LEVERHULME TRUST

RMCL2 Workshop

Progress on Soft Photon Resummation Implementation in Phokhara

Jérémy Paltrinieri

08/05/25

Motivation of the talk

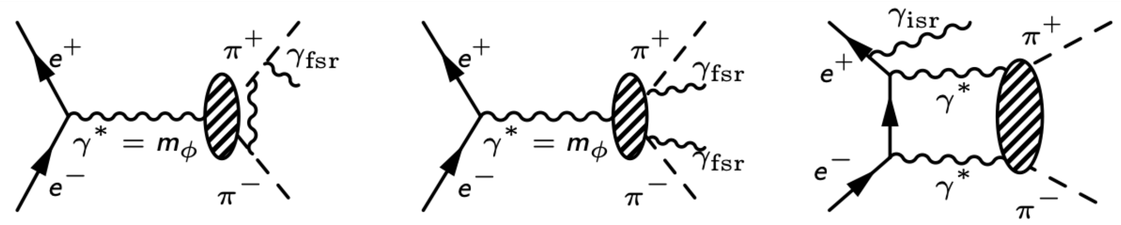
- Importance of hadron cross-sections measurements in g-2 determination
- Radiative corrections studies must be undertaken in order to clear up the tensions between experiments
- From radiative return experiments like KLOE, this means taking care of NNLO and all-order exponentiation effects: many challenges!
- In Phokhara, pursing NNLO accuracy on one side and approximate all order accuracy in the soft photon regime on the other side, for radiative return processes such as $e^+e^- \to \mu^+\mu^-\gamma$

Status of Phokhara

- Phokhara simulates electron-positron annihilation into hadrons (and muons) at NLO accuracy. This includes virtual and soft photon corrections to one photon emission events and the emission of two real hard photons.
- Case of muons: complete NLO - [arXiv:1312.3610]
- Case of pions: FxsQED for the pion treatment - [Szymon Tracz PhD Thesis]
- Disclaimer: not discussing many other hadronic final states: 3π, 4π, NN, KK…

Status of Phokhara

- Current version: Phokhara v10.0: https://looptreeduality.csic.es/phokhara/ (October 2020)
- Last update: two remaining gauge-invariant contributions: FSRNLO & TVP
- Strong2020 paper [2410.22882] and website

compl. NLO (`fsr=1, nlo=1 nlo2=1, fsrnlo=1`)	IFSNLO +	

Roadmap for Phokhara

$e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$ @ NLO
sQED for pions

Roadmap for Phokhara

CEEX formalism implemented in Phokhara, with the goal of implementing resummed $e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$

valid at $\mathcal{O}(\alpha^1)_{\text{CEEX}}$ in the language of KKMC

$e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$ @ NLO
sQED for pions

Roadmap for Phokhara

CEEX formalism implemented in Phokhara, with the goal of implementing resummed $e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$

valid at $\mathcal{O}(\alpha^1)_{\text{CEEX}}$ in the language of KKMC

Treatment of pions: see Pau's talk (insertion of GVMD model in Phokhara)

$e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$ @ NLO
sQED for pions

Roadmap for Phokhara

$e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$ @ NNLO
See William's talk

Treatment of pions: see Pau's talk (insertion of GVMD model in Phokhara)

CEEX formalism implemented in Phokhara, with the goal of implementing resummed $e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$

valid at $\mathcal{O}(\alpha^1)_{\text{CEEX}}$ in the language of KKMC

$e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$ @ NLO
sQED for pions

Roadmap for Phokhara

Resummed CEEX results for
$e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$
Valid at $\mathcal{O}(\alpha^2)_{\text{CEEX}}$?

$e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$@NNLO
See William's talk

Treatment of pions: see Pau's talk (insertion of GVMD model in Phokhara)

