`SignatureTensors.jl`: A Package for Signature Tensors in Julia

Gabriel Riffo, Leonard Schmitz
TU Berlin

ISSAC 2026 Oldenburg
2026-07-16

Chen's signature

For a (smooth enough) *path* $X : [0, 1] \to \mathbb{R}^d$ consider the *k-th signature tensor* $\sigma^{(k)} \in (\mathbb{R}^d)^{\otimes k}$ with entries

$$\sigma^{(k)}_{i_1, i_2, \ldots, i_k} := \int_0^1 \int_0^{t_k} \cdots \int_0^{t_2} \dot{X}_{i_1}(t_1)\dot{X}_{i_2}(t_2) \ldots \dot{X}_{i_k}(t_k)\,\mathrm{d}t_1 \mathrm{d}t_2 \ldots \mathrm{d}t_k$$

The *(k-truncated) signature*

$$\sigma^{(\leq k)} := 1 \oplus \sigma^{(1)} \oplus \sigma^{(2)} \oplus \cdots \oplus \sigma^{(k)}$$

is a non-commutative feature used in stochastic analysis, finance, optimization, machine learning, ...

[C54] **Chen** *"Iterated integrals and exponential homomorphisms "* Proc. Lond. Math. Soc., **1954**

[C+26] **Cuchiero**, **Gazzani**, **Möller**, **Svaluto-Ferro** *"Signature-Based Models in Finance "*, Springer, **2026**

[CK26] **Chevyrev**, **Kormilitzin** *"A Primer on the Signature Method in Machine Learning "*, Springer, **2026**

[B+23] **Bayer**, **Hager**, **Riedel**, **Schoenmakers** *"Optimal stopping with signatures "*, Ann. Appl. Probab., **2023**

Our package `SignatureTensors.jl`

- in `julia`
- compatible with `OSCAR` [DEFHJ25]
- relies on multivariate arrays, Lie theory, non-commutative polynomials, Gröbner bases, ...
- implements signatures of piecewise linear paths, polynomials, ...
- registered in the `julia` general registry

```
julia> using SignatureTensors
 | Package not found, but available from the registry.
 | Install package?
 |   (@v1.12) pkg> add SignatureTensors
 + (y/n/o) [y]: y
   Resolving package versions...
   Installed SignatureTensors - v1.0.0
```

[DEFHJ25] **Decker**, **Eder**, **Fieker**, **Horn**, **Joswig** *"The Computer Algebra System OSCAR"*, Springer, **2025**

Truncated tensor sequences

Signatures are (truncated) sequences of tensors in

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

which we implement via

```
struct TruncatedTensorAlgebra{R}
   base_algebra::R
   base_dimension::Int
   truncation_level::Int
end
```

```
struct TruncatedTensorAlgebraElem{R,E}
   parent::TruncatedTensorAlgebra{R}
   elem::Vector{Array{E}}
end
```

Piecewise linear paths

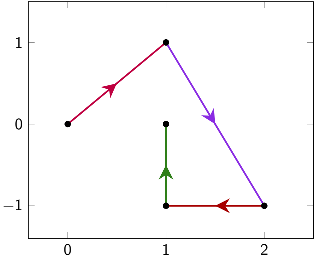
```
julia> T = TruncatedTensorAlgebra(QQ,2,3);
julia> A = QQ[1  1 -1  0;
              1 -2  0  1];
julia> sig(T,:pwln,coef=A)
0-dimensional Array{QQFieldElem, 0}:
1
⊕
2-element Vector{QQFieldElem}:
 1
 0
⊕
2×2 Matrix{QQFieldElem}:
 1//2  -3//2
 3//2  0
⊕
2×2×2 Array{QQFieldElem, 3}:
[:, :, 1] =
 1//6   11//6
 -1//6  -1//6

[:, :, 2] =
 -5//3  -1//6
 1//3   0
```

We encode m-segment paths in $\mathbb{R}^d$ by $(d \times m)$-matrices, where the i-th column represents the i-th step, e.g.

$$\begin{bmatrix} 1 & 1 & -1 & 0 \\ 1 & -2 & 0 & 1 \end{bmatrix}$$

represents



Two algorithms

Classically, one computes signatures via **Chen's identity and recursion** [C54]. Furthermore, we implement the approach via **dictionaries and (symmetric) Tucker decompositions** [PSS19].

$d \backslash m$	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**

Table: Benchmarks on MacBook Pro (M5 chip, 10 CPU cores, 24 GB RAM) **in miliseconds** for truncation $k = 4$ and `base_algebra=Float64`

[C54] **Chen** *"Iterated integrals and exponential homomorphisms"* Proc. Lond. Math. Soc., **1954**

[PSS19] **Pfeffer**, **Seigal**, **Sturmfels** *"Learning paths from signature tensors"*, SIMAX, **2019**

Benchmarks

We compare **the iisignature package** [RG20] based on numpy against **our SignatureTensors.jl package**

$d \backslash m$	10	20	30	40	50	60
10	0, 0	**0**, 1	**0**, 1	**0**, 2	**0**, 3	**0**, 3
20	1, **0**	**2**, 4	**3**, 20	**4**, 40	**5**, 49	**6**, 57
30	5, **1**	11, **6**	**18**, 23	**24**, 82	**30**, 200	**35**, 458
40	16, **3**	35, **9**	55, **31**	**74**, 102	**94**, 219	**112**, 566
50	43, **11**	94, **23**	145, **66**	197, **120**	**248**, 311	**301**, 582
60	93, **39**	205, **67**	316, **96**	429, **153**	539, **434**	647, **636**

