tiktaalik: Finite Element Code for the Evolution of Generalized Parton Distributions

Adam Freese
Thomas Jefferson National Accelerator Facility

November 19, 2024

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**.

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`
  - First release only leading order; NLO in progress.
- This talk is about the finite element method behind the code.

Building kernel matrices

Integral discretization

- First step is to discretize the integral:

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

- Kernel made up of three distributions; must be integrated separately:

$$K(x,y,\xi,Q^2) = K_R(x,y,\xi,Q^2) + [K_P(x,y,\xi,Q^2)]_+ + K_C(Q^2)\delta(y-x)$$

  - **Regular piece**—just a normal integral:

$$\int_{-1}^{+1} \mathrm{d}y\, K_R(x,y,\xi,Q^2) H(y,\xi,t,Q^2)$$

  - **Plus distribution piece**:

$$\int_{-1}^{+1} \mathrm{d}y\, [K_P(x,y,\xi,Q^2)]_+ H(y,\xi,t,Q^2) \equiv \int_{-1}^{+1} \mathrm{d}y\, K_P(x,y,\xi,Q^2)\Big(H(y,\xi,t,Q^2) - H(x,\xi,t,Q^2)\Big)$$

$$+ H(x,\xi,t,Q^2) \int_{-1}^{+1} \mathrm{d}y\, \Big(K_P(x,y,\xi,Q^2) - K_P(y,x,\xi,Q^2)\Big)$$

  - **Constant piece** (or delta distribution piece):

$$\int_{-1}^{+1} \mathrm{d}y\, K_C(Q^2)\delta(y-x) H(y,\xi,t,Q^2) \equiv K_C(Q^2) H(x,\xi,t,Q^2)$$

Regular piece

- Regular piece 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.
  - 15-point quadrature used inside each region.

$$S_R(x, \xi, t, Q^2) \approx \sum_{g=1}^{N_g = 6\times 15} w_g K_R(x, y_g, \xi, Q^2) H(y_g, \xi, t, Q^2)$$

  - 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.

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:

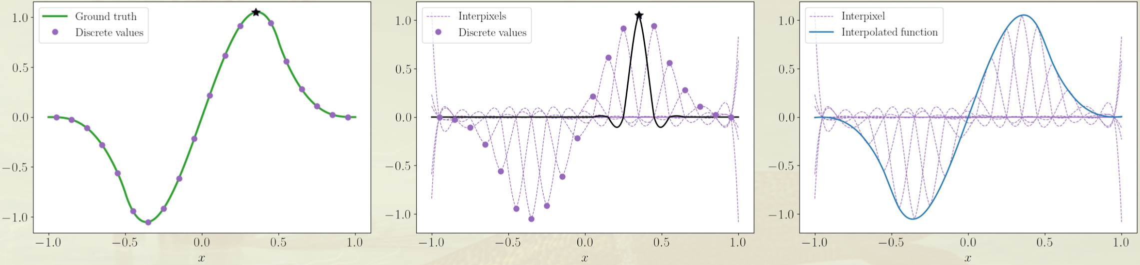
$$\boldsymbol{H} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix} = h_1 \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + h_2 \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \ldots + h_n \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \equiv h_1 \hat{e}_1 + h_2 \hat{e}_2 + \ldots + h_n \hat{e}_n$$

  - 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.

Regular piece: now with interpixels!

- GPD at Gaussian weight points from piecewise polynomial interpolation:

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

  - Interpolation decomposed into basis functions (**interpixels**).
- Integral is only over interpixels:

$$S_R(x,\xi,t,Q^2) \approx \sum_{j=1}^{N_x} \underbrace{\left(\sum_{g=1}^{N_g} w_g K_R(x_i,y_g,\xi,Q^2) P[\hat{e}_j](y_g) \right)}_{\left(K_R(\xi,Q^2)\right)_{ij}} H_j(\xi,t,Q^2)$$

  - Absorb interpixel into kernel matrix.
  - Integral over interpixel **independent of specific GPD**.
  - (Can be generalized: e.g., to adaptive integration.)