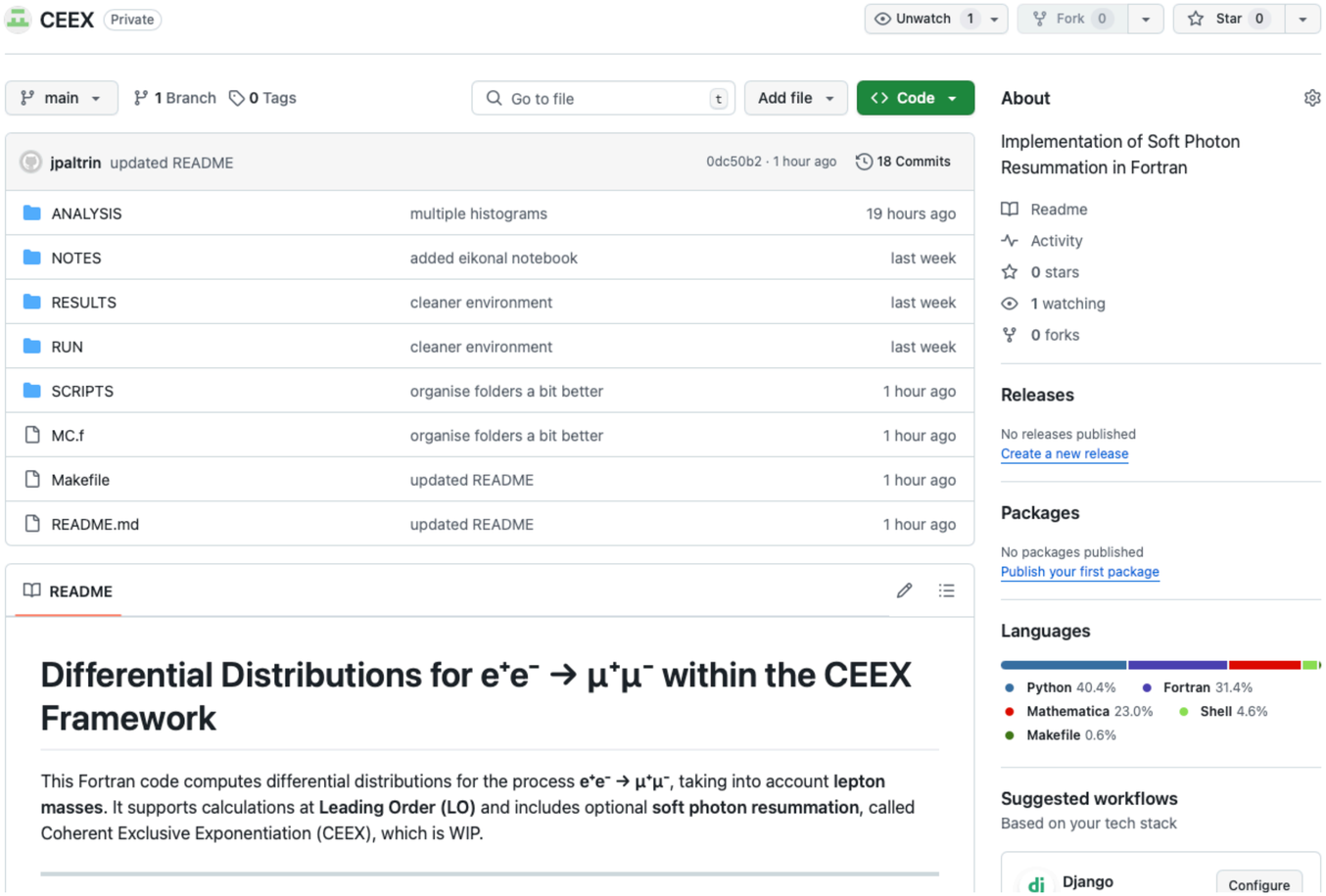
CEEX formalism implemented in Phokhara, with the goal of implementing resummed
$e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$

valid at $\mathcal{O}(\alpha^1)_{\text{CEEX}}$ in the language of KKMC

$e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$@NLO
sQED for pions

Towards CEEX in Phokhara

- Sheds another light on missing higher-order effects
- YFS formalism: resummation of soft photons effects
- 2-loop integrals are notoriously hard: approximations provide insight!



Progress on CEEX in Fortran

Different building blocks needed to achieve this:

$$\sigma^{(r)} = \frac{1}{\mathrm{flux}(s)} \sum_{n=0}^{\infty} \int d\mathrm{Lips}_{n+2}(p_a + p_b; p_c, p_d, k_1, \ldots, k_n)\ \rho^{(r)}_{\mathrm{CEEX}}(p_a, p_b, p_c, p_d, k_1, \ldots, k_n)$$

$$\rho^{(r)}_{\mathrm{CEEX}}(p_a, p_b, p_c, p_d, k_1, k_2, \ldots, k_n) = \frac{1}{n!} e^{Y(\Omega; p_a, \ldots, p_d)}\ \bar{\Theta}(\Omega) \sum_{\sigma_i = \pm 1} \sum_{\lambda_i, \bar{\lambda}_i = \pm 1}$$

$$\sum_{i,j,l,m=0}^{3} \hat{\varepsilon}^i_a \hat{\varepsilon}^j_b\ \sigma^i_{\lambda_a \bar{\lambda}_a} \sigma^j_{\lambda_b \bar{\lambda}_b} \mathfrak{M}^{(r)}_n \begin{pmatrix} p\, k_1\, k_2 \\ \lambda \sigma_1 \sigma_2 \end{pmatrix} \cdots \begin{matrix} k_n \\ \sigma_n \end{matrix} \Big) \Big[\mathfrak{M}^{(r)}_n \begin{pmatrix} p\, k_1\, k_2 \\ \bar{\lambda} \sigma_1 \sigma_2 \end{pmatrix} \cdots \begin{matrix} k_n \\ \sigma_n \end{matrix} \Big) \Big]^{\star} \sigma^l_{\bar{\lambda}_c \lambda_c} \sigma^m_{\bar{\lambda}_d \lambda_d} \hat{h}^l_c \hat{h}^m_d$$

