The Journal of Open Source Software

GroupFunctions.jl: computing individual entries of the irreducible representations of the unitary group U(d)

David Amaro-Alcalá [1] **and Konrad Szymański** [1]

1 Research Center for Quantum Information, Institute of Physics, Slovak Academy of Sciences, Dúbravská cesta 9, Bratislava, Slovakia

DOI: N/A

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: 01 January 1970

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

`GroupFunctions.jl`[1] is a Julia library that computes individual matrix entries of polynomial irreducible representations of the unitary group $\mathrm{U}(d)$. A representation assigns a matrix to each group element and preserves multiplication. An irreducible representation has no nonzero proper subspace invariant under all its matrices. In a fixed basis, each matrix entry, viewed as a scalar function of the group element, is called a group function or D-function. For each requested group function, the user can choose either a symbolic expression or a numerical value at a specified input matrix. For $\mathrm{SU}(2)$, they reduce to the Wigner D-functions.

The basis vectors are labelled by Gelfand-Tsetlin patterns (Gelfand & Tsetlin, 1950). The library also computes entire representation matrices, constructs input unitaries from parameterisations common in quantum optics, translates Gelfand-Tsetlin patterns into occupation-number kets, and computes the associated Schur functions.

Statement of need

The library serves researchers in mathematics and quantum physics who need symbolic expressions or numerical values for representation matrix entries. For $U \in \mathrm{U}(d)$, let $D^{(\lambda)}(U)$ denote the matrix representing U in a polynomial irreducible representation labelled by λ. The entry with row label out and column label init has equivalent expressions in inner-product and bra-ket notation:

$$D^{(\lambda)}_{\mathrm{out},\mathrm{init}}(U) = (D^{(\lambda)}(U)\mathrm{init}, \mathrm{out}) = \langle \mathrm{out} \mid D^{(\lambda)}(U) \mid \mathrm{init} \rangle,$$

where init and out are vectors in the chosen orthonormal basis, and the inner product $(\cdot,\cdot)$ is linear in its first argument.

In mathematics, summing the diagonal group functions gives the character of the representation, namely the trace of the representation matrix of U. This character is the Schur polynomial s_λ evaluated at the eigenvalues of U, a central object in algebraic combinatorics and symmetric function theory.

In quantum optics, group functions give photon transition amplitudes through linear optical networks (de Guise et al., 2014). They also characterise the performance of quantum devices (Amaro-Alcalá, 2026) and describe the symmetry properties of states (Otto & Szymański, 2024). In BosonSampling (Aaronson & Arkhipov, 2011), the transition amplitude is proportional to a

[1] David Amaro-Alcalá developed the package, its algorithms, and its test suite, and prepared the initial documentation. Konrad Szymański substantially revised and expanded the documentation, developed additional examples, and contributed to the standardisation of the API.

permanent whose evaluation is classically computationally hard. Whether boson samplers retain a computational advantage over classical sampling algorithms under realistic noise remains an open question.

These tasks can require numerical or symbolical group functions. Both are supported by `GroupFunctions.jl`; the following comparison concerns the symbolic case. To our knowledge, none of the related packages discussed below is designed to compute a single representation-matrix entry as an exact symbolic D-function. We therefore compare the cost of computing one entry directly, via the method employed by this library (the Grabmeier-Kerber formula (Grabmeier & Kerber, 1987), see the Software design section), with the conventional procedure of constructing the Lie-algebra representation matrices and exponentiating the resulting full matrix.

