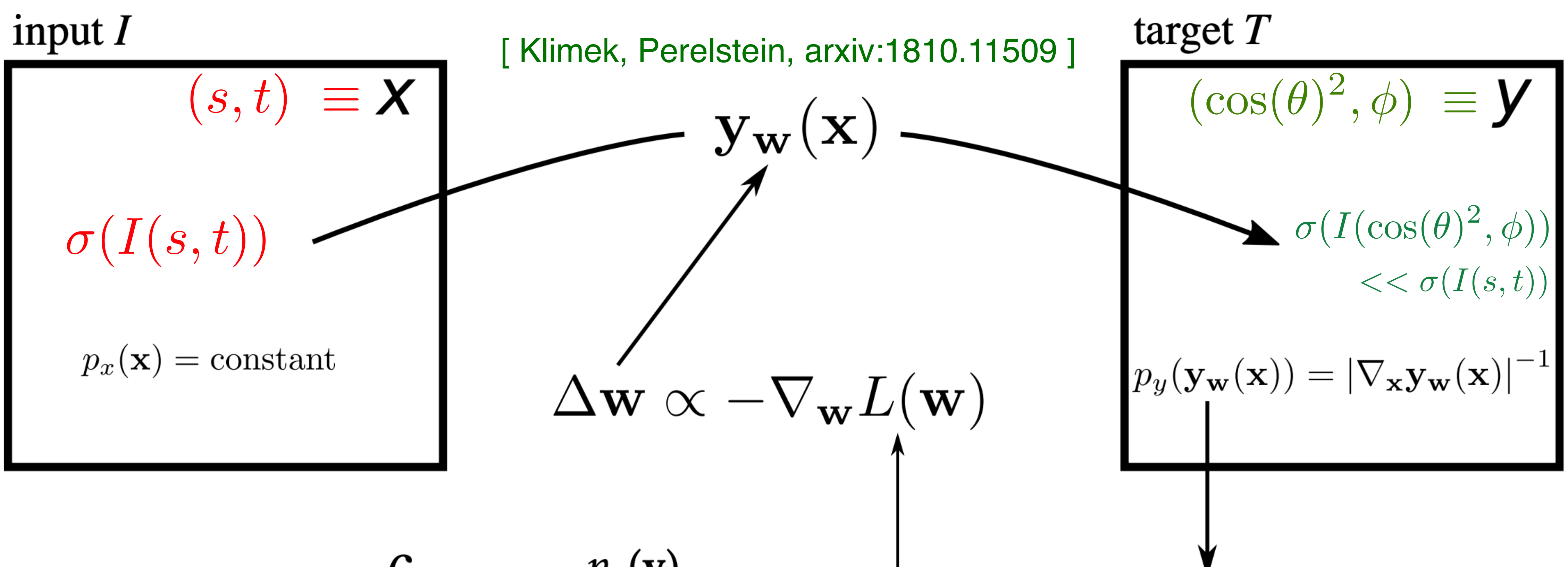
$V :=$ original integrand

$\tilde{V}_k :=$ NN approximant @ iteration k

$$I_k(V) = \frac{1}{p_k} \sum_{i=1}^{\boxed{p_k}} \Phi\left(\phi_n^{(k,i)}\right) \boxed{\widetilde{V}_k\left(\phi_n^{(k,i)}\right)} + \frac{1}{p_k f_k} \sum_{i=1}^{\boxed{p_k f_k}} \Phi\left(\phi_n^{(k,i)}\right) \left[\boxed{V\left(\phi_n^{(k,i)}\right)} - \boxed{\widetilde{V}_k\left(\phi_n^{(k,i)}\right)} \right]$$

NN Integration replacing VEGAS

If **integrand faster than NN inference** then all is not lost:



$$D_{\mathrm{KL}}[p_y(\mathbf{y}); f(\mathbf{y})] \equiv \int p_y(\mathbf{y}) \log \frac{p_y(\mathbf{y})}{f(\mathbf{y})}\, d\mathbf{y} \qquad L(\mathbf{w}) = D_{\mathrm{KL}}\left(p_y(\mathbf{y}_{\mathbf{w}}(\mathbf{x})); f(\mathbf{y}_{\mathbf{w}}(\mathbf{x}))\right)$$

Consider **a generative NN model** effectively learning a change of variables.

Contrary to **VEGAS**, it is not a piece-wise ansatz: no factorised approx.

Saturation of the **variance reduction** much delayed.