Progress on CEEX in Fortran

Different building blocks needed to achieve this:

$$\sigma^{(r)} = \frac{1}{\mathrm{flux}(s)} \sum_{n=0}^{\infty} \int d\mathrm{Lips}_{n+2}(p_a + p_b; p_c, p_d, k_1, \ldots, k_n)\, \rho^{(r)}_{\mathrm{CEEX}}(p_a, p_b, p_c, p_d, k_1, \ldots, k_n)$$

Multi-Photon Phase-Space Generation

$$\rho^{(r)}_{\mathrm{CEEX}}(p_a, p_b, p_c, p_d, k_1, k_2, \ldots, k_n) = \frac{1}{n!} e^{Y(\Omega; p_a, \ldots, p_d)}\, \bar{\Theta}(\Omega) \sum_{\sigma_i = \pm 1} \sum_{\lambda_i, \bar{\lambda}_i = \pm 1}$$

$$\sum_{i,j,l,m=0}^{3} \hat{\varepsilon}^i_a \hat{\varepsilon}^j_b\, \sigma^i_{\lambda_a \bar{\lambda}_a} \sigma^j_{\lambda_b \bar{\lambda}_b} \mathfrak{M}^{(r)}_n \left(\begin{smallmatrix} p\, k_1\, k_2 \\ \lambda \sigma_1 \sigma_2 \end{smallmatrix} \cdots \begin{smallmatrix} k_n \\ \sigma_n \end{smallmatrix}\right) \left[\mathfrak{M}^{(r)}_n \left(\begin{smallmatrix} p\, k_1\, k_2 \\ \bar{\lambda} \sigma_1 \sigma_2 \end{smallmatrix} \cdots \begin{smallmatrix} k_n \\ \sigma_n \end{smallmatrix}\right)\right]^{\star} \sigma^l_{\bar{\lambda}_c \lambda_c} \sigma^m_{\bar{\lambda}_d \lambda_d} \hat{h}^l_c \hat{h}^m_d$$

Progress on CEEX in Fortran

Different building blocks needed to achieve this:

$$\sigma^{(r)} = \frac{1}{\mathrm{flux}(s)} \sum_{n=0}^{\infty} \int d\mathrm{Lips}_{n+2}(p_a + p_b; p_c, p_d, k_1, \ldots, k_n)\, \rho^{(r)}_{\mathrm{CEEX}}(p_a, p_b, p_c, p_d, k_1, \ldots, k_n)$$

Multi-Photon Phase-Space Generation

YFS form factor

$$\rho^{(r)}_{\mathrm{CEEX}}(p_a, p_b, p_c, p_d, k_1, k_2, \ldots, k_n) = \frac{1}{n!} e^{Y(\Omega; p_a, \ldots, p_d)} \bar{\Theta}(\Omega) \sum_{\sigma_i = \pm 1} \sum_{\lambda_i, \bar{\lambda}_i = \pm 1}$$

$$\sum_{i,j,l,m=0}^{3} \hat{\varepsilon}^i_a \hat{\varepsilon}^j_b\, \sigma^i_{\lambda_a \bar{\lambda}_a} \sigma^j_{\lambda_b \bar{\lambda}_b} \mathfrak{M}^{(r)}_n \left(\begin{smallmatrix} p\, k_1\, k_2 \ldots k_n \\ \lambda \sigma_1 \sigma_2 \ldots \sigma_n \end{smallmatrix}\right) \left[\mathfrak{M}^{(r)}_n \left(\begin{smallmatrix} p\, k_1\, k_2 \ldots k_n \\ \bar{\lambda} \sigma_1 \sigma_2 \ldots \sigma_n \end{smallmatrix}\right)\right]^{\star} \sigma^l_{\bar{\lambda}_c \lambda_c} \sigma^m_{\bar{\lambda}_d \lambda_d} \hat{h}^l_c \hat{h}^m_d$$

Progress on CEEX in Fortran

Different building blocks needed to achieve this:

$$\sigma^{(r)} = \frac{1}{\mathrm{flux}(s)} \sum_{n=0}^{\infty} \int d\mathrm{Lips}_{n+2}(p_a + p_b; p_c, p_d, k_1, \ldots, k_n)\, \rho^{(r)}_{\mathrm{CEEX}}(p_a, p_b, p_c, p_d, k_1, \ldots, k_n)$$

