tiktaalik: fast code for NLO GPD evolution

Adam Freese
May 15, 2026
Comp. Phys. Comm. 311 (2025) 109552

CENTER for NUCLEAR FEMTOGRAPHY

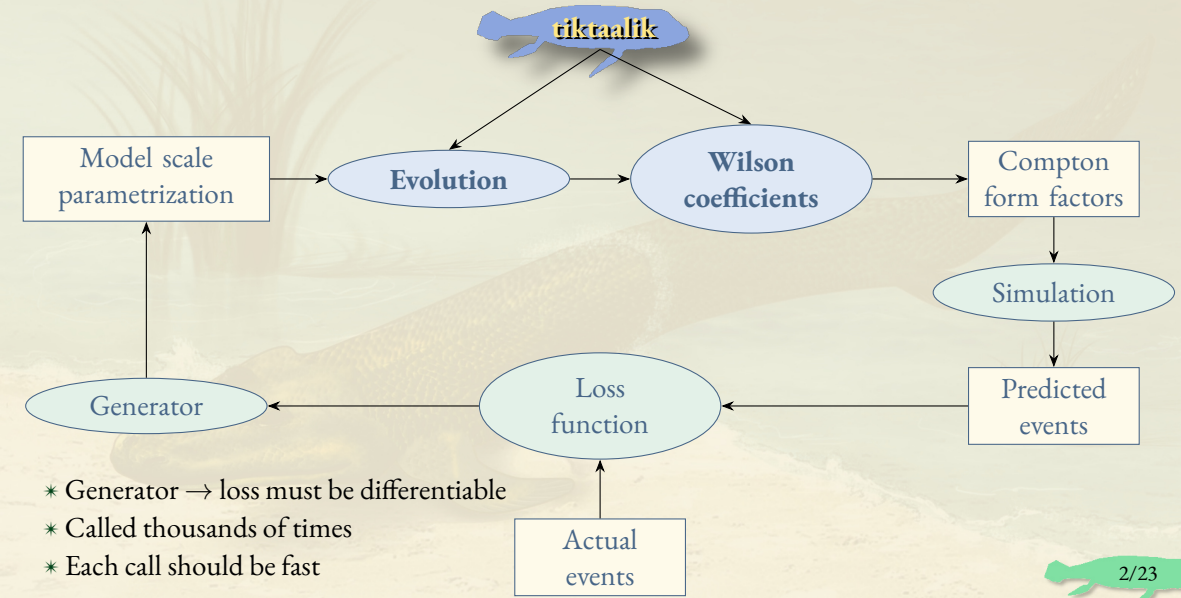
- **tiktaalik** is a Fortran+Python code package to:
  1. Evolve generalized parton distributions (GPDs) in x-space
  2. Convert evolved GPDs into Compton form factors (CFFs)
- tiktaalik does both via **matrix multiplication**

$$H(x_a, \xi_c, Q_d^2) = \sum_b M_{ab}(\xi_c, Q_0^2 \to Q_d^2) H(x_b, \xi_c, Q_0^2)$$

$$\mathcal{H}(\xi_c, Q_d^2) = \sum_a C_a(\xi_c, Q_d^2) H(x_a, \xi_c, Q_d^2)$$

  - ⇨ Fast (microseconds on GPUs)
  - ⇨ Auto-differentiable
  - ⇨ Massively parallelizable (especially on GPUs)
- tiktaalik **creates the matrices**
  - ⇨ Matrices can be plugged into existing framework
  - ⇨ Easy-to-use Python interface

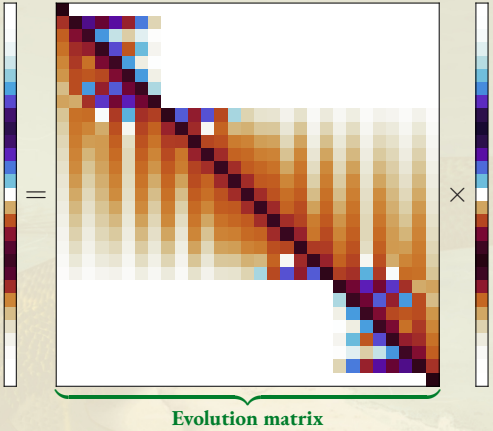
Why matrix multiplication?



- Generator → loss must be differentiable
- Called thousands of times
- Each call should be fast

Evolution matrices: what they do

- **Evolution matrix** evolves discretized GPDs via matrix multiplication



- **Evolution matrices** are *independent of* the specific GPD
  - ⇨ Compute once, store in memory
- Evolving a GPD takes **microseconds** on a GPU!
  - ⇨ Can easily be incorporated into global fits with AI/ML


Interpixels as finite elements

Interpixels

- **Interpixels** (**interp**olated **pixel**): interpolation basis functions.
  - Exploit linearity of polynomial interpolation:

$$P[y_1 + y_2](x) = P[y_1](x) + P[y_2](x)$$

  - GPD pixelation is a sum of pixels:

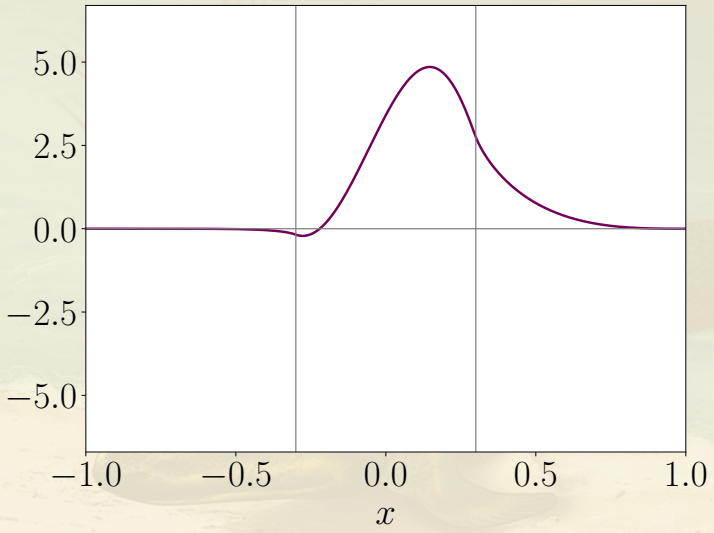
$$\boldsymbol{H} = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_n \end{bmatrix} = H_1 \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + H_2 \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \ldots + H_n \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \equiv H_1 \hat{e}_1 + H_2 \hat{e}_2 + \ldots + H_n \hat{e}_n$$