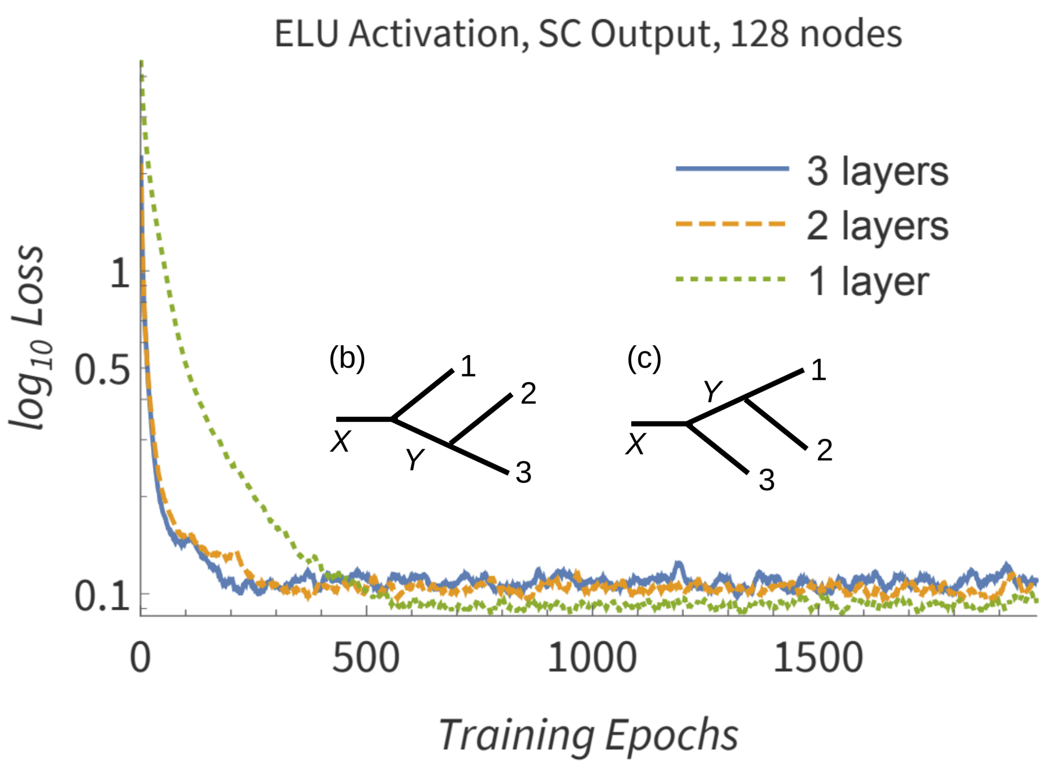
If perfectly trained, then V=0 and a single evaluation yields the exact integral.

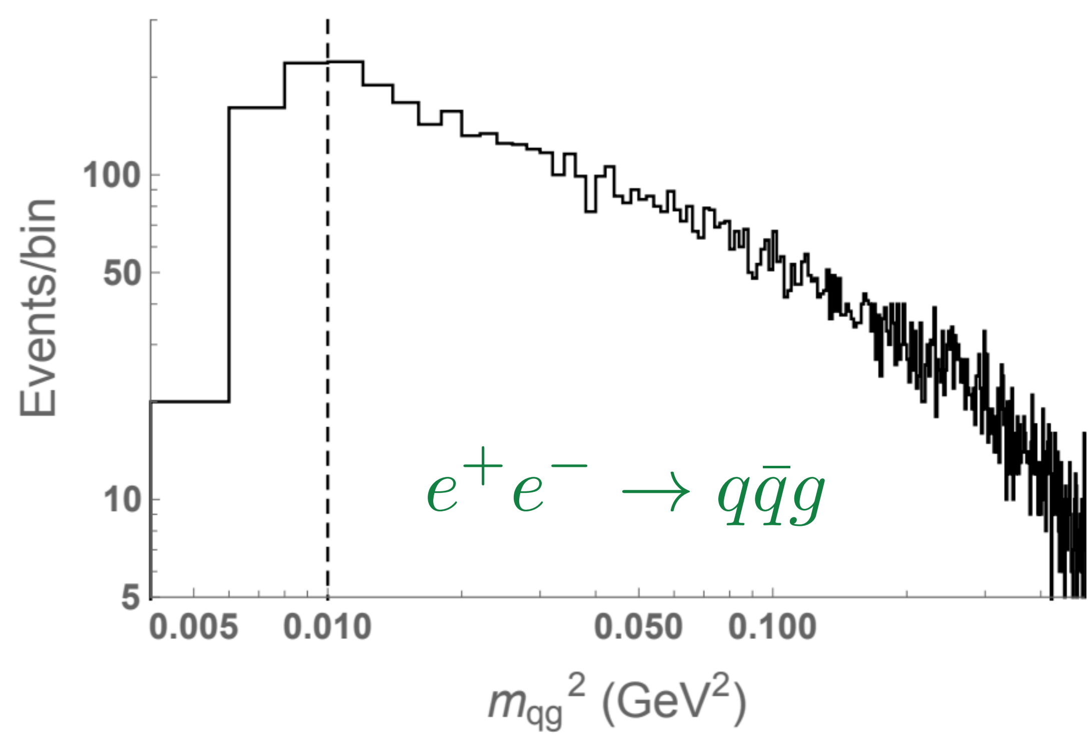
"NNVegas" Results

Algorithm	# of Func. Evals	$\sigma_w / < w >$	σ_I / I (2e6 add. evts)
VEGAS	1,500,000	19	$\pm 1.3 \times 10^{-2}$
Generative BDT	3,200,000	0.63	$\pm 4.5 \times 10^{-4}$
Generative BDT (staged)	3,200,000	0.31	$\pm 2.2 \times 10^{-4}$
Generative DNN	294,912	0.15	$\pm 1.1 \times 10^{-4}$
Generative DNN (staged)	294,912	0.081	$\pm 5.7 \times 10^{-5}$

9-dimensional camel function

[Bendavid arxiv:1707.00028]





[Klimek, Perelstein, arxiv:1810.11509]

Plan

- Type of integrals appearing in HEP
- Standard phase-space integration techniques
- Current numerical methods for loop integrals
- Machine learning assisted MC-integration
- Prospects

Plan

- Type of integrals appearing in HEP
- Standard phase-space integration techniques
- Current numerical methods for loop integrals
- Machine learning assisted MC-integration
- Prospects

Prospects

- Time is right for something like "**NNVegas**"
  ➡ Make applicable by non-experts, automatically adjusting hyperparams given case at hand.

- Explore different **models** (e.g. Boltzmann Machines).

- Explore idea of feeding **redundant** inputs to the **NN**, thereby feeding it **prior integrand knowledge**.

- **Study NN integration** thoroughly by setting up an **interface** to a generic HEP simulation tool (e.g. **MG5aMC**)