Multi-Photon Phase-Space Generation

YFS form factor

$$\rho^{(r)}_{\mathrm{CEEX}}(p_a, p_b, p_c, p_d, k_1, k_2, \ldots, k_n) = \frac{1}{n!} e^{Y(\Omega; p_a, \ldots, p_d)} \bar{\Theta}(\Omega) \sum_{\sigma_i = \pm 1} \sum_{\lambda_i, \bar{\lambda}_i = \pm 1}$$

$$\sum_{i,j,l,m=0}^{3} \hat{\varepsilon}^i_a \hat{\varepsilon}^j_b\, \sigma^i_{\lambda_a \bar{\lambda}_a} \sigma^j_{\lambda_b \bar{\lambda}_b} \mathfrak{M}^{(r)}_n \left({}^{p\, k_1\, k_2 \ldots k_n}_{\lambda \sigma_1 \sigma_2 \ldots \sigma_n} \right) \left[\mathfrak{M}^{(r)}_n \left({}^{p\, k_1\, k_2 \ldots k_n}_{\bar{\lambda} \sigma_1 \sigma_2 \ldots \sigma_n} \right) \right]^{\star} \sigma^l_{\bar{\lambda}_c \lambda_c} \sigma^m_{\bar{\lambda}_d \lambda_d} \hat{h}^l_c \hat{h}^m_d$$

Beta functions (IR safe Matrix Elements)

Progress on CEEX in Fortran

Different building blocks needed to achieve this:

$$\sigma^{(r)} = \frac{1}{\mathrm{flux}(s)} \sum_{n=0}^{\infty} \int d\mathrm{Lips}_{n+2}(p_a + p_b; p_c, p_d, k_1, \ldots, k_n)\, \rho^{(r)}_{\mathrm{CEEX}}(p_a, p_b, p_c, p_d, k_1, \ldots, k_n)$$

Analysis Framework

Multi-Photon Phase-Space Generation

YFS form factor

$$\rho^{(r)}_{\mathrm{CEEX}}(p_a, p_b, p_c, p_d, k_1, k_2, \ldots, k_n) = \frac{1}{n!} e^{Y(\Omega; p_a, \ldots, p_d)} \bar{\Theta}(\Omega) \sum_{\sigma_i = \pm 1} \sum_{\lambda_i, \bar{\lambda}_i = \pm 1}$$

$$\sum_{i,j,l,m=0}^{3} \hat{\varepsilon}^i_a \hat{\varepsilon}^j_b\, \sigma^i_{\lambda_a \bar{\lambda}_a} \sigma^j_{\lambda_b \bar{\lambda}_b} \mathfrak{M}^{(r)}_n \left({}^{p\, k_1\, k_2 \ldots k_n}_{\lambda \sigma_1 \sigma_2 \ldots \sigma_n} \right) \left[\mathfrak{M}^{(r)}_n \left({}^{p\, k_1\, k_2 \ldots k_n}_{\bar{\lambda} \sigma_1 \sigma_2 \ldots \sigma_n} \right) \right]^{\star} \sigma^l_{\bar{\lambda}_c \lambda_c} \sigma^m_{\bar{\lambda}_d \lambda_d} \hat{h}^l_c \hat{h}^m_d$$

Beta functions (IR safe Matrix Elements)

Progress on CEEX in Fortran

Multi-Photon Phase-Space Generation

↓

Module PS

↓

Multi-photon Phase-Space event generator is necessary for resummed results. Can be cross-checked with NLO generator in Phokhara.

Progress on CEEX in Fortran

Multi-Photon Phase-Space Generation

↓

Module PS

↓

Multi-photon Phase-Space event generator is necessary for resummed results. Can be cross-checked with NLO generator in Phokhara.

Input:
- n: integer, number of photons

Outputs:
- momenta (modified in-place)
- weight: phase space weight (modified in-place)

Step 1: Initialise incoming momenta
- Generate initial energy and momentum
- Add positron and electron momenta
- Update weight

Step 2: If no photons
- Reconstruct final state muon-antimuon pair momenta

Step 3: If photons are emitted
Repeat until a valid event is generated:
- For each photon:
  - Generate massless photon momentum
  - Reconstruct final state muon-antimuon pair momenta
  - Check kinematic validity:
- Update weight

Progress on CEEX in Fortran

Multi-Photon Phase-Space Generation

↓

Module PS

↓

Multi-photon Phase-Space event generator is necessary for resummed results. Can be cross-checked with NLO generator in Phokhara.

Input:
- n: integer, number of photons

Outputs:
- momenta (modified in-place)
- weight: phase space weight (modified in-place)

Step 1: Initialise incoming momenta
- Generate initial energy and momentum
- Add positron and electron momenta
- Update weight

Step 2: If no photons
- Reconstruct final state muon-antimuon pair momenta

Step 3: If photons are emitted

Repeat until a valid event is generated:
- For each photon:
  - Generate massless photon momentum
  - Reconstruct final state muon-antimuon pair momenta
  - Check kinematic validity:
- Update weight