  - Interpolated pixelation is a sum of interpixels!

$$P[\boldsymbol{H}](x) = H_1 P[\hat{e}_1](x) + H_2 P[\hat{e}_2](x) + \ldots + H_n P[\hat{e}_n](x)$$

- Interpixels are an example of a **finite element**.
  - Used previously in some PDF evolution codes, e.g., HOPPET and APFEL.

Interpixel demo

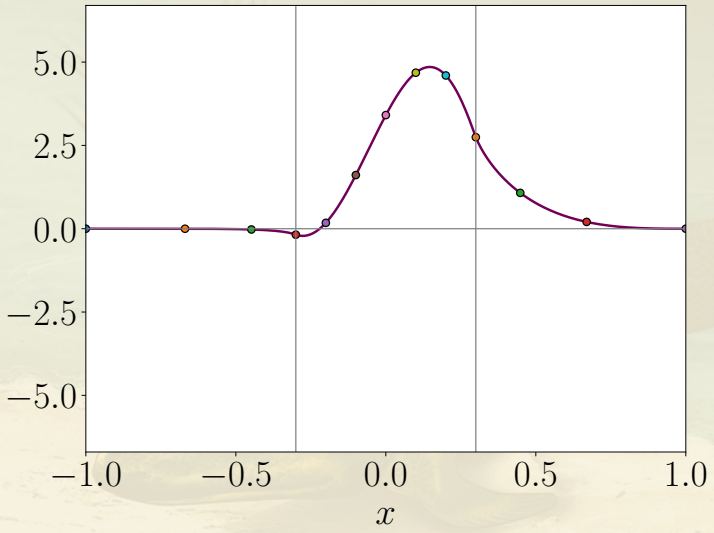


- Model H_u for **ground truth**

Example with $N_x = 13$

Model from: Kroll, Moutarde & Sabatie, Eur Phys J C 73 (2013) 2278

Interpixel demo

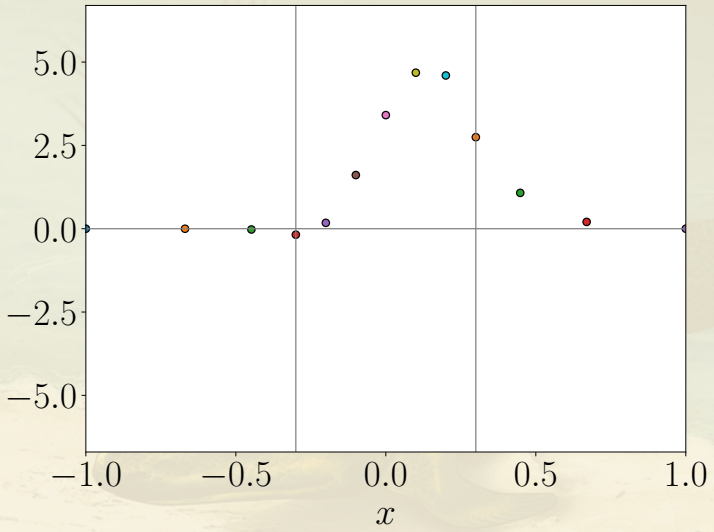


- Model H_u for **ground truth**
- Select **discrete values** from grid

Example with $N_x = 13$

Model from: Kroll, Moutarde & Sabatie, Eur Phys J C 73 (2013) 2278

Interpixel demo

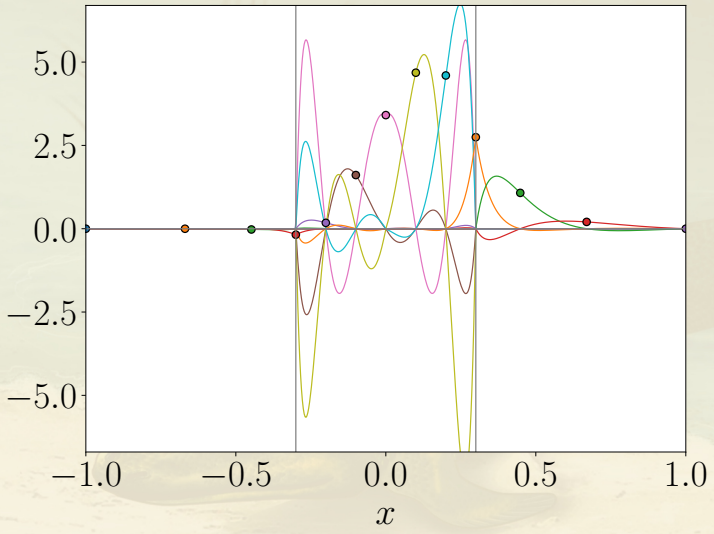


- Model H_u for **ground truth**
- Select **discrete values** from grid

Example with $N_x = 13$

Model from: Kroll, Moutarde & Sabatie, Eur Phys J C 73 (2013) 2278

Interpixel demo

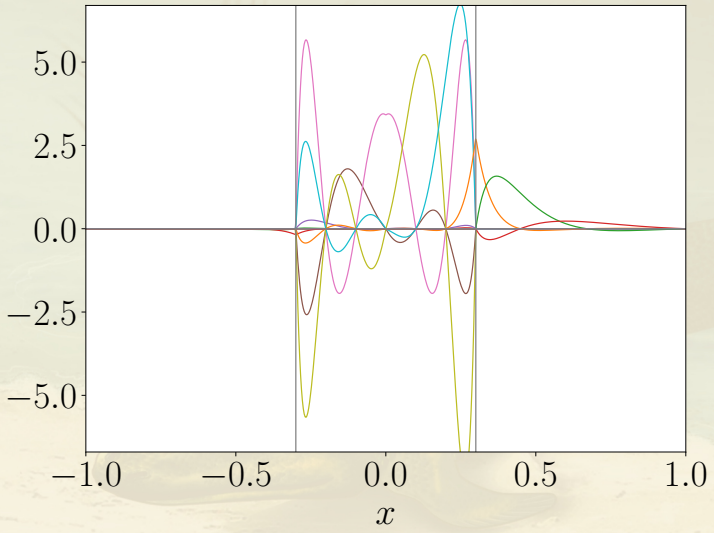


- Model H_u for **ground truth**
- Select **discrete values** from grid
- Build **interpixels** from each discrete value (highly oscillatory)
  - ⇨ Treat $x = \pm\xi$ as *edges*