The cost of forming the full matrix depends on the representation dimension $r_\lambda(d)$, the number of rows and columns in $D^{(\lambda)}(U)$. Fix a Young diagram λ with $N = \sum_i \lambda_i > 1$ boxes and at most d rows. The basis vectors are indexed by semistandard Young tableaux of this shape with entries in $\{1, \dots, d\}$. The hook-content formula counts these tableaux (Fulton, 1997, sec. 4.3, p. 55). The dimension $r_\lambda(d)$ of the irreducible representation is also the Schur polynomial s_λ evaluated at d arguments equal to 1, because a representation's character at the identity equals its dimension:

$$r_\lambda(d) = s_\lambda(\underbrace{1, \dots, 1}_{d}) = \prod_{(i,j)\in\lambda} \frac{d + j - i}{h_{ij}}.$$

The box (i, j) in row i and column j has content $j - i$. Its hook length h_{ij} counts the box itself, those to its right in the same row, and those below it in the same column. For fixed λ, the product has N factors linear in d and a constant denominator, giving $r_\lambda(d) = \Theta(d^N)$. Here $\Theta(d^k)$ denotes growth bounded above and below by positive constant multiples of d^k as $d \to \infty$. For example, $\lambda = (2)$ gives $r_{(2)}(d) = d(d + 1)/2$.

To compute one group function directly, we keep the two tableau labels fixed as d grows and supply the corresponding basis vectors, avoiding enumeration of the full basis. Under these assumptions, the current symbolic implementation has time complexity $\Theta(d^3)$ and space complexity $\Theta(d^2)$. The cubic cost arises from constructing the tables that index the monomial sum in the Grabmeier-Kerber formula: recursively assembling each length-d row copies successively longer arrays, requiring $\Theta(d^2)$ operations per row across d rows. With N fixed, the number of tables and permutations remains bounded as d grows. The mathematical sum runs over the $N!$ permutations of S_N, grouped into double cosets. The implementation pads tableau contents when $d > N$, so the enumeration can include additional permutations; for example, $\lambda = (2, 0, 0)$ yields six rather than two. This padding leaves the permutation count bounded for fixed N as d grows. The operation count includes table construction, monomial preparation, and evaluation of the final sum. Scalar arithmetic operations, array accesses, and copies of individual entries are each assigned cost $O(1)$. Space complexity is the maximum number of scalar entries stored simultaneously. Constants may depend on N; bit complexity is excluded.

For comparison, consider constructing the representation matrix by exponentiating a Lie-algebra matrix. Assume that this $r_\lambda(d) \times r_\lambda(d)$ Lie-algebra matrix has already been constructed (Barut & Rączka, 1986, p. 280). Numerical scaling and squaring with a Padé approximant uses dense matrix products and a linear solve (Higham, 2005). With the Padé degree and number of squarings fixed, standard dense multiplication and elimination require $\Theta(r_\lambda(d)^3) = \Theta(d^{3N})$ scalar arithmetic operations and have space complexity $\Theta(r_\lambda(d)^2) = \Theta(d^{2N})$. Each scalar addition, subtraction, multiplication, or division has unit cost; numerical implementations use fixed-precision arithmetic. For $\lambda = (2)$, these costs grow as d^6 and d^4, respectively, compared with d^3 and d^2 for the direct calculation. Exponentiation produces the full matrix at a chosen U; the direct calculation produces one exact polynomial in its entries. The comparison

quantifies the cost of forming the full matrix when only one entry is needed, with N fixed. Numerical methods designed to compute individual entries are outside this comparison.

State of the field

Related packages support representation theory and quantum optics, with different computational goals. `SUNRepresentations.jl` (QuantumKitHub contributors, 2020) and the algorithm in the appendix of (Alex et al., 2011) compute $\mathrm{SU}(d)$ Clebsch-Gordan coefficients. `RepLAB` (Rosset et al., 2021) manipulates irreducible representations of various groups, including $\mathrm{U}(d)$, but provides only indirect numerical access to group functions. `IntegrateUnitary.jl` (Pawela & Puchała, 2026) performs symbolic integration over compact groups, and `haarpy` (Cardin et al., 2026) implements Weingarten-calculus methods. In quantum optics, `BosonSampling.jl` (Seron & Restivo, 2024), `Perceval` (Heurtel et al., 2023), and `QOptCraft` (Gómez Aguado et al., 2023) numerically model linear optical devices, while `The Walrus` (Gupt et al., 2019) computes amplitudes for Gaussian boson sampling. None of these packages targets the symbolic computation of individual representation-matrix entries, the primary purpose of `GroupFunctions.jl`.

