cppTPSA/pyTPSA: a C++/python package for truncated power series algebra

He zhang[*1]

1 Thomas Jefferson National Accelerator Facility, Newport News, VA 23606, USA

DOI:

Software

- Review
- Repository
- Archive

Summary

The truncated power series algebra (TPSA), also referred to as differential algebra (DA), is a well established and widely used method in particle accelerator physics and astronomy. The most straightforward usage of TPSA/DA is to calculate the Taylor expansion up to order n of a given function at a specific point, based on which more sophisticated methods have been developed, *e.g.* symplectic tracking, normal form analysis, verified integration, global optimization, *etc.* TPSA/DA can also be used in the fast multipole method for pairwise interactions between particles and in the image reconstruction algorithm. This package implements the TPSA/DA in C++11 and provides the developers an lib to build the advanced TPSA/DA-based method. A Python lib has also been developed based on the C++ lib and is available in a separate github repository.

Background

In the following, we give a very brief introduction from the perspective of computation and practice. Please refer to [] for the theory and more details.

The fundamental element in DA is the DA vector and first we have to define what a DA vector is. To make the concept easier to understand, we can take a DA vector as the Taylor expansion of a function at a specific point.

Considering a function $f(\mathbf{x})$ and its Taylor expansion $f_{\mathrm{T}}(\mathbf{x}_0)$ up to the order n, we can define a define an equivalence relation between the Taylor expansion and the DA vector as follows

$$f_{\mathrm{T}}(\mathbf{x}_0) = [f]_n = \sum C_{n_1,n_2,\ldots,n_v} \cdot d_1^{n_1} \cdot \cdots \cdot d_v^{n_v},$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_v)$, and $n \geq n_1 + n_2 + \cdots + n_v$. Here d_i is a special number and it represents a small variance in x_i. Generally one can define a DA vector by setting values to respective terms, without define the function f. The addition and multiplication of two DA vectors can be defined straightforwardly. To add two DA vectors, we simply add the coefficients of the like terms in them. To multiply two DA vectors, we calculate the multiplication of each term in the first one with all the terms in the second one and combine like terms ignoring all the terms with an order above n. So given two DA vectors $[a]_n$ and $[b]_n$ and a scalar c, we have the following formulas:

*corresponding author

$$\begin{aligned} [a]_n + [b]_n &:= [a+b]_n, \\ c \cdot [a]_n &:= [c \cdot a]_n, \\ [a]_n \cdot [b]_n &:= [a \cdot b]_n, \end{aligned} \tag{1}$$

According to the fixed point theorem, the inverse of a DA vector that is not infinitely small can be calculated iteratively in a limit number of iterations.

The derivation operator ∂_v with respect to the v^{th} variable can be defined as

$$\partial_v [a]_n = \left[\frac{\partial}{\partial x_v} a \right]_{n-1},$$

which can be carried out term by term on $[a]_n$. The operator ∂_v satisfies the chain rule:

$$\partial_v([a] \cdot [b]) = [a] \cdot (\partial_v[b]) + (\partial_v[a]) \cdot [b].$$

The inverse operator ∂_v^{-1} can also be defined and carried out easily in a term-by-term manner.

Statement of need

`Gala` is an Astropy-affiliated Python package for galactic dynamics. Python enables wrapping low-level languages (e.g., C) for speed without losing flexibility or ease-of-use in the user-interface. The API for `Gala` was

Features

Single dollars ($) are required for inline mathematics e.g. $f(x) = e^{\pi/x}$

Double dollars make self-standing equations:

Figures

Figures can be included like this:

Acknowledgements

We acknowledge contributions from Brigitta Sipocz, Syrtis Major, and Semyeong Oh, and support from Kathryn Johnston during the genesis of this project.

Appendix

In the following we give a brief introduction of the DA method. The basic concepts and the deductions will be presented directly without proof.

References