Undergraduate Research Opportunity Programme in Science

Canonical basis of the quantized Fock space

Hu Yi*, Shi Cong† and Tan Kai Meng‡

Department of Mathematics, National University of Singapore

2, Science Drive 2, Singapore 117543

1 The quantized Fock space

Let $\mathcal{P}$ denote the set of partitions of non-negative integers, and let q be an indeterminate. The quantized Fock space $\mathcal{F}$ is a vector space over $\mathbb{C}(q)$ having two distinguished bases, the standard basis $\{s(\lambda)|\lambda \in \mathcal{P}\}$ and the canonical basis $\{G(\lambda)|\lambda \in \mathcal{P}\}$. Write $G(\lambda) = \sum_\mu d_{\lambda,\mu}(q)s(\mu)$ and let $\mathbf{D}$ be the matrix $[d_{\lambda,\mu}(q)]$. The aim of this project is to implement an algorithm that computes the matrix $\mathbf{D}$.

Due to the space constraint of this abstract, we shall not include the combinatorial definitions and refer interested readers to Section 2.7 of [1].

2 Theoretical algorithm

The algorithm can be divided into three main steps. A matrix whose entries are Laurent polynomials with single-variable q is obtained at the end of each step. Denote these three matrices by $\mathbf{T}$, $\mathbf{A}$ and $\mathbf{D}$ respectively.

1. Matrix $\mathbf{T}$

   We are given a partition λ of integer N.

   If $\lambda = (1^{m_1}, \cdots, r^{m_r})$, we write $\alpha = (1^{a_1}, \cdots, r^{a_r})$ and $\beta = (1^{b_1}, \cdots, r^{b_r})$, where $m_i = na_i + b_i$ and $0 \leqslant b_i < n$. Here β is n-regular and has a ladder decomposition $L_1, \cdots, L_s$, where each ladder L_i has k_i nodes, all having n-residue r_i. Set

   $$t(\lambda) = F_{r_s}^{(k_s)} F_{r_{s-1}}^{(k_{s-1})} \cdots F_{r_1}^{(k_1)} V_1^{a_1} V_1^{a_2} \cdots V_r^{a_r} s(\emptyset).$$

   The action of V_k is defined below, in Section 3. We describe the action of F_i here. Let λ be a partition and let μ be the partition obtained by adding an indent

*Student

†Student

‡Assistant Professor

i-node γ. Let $N_i^r(\lambda,\mu) = I_i^r(\lambda,\mu) - R_i^r(\lambda,\mu)$ where $I_i^r(\lambda,\mu)$ (resp. $R_i^r(\lambda,\mu)$) denotes the number of indent i-nodes of λ (resp. of removable i-nodes of λ) situated to the right of γ (γ not included). Then

$$F_i s(\lambda) = \sum_{\mu} q^{N_i^r(\lambda,\mu)} s(\mu),$$

where the sum is over all partitions μ obtained from λ by adding an indent i-node. Furthermore,

$$F_i^{(r)} = \frac{F_i^r}{[r]!}, \text{ where } [r]! = \prod_{i=1}^{r} \frac{q^i - q^{-i}}{q - q^{-1}}.$$

Let $t(\lambda) = \sum_{\mu} t_{\lambda,\mu}(q)s(\mu)$. Then $t_{\lambda,\mu}(q) \in \mathbb{Z}[q,q^{-1}]$ and is non-zero only if λ and μ has the same n-core and n-weight. Fix an n-core and n-weight, and let Q be the set of all partitions having this n-core and this n-weight.

The matrix $\mathbf{T} = [t_{\lambda,\mu}(q)]_{\lambda,\mu\in Q}$. We order the partition indices lexicographically decreasing.

2. Matrix **A**

   (a) Calculate the matrix $\overline{\mathbf{T}} = [t_{\lambda,\mu}(q^{-1})]_{\lambda,\mu\in Q}$.

   (b) Obtain matrix **A** by the formula $\mathbf{A} = \mathbf{T}(\overline{\mathbf{T}})^{-1}$.

   The matrix **A** is known to be lower triangular with all diagonal entries equal to 1.

   It seems easy to obtain **A** from **T**. However, implementing Step 2(b) is in fact the most challenging part of this project.

3. Matrix **D**

   From the proof of the existence of the basis $\{G(\lambda)|\lambda \in \mathcal{P}\}$ (see Theorem 9 of [1]), we can devise a method to compute the $d_{\lambda,\mu}(q)$ from the entries in matrix **A**, where

$$G(\mu) = \sum_{\lambda} d_{\lambda,\mu}(q)s(\lambda).$$

   The matrix $\mathbf{D} = [d_{\lambda,\mu}(q)]$ is also known to be lower triangular with all diagonal entries equal to 1. All the other non-zero entries are polynomials with non-negative integer coefficients, and having no constant term.

3 Implementation

Due to the limited space of this abstract, we shall only include the following three parts, which highlight the main challenges we encountered in this project.

