The Journal of Open Source Software

tParton: A Python package for next-to-leading order evolution of transversity parton distribution functions

Congzhou M Sha [1¶] and **Bailing Ma** [2]

1 Penn State College of Medicine, Hershey, PA 17033, USA **2** Wake Forest University School of Medicine, Winston-Salem, NC 27101, USA ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Parton distribution functions (PDFs) describe the probability of finding quarks and gluons (collectively called partons) within hadrons such as protons and neutrons. These functions are fundamental to our understanding of quantum chromodynamics (QCD) and are essential for interpreting high-energy physics experiments. The transversity PDF, which encodes information about the transverse spin structure of hadrons, is particularly challenging to measure experimentally and has been less studied computationally compared to unpolarized and helicity PDFs.

`tParton` is a Python package that implements two distinct methods for solving the Dokshitzer–Gribov–Lipatov–Altarelli–Parisi (DGLAP) evolution equations for transversity PDFs at leading order (LO) and next-to-leading order (NLO) in perturbative QCD. The package provides both a command-line interface and a Python API, making it accessible for both quick calculations and integration into larger analysis workflows.

Statement of need

PDFs must be evolved from one energy scale to another to enable comparisons between different experiments and theoretical predictions. While numerous codes exist for evolving unpolarized and helicity PDFs (such as QCDNUM (Botje, 2011), EKO (Candido et al., 2022), HOPPET (Salam & Rojo, 2009), and APFEL++ (Bertone et al., 2014, 2017)), options for transversity PDF evolution are limited. The original Fortran implementation by Hirai et al. (Hirai et al., 1998) is nearly 30 years old and no longer accessible. APFEL++ (Bertone et al., 2017) provides an implementation using direct numerical integration, but no publicly available code has implemented the alternative Mellin moment method proposed by Vogelsang (Vogelsang, 1998).

`tParton` fills this gap by providing:

1. **Two complementary methods**: A direct integration method (following Hirai et al.) and a Mellin moment method (following Vogelsang), allowing users to choose based on their accuracy and computational needs.
2. **Modern Python implementation**: Built on NumPy (Harris et al., 2020) and SciPy (Virtanen et al., 2020), `tParton` is easy to install via pip and integrates seamlessly with the Python scientific computing ecosystem.
3. **Comprehensive validation**: The package includes extensive examples and validation against both Mathematica implementations and APFEL++ results, with detailed discussion of discretization effects and method comparisons.

4. **Dual interface**: Both command-line tools for standalone use and importable modules for integration into larger projects.

The package is aimed at researchers in hadronic physics, particularly those analyzing semi-inclusive deep inelastic scattering experiments and studying nucleon spin structure. It has been validated and documented in a detailed preprint (Sha & Ma, 2025).

Implementation

`tParton` implements the DGLAP evolution equation for the transversity PDF:

$$\frac{\partial}{\partial t}\Delta_T q^{\pm}(x,t) = \frac{\alpha_s(t)}{2\pi}\Delta_T P_{q^{\pm}}(x) \otimes \Delta_T q^{\pm}(x,t)$$

where $t = \ln Q^2$, Q^2 is the energy scale, $\Delta_T P_{q^{\pm}}$ is the transversity splitting function, and $\otimes$ denotes Mellin convolution defined by:

$$f(x) \otimes g(x) := \int_x^1 \frac{dy}{y} f\left(\frac{x}{y}\right) g(y)$$

Method 1: Direct integration (Hirai method)

The first method discretizes both the momentum fraction x and energy scale Q^2 into grids and solves the integro-differential equation using the Euler method for Q^2 evolution and Simpson's rule for x integration. This approach is straightforward but can be computationally expensive for fine grids.

Method 2: Mellin moment method (Vogelsang method)

The second method exploits the convolution theorem for Mellin transforms. The solution is expressed in terms of Mellin moments:

$$\mathcal{M}[\Delta_T q^{\pm}](Q^2; s) = K(s, Q^2, Q_0^2)\mathcal{M}[\Delta_T q^{\pm}](Q_0^2; s)$$

where K contains the evolution kernel depending on the splitting function moments. Since analytic expressions for the evolution kernel are available, this method obviates the need to solve the ODE. The evolved PDF is reconstructed via inverse Mellin transform using the Cohen contour method. This approach is typically faster for evaluation of the transversity PDF at single points, and less sensitive to discretization for smooth PDFs.

Both methods support LO and NLO evolution with numerically exact or analytical forms of the running coupling constant $\alpha_s(Q^2)$. See our arXiv preprint for detailed computational complexity analysis (Sha & Ma, 2025).