Plus distribution piece

- Plus distribution piece is a sum of two integrals:

$$S_P(x,\xi,t,Q^2) \equiv \int_{-1}^{+1} \mathrm{d}y\, [K_P(x,y,\xi,Q^2)]_+ H(y,\xi,t,Q^2) = S_P^{(1)}(x,\xi,t,Q^2) + S_P^{(2)}(x,\xi,t,Q^2)$$

$$S_P^{(1)}(x,\xi,t,Q^2) = \int_{-1}^{+1} \mathrm{d}y\, K_P(x,y,\xi,Q^2)\Big(H(y,\xi,t,Q^2) - H(x,\xi,t,Q^2)\Big)$$

$$S_P^{(2)}(x,\xi,t,Q^2) = H(x,\xi,t,Q^2)\int_{-1}^{+1} \mathrm{d}y\, \Big(K_P(x,y,\xi,Q^2) - K_P(y,x,\xi,Q^2)\Big)$$

- Presents numerical difficulties because of $1/(y-x)$ factors in K_P.

Plus distribution piece (continued)

- Do first integral via Gauss-Kronrod rule still.
  - Break into same six integration regions.
  - Use same fifth-order Lagrange interpolation.
- **Matrix implementation**:

$$S_P^{(1)}(x_i,\xi,t,Q^2) \approx \sum_{j=1}^{N_x} \underbrace{\left(\sum_{g=1}^{N_g} w_g K_P(x_i,y_g,\xi,Q^2) \Big[P[\hat{e}_j](y_g) - \delta_{ij} \Big] \right)}_{\left(K_P^{(1)}(\xi,Q^2)\right)_{ij}} H_j(\xi,t,Q^2)$$

- Second integral (independent of GPD) done analytically:

$$S_P^{(2)}(x_i,\xi,t,Q^2) = \sum_{j=1}^{N_x} \underbrace{\int_{-1}^{+1} \mathrm{d}y \left(K_P(x_i,y,\xi,Q^2) - K_P(y,x_i,\xi,Q^2) \right) \delta_{ij}}_{\left(K_P^{(2)}(\xi,Q^2)\right)_{ij}} H_j(\xi,t,Q^2)$$

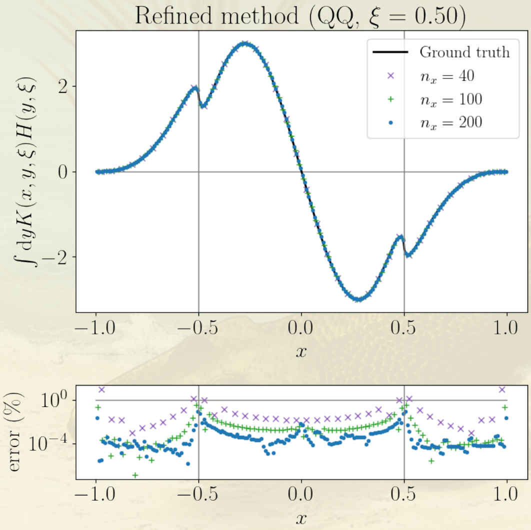
Constant piece

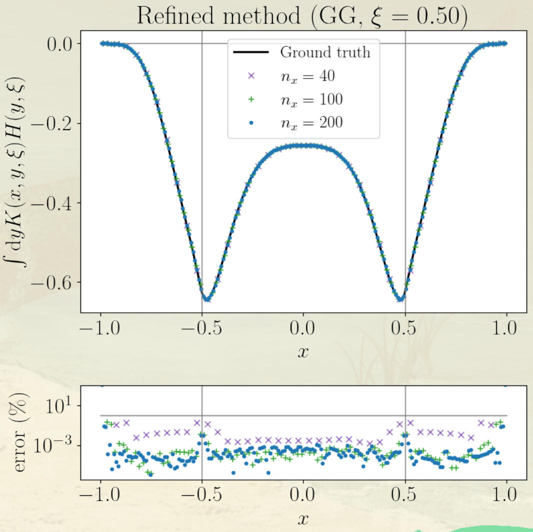
- The constant piece (delta distribution piece) is trivial.

$$S_C(x_i, \xi, t, Q^2) = \int_{-1}^{+1} \mathrm{d}y\, K_C(Q^2)\delta(y - x_i) H(y, \xi, t, Q^2)$$

$$= \sum_{j=1}^{N_x} \underbrace{\left(\delta_{ij} K_C(Q^2)\right)}_{\left(K_C(Q^2)\right)_{ij}} H_j(\xi, t, Q^2)$$