Example with $N_x = 13$

Model from: Kroll, Moutarde & Sabatie, Eur Phys J C 73 (2013) 2278

Interpixel demo

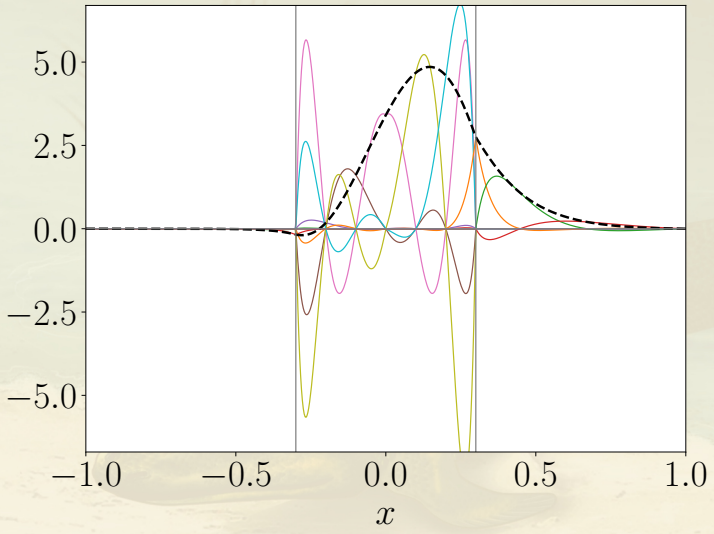


- Model H_u for **ground truth**
- Select **discrete values** from grid
- Build **interpixels** from each discrete value (highly oscillatory)
  - ⇨ Treat $x = \pm\xi$ as *edges*

Example with $N_x = 13$

Model from: Kroll, Moutarde & Sabatie, Eur Phys J C 73 (2013) 2278

Interpixel demo

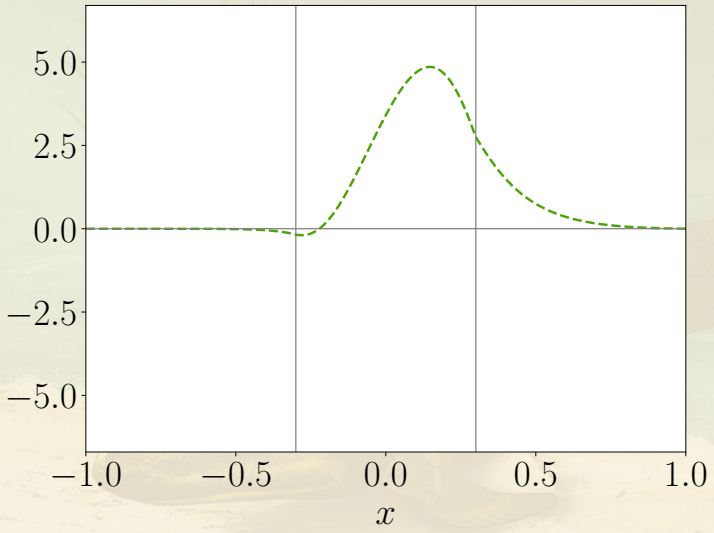


- Model H_u for **ground truth**
- Select **discrete values** from grid
- Build **interpixels** from each discrete value (highly oscillatory)
  - ⇨ Treat $x = \pm\xi$ as *edges*
- **Summing interpixels** approximates the **ground truth** (oscillations cancel in sum)

Example with $N_x = 13$

Model from: Kroll, Moutarde & Sabatie, Eur Phys J C 73 (2013) 2278

Interpixel demo

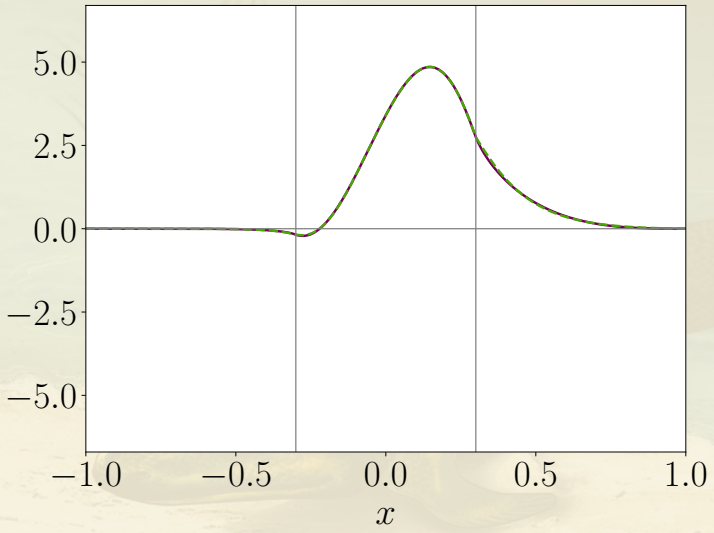


- Model H_u for **ground truth**
- Select **discrete values** from grid
- Build **interpixels** from each discrete value (highly oscillatory)
  - ⇨ Treat $x = \pm\xi$ as *edges*
- **Summing interpixels** approximates the **ground truth** (oscillations cancel in sum)

