

# 1 PhotonWeave

2 **Simon Sekavčník** <sup>1</sup>, **Kareem H. El-Safty** <sup>1</sup>, and **Janis Nötzel** <sup>1</sup>

3 **1** Technical University of Munich, Theoretical Quantum System Design, Munich, Germany

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

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

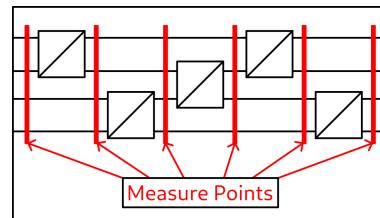
## 4 Summary

5 PhotonWeave is a quantum systems simulator designed to offer intuitive abstractions for  
6 simulating quantum photonic systems and their interactions in Fock spaces ([Fock, 1932](#)) and  
7 any custom Hilbert spaces. The simulator focuses on simplifying complex quantum state  
8 representations, such as continuous photonic states with polarization using envelopes that  
9 can mimic pulses, making it more approachable for specific quantum simulations. While  
10 general-purpose quantum simulation libraries such as QuTiP ([Johansson et al., 2012](#)) provide  
11 robust tools for quantum state manipulations, some require advanced software skills to  
12 manipulate complex system interactions. PhotonWeave addresses this by abstracting such  
13 details, streamlining the simulation process, and allowing quantum systems to interact naturally  
14 as the simulation progresses.

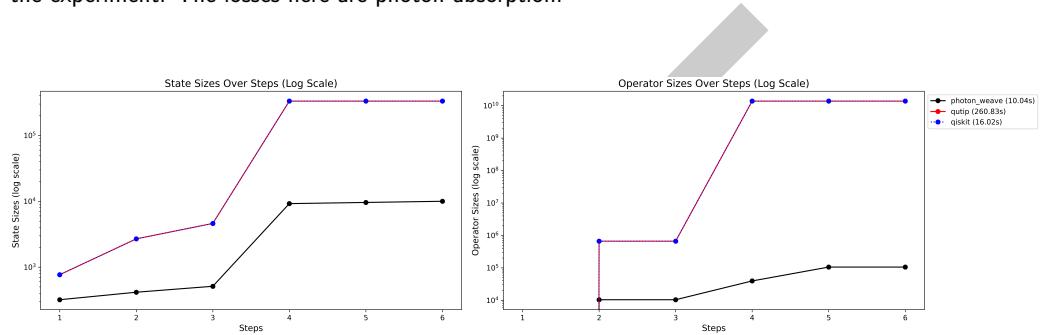
15 In contracts to other frameworks such as Qiskit ([Javadi-Abhari et al., 2024](#)), which are  
16 primarily designed for qubit-based computations, PhotonWeave excels at simulating continuous-  
17 variable quantum systems, mainly photons, as well as custom quantum states that can interact  
18 dynamically. Furthermore, PhotonWeave offers a balance of flexibility and automation by  
19 deferring the joining of quantum spaces using State Containers until necessary, enhancing  
20 computational efficiency. The simulator supports CPU and GPU execution, ensuring scalability  
21 and performance for large-scale simulations. This is achieved by using the JAX ([Bradbury et  
22 al., 2018](#)) library.

## 23 Statement of Need

24 Tools like QuTiP, Qiskit, Piquasso, StrawberryFields ([Killoran et al., 2019; Kolarovszki et al.,  
25 2024](#)), and SOQCS ([Osca & Vala, 2024](#)) already exist for modeling quantum phenomena.  
26 However, many of them either require extensive user control (QuTiP) or enforce rigid circuit  
27 structures (StrawberryFields, SOQCS). Researchers in quantum optics and related fields need  
28 a tool that simplifies photonic systems simulations and supports dynamic interactions between  
29 custom quantum systems. PhotonWeave introduces such features without restricting itself to  
30 the circuit model, allowing researchers to focus on component development. Such a tool could  
31 generate a library of components and gates that closely model real-world devices, fostering  
32 greater collaboration among scientists in these fields. In Fig. 1, a complex scenario of lossy  
33 beam splitters is depicted, and in Fig. 2, the performance is compared to Qiskit and QuTiP.



**Figure 1:** The simulation of lossy Beam Splitters. The simulation tracks the state evolution throughout the experiment. The losses here are photon absorption.



**Figure 2:** Comparison between PhotonWeave, Qiskit, and QuTip regarding simulation time and the required space to simulate the experiment in Figure 1. The steps are the executed operations.

## 34 PhotonWeave Overview

35 PhotonWeave is a quantum simulation library designed for simulating any system, provided  
 36 that simulating hardware meets the resource requirements. This simulator allows users to  
 37 easily create, manipulate, and measure quantum systems.

## 38 PhotonWeave Implementation Details

39 In the following sections, we will describe the main features of PhotonWeave; details about  
 40 implementations and usage can be found in [the documentation](#).

### 41 State Containers

42 PhotonWeave's core functionality revolves around quantum state containers. States can be  
 43 represented in three forms: Label, Vector, or Matrix, which progressively require more  
 44 memory. PhotonWeave automatically manages these representations, reducing representations  
 45 where applicable to save resources. The framework provides state containers such as Fock,  
 46 Polarization, Envelope, and CustomState. Fock, Polarization, and CustomState are  
 47 essential state containers that hold the quantum state in any valid representation until the  
 48 state interacts with other states. When states interact, these containers store references to the  
 49 Envelope, CompositeEnvelope, or both. This allows each container to understand its place  
 50 within a larger product space and how it evolves mathematically.