Software design

The library computes matrix entries of polynomial irreducible representations of $\mathrm{U}(d)$ specified by integer partitions of length at most d, using a unified method that also supports symbolic input matrices. Several computational routes are possible. For symmetric irreducible representations, which model fully indistinguishable bosons, states can be manipulated as polynomials of creation operators applied to the vacuum. Another approach constructs and exponentiates the Lie algebra generators in the chosen representation. Both approaches are computationally expensive. More restricted methods use generating functions (Prakash, 1996).

The authors chose the Grabmeier-Kerber formula (Grabmeier & Kerber, 1987) for its generality. The formula expresses a requested matrix entry as a sum of monomials in the input matrix entries, with coefficients determined by the irreducible representation and the chosen basis vectors. Our implementation optimises the enumeration of the double cosets that index this sum by grouping permutations that contribute the same monomial. The library's main function, `group_function`, evaluates the resulting sum.

Internally, the algorithm represents basis vectors as semistandard Young tableaux. The public interface exposes the equivalent Gelfand-Tsetlin patterns through the `GTPattern` data structure and provides utilities such as `occupation_number` for common quantum optics calculations.

Research impact statement

`GroupFunctions.jl` has been used to evaluate matrix entries and group characters in published work on filtered randomized benchmarking (Amaro-Alcalá, 2026). Development has continued since 2020, with tests run through continuous integration. The package is registered in the Julia General registry, distributed under the MIT licence, and installed with the following command:

```
] add GroupFunctions
```

The following example computes a symbolic matrix entry between two basis vectors of a mixed-symmetry irreducible representation of $\mathrm{U}(4)$.

```
λ = [2,2,1,0]; basis=basis_states(λ); # integer partition and basis
group_function(λ, basis[1], basis[end]) # symbolic matrix entry
```

The result is a polynomial in the entries of the $\mathrm{U}(4)$ matrix. For symmetric irreducible representations, matrix entries are proportional to permanents, which symbolic algebra packages also compute. For mixed-symmetry representations, no other software computes these entries symbolically. Further examples, applications, and mathematical background are available in the documentation.

AI usage disclosure

OpenAI Codex (GPT-5.3) assisted in producing a faster version of the double-coset enumeration. An earlier, AI-free implementation remains in the repository history. The authors developed the mathematical design and proof of correctness of the optimised algorithm. Anthropic Claude (Opus 4.8) assisted with code review and language editing of the documentation and manuscript. The authors reviewed and edited all AI-assisted changes.

Acknowledgements

We thank Dr. Hubert de Guise for discussions and bibliographic suggestions, and Dr. Alonso Botero for suggestions on the presentation. Mitacs CALAREO, DeQHOST APVV-22-0570, QUAS VEGA 2/0164/25, Postdokgrant APD0161, and the Štefan Schwarz programme supported this work. David Amaro-Alcalá also acknowledges indirect support from the Government of Alberta and NSERC during his PhD studies at the University of Calgary.

References

Aaronson, S., & Arkhipov, A. (2011). The computational complexity of linear optics. *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, 333–342. https://doi.org/10.1145/1993636.1993682

Alex, A., Kalus, M., Huckleberry, A., & Delft, J. von. (2011). A numerical algorithm for the explicit calculation of SU(N) and SL(N,C) Clebsch-Gordan coefficients. *J. Math. Phys.*, *52*(2), 023507. https://doi.org/10.1063/1.3521562

Amaro-Alcalá, D. (2026). Kostant relation in filtered randomized benchmarking for passive bosonic devices. *Phys. Rev. A*, *113*(5), 052602. https://doi.org/10.1103/xmlq-vssg

Barut, A. O., & Rączka, R. (1986). *Theory of group representations and applications*. PWN-Polish Scientific Publishers.

