tiktaalik: code for ultrafast GPD evolution

Adam Freese
May 22, 2025
Comp. Phys. Comm. 311 (2025) 109552

Jefferson Lab

GPD evolution code: the needs

- Needs for x-space evolution code:
  - ⇨ **Fast**: for use in global analysis.
  - ⇨ **Differentiable**: for machine learning applications.
  - ⇨ **Standalone**: to be easily usable by anyone (for model calculations, lattice QCD, ...)
- General form of evolution equation:

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

- Numerically solve by discretizing (pixelizing) in x:

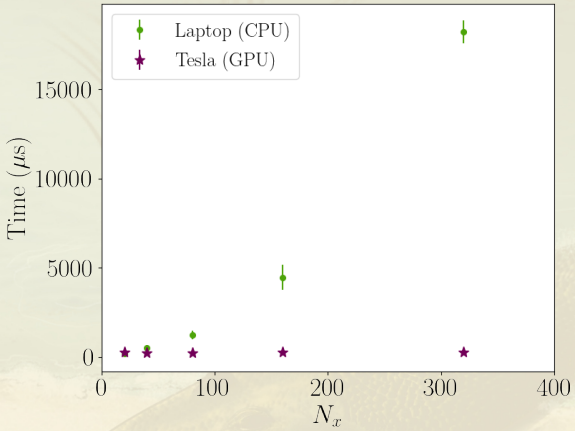
$$\frac{\mathrm{d}H_i(\xi,Q^2)}{\mathrm{d}\log(Q^2)} \approx \sum_j K_{ij}(\xi,Q^2) H_j(\xi,Q^2)$$

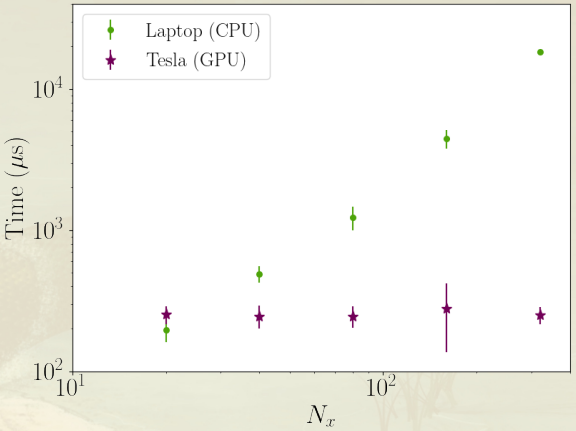
  - ⇨ Becomes a **matrix equation**!
- Solution found via **evolution matrices**:

$$H_i(\xi,Q^2) = \sum_j M_{ij}(\xi, Q_0^2 \to Q^2) H_j(\xi,Q_0^2)$$

  - ⇨ Evolution matrix is **independent of model-scale GPD**.

How fast is fast?