1. Given a partition λ of integer N, generate all the partitions which share the same n-core and n-weight as λ.

   - Set $core(\lambda, n) := n$-core of λ, $w := n$-weight of λ.

     w is the total number of moves that we have to push the beads down the abacus. Prior to any move, the beads in an abacus are arranged in such a way that represents $core(\lambda, n)$. Thus, if the abacus has n columns, we split w into n non-negative integers and assign these moves to the n columns accordingly.

   - Generate all partitions λ_w of w, such that $\lambda_w = (\lambda_{w_1}, \lambda_{w_2}, \cdots, \lambda_{w_r})$, where $r \leqslant n$.

   - Add zeros to each partition λ_w if necessary, such that $\lambda'_w = (\lambda_{w'_1}, \lambda_{w'_2}, \cdots, \lambda_{w'_n})$, where
     $$\lambda_{w'_i} = \begin{cases} \lambda_{w_i}, & \textit{if } 1 \leqslant i \leqslant r; \\ 0, & \textit{if } r < i \leqslant n. \end{cases}$$

   - Permute each λ'_w and assign $\lambda_{w'_i}$ moves to the ith column. After w moves are made, a partition of integer N is generated.

   - Declare a map $\widetilde{P}$ to store all the partition λ's generated, using λ as the key, index of λ as data and following lexicographically decreasing order.

2. Action V_k on a standard basis element $s(\xi)$.

   - We construct the abacus representing $s(\xi)$ and make k bead-moves in it. The first move can be made on any bead in the abacus that has an empty slot below. The number of beads being crossed[1] by this bead adds to the total number of spins.

   - Each subsequent move must be made on a bead that lies after the original position of the bead which has just been moved. Spin number accumulates in the same manner. Stop after completing k moves.

   - Each series of k moves produces a partition μ with spin number γ. The product $(-q)^{-\gamma} s(\mu)$ is a term in the final of result of this action V_k on $s(\xi)$. Exhaust all such possible series of k moves and sum all such products to obtain $V_k s(\xi)$.

3. Generating matrix $\mathbf{A}$ from $\mathbf{T}$

   We use the formula $\mathbf{A} = \mathbf{T}(\overline{\mathbf{T}})^{-1}$. Although $\mathbf{T}$ and $\mathbf{A}$ are matrices whose entries are Laurent polynomials. $\overline{\mathbf{T}}$ is a matrix whose entries are rational functions. When the size of these matrices are large, the symbolic calculations becomes extremely difficult to handle. Even powerful symbolic computation softwares,

[1] Bead B is said to be crossed by A if B lies between A's original and final positions.

such as MatLab, Mathematica and Maxima, failed to produce the correct matrix $\mathbf{A}$ when the matrix size is larger than 40.

We sought for alternative methods which avoids direct symbolic computation of matrix $(\overline{\mathbf{T}})^{-1}$. We realized that inversion of numerical matrices is much simpler to compute than that of symbolic matrices, especially with the help from the Matrix Template Library [3]. Thus we need to compute a number of numerical matrices and recover the Laurent polynomial entries from them.

We notice that each entry $a_{\lambda,\mu}(q)$ in matrix $\mathbf{A}$ is a Laurent polynomial with boundaries on its powers[2]. The upper bound is n and the lower bound is $(1-n)w$, i.e. each $a_{\lambda,\mu}(q)$ can have at most $nw+1$ non-zero coefficients. Therefore, we can set q to $nw+1$ different numerical values, compute the corresponding numerical matrices $\mathbf{T}$, $\overline{\mathbf{T}}$ and $\mathbf{A}$. Now for each entry in $\mathbf{A}$, we have $nw+1$ pairs of $(q_i, a_{\lambda,\mu}(q_i))$ values. We can then recover the coefficients of $a_{\lambda,\mu}(q)$ using a modified version of the Lagrange's interpolation formula[3].

$$a_{\lambda,\mu}(q) = q^{(1-n)w} \sum_i q_i^{(n-1)w} a_{\lambda,\mu}(q_i) \prod_{j \neq i} \frac{q - q_j}{q_i - q_j}$$

4 Possible Improvements

It is obvious that the bottleneck of this algorithm is inverting matrix $\overline{\mathbf{T}}$ and computing matrix $\mathbf{A}$. Our numerical approach can generate fast and accurate results for matrices of smaller sizes but produce some wrong entries when the size of the matrix exceeds 1200. A possible way of improving it is to insert $nw+1$ complex numbers instead of real numbers. However, it is not implemented in our project due to time constraint.

References

[1] G. James and A. Kerber, *The representation theory of the symmetric group*, Encyclopedia of mathematics and its applications, vol 16. Addison-Wesley, 1981.

[2] B. Leclerc, *Symmetric functions and the Fock space representation of* $U_q(\widehat{\mathfrak{sl}}_n)$, Lectures given at Isaac Newton Institute, June 2001.

[3] The Matrix Template Library, `http://www.osl.iu.edu/research/mtl/` .

[2]The value of the lower bound still remains as a conjecture, but it worked well for all practical purposes in this project.

[3]The original Lagrange's interpolation formula only works for polynomial.