Cardin, Y., Guise, H. de, & Quesada, N. (2026). Haarpy: A python library for weingarten calculus and integration over classical compact groups, their associated circular ensembles, the permutation and centered permutation groups, and quantum groups. *J. Open Source Softw.*, *11*(124), 11002. https://doi.org/10.21105/joss.11002

de Guise, H., Tan, S.-H., Poulin, I. P., & Sanders, B. C. (2014). Coincidence landscapes for three-channel linear optical networks. *Phys. Rev. A*, *89*(6), 063819. https://doi.org/10.1103/physreva.89.063819

Fulton, W. (1997). *Young tableaux: With applications to representation theory and geometry* (Vol. 35). Cambridge University Press. https://doi.org/10.1017/CBO9780511626241

Gelfand, I. M., & Tsetlin, M. L. (1950). Finite-dimensional representations of the group of unimodular matrices. *Dokl. Akad. Nauk SSSR*, *71*, 825–828.

Gómez Aguado, D., Gimeno, V., Moyano-Fernández, J. J., & Garcia-Escartin, J. C. (2023). QOptCraft: A Python package for the design and study of linear optical quantum systems. *Comput. Phys. Commun.*, *282*, 108511. https://doi.org/10.1016/j.cpc.2022.108511

Grabmeier, J., & Kerber, A. (1987). The evaluation of irreducible polynomial representations of the general linear groups and of the unitary groups over fields of characteristic 0. *Acta Appl. Math.*, *8*(3), 271–291. https://doi.org/10.1007/bf00046717

Gupt, B., Izaac, J., & Quesada, N. (2019). The Walrus: A library for the calculation of hafnians, Hermite polynomials and Gaussian boson sampling. *J. Open Source Softw.*, *4*(44), 1705. https://doi.org/10.21105/joss.01705

Heurtel, N., Fyrillas, A., Gliniasty, G. de, Le Bihan, R., Malherbe, S., Pailhas, M., Bertasi, E., Bourdoncle, B., Emeriau, P.-E., Mezher, R., Music, L., Belabas, N., Valiron, B., Senellart, P., Mansfield, S., & Senellart, J. (2023). Perceval: A software platform for discrete variable photonic quantum computing. *Quantum*, *7*, 931. https://doi.org/10.22331/q-2023-02-21-931

Higham, N. J. (2005). The scaling and squaring method for the matrix exponential revisited. *SIAM J. Matrix Anal. Appl.*, *26*(4), 1179–1193. https://doi.org/10.1137/04061101X

Otto, F., & Szymański, K. (2024). Rotational covariance restricts available quantum states. *Phys. Rev. A*, *110*(2), 022440. https://doi.org/10.1103/PhysRevA.110.022440

Pawela, Ł., & Puchała, Z. (2026). *IntegrateUnitary.jl: A Julia package for symbolic integration over Haar measures*. https://arxiv.org/abs/2605.23830

Prakash, J. S. (1996). Wigner's D-matrix elements for SU(3)-a generating function approach. *J. Phys. A: Math. Gen.*, *29*(18), 6059–6072. https://doi.org/10.1088/0305-4470/29/18/032

QuantumKitHub contributors. (2020). *SUNRepresentations.jl*. https://github.com/QuantumKitHub/SUNRepresentations.jl.

Rosset, D., Montealegre-Mora, F., & Bancal, J.-D. (2021). RepLAB: A computational/numerical approach to representation theory. In M. B. Paranjape, R. MacKenzie, Z. Thomova, P. Winternitz, & W. Witczak-Krempa (Eds.), *Quantum theory and symmetries* (pp. 643–653). Springer International Publishing. https://doi.org/10.1007/978-3-030-55777-5_60

Seron, B., & Restivo, A. (2024). BosonSampling.jl: A Julia package for quantum multi-photon interferometry. *Quantum*, *8*, 1378. https://doi.org/10.22331/q-2024-06-18-1378