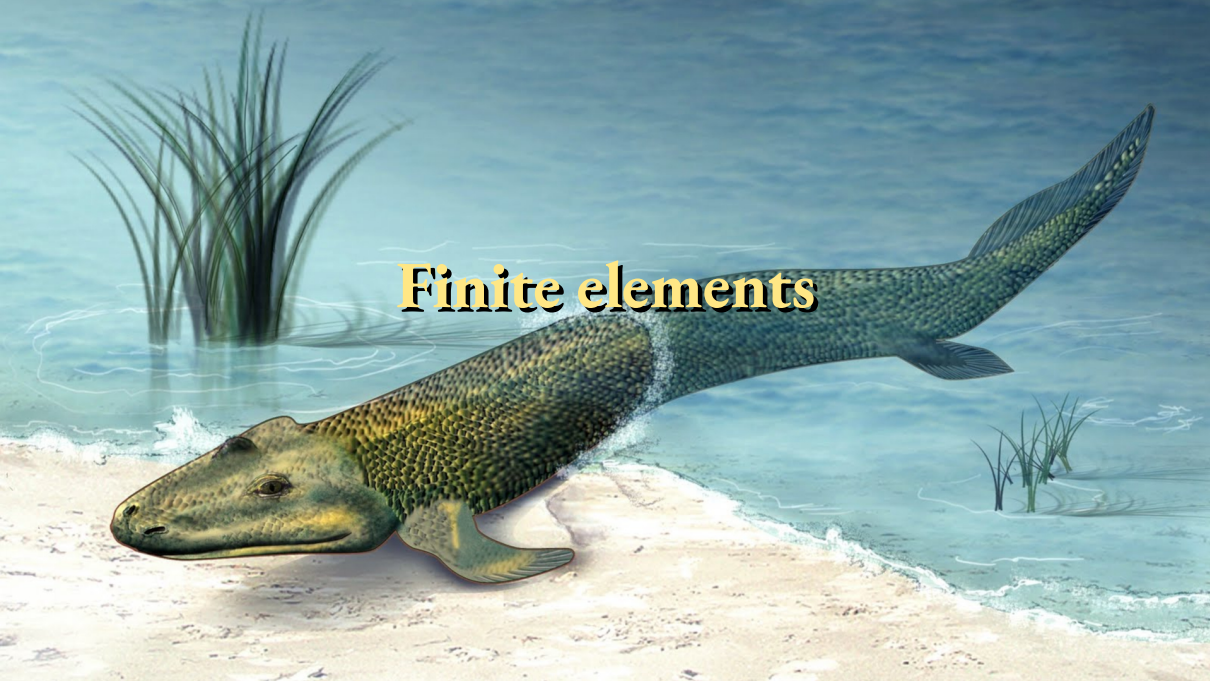
- On a GPU: **microseconds** to evolve a GPD.
- Evolution is just matrix multiplication.
  - Takes more time (milliseconds-seconds) to build matrices ...
  - ...**but this only needs to be done once**.

tiktaalik: code to make evolution matrices

$$H_i(\xi, Q^2) = \sum_j M_{ij}(\xi, Q_0^2 \to Q^2) H_j(\xi, Q_0^2)$$

- **tiktaalik** is code that builds matrices M_{ij} to evolve GPDs.
  - Evolution done in x-space.
  - Method based on finite elements.
  - Easy-to-use Python interface.

- The code is available online!
  - `https://github.com/quantom-collab/tiktaalik`
  - **Can do leading order (LO) and next-to-leading order (NLO) evolution!**


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:

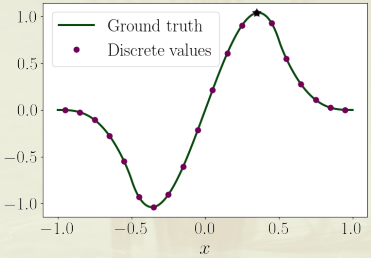
$$\boldsymbol{H} = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_n \end{bmatrix} = H_1 \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + H_2 \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \ldots + H_n \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \equiv H_1\hat{e}_1 + H_2\hat{e}_2 + \ldots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

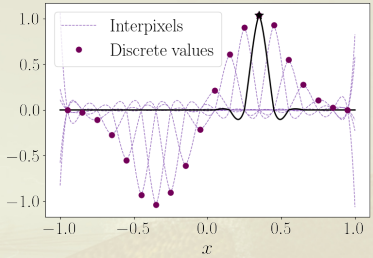
  - Interpolated pixelation is a sum of interpixels!

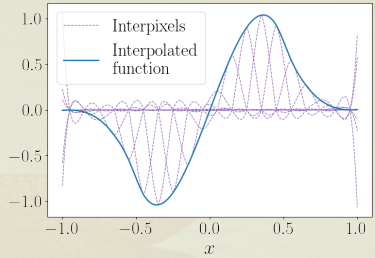
$$P[\boldsymbol{H}](x) = H_1 P[\hat{e}_1](x) + H_2 P[\hat{e}_2](x) + \ldots + H_n P[\hat{e}_n](x)$$