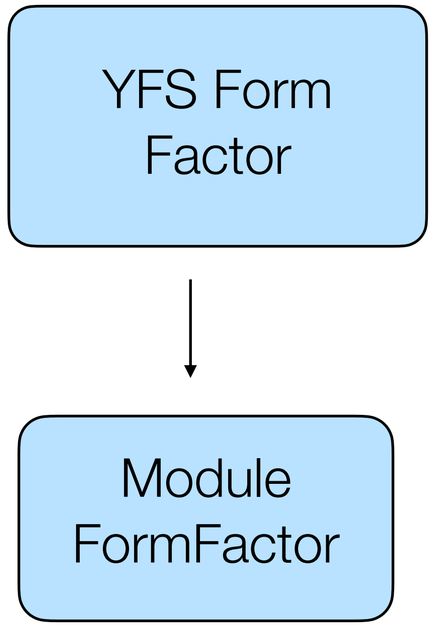
E	px	py	pz	m2
0.500000	0.000000	0.000000	0.499975	0.000025
0.500000	0.000000	0.000000	-0.499975	0.000025
0.500000	0.051492	0.043576	-0.484031	0.011164
0.500000	-0.051492	-0.043576	0.484031	0.011164
Total momentum				
0.000000	0.000000	0.000000	0.000000	

0-photon event

E	px	py	pz	m2
0.500000	0.000000	0.000000	0.499975	0.000025
0.500000	0.000000	0.000000	-0.499975	0.000025
0.278691	-0.087642	0.088248	-0.225912	0.011164
0.223409	-0.027450	-0.008661	0.194729	0.011164
0.131952	-0.018749	0.024650	-0.128266	-0.000000
0.020522	0.001769	-0.001243	-0.020407	0.000000
0.047267	0.023795	-0.004575	0.040584	0.000000
0.027761	-0.010782	-0.000331	-0.025580	-0.000000
0.167504	0.127600	-0.088527	0.062760	-0.000000
0.102894	-0.008541	-0.009559	0.102093	0.000000
Total momentum				
0.000000	0.000000	0.000000	-0.000000	

6-photon event

Progress on CEEX in Fortran



Progress on CEEX in Fortran

YFS Form Factor

↓

Module FormFactor

Input:
- p1: momenta particle 1
- p2: momenta particle 2

Outputs:
- YFS formfactor B(p1,p2)

[2203.10948]
Sherpa YFS

Virtual IR Function

Here we present the expression for the virtual part of the YFS for any two charged massive particles.

$$
\begin{aligned}
&2\alpha\mathcal{R}B(p_1,p_2)\\
&= -Z_iZ_j\theta_i\theta_j\frac{\alpha}{\pi}\Bigg[\left(\frac{1}{\rho}\ln\frac{\mu(1+\rho)}{m_1m_2}-1\right)\ln\frac{m_\gamma^2}{m_1m_2}+\frac{\mu\rho}{s}\ln\frac{\mu(1+\rho)}{m_1m_2}+\frac{m_1^2-m_2^2}{2s}\ln\frac{m_1}{m_2}-1\\
&\quad+\frac{1}{\rho}\left(\pi^2-\frac{1}{2}\ln\frac{\mu(1+\rho)}{m_1^2}\ln\frac{\mu(1+\rho)}{m_2^2}-\frac{1}{2}\ln^2\frac{m_1^2+\mu(1+\rho)}{m_2^2+\mu(1+\rho)}-\mathrm{Li}_2(\zeta_1)-\mathrm{Li}_2(\zeta_2)\right)\Bigg],
\end{aligned}
\tag{A.1}
$$

where,

$$
\mu=p_1p_2,\quad s=2\mu+m_1^2+m_2^2,\quad \rho=\sqrt{1-\left(\frac{m_1m_2}{\mu}\right)^2},\quad \zeta_i=\frac{2\mu\rho}{m_i^2+\mu(1+\rho)}. \tag{A.2}
$$

Real IR Function

Here we present an expression for the IR function $\tilde{B}$ which corresponds to the emission of a real photon $k\in\Omega$ from a dipole consisting of two charged massive particles p_1 and p_2.

$$
\begin{aligned}
2\alpha\tilde{B}(p_1,p_2) &= -Z_iZ_j\theta_i\theta_j\frac{\alpha}{\pi}\Bigg[\left(\frac{1}{\rho}\ln\frac{\mu(1+\rho)}{m_1m_2}-1\right)\ln\frac{\omega}{m_\gamma^2}+\frac{1}{2\beta_1}\ln\frac{1+\beta_1}{1-\beta_1}\\
&\quad+\frac{1}{2\beta_2}\ln\frac{1+\beta_2}{1-\beta_2}+\mu G(p_1,p_2)\Bigg],
\end{aligned}
\tag{A.3}
$$

where $\beta_i=\frac{|\vec{p}_i|}{E_i}$ and μ,ρ are defined in Eq. (A.2) and ω is the momentum cut-off specifying Ω in the frame $\tilde{B}$ is to be evaluated in. $G(p_1,p_2)$ is a complicated function that can be expressed as a combination of logarithms and dilogarithms,

$$
G(p_1,p_2) = \frac{1}{\sqrt{(Q^2+M^2)(Q^2+\delta^2)}}\left[\ln\frac{\sqrt{\Delta^2+Q^2}-\Delta}{\sqrt{\Delta^2+Q^2}+\Delta}\left[\chi^{14}_{23}(\eta_1)-\chi^{14}_{23}(\eta_0)\right]+Y(\eta_1)-Y(\eta_0)\right],
\tag{A.4}
$$