Example with $N_x = 13$

Model from: Kroll, Moutarde & Sabatie, Eur Phys J C 73 (2013) 2278

Interpixel demo



- Model H_u for **ground truth**
- Select **discrete values** from grid
- Build **interpixels** from each discrete value (highly oscillatory)
  - ⇨ Treat $x = \pm\xi$ as *edges*
- **Summing interpixels** approximates the **ground truth** (oscillations cancel in sum)

Example with $N_x = 13$

Model from: Kroll, Moutarde & Sabatie, Eur Phys J C 73 (2013) 2278

Discretizing the integral

- Right-hand size of evolution equation is an integral:

$$\frac{\mathrm{d}H(x,\xi,Q^2)}{\mathrm{d}\log(Q^2)} = \int_{-1}^{+1} \mathrm{d}y\, K(x,y,\xi,Q^2)H(y,\xi,Q^2)$$

- Integral in evolution equation approximated using **Gauss-Kronrod quadrature**.
  - The domain $[-1,1]$ is broken into **six regions** with boundaries:

$$-1 < \min(-\xi,-|x|) < \max(-\xi,-|x|) < 0 < \min(\xi,|x|) < \max(\xi,|x|) < 1$$

  - If $x = \pm\xi$, reduces to four regions.
  - 21-point quadrature used inside each region. (First release & paper used 15-point rule)

$$\int_{-1}^{+1} \mathrm{d}y\, K(x,y,\xi,Q^2)H(y,\xi,Q^2) \approx \sum_{g=1}^{N_g=6\times21} w_g K(x,y_g,\xi,Q^2)H(y_g,\xi,Q^2)$$

  - Discretized grid $\{x_i\}$ and quadrature grid $\{y_g\}$ are not the same.
  - x_i- and ξ-dependent interpolation must be done.
  - **Interpixels** are used for interpolation.

Integral discretization: now with interpixels!

- GPD at Gaussian weight points from piecewise polynomial interpolation:

$$H(y_g, \xi, Q^2) \approx \sum_{j=1}^{N_x} H_j(\xi, Q^2) P[\hat{e}_j](y_g)$$

  - ⇨ Interpolation decomposed into basis functions (**interpixels**).

- Integral is only over interpixels:

$$\int_{-1}^{+1} \mathrm{d}y\, K(x, y, \xi, Q^2) H(y, \xi, Q^2) \approx \sum_{j=1}^{N_x} \underbrace{\left(\sum_{g=1}^{N_g} w_g K(x_i, y_g, \xi, Q^2) P[\hat{e}_j](y_g) \right)}_{\big(K(\xi, Q^2)\big)_{ij}} H_j(\xi, Q^2)$$

  - ⇨ Absorb interpixel into kernel matrix.
  - ⇨ Integral over interpixel **independent of specific GPD**.
  - ⇨ Method can be generalized to distributions (plus prescription etc.)

Differential matrix equation

- Discretization+interpixels turns the evolution equation into a **matrix differential equation**:

$$\frac{\mathrm{d}H_i(\xi,Q^2)}{\mathrm{d}\log(Q^2)} = \sum_{j=1}^{N_x} K_{ij}(\xi,Q^2)H_j(\xi,Q^2)$$

- Can be solved with standard techniques, like Runge-Kutta.

$$H_i(\xi,t,Q^2_{\text{fin}}) = \sum_{j=1}^{N_x} M_{ij}(\xi,Q^2_{\text{ini}}\to Q^2_{\text{fin}})H_j(\xi,Q^2_{\text{ini}})$$

☞ Only K_{ij}—not H_j itself—is needed to build M_{ij}.


Features and limitations of tiktaalik

Features of tiktaalik

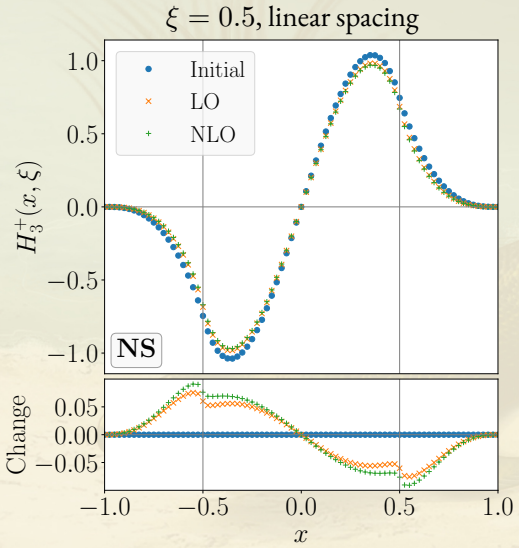
- Two grid types:
  1. Linear x spacing—ideal for $\xi \gtrsim 0.2$
  2. Hybrid log-linear spacing—ideal for $\xi \lesssim 0.2$
     - Logarithmic in DGLAP region, linear in ERBL region.

