SignatureTensors.jl: A Package for Signature Tensors in Julia

Gabriel Riffo and Leonard Schmitz
Institut für Mathematik, Technische Universität Berlin
10623 Berlin, Germany
riffo@tu-berlin.de , lschmitz@math.tu-berlin.de

Abstract

We introduce `SignatureTensors.jl`, a new package for computing signature tensors of paths in `julia`. We present its core functionality and demonstrate its use through illustrative examples. The package is compatible with the computer algebra system `OSCAR`, enabling both exact and numerical computations with signatures.

1 Introduction

Path signatures are fundamental objects in rough path theory [FV10] and serve as a non-commutative feature that captures the essential geometry of sequential data. Recently, a tangible link to algebraic geometry [AFS19] was established through the study of signature varieties associated with specific families of paths. This viewpoint proved particularly useful for studying the problem of learning paths from their signature tensors [PSS19].

For this purpose, a practical and easily extendable package within a modern computer algebra system is required, providing access to multivariate arrays, Gröbner bases, Lie theory, non-commutative polynomials, and other structures. We introduce `SignatureTensors.jl`, a new package that leverages the symbolic computation capabilities of `OSCAR` [DEF+24], a modern open source computer algebra system written in `julia`. The package provides a general framework for computing and manipulating path signatures using algebraic methods while seamlessly interacting with the `OSCAR` ecosystem.

The previous approach [RG20] based on the structural properties of free Lie algebras develops a `python` package to efficiently compute signatures of paths. Several frameworks for machine learning tasks are provided in [DK24, DISW26]. Signature barycenters are efficiently computed using log-coordinates and symbolic preprocessing in [CDM+24], based on BCH series and Gröbner bases of modules in `sage`. The recent `Macaulay2` package [ASLF26] is close to our approach and implements several parts of [AFS19]. Our package is compatible with the `OSCAR` code accompanying [AS25, Sch25]. Our long-term goal is for `SignatureTensors.jl` to become a foundational tool for theoretical research on signature tensors, while providing a flexible base framework for interdisciplinary application, such as time series, spatial data, and more. The code and documentation are available at https://leonardschmitz.github.io/SignatureTensors.jl. The package is registered in the Julia General Registry and can be installed via `Pkg.add("SignatureTensors")`.

2 Path signatures

We give a short introduction of our implementation of signatures. For more details on signatures in general, we refer to [FV10]. For fixed dimension $d \geq 2$ and truncation level $k \in \mathbb{N}$ we consider the truncated tensor algebra

$$T_{d,k} := \bigoplus_{\ell=0}^{k} (\mathbb{R}^d)^{\otimes \ell} = \mathbb{R} \oplus \mathbb{R}^d \oplus \mathbb{R}^{d \times d} \oplus \cdots \oplus \mathbb{R}^{d \times \cdots \times d}$$

serving as our $\frac{d^{k+1}-1}{d-1}$ dimensional ambient space. Let $X : [0,1] \to \mathbb{R}^d$ be a (smooth enough) continuous *path* such that all integrals in Equation (1) below are defined. The iterated-integrals signature is a sequence of tensors $\sigma^{\leq k}(X) := 1 \oplus \sigma^{(1)}(X) \oplus \cdots \oplus \sigma^{(k)}(X) \in T_{d,k}$ with entries,

$$\sigma^{(\ell)}(X)_{w_1,\ldots,w_\ell} := \int_{0 \leq t_1 \leq \cdots \leq t_\ell \leq 1} \dot{X}_{w_1}(t_1) \ldots \dot{X}_{w_\ell}(t_\ell) \, \mathrm{d}t_1 \ldots \mathrm{d}t_\ell \tag{1}$$

for $1 \leq w_1, \ldots, w_\ell \leq d$. For every path X, its signature $\sigma^{\leq k}(X)$ lies in the free nilpotent Lie group $\mathcal{G}_{d,k}$ of step k over $\mathbb{R}^d$; see [FV10, Theorem 7.30]. We introduce two new structures `TruncatedTensorAlgebra{R}` and `TruncatedTensorAlgebraElem{R,E}` that implement the algebra of truncated tensor sequences $T_{d,k}$ and its elements, with a flexible type for its coefficients. The constructors allow several base algebras, e.g., rational numbers or polynomial rings in `OSCAR`. We also support `julia` floats.