Progress on CEEX in Fortran

YFS Form Factor

↓

Module FormFactor

Input:
- p1: momenta particle 1
- p2: momenta particle 2

Outputs:
- YFS formfactor B(p1,p2)

[2203.10948]
Sherpa YFS

Virtual IR Function

Here we present the expression for the virtual part of the YFS for any two charged massive particles.

$$
\begin{aligned}
&2\alpha\mathcal{R}B(p_1,p_2)\\
&= -Z_iZ_j\theta_i\theta_j\frac{\alpha}{\pi}\Bigg[\left(\frac{1}{\rho}\ln\frac{\mu(1+\rho)}{m_1m_2}-1\right)\ln\frac{m_\gamma^2}{m_1m_2}+\frac{\mu\rho}{s}\ln\frac{\mu(1+\rho)}{m_1m_2}+\frac{m_1^2-m_2^2}{2s}\ln\frac{m_1}{m_2}-1\\
&\quad+\frac{1}{\rho}\left(\pi^2-\frac{1}{2}\ln\frac{\mu(1+\rho)}{m_1^2}\ln\frac{\mu(1+\rho)}{m_2^2}-\frac{1}{2}\ln^2\frac{m_1^2+\mu(1+\rho)}{m_2^2+\mu(1+\rho)}-\mathrm{Li}_2(\zeta_1)-\mathrm{Li}_2(\zeta_2)\right)\Bigg],
\end{aligned}
\tag{A.1}
$$

where,

$$
\mu=p_1p_2,\quad s=2\mu+m_1^2+m_2^2,\quad \rho=\sqrt{1-\left(\frac{m_1m_2}{\mu}\right)^2},\quad \zeta_i=\frac{2\mu\rho}{m_i^2+\mu(1+\rho)}. \tag{A.2}
$$

Real IR Function

Here we present an expression for the IR function $\tilde{B}$ which corresponds to the emission of a real photon $k\in\Omega$ from a dipole consisting of two charged massive particles p_1 and p_2.

$$
\begin{aligned}
2\alpha\tilde{B}(p_1,p_2) &= -Z_iZ_j\theta_i\theta_j\frac{\alpha}{\pi}\Bigg[\left(\frac{1}{\rho}\ln\frac{\mu(1+\rho)}{m_1m_2}-1\right)\ln\frac{\omega}{m_\gamma^2}+\frac{1}{2\beta_1}\ln\frac{1+\beta_1}{1-\beta_1}\\
&\quad+\frac{1}{2\beta_2}\ln\frac{1+\beta_2}{1-\beta_2}+\mu G(p_1,p_2)\Bigg],
\end{aligned}
\tag{A.3}
$$

where $\beta_i=\frac{|\vec{p}_i|}{E_i}$ and μ,ρ are defined in Eq. (A.2) and ω is the momentum cut-off specifying Ω in the frame $\tilde{B}$ is to be evaluated in. $G(p_1,p_2)$ is a complicated function that can be expressed as a combination of logarithms and dilogarithms,

$$
G(p_1,p_2) = \frac{1}{\sqrt{(Q^2+M^2)(Q^2+\delta^2)}}\left[\ln\frac{\sqrt{\Delta^2+Q^2}-\Delta}{\sqrt{\Delta^2+Q^2}+\Delta}\left[\chi_{23}^{14}(\eta_1)-\chi_{23}^{14}(\eta_0)\right]+Y(\eta_1)-Y(\eta_0)\right],
\tag{A.4}
$$

✅ Validated with Mathematica

See YFS.nb notebook in the NOTES folder of the git repository

Jérémy Paltrinieri

Progress on CEEX in Fortran

ANALYSIS

Modules
HISTO, VARIABLES

Progress on CEEX in Fortran

ANALYSIS

↓

Modules
HISTO, VARIABLES

This module allows to **create, fill, display, and save histograms at running time** (MC event generation)**.**

What each part does:

- **init_histo**:
  Sets up empty histogram bins between a lower and upper limit. Each bin gets a range and starts with a value of zero.
- **fill_histo**:
  Adds the *weight* of the event to the correct bin based on the input data. It figures out which bin the data point belongs to and increases that bin's count.
- **finalize_histo**:
  Finishes the histogram by adjusting all the values so they're properly scaled.
- **print_histo**:
  Simply prints out the histogram — showing the lower and upper edge of each bin and the value it holds.
- **export_histogram_to_csv**:
  Saves the histogram to a CSV file that can be compared to other MC generators.