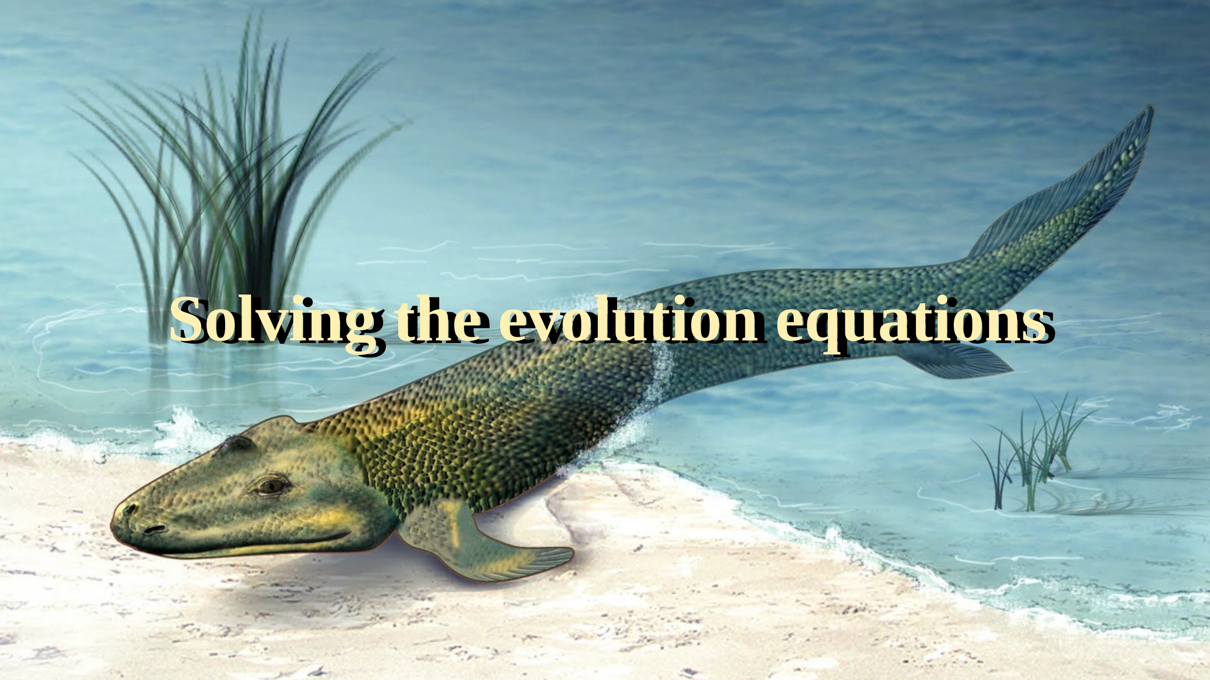
Accuracy benchmarks





- Excellent accuracy, but spikes to $\sim 1\%$ at $x \approx \pm\xi$.

Solving the evolution equations



Differential matrix equation

- Combining pieces gives a matrix form of the evolution kernel:

$$K_{ij}(\xi, Q^2) = \big(K_R(\xi, Q^2)\big)_{ij} + \big(K_P^{(1)}(\xi, Q^2)\big)_{ij} + \big(K_P^{(2)}(\xi, Q^2)\big)_{ij} + \big(K_C(Q^2)\big)_{ij}$$

- Turns evolution equation into a **matrix differential equation**:

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

- This can be solved using Runge-Kutta.