### 51 Envelopes

52 PhotonWeave places a particular emphasis on the Envelope concept. An Envelope represents  
 53 a pulse of light, where all photons are indistinguishable and share the same polarization,  
 54 representing the  $\mathcal{F} \otimes \mathcal{P}$  space where  $\mathcal{F}$  represents the Fock space and  $\mathcal{P}$  represents the  
 55 Polarization space. Initially, when the states are separable, they are stored in the respective  
 56 Fock and Polarization containers. In addition to the states, an Envelope holds essential  
 57 metadata such as wavelength and temporal profile.

58    **Composite Envelopes**

59    When envelopes interact, for example, using a beam-splitter (Xiang-bin, 2002), their states  
60    must be joined. The necessary state data are extracted from their respective containers, and  
61    their Hilbert spaces form a product space in these cases. A CompositeEnvelope can contain  
62    multiple product spaces, which can be accessed from any of the contributing state contain-  
63    ers. Additionally, CompositeEnvelope instances can be merged, allowing states within both  
64    envelopes to interact. CustomState instances can also be included in a CompositeEnvelope  
65    since any custom state can, in principle, interact with any other state.

66    **Operations**

67    PhotonWeave provides several ways to perform operations on quantum states. All operations  
68    are created using the Operation type as well as one of the Enums: FockOperationType,  
69    PolarizationOperationType, CustomStateOperationType, and CompositeOperationType  
70    to further define what on which type of a state the operation will operate. Operations can be  
71    manually constructed or generated using expressions with a context along with the predefined  
72    ones. PhotonWeave supports photonic operators such as Squeezing, Displacement, Phase  
73    Shift, Beam Splitter, and non-linear operations. It also supports Pauli operators.

74    PhotonWeave optimizes resource usage by automatically adjusting the dimensionality of  
75    the Fock space when necessary, even within product states. This ensures that only the  
76    minimal required space is used, dynamically resizing the quantum state representation to avoid  
77    unnecessary memory consumption.

78    Once an operation is defined, it can be applied to an appropriate state at any level. If a state  
79    is a part of a product space, PhotonWeave ensures that the operation is applied to the correct  
80    subspace. Additionally, Kraus operators can be applied to any desired state space. This allows  
81    the user to simulate losses at any level.

82    **Measurements**

83    PhotonWeave offers a robust measurement framework for any state. By default, Fock spaces  
84    are measured on a number basis, Polarization spaces are measured on a computational basis,  
85    and CustomState is measured on a respective basis. PhotonWeave also supports more precise  
86    measurement definitions, such as Positive Operator Valued Measurement (POVM).

87    **Conclusion**

88    PhotonWeave is an open-source quantum system simulator under the Apache-2.0 license,  
89    targeting researchers and developers who need an easy-to-use yet powerful simulation tool.  
90    One of the intended outcomes is to build a library of interoperable quantum device models  
91    powered by the PhotonWeave framework.

92    **Acknowledgments**

93    The authors acknowledge the financial support by the Federal Ministry of Education and  
94    Research of Germany in the programme of "Souverän. Digital. Vernetzt.". Joint project  
95    6G-life, project identification number: 16KISK002 and via grants 16KIS1598K, 16KISQ039,  
96    16KISQ077, 16KISQ093 and that of the DFG via grant 1129/2-1. This research is part of  
97    the Munich Quantum Valley, which is supported by the Bavarian state government with funds  
98    from the Hightech Agenda Bayern Plus and received support from the Bavarian Ministry for  
99    Economic Affairs (StMWi) via the project 6GQT.

## References

- 100
- 101 Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G.,  
102 Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable  
103 transformations of Python+NumPy programs* (Version 0.3.13). [http://github.com/jax-ml/  
104 jax](http://github.com/jax-ml/jax)
- 105 Fock, V. (1932). Konfigurationsraum und zweite quantelung. *Zeitschrift f r Physik*, 75(9–10),  
106 622–647. <https://doi.org/10.1007/bf01344458>
- 107 Javadi-Abhari, A., Treinish, M., Krsulich, K., Wood, C. J., Lishman, J., Gacon, J., Martiel, S.,  
108 Nation, P. D., Bishop, L. S., Cross, A. W., Johnson, B. R., & Gambetta, J. M. (2024).  
109 *Quantum computing with qiskit*. <https://doi.org/10.48550/arXiv.2405.08810>
- 110 Johansson, J. R., Nation, P. D., & Nori, F. (2012). QuTiP: An open-source python framework  
111 for the dynamics of open quantum systems. *Computer Physics Communications*, 183(8),  
112 1760–1772. <https://doi.org/10.1016/j.cpc.2012.02.021>
- 113 Killoran, N., Izaac, J., Quesada, N., Bergholm, V., Amy, M., & Weedbrook, C. (2019).  
114 Strawberry fields: A software platform for photonic quantum computing. *Quantum*, 3, 129.  
115 <https://doi.org/10.22331/q-2019-03-11-129>
- 116 Kolarovszki, Z., Rybotycki, T., Rakyta, P., Kaposi, Á., Poór, B., Jóczik, S., Nagy, D. T.  
117 R., Varga, H., El-Safty, K. H., Morse, G., Oszmaniec, M., Kozsik, T., & Zimborás, Z.  
118 (2024). *Piquasso: A photonic quantum computer simulation software platform*. <https://doi.org/10.48550/arXiv.2403.04006>
- 119
- 120 Osca, J., & Vala, J. (2024). SOQCS: A stochastic optical quantum circuit simulator. *SoftwareX*,  
121 25, 101603. <https://doi.org/10.1016/j.softx.2023.101603>
- 122 Xiang-bin, W. (2002). Theorem for the beam-splitter entangler. *Physical Review A*, 66(2).  
123 <https://doi.org/10.1103/physreva.66.024303>