Progress on CEEX in Fortran

ANALYSIS

↓

Modules
HISTO, VARIABLES

This module allows to **create, fill, display, and save histograms at running time** (MC event generation)**.**

What each part does:

- **init_histo**:
  Sets up empty histogram bins between a lower and upper limit. Each bin gets a range and starts with a value of zero.
- **fill_histo**:
  Adds the *weight* of the event to the correct bin based on the input data. It figures out which bin the data point belongs to and increases that bin's count.
- **finalize_histo**:
  Finishes the histogram by adjusting all the values so they're properly scaled.
- **print_histo**:
  Simply prints out the histogram — showing the lower and upper edge of each bin and the value it holds.
- **export_histogram_to_csv**:
  Saves the histogram to a CSV file that can be compared to other MC generators.

Validated at LO with other generators (KKMC)

Finalized Histogram

Low	Up	Value
0.000000	0.200000	0.000000E+00
0.200000	0.400000	0.000000E+00
0.400000	0.600000	0.000000E+00
0.600000	0.800000	0.000000E+00
0.800000	1.000000	0.000000E+00
1.000000	1.200000	0.433490E+06
1.200000	1.400000	0.000000E+00
1.400000	1.600000	0.000000E+00
1.600000	1.800000	0.000000E+00
1.800000	2.000000	0.000000E+00

$m^2_{\mu\mu}$ ***invariant mass of LO events***

Validation of Fixed-Order Modules

As is Phokhara, everything can be run in parallel, and of course for LO events the generation and integration is very fast.

```
[jpaltrinieri@kappa CEEX]$ ./MC 1 10000000

 ***********************************************
 RUNNING FORTRAN CEEX PROGRAM
 Seed set to:           1
 NbEvents:    10000000

 ***********************************************
 PRINTING RESULTS
 Monte Carlo Integral Result (pb):    86788.666725620918
 Analytical Result Massive (pb):    86788.525413961281
 Analytical Result Massless (pb):    86854.463232265436
 Ratio =    1.0000016282297570

 PROGRAM COMPLETE
 ***********************************************
```

Validation of Fixed-Order Modules

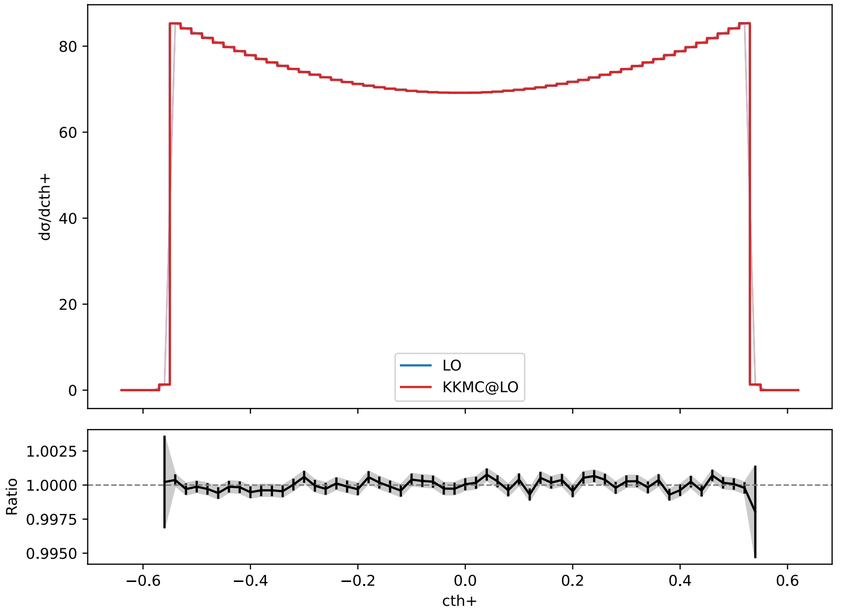
As is Phokhara, everything can be run in parallel, and of course for LO events the generation and integration is very fast.

```
[jpaltrinieri@kappa CEEX]$ ./MC 1 10000000

 ************************************************
 RUNNING FORTRAN CEEX PROGRAM
 Seed set to:            1
 NbEvents:    10000000

 ************************************************
 PRINTING RESULTS
 Monte Carlo Integral Result (pb):    86788.666725620918
 Analytical Result Massive (pb):    86788.525413961281
 Analytical Result Massless (pb):    86854.463232265436
 Ratio =    1.0000016282297570

 PROGRAM COMPLETE
 ************************************************
```



$e^+e^- \to \mu^+\mu^-$ **@LO**

Which showcases the validation of the LO modules (Spinor-helicity variables, event generator)