Examples and validation

The package includes extensive Jupyter notebooks in the `examples/` directory that:

- Generate initial transversity distributions based on literature models (Hirai et al., 1998)
- Compare both evolution methods against each other and against APFEL++
- Demonstrate sensitivity to numerical parameters (grid resolution, timesteps)
- Reproduce figures from the associated preprint (Sha & Ma, 2025)

A separate Mathematica notebook validates the analytical expressions for the Mellin moments of the splitting functions, providing an independent check of the theoretical framework. The key validation figures from our preprint are shown below:

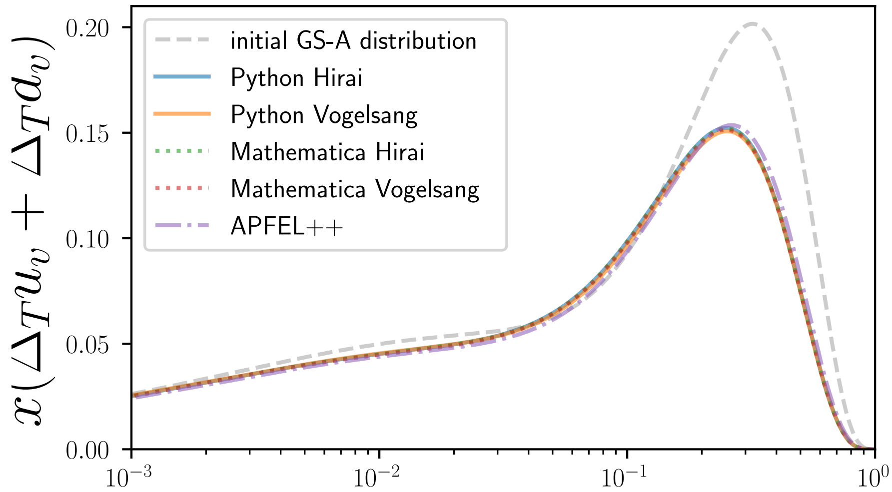

Figure 1: Evolution of the initial GS-A type PDF with numerically evolved $\alpha_s(Q^2)$, comparing the Hirai method (Python), the Vogelsang method (Python), Mathematica validation, and APFEL++ evolution. APFEL++ evolution data was provided by V Bertone.

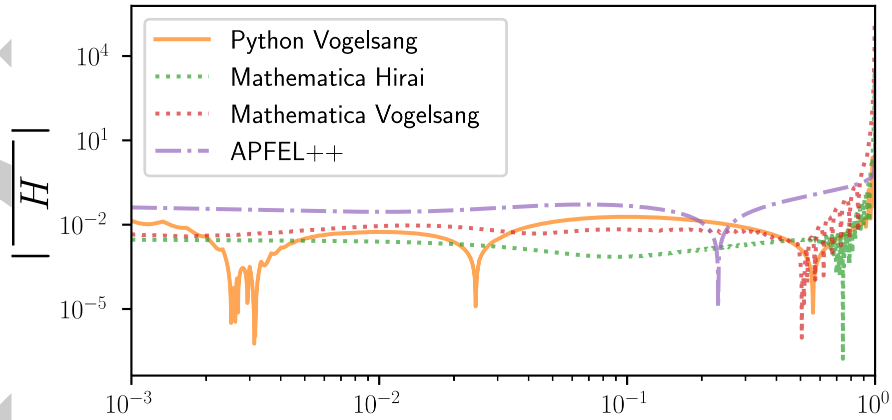

Figure 2: Relative error of the various methods, with the Hirai method (Python) as control

Users can evolve a PDF with a single command:

```
python -m tparton m input.dat 3.1 10.6 --morp plus -o output.dat
```

Or import and use the package programmatically:

```python
from tparton.m_evolution import evolve
result = evolve(input_pdf, Q0_squared=3.1, Q_squared=10.6,
                morp='plus', order='NLO')
```

Complete online documentation of the API and detailed examples are available on the GitHub Pages associated with the repository.

Acknowledgements

We acknowledge helpful discussions with colleagues in the hadronic physics community and thank the maintainers of APFEL++ for providing comparison benchmarks.

References

Bertone, V., Carrazza, S., & Nocera, E. R. (2017). APFEL++: A new PDF evolution library in C++14. *European Physical Journal C*, *77*(8), 516. https://doi.org/10.1140/epjc/s10052-017-5088-y

Bertone, V., Carrazza, S., & Rojo, J. (2014). APFEL: A PDF evolution library with QED corrections. *Computer Physics Communications*, *185*, 1647–1668. https://doi.org/10.1016/j.cpc.2014.03.007

Botje, M. (2011). QCDNUM: Fast QCD evolution and convolution. *Computer Physics Communications*, *182*(2), 490–532. https://doi.org/10.1016/j.cpc.2010.10.020

Candido, A., Hekhorn, F., & Magni, G. (2022). EKO: Evolution kernel operators. *European Physical Journal C*, *82*(10), 976. https://doi.org/10.1140/epjc/s10052-022-10878-w

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hirai, M., Kumano, S., & Miyama, M. (1998). Numerical solution of Q2 evolution equation for the transversity distribution $\Delta_T q$. *Computer Physics Communications*, *111*(1), 150–166. https://doi.org/10.1016/S0010-4655(98)00028-9

Salam, G. P., & Rojo, J. (2009). A higher order perturbative parton evolution toolkit (HOPPET). *Computer Physics Communications*, *180*, 120–156. https://doi.org/10.1016/j.cpc.2008.08.010

Sha, C. M., & Ma, B. (2025). *tParton: Implementation of next-to-leading order evolution of transversity parton distribution functions*. https://arxiv.org/abs/2409.00221

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Vogelsang, W. (1998). Next-to-leading order evolution of transversity distributions and Soffer's inequality. *Physical Review D*, *57*, 1886–1894. https://doi.org/10.1103/PhysRevD.57.1886