- Leading order (LO) and next-to-leading order (NLO) GPD evolution
  - Splitting functions from Belitsky, Freund & Müller, Nucl Phys B 574 (2000) 347

- LO and NLO Wilson coefficients for Compton form factors (CFFs)
  - Wilson coefficients from Braun, Yao Ji & Schoenleber, Phys Rev Lett 129 (2022) 172001
  - GPD $\rightarrow$ CFF conversion is *also* matrix multiplication!
  - Only implemented in dev branch—will pull into master branch in near future

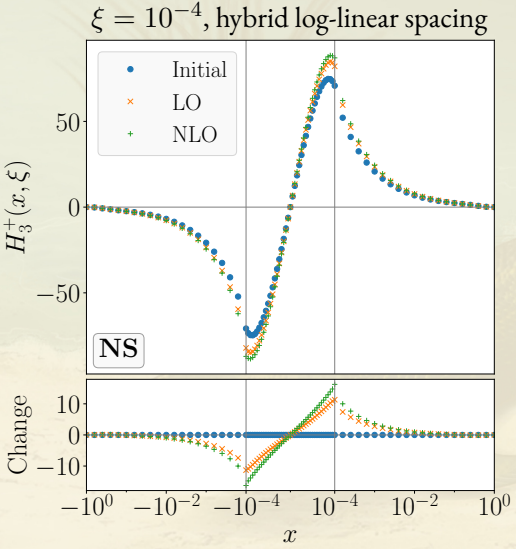
- Easy installation via pip.

Linear x grid



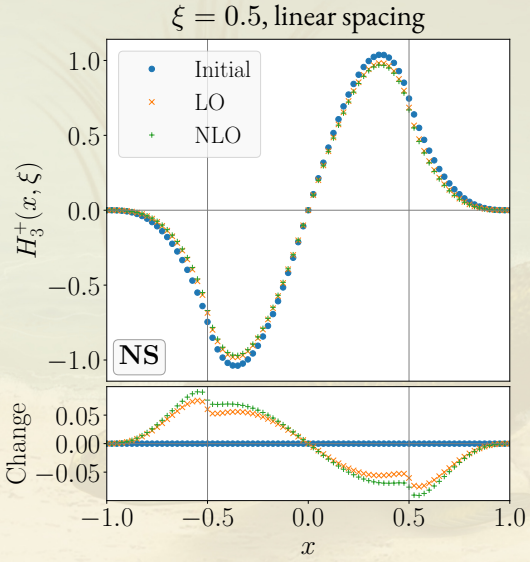
- User chooses x grid type, via `grid_type` kwarg, when calling `tiktaalik.matrices.set_x_xi_grids`
- `grid_type=1` (default) gives linear spacing.
- Great for $\xi \gtrsim 0.2$
- Awful at $\xi \lesssim 0.1$

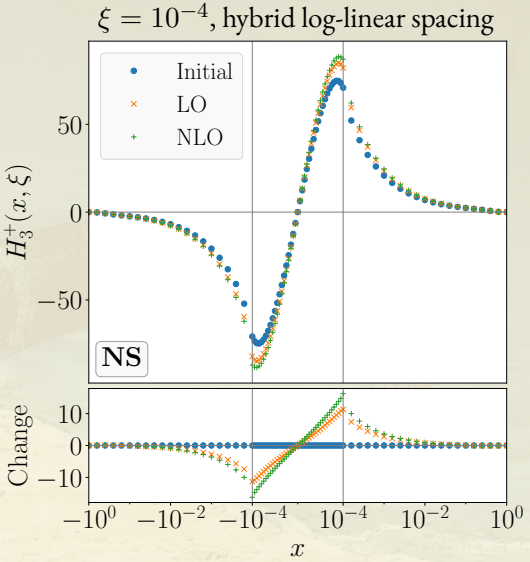
Hybrid log-linear x grid



- User chooses x grid type, via `grid_type` kwarg, when calling `tiktaalik.matrices.set_x_xi_grids`
- `grid_type=2` gives the hybrid spacing.
  - ⇨ Logarithmic when $|x| > |\xi|$
  - ⇨ Linear when $|x| < |\xi|$
- Great for $\xi \lesssim 0.1$
- Okay at $\xi > 0.2$, except for $|x| \to 1$
- Best choice for HERMES & EIC analyses!