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

Interpixel demo



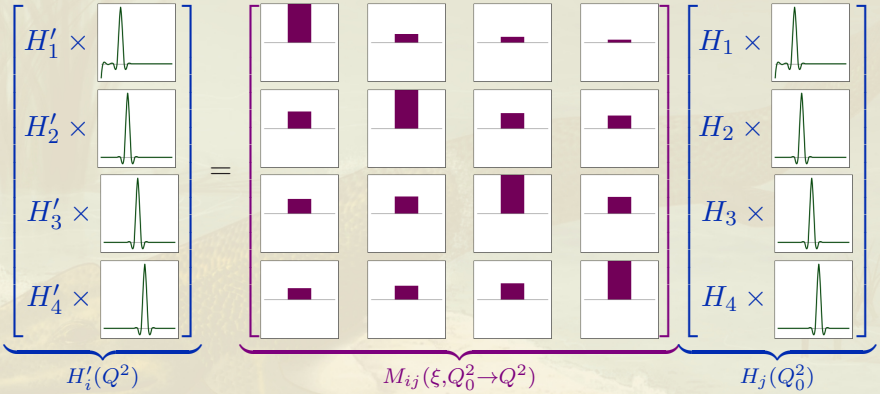




- Interpixel is a *piecewise* polynomial of fixed order.
  - Increase N_x *without* increasing interpolation order (avoids Runge phenomenon).
  - I'm using fifth-order Lagrange interpolation.
  - Knots at the discrete x_i grid points.
- Each interpixel has oscillations.
  - Oscillations cancel in sum.

Evolution matrices: what they do

- **Evolution matrix** says how **interpixels** evolve into **interpixels**.



- **The matrix** depends only on **the interpixels**—independent of **the GPD** itself.
- The **same matrix** can be used for *any* **model GPD**.

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 pieces** with boundaries:

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

  - x and ξ grids must be misaligned.
  - 21-point quadrature used inside each region. (First release & paper used 15-point rule)

$$\int_{-1}^{+1} \mathrm{d}y\, K(x,y,\xi,Q^2) H(y,\xi,Q^2) \approx \sum_{g=1}^{N_g=6\times 21} w_g K(x,y_g,\xi,Q^2) H(y_g,\xi,Q^2)$$

  - 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

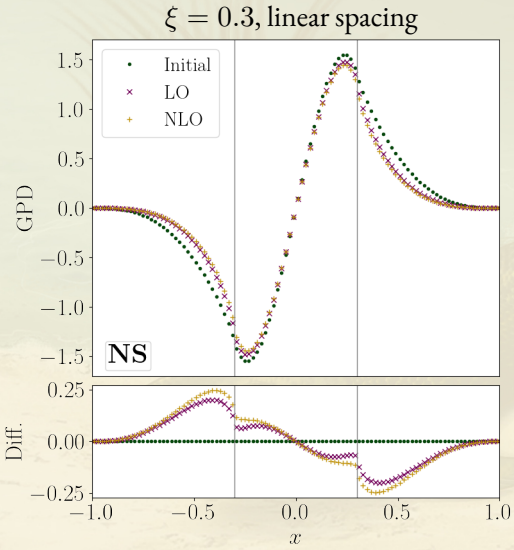
Features of tiktaalik

- 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.
  - Pick grid type when initializing kernel matrices.

- Leading order (LO) and next-to-leading order (NLO) GPD evolution
  - Set evolution order when initializing evolution matrices.
  - Kernel initialization prepares both LO & NLO kernels.
  - Splitting functions from Belitsky, Freund & Müller, Nucl Phys B 574 (2000) 347

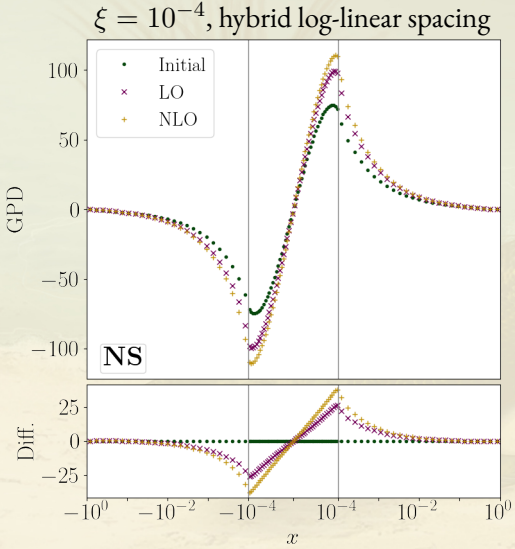
- Easy installation via pip.

