TTSTools

MATLAB® Functions for Tensor-Valued Time-Series Processing

João P. Hespanha

November 11, 2014

Abstract

This toolbox provides functions to process tensor-valued time series, including differentiations, integration, resampling, etc. These functions can be applied to numerical values or to TensCalcTools Symbolic Tensor-Valued Expressions (STVEs).

Contents

1 Tensor-Valued Time Series

Tensors are essentially multi-dimensional arrays. Specifically, an α*-index tensor* is an array in $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_\alpha}$ where α is an integer in $\mathbb{Z}_{\geq 0}$. By convention, the case $\alpha = 0$ corresponds to a *scalar* in $\mathbb{R}$. We use the terminology *vector* and *matrix* for the cases $\alpha = 1$ and $\alpha = 2$, respectively. The integar α is called the *index* of the tensor and the vector of integers $[n_1, n_2, \dots, n_\alpha]$ (possibly empty for $\alpha = 0$) is called the *dimension* of the tensor.

A *tensor-valued time series* (TTTS) is a sampled-based representation of a time-varying tensor, i.e., a function $F : \mathbb{R} \to \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_\alpha}$. A TTTS represents F through a pair (X, T) where $T \in \mathbb{R}^{n_t}$ is a vector of sample times and $X \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_\alpha \times n_t}$ is an $\alpha + 1$*-index tensor* with the understanding that $i \in \{1, 2, \dots, n_t\}$,

$$F(T_i) = [X_{i_1, i_2, \dots, i_\alpha, i}]_{i_1=1, i_2=1, \dots, i_\alpha=1}^{i_1=n_1, i_2=n_2, \dots, i_\alpha=n_\alpha}, \quad \forall i \in \{1, 2, \dots, n_t\},$$

i.e., the first α indices of X represent the value of F and the last index represents time.

2 Functions provided by TTSTools

tsDerivative

```
[dx,ts]=tsDerivative(x,ts)
```

This function returns a TTTS (dx,ts) that represents the time derivative of the input TTTS (x,ts). The output sampling times ts are equal to the input sampling times ts and therefore the size of derivatine dx is equal to the size of the input x. The time derivative is computed assuming that the input time-series is piecewise quadratic.

tsDerivative2

```
[ddx,ts]=tsDerivative2(x,ts)
```

This function returns a TTTS (ddx,ts) that represents the second time derivative of the input TTTS (x,ts). The output sampling times ts are equal to the input sampling times ts and therefore the size of derivatine dx is equal to the size of the input x. The time derivatives are computed assuming that the input time-series is piecewise quadratic.

tsIntegral

```
y=tsIntegral(x,ts)
```

This function returns a tensor y that represents the time integral of the input TTTS (x,ts). The size of the integral y is equal to the size of the input x with the last (time) dimension removed. The integral is computed assuming that the input time-series is piecewise quadratic.

tsDot

```
[y,ts]=tsDot(x1,x2,ts)
```

This function returns a scalar-valued time-series (y,ts) that represents the dot product of two n-vector time-series (x1,ts) and (x2,ts):

$$y = x1'.*x2$$

The size of the output y is equal to the size of ts.

tsCross

```
[y,ts]=tsCross(x1,x2,ts)
```

This function returns a 3-vector time-series (y,ts) that represents the cross product of two 3-vector time-series (x1,ts) and (x2,ts):

$$y = cross(x1, x2)$$

The size of the output y is equal to the size of the inputs x1 and x2.

tsQdot

```
[y,ts]=tsQdot(q1,q2,ts)
```

This function returns a 4-vector time-series `(y,ts)` that represents the product of two quaternions `(q1,ts)`, `(q2,ts)`:

$$y = q1 \cdot q2$$

The size of the output `y` is equal to the size of the inputs `x1` and `x2`. However, if either `(q1,ts)` or `(q2,ts)` is a 3-vector time series, then the corresponding input quaternion is assumed pure, but the output quaternion is always a 4-vector time series.

tsQdotStar

```
[y,ts]=tsQdotStar(q1,q2,ts)
```

This function returns a 4-vector time-series `(y,ts)` that represents the product of two quaternions `(q1,ts)*`, `(q2,ts)`:

$$y = q1^* \cdot q2$$

The size of the output `y` is equal to the size of the inputs `x1` and `x2`. However, if either `(q1,ts)` or `(q2,ts)` is a 3-vector time series, then the corresponding input quaternion is assumed pure, but the output quaternion is always a 4-vector time series.

tsRotation

```
[y,ts]=tsRotation(q,x,ts)
```

This function returns a 3-vector time-series `(y,ts)` that represents the rotation of a 3-vector time-series `(x,ts)` by a 4-vector time-series `(q,ts)` representing a quaternion:

$$y = q \cdot x \cdot q^*$$

The size of the output `y` is equal to the size of the input `x`.

tsRotationT

```
[y,ts]=tsRotationT(q,x,ts)
```

This function returns a 3-vector time-series `(y,ts)` that represents the rotation of a 3-vector time-series `(x,ts)` by a 4-vector time-series `(q,ts)*` representing a quaternion:

$$y = q^* \cdot x \cdot q$$

The size of the output `y` is equal to the size of the input `x`.