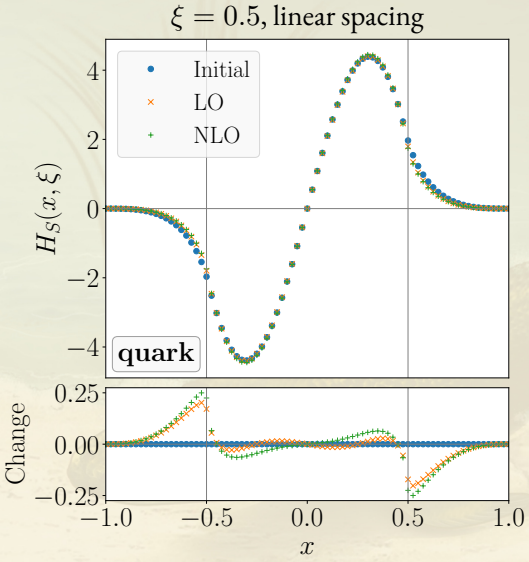
NLO evolution: non-singlet demo

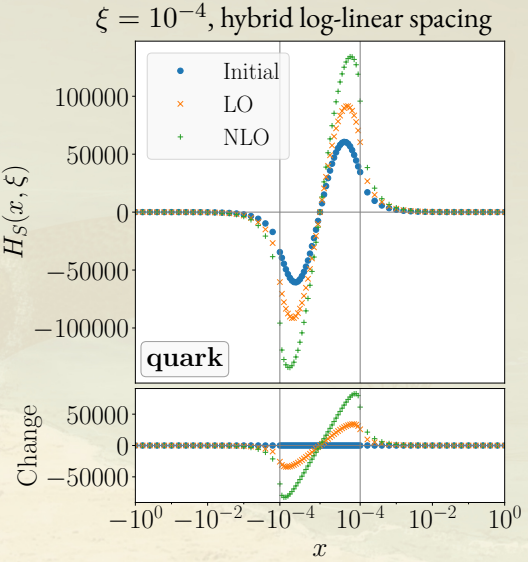




- **tiktaalik features next-to-leading order (NLO) evolution!**

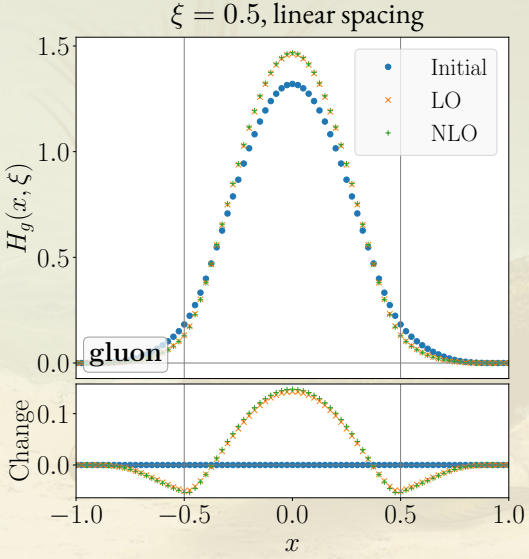
NLO evolution: quark singlet demo

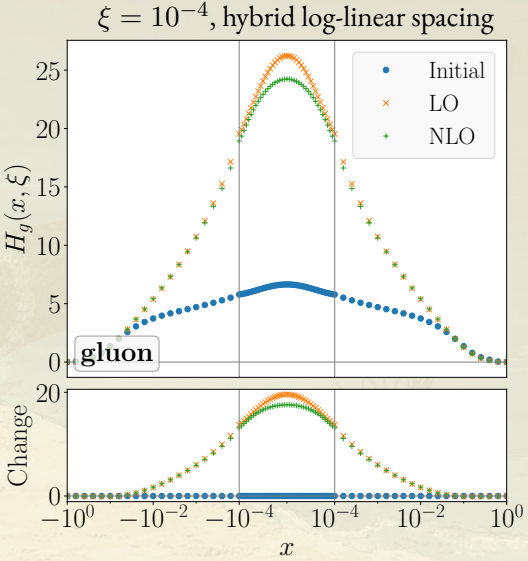




- NLO has a massive impact on singlet quark distribution at small ξ

NLO evolution: gluon demo





- Gluon distribution smaller at NLO

Limitations of tiktaalik

- Lower bound on ξ values where tiktaalik is reliable.
  - **Leading order**: $\xi \gtrsim 3 \times 10^{-6}$
  - **Next-to-leading order**: $\xi \gtrsim 2 \times 10^{-5}$
  - You should use hybrid grid for such low ξ values
- master branch requires $x \neq \pm\xi$
  - Allow $x = \pm\xi$ in dev branch, but more testing needed before we can pull the changes
- The dev branch requires $N_x = 4n + 1$
  - Ensures $x = \pm\xi$ are included on grid for hybrid spacing
  - This requirement *will* be pulled into master branch soon
- Accuracy limitations in DVCS Wilson coefficients
  - Need $N_x \gtrsim 401$ for sub-percent error in LO real part
  - Cannot get sub-percent error in NLO real part
  - Imaginary part under control
- tiktaalik assumes $3 \leq n_{\text{fl.}} \leq 5$


Accuracy benchmarks

Defining accuracy benchmarks

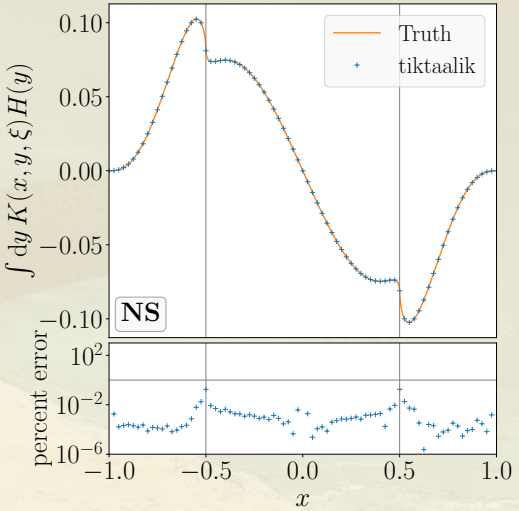
- Accuracy benchmarks done on the integral:

$$S(x,\xi) = \int_{-1}^{+1} \mathrm{d}y\, K(x,y,\xi) H(y,\xi)$$

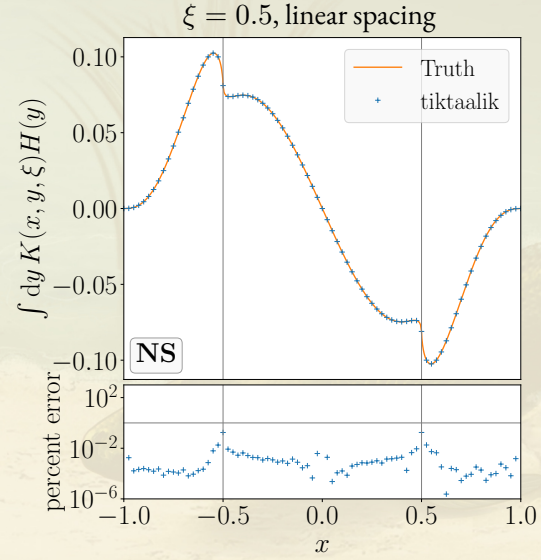
- "Ground truth" from different numerical method.
  - ⇨ We don't have an analytic solution.
  - ⇨ Adaptive quadrature for ground truth.
- Percent error estimated via:

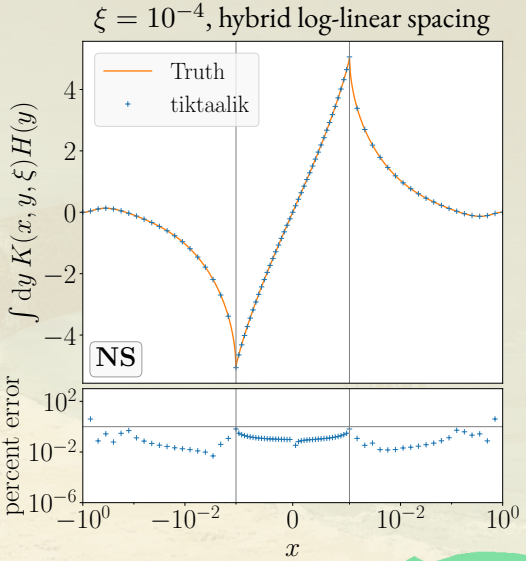
$$100\% \times \frac{|S_{\text{truth}}(x,\xi) - S_{\text{tiktaalik}}(x,\xi)|}{|S_{\text{truth}}(x,\xi)|}$$

  - ⇨ Artificial spikes when truth goes to zero.
- Goloskokov-Kroll model used in benchmarks.



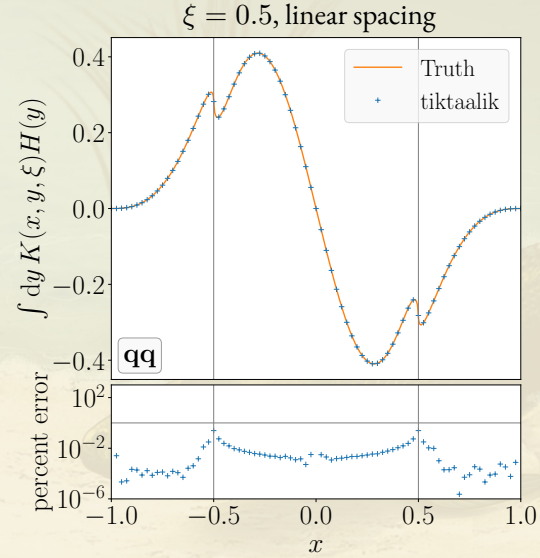
Accuracy benchmarks: non-singlet

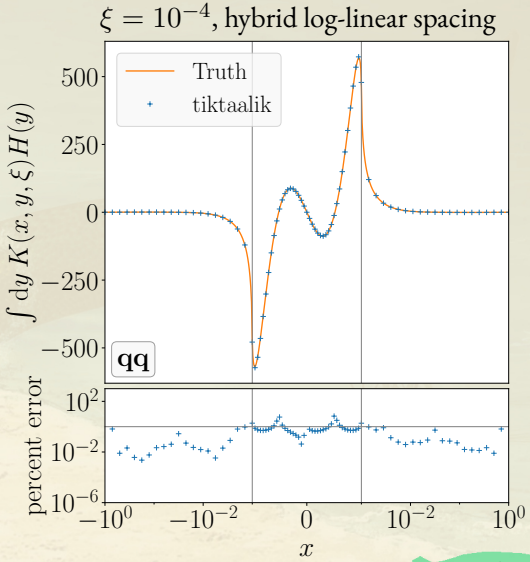




- Benchmarks at **next-to-leading order** (NLO), & $N_x = 81$.

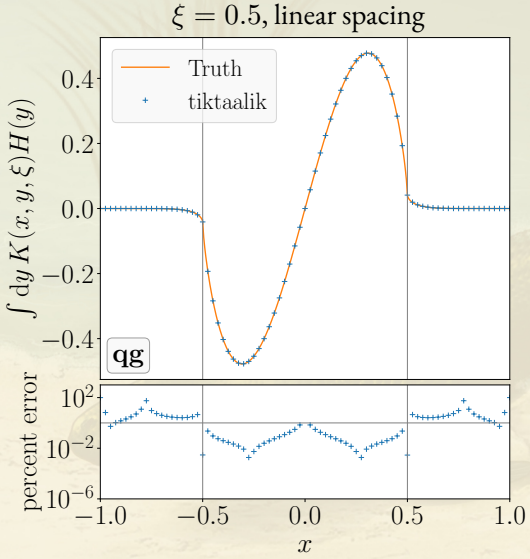
Accuracy benchmarks: quark-from-quark

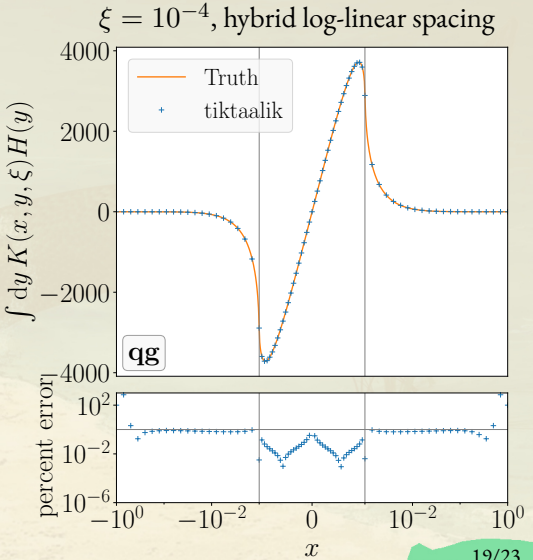




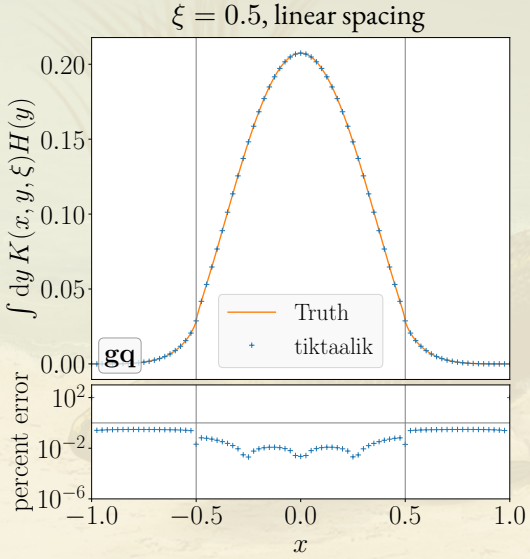
- Some error spikes are artifacts of zero crossings.