Linear x grid



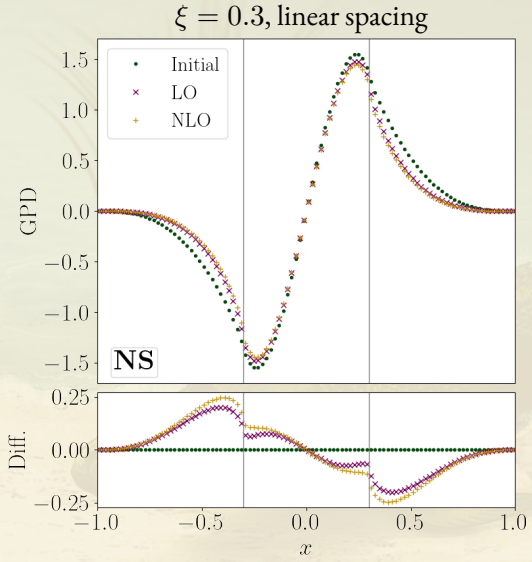
- User chooses x grid type, via `grid_type` kwarg.
- `grid_type=1` (default) gives linear spacing.
- Great for $\xi \gtrsim 0.2$
- Awful at $\xi \lesssim 0.1$

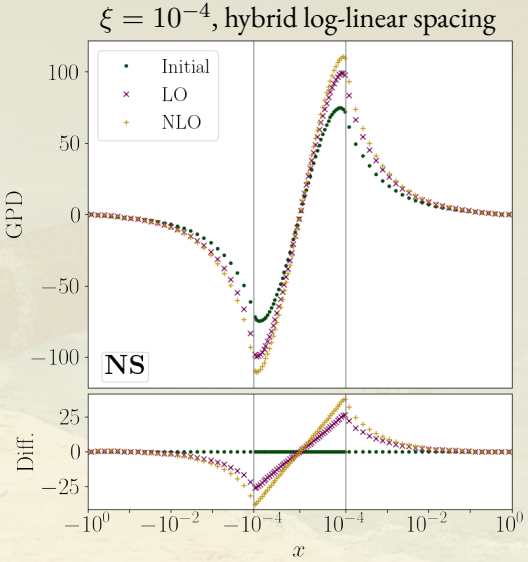
Hybrid log-linear x grid



- User chooses x grid type, via `grid_type` kwarg.
- `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

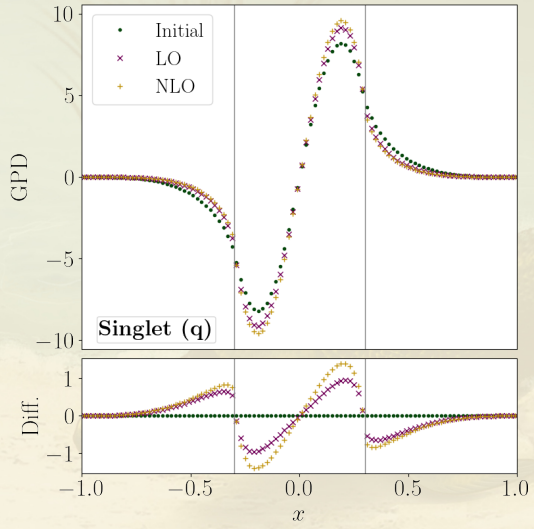




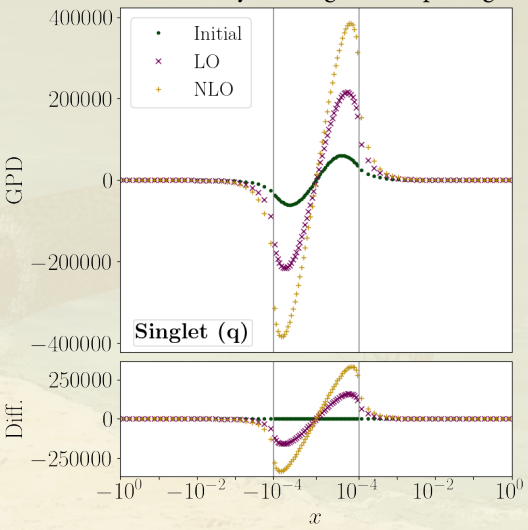
- **tiktaalik features next-to-leading order (NLO) evolution!**