Evolution matrices

- Solution to the evolution equation, via RK4:

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

  - **Evolution matrix**:

$$M_{ij}(\xi, Q^2_{\text{ini}} \to Q^2_{\text{fin}}) = \delta_{ij} + \frac{1}{6} \log \frac{Q^2_{\text{fin}}}{Q^2_{\text{ini}}} \left(M^{(1)}_{ij}(\xi) + 2M^{(2)}_{ij}(\xi) + 2M^{(3)}_{ij}(\xi) + M^{(4)}_{ij}(\xi) \right)$$

  - Build using RK4:

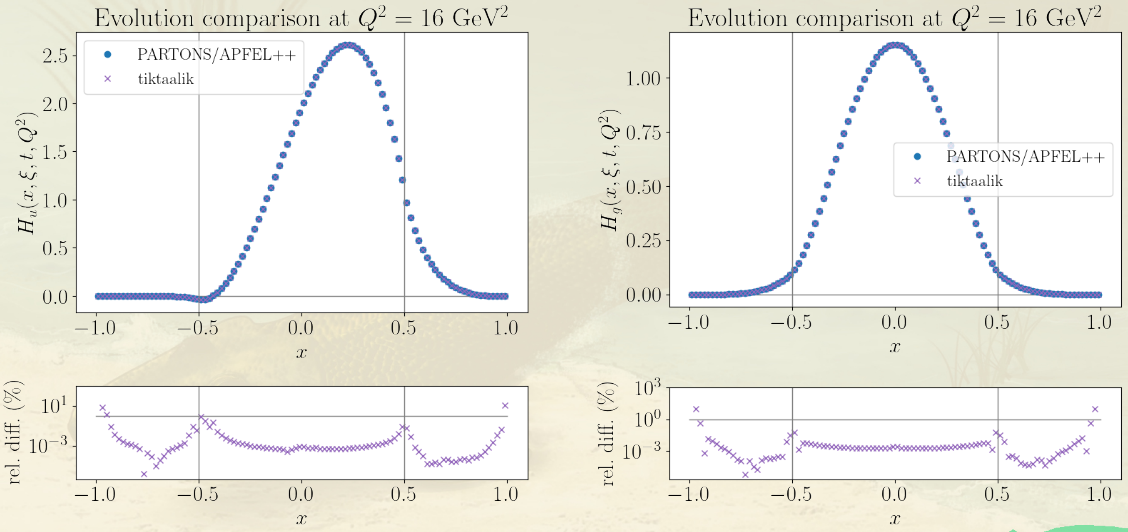
$$M^{(1)}_{ij}(\xi) = K_{ij}(\xi, Q^2_{\text{ini}})$$

$$M^{(2)}_{ij}(\xi) = \sum_{l=1}^{N_x} K_{il}(\xi, Q^2_{\text{mid}}) \left(\delta_{lj} + \frac{1}{2} \log \frac{Q^2_{\text{fin}}}{Q^2_{\text{ini}}} M^{(1)}_{lj}(\xi) \right)$$

$$M^{(3)}_{ij}(\xi) = \sum_{l=1}^{N_x} K_{il}(\xi, Q^2_{\text{mid}}) \left(\delta_{lj} + \frac{1}{2} \log \frac{Q^2_{\text{fin}}}{Q^2_{\text{ini}}} M^{(2)}_{lj}(\xi) \right)$$

$$M^{(4)}_{ij}(\xi) = \sum_{l=1}^{N_x} K_{il}(\xi, Q^2_{\text{fin}}) \left(\delta_{lj} + \log \frac{Q^2_{\text{fin}}}{Q^2_{\text{ini}}} M^{(3)}_{lj}(\xi) \right)$$

Numerical results—comparison to PARTONS/APFEL++



- Excellent agreement, but differences $\sim 1\%$ at $x \approx \pm\xi$.

Outlook

Future work

- Deal with error spikes at $x \approx \pm\xi$.
  - Due to approximating non-analytic function (true GPD) with an analytic function (polynomial).
  - Knots in interpixels are non-analytic; changing grid might help.
  - Could also have non-analytic map between x space and grid space.
- Improve accuracy at small ξ.
  - Code currently only accurate for $\xi \gtrsim 0.1$.
  - Due to using linear x spacing. (Currently exploring alternatives.)
- Include next-to-leading order (NLO) corrections.
- First paper in preparation!
  - Daniel Adamiak, Ian Cloët, Adam Freese, Wally Melnitchouk, Jianwei Qiu, Nobuo Sato, and Marco Zaccheddu, arxiv:2412.xxxx

The End

- Code package **tiktaalik** is public!
  - `https://github.com/quantom-collab/tiktaalik`
  - First release only leading order; NLO in progress.

Thank you for your time!