We implement (truncated) signatures of several path types, such as moment, axis, and polynomial paths, via the function `sig`. Some of these path types require additional arguments. For example, the option `geom_type=:pwln` for piecewise linear paths requires the mandatory argument `coef` that contains the coefficients for the linear segments. Here, the optional keyword `algorithm::Symbol` specifies two algorithms: by default, we use `algorithm=:Chen`, which iteratively applies Chen's identity [FV10, Theorem 7.11], requiring $\mathcal{O}(md^k)$ operations. If `algorithm=:congruence`, we use matrix–tensor congruence [PSS19, Equation (3)] and require $\mathcal{O}(m^k + md^k)$ elementary operations. We illustrate wall clock times for these options in Section 3. More generally, the option `geom_type=:spline` implements piecewise polynomial paths with regularity constraints. Here we require the mandatory arguments `coef::AbstractMatrix` and `composition::Vector{Int}`, containing the coefficients using matrix–tensor congruence; see [AFS19, Corollary 4.11].

An important inverse problem is the task of recovering a path from its signature. Recall from [AFS19] that the image of the signature under piecewise linear paths with m segments is a semi-algebraic set in $T_{d,k}$. From [PSS19] we reduce the question whether an element S is the signature of a piecewise linear path with m segments to a polynomial system in md variables with $(d^{k+1}-1)/(d-1)$ equations, each of degree k or lower. For further details, we refer to [AS25, Section 6] with a notation closest to ours. Note that we learn with respect to the truncated sequence of signature tensors. With [AFS19, Corollary 6.2] we obtain this sequence from its highest projection; see also [ALS26, Remark 3.5] for details.

In the following, we demonstrate learning within our package. This example shows that there are four (potentially complex) solutions corresponding to piecewise linear paths with four segments and the same signature as S.

```
julia> using Oscar, SignatureTensors;
julia> A = QQ[ 6  -2  6   -10; 7  -4  10  -4];          # dimension d=2, m=4 segments
julia> R, a = polynomial_ring(QQ, :a => (1:2, 1:4));
julia> T = TruncatedTensorAlgebra(QQ,2,4);              # truncation level k=4
julia> S = sig(T,:pwln,coef=A);                  # S signature of piecewise linear path
julia> aC = sig(T,:pwln,coef=a);                 # aC signature with variable segments
julia> I = ideal(R,vec(S-aC));                   # ideal generated by 31=2^5-1 relations
julia> dim(I), degree(I)                         # dimension and degree of I
(0, 4)
```

If the dimension of the resulting vanishing ideal is zero, and its degree one, then we have a unique path recovery. This has recently been formalized by the notion of *path recovery degree*; see [ALS26, Definition 6.1]. Then, the command `recover` produces the coefficients of the piecewise linear path. The optional argument `core` allows other core tensors. If $k = 3$ and the coefficient matrix is invertible, then the solution is unique by [PSS19, Theorem 6.2]. With the option `algorithm=:Sch25` we support the algorithm from [Sch25] which requires $\mathcal{O}(d^4)$ elementary operations in expectation.

3 Benchmarks

Benchmarks were conducted on a MacBook Pro (Model Identifier: Mac17,2) equipped with an Apple M5 chip with 10 CPU cores (4 performance and 6 efficiency cores) and 24 GB of RAM, running macOS 26.0 and Julia 1.12.3. The reported timings correspond to wall-clock times in milliseconds (rounded down). Timings were measured using `BenchmarkTools.jl`. For each parameter configuration we report the median wall-clock time across 100 samples. All experiments were carried out over `base_algebra=Float64`. For path signatures and for each triple (d, m, k), we generated random $d \times m$ integer matrices for `coef` with entries sampled uniformly from $[-20, 20]$. In Table 1 we report on the wall-clock times for `sig` with different algorithms. Further benchmarks are provided in the associated repository.

$d\backslash m$	10	20	30	40	50	60
10	0, 0	0, 0	**0**, 1	**0**, 2	**0**, 5	**1**, 9
20	0, 0	0, 0	**0**, 1	**1**, 2	**1**, 5	**1**, 9
30	0, 0	1, **0**	2, **1**	2, 2	**3**, 5	**4**, 9
40	1, **0**	2, **0**	3, **1**	5, **3**	6, **5**	**7**, 9
50	2, **0**	5, **0**	7, **1**	10, **3**	20, **5**	39, **9**
60	4, **0**	9, **0**	21, **1**	41, **3**	55, **6**	60, **10**