NLO evolution: quark singlet demo

$\xi = 0.3$, linear spacing



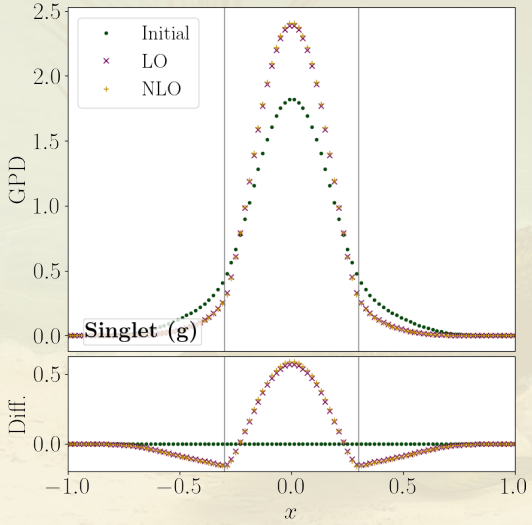
$\xi = 10^{-4}$, hybrid log-linear spacing



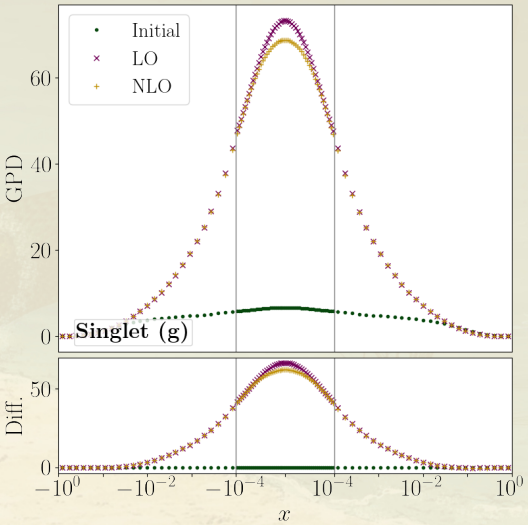
- NLO has a massive impact on singlet quark distribution at small ξ

NLO evolution: gluon demo

$\xi = 0.3$, linear spacing



$\xi = 10^{-4}$, hybrid log-linear spacing



- Gluon distribution smaller at NLO—not sure why.

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 these.

- For linear grids, $x = \xi$ *must be avoided*.
  - Partly due to how integrals are broken up.
  - Partly due to undefined behavior (nans) in function definitions.
  - The hybrid log-linear grids automatically avoid $x = \xi$.

- tiktaalik assumes $3 \leq n_{\text{fl.}} \leq 5$


Benchmarks

Defining accuracy benchmarks

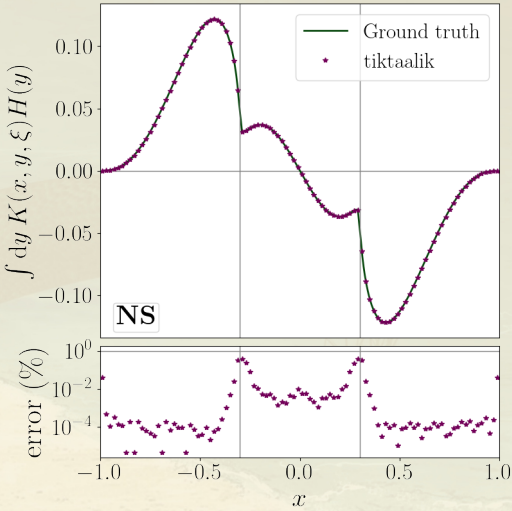
- Accuracy benchmarks done on the integral:

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

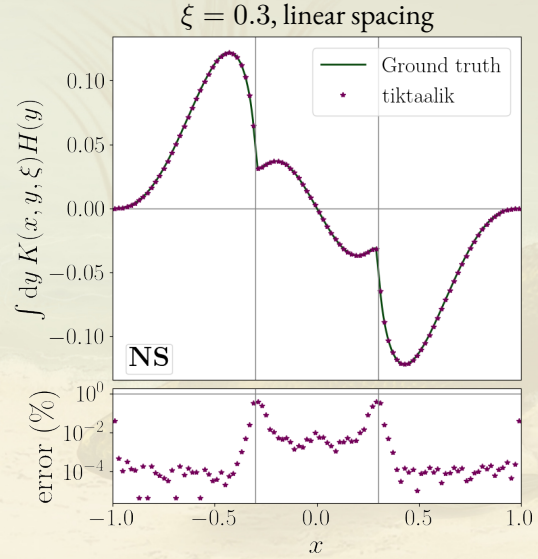
- "Ground truth" from different numerical method.
  - ⇨ We don't have an analytic solution.
  - ⇨ Adaptive quadrature for ground truth.
- Percent error estimated via:

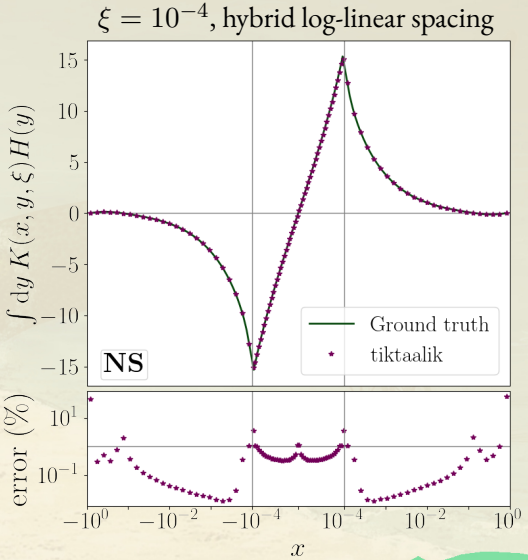
$$100\% \times \frac{|S_{\text{truth}}(x,\xi) - S_{\text{tiktaalik}}(x,\xi)|}{|S_{\text{truth}}(x,\xi)|}$$

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



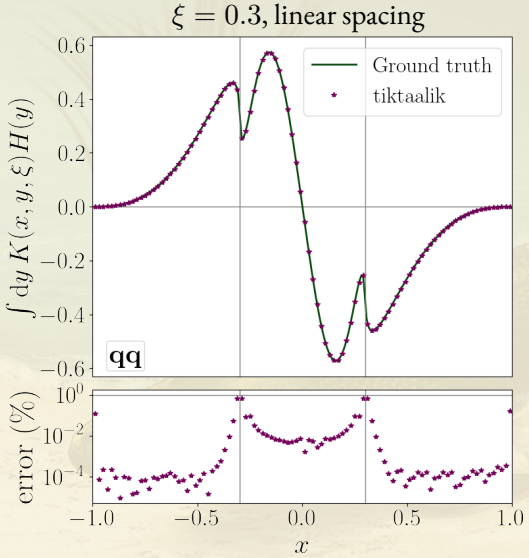
Accuracy benchmarks: non-singlet

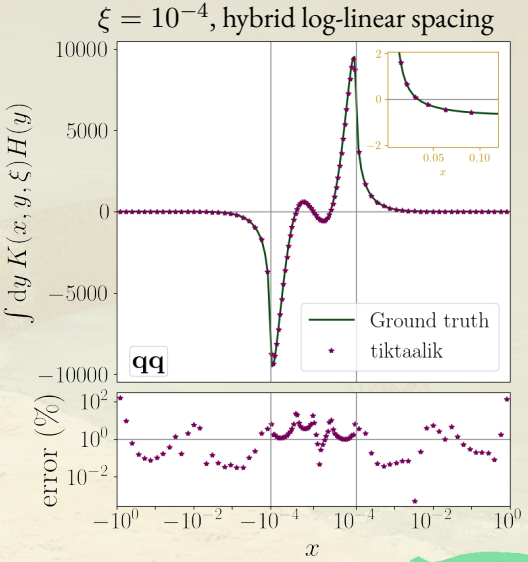




- Benchmarks at **next-to-leading order** (NLO), & $N_x = 100$.