Fortran modules in Phokhara

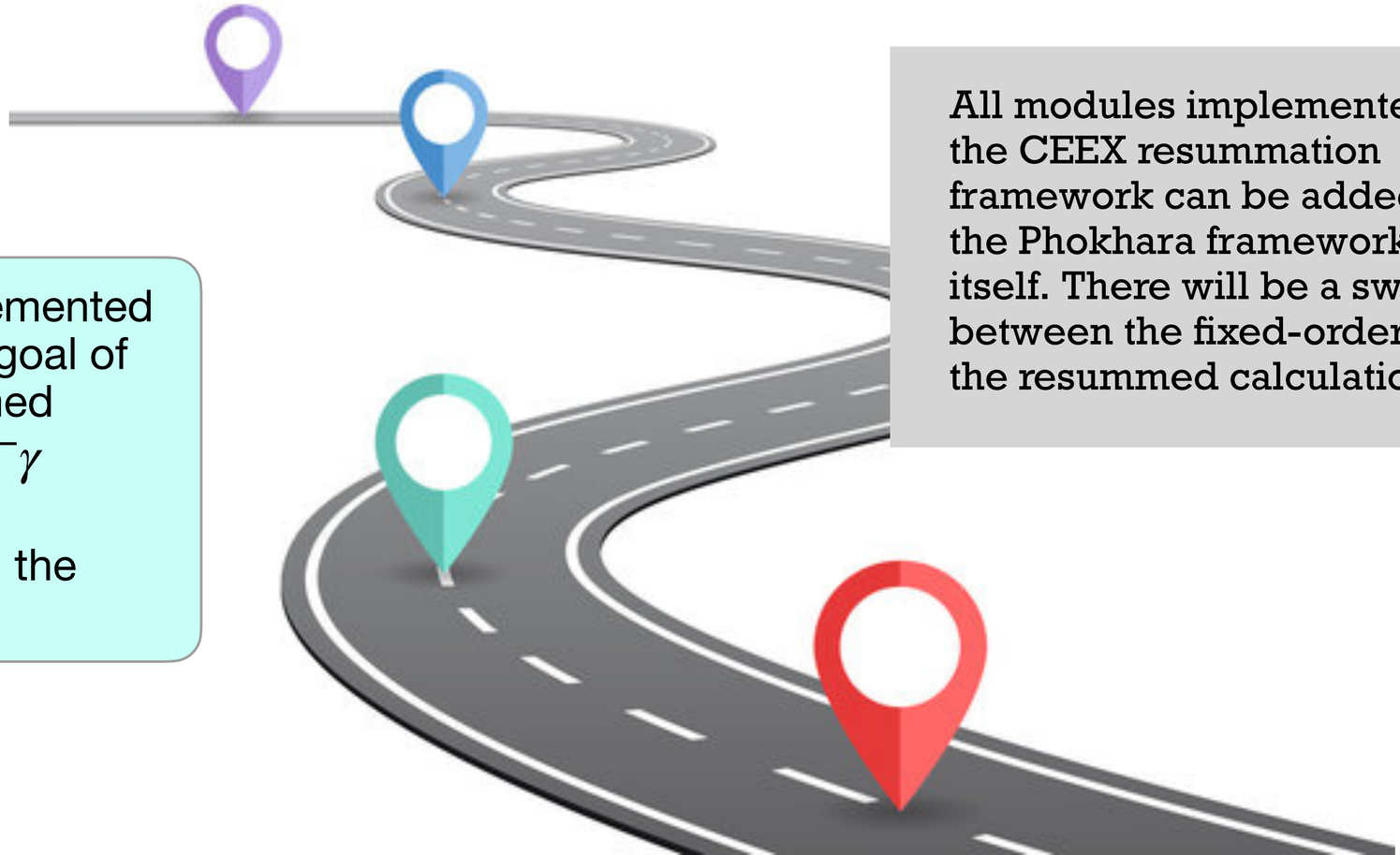
$e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$@NNLO
See William's talk

CEEX formalism implemented in Phokhara, with the goal of implementing resummed $e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$

valid at $\mathcal{O}(\alpha^1)_{\text{CEEX}}$ in the language of KKMC

Fortran modules in Phokhara

$e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$@NNLO
See William's talk

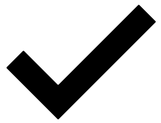


CEEX formalism implemented in Phokhara, with the goal of implementing resummed $e^+e^- \to \mu^+\mu^-\gamma, \pi^+\pi^-\gamma$

valid at $\mathcal{O}(\alpha^1)_{\text{CEEX}}$ in the language of KKMC

All modules implemented for the CEEX resummation framework can be added to the Phokhara framework itself. There will be a switch between the fixed-order and the resummed calculation

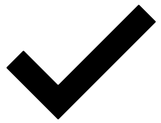
Progress on CEEX in Phokhara



Summary

- Framework in Fortran established
- Fixed-Order Modules working
- Debugging multi-photon Modules

Progress on CEEX in Phokhara



Summary

- Framework in Fortran established
- Fixed-Order Modules working
- Debugging multi-photon Modules

What’s next?

- Validation of KKMC-like exponentiation
- Incorporation in Phokhara
- Improving the beta functions, rewrite the procedure in dim-reg
- GVMD@NLO for Pions, but no plans for GVMD@NNLO