$k = 3$

$d\backslash m$	10	20	30	40	50	60
10	0, 0	**1**, 4	**1**, 20	**2**, 64	**3**, 188	**3**, 525
20	4, **0**	8, **4**	**20**, 21	**40**, 76	**49**, 189	**57**, 541
30	58, **1**	243, **6**	337, **23**	372, **82**	412, **200**	**458**, 558
40	292, **3**	360, **9**	470, **31**	726, **102**	839, **219**	1104, **566**
50	444, **11**	913, **23**	1307, **66**	1742, **120**	2127, **311**	2437, **582**
60	1159, **39**	2126, **67**	2860, **96**	4536, **153**	7148, **434**	8602, **636**

$k = 4$

Table 1: Timings (in milliseconds) for signatures of piecewise linear paths with m segments using `algorithm=:Chen` (first number) and `algorithm=:congruence` (second number).

Faster timings are highlighted in bold; Congruence outperforms Chen in high dimensions. Our timings are comparable with [RG20], e.g., the `numpy` library `iisignature` requires 6 and 654 milliseconds in the above setup for $d = m = 60$ and $k \in \{3, 4\}$. When moving to exact arithmetic over the rationals we use `OSCAR` and outperform the `M2` package `PathSignatures` from [ASLF26]. For example, for $k = 3$ and $d = m \in \{10, 20\}$ the package `PathSignatures` requires 1469 and 484595, whereas we need 1 and 14 miliseconds, respectively.

Acknowledgements

We thank Carlos Améndola, Michael Joswig, Benjamin Lorenz, and Felix Lotter for several suggestions and comments on earlier versions of our package. The authors acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – CRC/TRR 388 "Rough Analysis, Stochastic Dynamics and Related Fields" – Project A04 and B01, 516748464.

References

[AFS19] Carlos Améndola, Peter Friz, and Bernd Sturmfels. Varieties of signature tensors. *Forum of Mathematics, Sigma*, 7, 2019.

[ALS26] Carlos Améndola, Felix Lotter, and Leonard Schmitz. Signature varieties of splines. *preprint arXiv:2602.13011, to appear in Proceedings of the 2026 International Symposium on Symbolic and Algebraic Computation*, 2026.

[AS25] Carlos Améndola and Leonard Schmitz. Learning barycenters from signature matrices. *preprint arXiv:2509.07815, to appear in SIAM Journal on Matrix Analysis and Applications*, 2025.

[ASLF26] Carlos Améndola, Angelo El Saliby, Felix Lotter, and Oriol Reig Fité. Computing Path Signature Varieties in Macaulay2. *Journal of Software for Algebra and Geometry*, 2026.

[CDM+24] Marianne Clausel, Joscha Diehl, Raphael Mignot, Leonard Schmitz, Nozomi Sugiura, and Konstantin Usevich. The Barycenter in Free Nilpotent Lie Groups and Its Application to Iterated-Integrals Signatures. *SIAM Journal on Applied Algebra and Geometry*, 8(3):519–552, 2024.

[DEF+24] Wolfram Decker, Christian Eder, Claus Fieker, Max Horn, and Michael Joswig, editors. *The Computer Algebra System OSCAR: Algorithms and Examples*, volume 32 of *Algorithms and Computation in Mathematics*. Springer, 1 edition, 8 2024.

[DISW26] Joscha Diehl, Rasheed Ibraheem, Leonard Schmitz, and Yue Wu. Tensor-to-tensor models with fast iterated sum features. *Neurocomputing*, 675:132884, 2026.

[DK24] Joscha Diehl and Richard Krieg. FRUITS: feature extraction using iterated sums for time series classification. *Data Mining and Knowledge Discovery*, 38(6):4122–4156, 2024.

[FV10] Peter K Friz and Nicolas B Victoir. *Multidimensional stochastic processes as rough paths: theory and applications*, volume 120. Cambridge University Press, 2010.

[PSS19] Max Pfeffer, Anna Seigal, and Bernd Sturmfels. Learning Paths from Signature Tensors. *SIAM Journal on Matrix Analysis and Applications*, 2019.

[RG20] Jeremy F. Reizenstein and Benjamin Graham. Algorithm 1004: The Iisignature Library: Efficient Calculation of Iterated-Integral Signatures and Log Signatures. *ACM Trans. Math. Softw.*, 46(1), March 2020.

[Sch25] Leonard Schmitz. An efficient algorithm for path recovery from signature tensors. *preprint arXiv:2512.14218*, 2025.