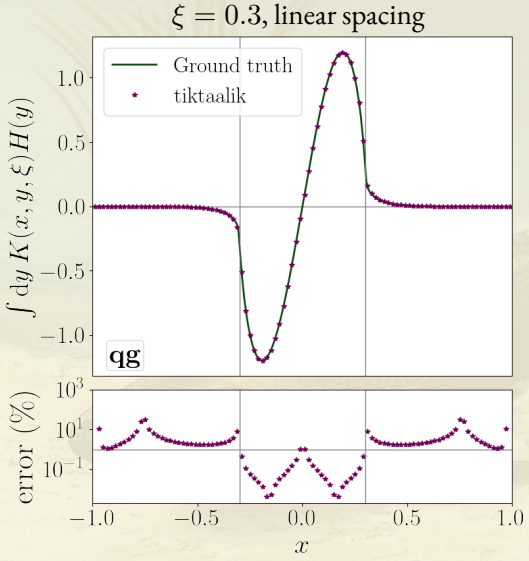
Accuracy benchmarks: quark-from-quark

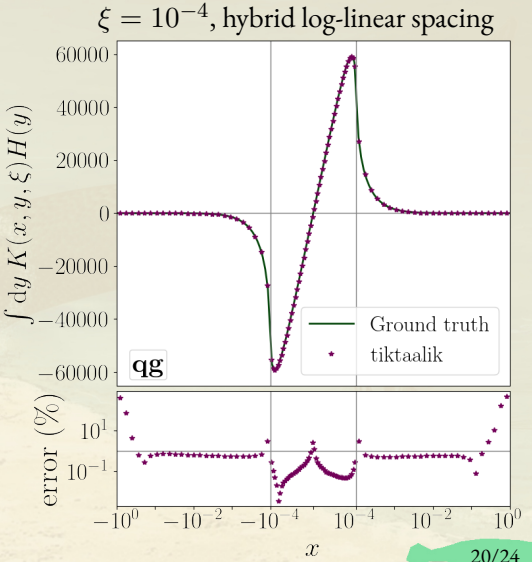




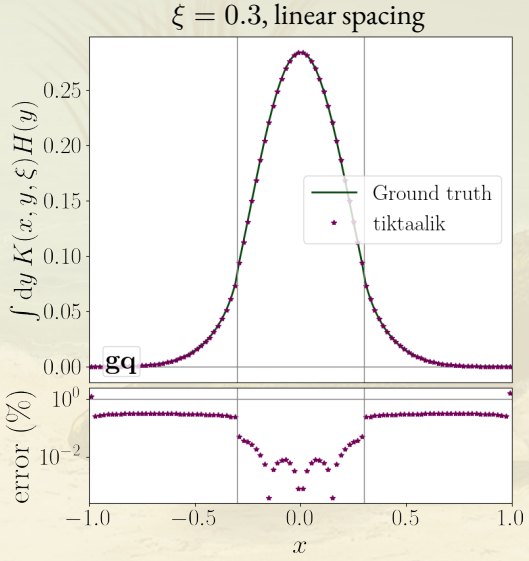
- Some error spikes are artifacts of zero crossings.

