PySymmPol - Symmetric Polynomials

Summary

PySymmPol is a Python package designed for efficient manipulation of symmetric polynomials. It provides functionalities for working with various types of symmetric polynomials, including elementary, homogeneous, monomial symmetric, (skew-) Schur, and Hall-Littlewood polynomials. In addition to polynomial operations, **PySymmPol** offers tools to explore key properties of integer partitions and Young diagrams, such as transposition, Frobenius coordinates, characters of symmetric groups and others.

This package originated from research conducted in the realm of integrable systems applied to string theory and the AdS/CFT correspondence. **PySymmPol** aims to facilitate computational tasks related to symmetric polynomials and their applications in diverse fields.

Statement of need

Symmetric polynomials play a crucial role across various domains of mathematics and theoretical physics due to their rich structure and broad applications. They arise naturally in combinatorics, algebraic geometry, representation theory, and mathematical physics [@Macdonald:1998, @Fulton:2004]. These polynomials encode essential information about symmetries and patterns, making them indispensable in the study of symmetric functions and their connections to diverse mathematical structures. Moreover, symmetric polynomials find extensive applications in theoretical physics, particularly in quantum mechanics, statistical mechanics, and quantum field theory [@Babelon:2003, @Korepin:1993, @Marino:2005, @Wheeler:2010]. Their utility extends to areas such as algebraic combinatorics, where they serve as powerful tools for solving combinatorial problems and understanding intricate relationships between different mathematical objects. Thus, tools and libraries like PySymmPol provide researchers and practitioners with efficient means to explore and manipulate symmetric polynomials, facilitating advancements in both theoretical studies and practical applications.

Main features and functionalities

PySymmPol is composed of several modules in two main packages. 1. Partitions 2. Polynomials Let us briefly explain them.

Integer Partitions and Young Diagrams

For integer partitions, we employ two primary representations. The first is the conventional representation as a monotonic decreasing sequence, denoted as $\lambda = (\lambda_1, \ldots, \lambda_m)$, where $\lambda_i \geq \lambda_{i+1}$. This representation enables basic manipulations of integer partitions, including determining the number of rows, columns, boxes, diagonal boxes, and Frobenius coordinates within the associated Young Diagram.

Additionally, Young diagrams are represented as cycles within the symmetric group $\mathfrak{S}_N$, denoted as $\lambda = (1^{k_1} 2^{k_2} \ldots)$, where $N = \sum_i \lambda_i = \sum_j j k_k$. These cycles serve as representatives of the conjugacy class of $\mathfrak{S}_N$, and as such, we refer to them as conjugacy class vectors. This alternative representation offers insights into the structural properties of integer partitions and facilitates their manipulation within the framework of group theory.

Our keen interest in exploring these representations stems from their relevance in the realm of two-dimensional Conformal Field Theories (CFTs). In this context, bosonic states are labeled by conjugacy class vectors, highlighting the importance of understanding and manipulating such representations. Conversely, fermions are represented using the standard representation, emphasizing the need to bridge the conceptual gap between these distinct frameworks. Investigating these representations not only sheds light on the fundamental properties of fermion states within CFTs but also provides valuable insights for theoretical developments in quantum field theory and related fields.

Some methods to deal with these bosonic and fermionic states are also available in the package, and the ACCELASC algorithm~[@Kelleher:2009] greatly improved the speed of these methods.

Symmetric Polynomials

One of the main goals of the package is to provide the definitions of the polynomials in terms of the Miwa coordinates, or power sums,

$$t_j = \frac{1}{j} \sum_{i=1}^{N} x_i^j$$

where $\vec{x} = (x_1, \ldots, x_N)$ and $\mathbf{t} = (t_1, t_2, \ldots)$.

In terms of these coordiantes, the *Complete Homogeneous* $h_n(\mathbf{t})$ and *Elementary Symmetric Polynomials* $e_n(\mathbf{t}) = (-1)^n h_n(-\mathbf{t})$ are defined via

$$h_n(\mathbf{t}) = \sum_{k_1 + 2k_2 + \cdots = n} \frac{t_1^{k_1}}{k_1} \frac{t_2^{k_2}}{k_2} \cdots$$

From these expressions, we obtain the *(skew-) Schur polynomials* $s_{\lambda/\mu}(\mathbf{t})$, where λ and μ are integer partitions, via **Jacobi-Trudi identity

$$s_{\lambda/\mu} = \det_{p,q}(h_{\lambda_q - \mu_p - q + p}(\mathbf{t})) \ .$$

The *Hall-Littlewood polynomials* are defined via

$$P_\lambda(x_1, \ldots, x_N; Q) = \prod_{i \geq 0} \prod_{j=1}^{p(i)} \frac{1-Q}{1-Q^j} \sum_{\omega \in \mathfrak{S}_N} \omega \left(x_1^{\lambda_1} \cdots x_n^{\lambda_n} \prod_{i<j} \frac{x_i - Q x_j}{x_i - x_j} \right)$$

where $p(i)$ is the number of rows of size i in λ, and $\mathfrak{S}_N$ is the symmetric group. The limit $Q = 0$ in the Hall-Littlewood polynomials gives the Schur polynomials, while $Q = 1$ returns the *Monomial Symmetric Polynomials* $m_\lambda(x_1, \ldots, x_N)$.

For more information on these topics, see REFERENCES.

Acknowledgements

I would like to thank Fapesp for finantial support, grant **2022/06599-0**. I would also like to thank the Free and Open Source Software community.

References