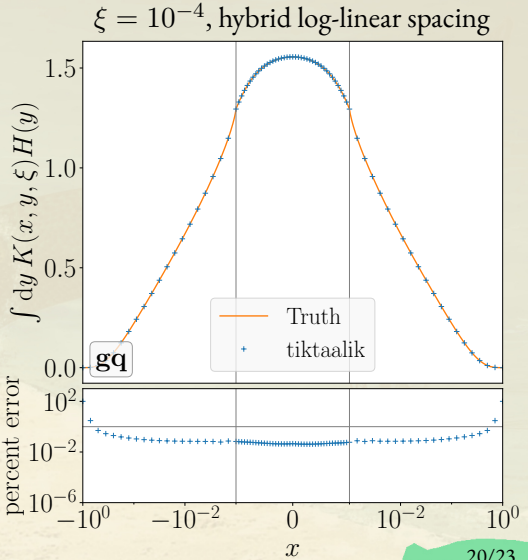
Accuracy benchmarks: quark-from-gluon



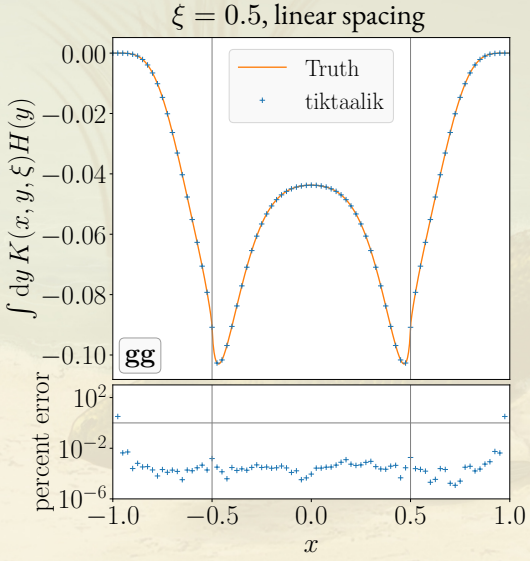


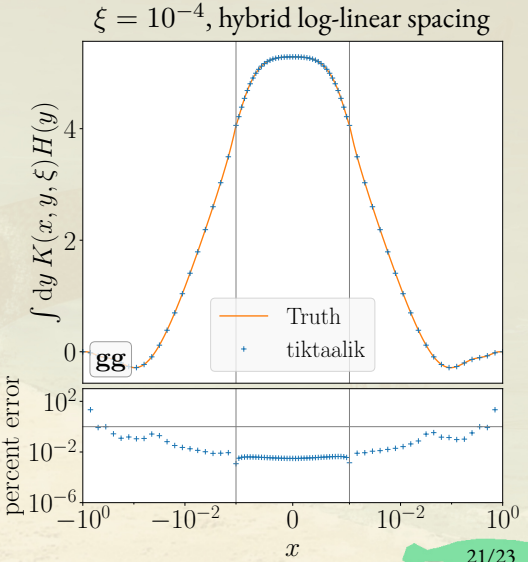
Accuracy benchmarks: gluon-from-quark



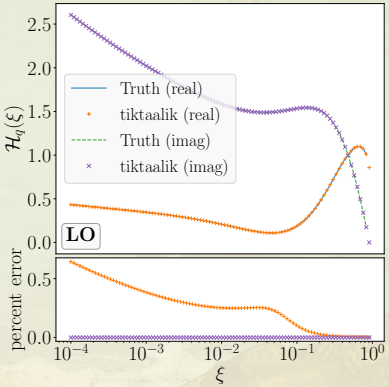


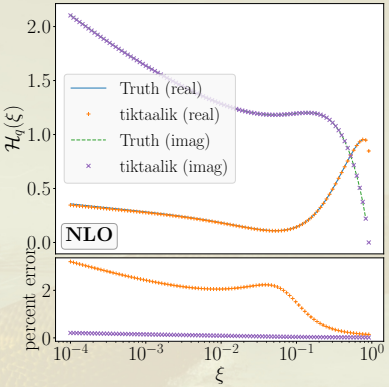
Accuracy benchmarks: gluon-from-gluon

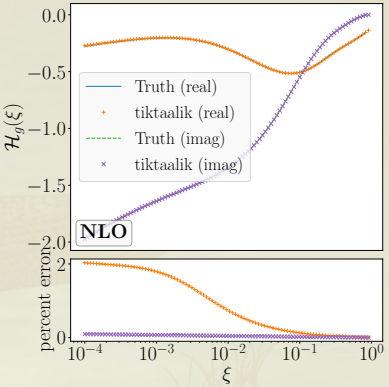




Accuracy benchmarks: Compton form factors







- Matrices to get LO and NLO CFFs available
- Need large $N_x \sim 401$ for sub-percent error in LO real part
- Can't get sub-percent error in NLO real part
- More work needed to improve CFF accuracy


Wrapping Up

The End

- First paper published!
  - Computer Physics Communications 311 (2025) 109552
  - Second paper (on NLO and new features) in progress
- Code package **tiktaalik** is public!
  - `https://github.com/quantom-collab/tiktaalik`
  - **tiktaalik** is open source, & all are welcome to contribute!
  - Latest features in dev branch, but dev branch is unstable.
- **Developers**:
  - AF (lead)
  - Daniel Adamiak
  - Ian Cloët
  - Jian-Wei Qiu
  - Nobuo Sato
  - Marco Zaccheddu

Thank you for your time!