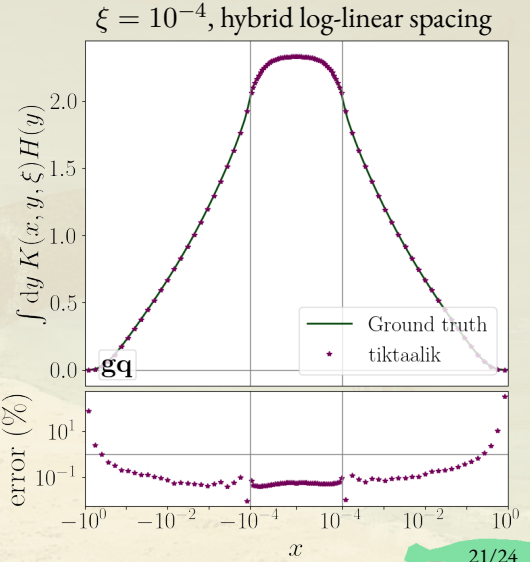
Accuracy benchmarks: quark-from-gluon



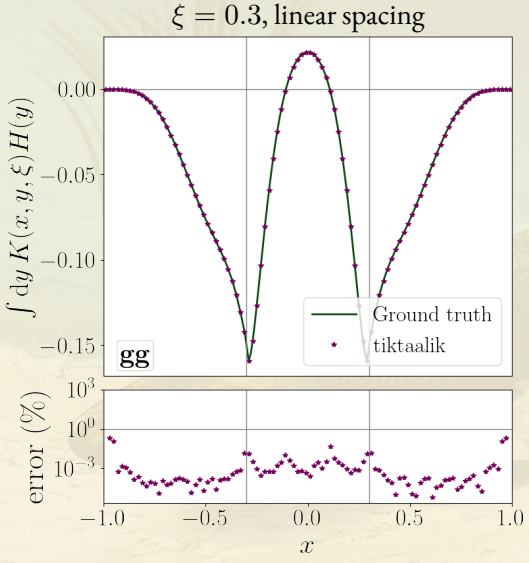


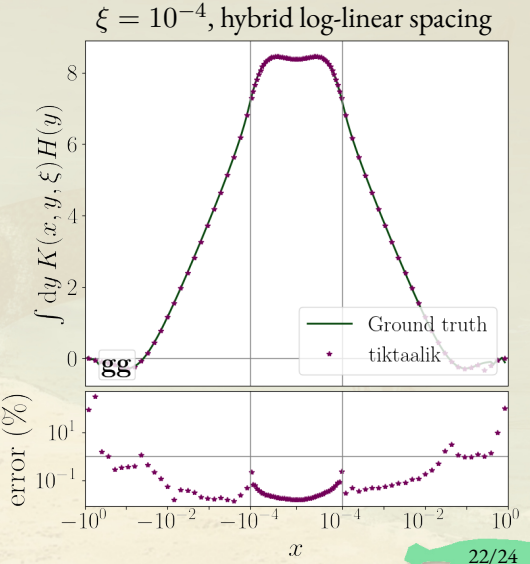
Accuracy benchmarks: gluon-from-quark



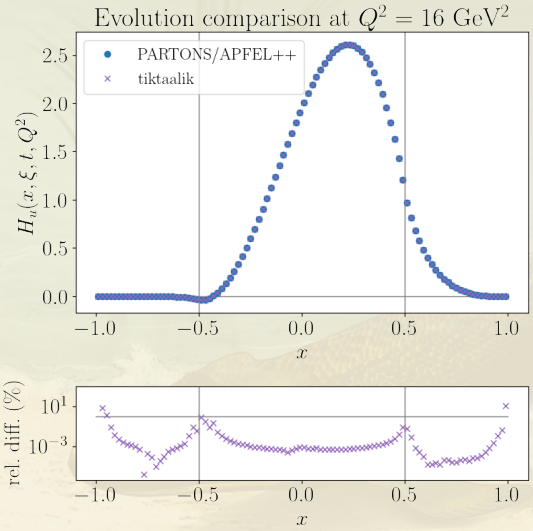


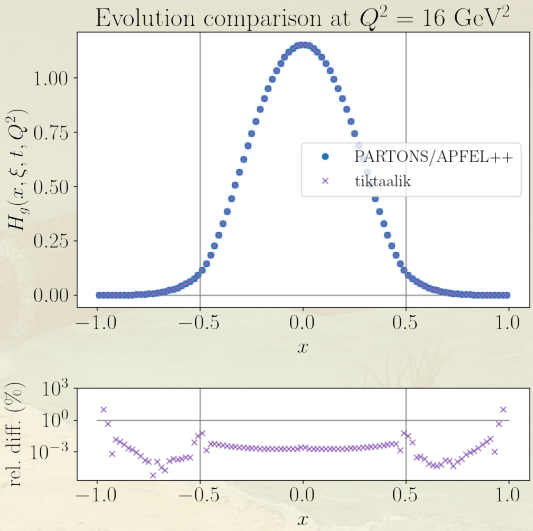
Accuracy benchmarks: gluon-from-gluon





Leading order comparison to PARTONS/APFEL++





- Excellent agreement, but differences $\sim 1\%$ at $x \approx \pm\xi$.
- Comparison done at leading order (LO).


Wrapping up

The End

- First paper published!
  - Computer Physics Communications 311 (2025) 109552
- Code package **tiktaalik** is public!
  - `https://github.com/quantom-collab/tiktaalik`
  - **tiktaalik** is open source, & all are welcome to contribute!
- **Developers**:
  - AF (lead)
  - Daniel Adamiak
  - Ian Cloët
  - Jian-Wei Qiu
  - Nobuo Sato
  - Marco Zaccheddu

Thank you for your time!