Table: $k = 4$ and `base_algebra=Float64`

We compare against **Macaulay2 package PathSignatures** [A+25]

$d = m$	10	20	30	40	50	60
$k = 3$	1469, **1**	484595, **14**	F, **73**	F, **569**	F, **1266**	F, **3750**
$k = 4$	357627, **19**	F, **1178**	F, **14373**	F, **192897**	F, F	F, F

Table: `base_algebra=QQ`

[RG20] **Reizenstein**, **Graham** *"The iisignature Library"*, TOMS, **2020**

[A+25] **Améndola**, **El Saliby**, **Lotter**, **Fité** *"Computing Path Signature Varieties in Macaulay2"*, JSAG, **2026**

Path recovery

Theorem [PSS19]. The 3rd signature tensor, when considered as a map $\sigma^{(3)} : \mathrm{GL}_d \to \mathbb{R}^{d\times d\times d}$, is injective.

Reformulated as a polynomial system with d^3 equations and given $C, G \in \mathbb{R}^{d\times d\times d}$, the d^2 unknowns $A_{1,1},\dots,A_{d,d}$ are uniquely determined by

$$\sum_{\alpha=1}^{d}\sum_{\beta=1}^{d}\sum_{\gamma=1}^{d} A_{i,\alpha}A_{j,\beta}A_{k,\gamma}C_{\alpha,\beta,\gamma} = G_{i,j,k} \tag{1}$$

Theorem [S25] We solve (1) in $\mathcal{O}(d^4)$ expected operations.

d	2	3	4	5	6	7	10	20	30	40	50	60
	0, 0	0, 0	0, 0	1, **0**	35, **0**	F, **0**	F, **0**	F, **0**	F, **1**	F, **5**	F, **16**	F, **29**

Table: Benchmarks in **seconds**

[PSS19] **Pfeffer**, **Seigal**, **Sturmfels** *"Learning paths from signature tensors"*, SIMAX, **2019**

[S25] **Schmitz** *"An Efficient Algorithm for Path Recovery from Signature Tensors"*, arXiv:2512.14218, **2025**

Barycenters of signatures

```
julia> R, a = polynomial_ring(QQ,:a=>(1:3,1:3));     # ambient dimension 3
julia> T = TruncatedTensorAlgebra(R,3,2);            # truncation level 2
julia> sX = sig(T,:pwln, coef=QQ[1 0; 0 1; 0 0]);    # signature of X
julia> sY = sig(T,:pwln, coef=QQ[0;0;1]);            # signature of Y
julia> sZ = sig(T,:pwln, coef=a);                 # signature of unknown path
julia> I = ideal(R,vec(sZ - bary([sX,sY])));         # recovery ideal
julia> dim(I), degree(I)                             # dimension and degree
(3, 2)
```

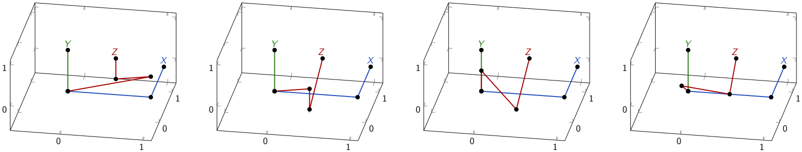


Figure: Given X and Y, recover Z from $\mathrm{bary}(\sigma^{(\leq 2)}(X), \sigma^{(\leq 2)}(Y)) = \sigma^{(\leq 2)}(Z)$

[AS25] **Améndola**, **Schmitz** *"Learning barycenters from signature matrices"*, SIMAX (to appear), **2026**

[BK81] **Buser**, **Karcher** *"Gromov's almost flat manifolds"*, Asté., **1981**

Documentation

https://leonardschmitz.github.io/SignatureTensors.jl/

SignatureTensors.jl

Search docs (Ctrl + /)

Home

Documentation

- Types
- Signature Constructors
- Tensor Learning / Path Recovery
- Barycenters

References

Version docs

Notes

- Only defined for `sequence_type == :iis`. Not supported for `:p2id` or `:p2`.
- The input `a` is expected to have level-0 component equal to `1` (i.e., `a` lives in the group, not the Lie algebra).
- `log` and `exp` are mutual inverses: `log(exp(x)) == x` and `exp(log(a)) == a`.
- See also: `Base.exp`, `Base.inv`, `Base.:^`.

source

> `Base.vec` — Function

> `Base.:(==)` — Function

Signature Constructors

The primary interface for computing signatures. `sig` dispatches on the `geom_type` symbol to select the appropriate path or membrane class, and accepts keyword arguments for coefficients, shape, composition, regularity, and algorithm choice. See the table below for supported geometry types.

`geom_type`	Description	Key arguments
`:point`	Constant path (unit in $G_{d,k}$)	—
`:axis`	Canonical axis path	—
`:mono`	Moment path	—
`:pwln`	Piecewise linear path	`coef`, `algorithm` (`:Chen` or `:congruence`)
`:poly`	Polynomial path	`coef`, `algorithm` (`:congruence`, `:ARS26` or `:LS26`)
`:pwmon`	Piecewise monomial path	`composition`, `regularity`

Thank you!