New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
17 March 2022

REVISED
9 August 2022

ACCEPTED FOR PUBLICATION
25 August 2022

PUBLISHED
14 September 2022

PAPER

CircuitQ: an open-source toolbox for superconducting circuits

Philipp Aumann[1,*], **Tim Menke**[2,3,4], **William D Oliver**[2,3,5,6] and **Wolfgang Lechner**[1,7]

[1] Institute for Theoretical Physics, University of Innsbruck, A-6020 Innsbruck, Austria
[2] Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139, United States of America
[3] Department of Physics, Massachusetts Institute of Technology, Cambridge, MA 02139, United States of America
[4] Department of Physics, Harvard University, Cambridge, MA 02138, United States of America
[5] MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02420, United States of America
[6] Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, United States of America
[7] Parity Quantum Computing GmbH, A-6020 Innsbruck, Austria
[*] Author to whom any correspondence should be addressed.

E-mail: philipp.aumann@uibk.ac.at and wolfgang.lechner@uibk.ac.at

Keywords: quantum physics, superconducting circuits, superconducting qubits, software toolbox, open-source, Python

Abstract

We introduce CircuitQ, an open-source toolbox for the analysis of superconducting circuits implemented in Python. It features the automated construction of a symbolic Hamiltonian of the input circuit and a dynamic numerical representation of the Hamiltonian with a variable basis choice. The software implementation is capable of choosing the basis in a fully automated fashion based on the potential energy landscape. Additional features include the estimation of the T_1 lifetimes of the circuit states under various noise mechanisms. We review previously established circuit quantization methods and formulate them in a way that facilitates the software implementation. The toolbox is then showcased by applying it to practically relevant qubit circuits and comparing it to specialized circuit solvers. Our circuit quantization is applicable to circuit inputs from a large design space, and the software is open-sourced. We thereby add an important resource for the design of new quantum circuits for quantum information processing applications.

1. Introduction

Superconducting circuits are one of the most versatile and promising platforms in the development of chip-based quantum processors [1, 2]. The development of cutting-edge qubit designs is currently driven by the effort to realize quantum computers with long coherence times as well as high-fidelity control and readout, all in a scalable design [3]. Combining these requirements is a grand challenge for quantum hardware design. Therefore, considerable effort is invested into the study of new and improved qubit circuits for quantum information processing applications [4–7].

A major part of the analysis of superconducting circuits is the construction of a quantum model to describe the system theoretically. Such a model is obtained by using general methods to construct the corresponding Hamiltonian [8–10]. A numerical implementation of the algebraic description is then needed to analyze the quantum properties of the circuit. A number of open source software packages has been developed for this purpose. The library scQubits [11], for example, simulates qubits from a specific set of circuits. The package QuCAT [12] offers a more general circuit input, as it permits a combination of Josephson junctions, inductances, capacitances and resonators. The quantization is performed in the basis of normal modes, which is suitable for weakly anharmonic systems. Another useful toolbox is provided by the SuperQuantPackage [13], which includes an algorithm to provide the user with a numerical representation of the Hamiltonian for a given input circuit. It performs a coordinate transformation prior to quantization. Qiskit Metal [14] and KQCircuits [15] enable the analysis of superconducting circuits based on their physical layout on the chip. While such software packages have been proven to be useful for specific circuit design tasks, we expand on prior work by presenting a toolbox that works for a generic variety of

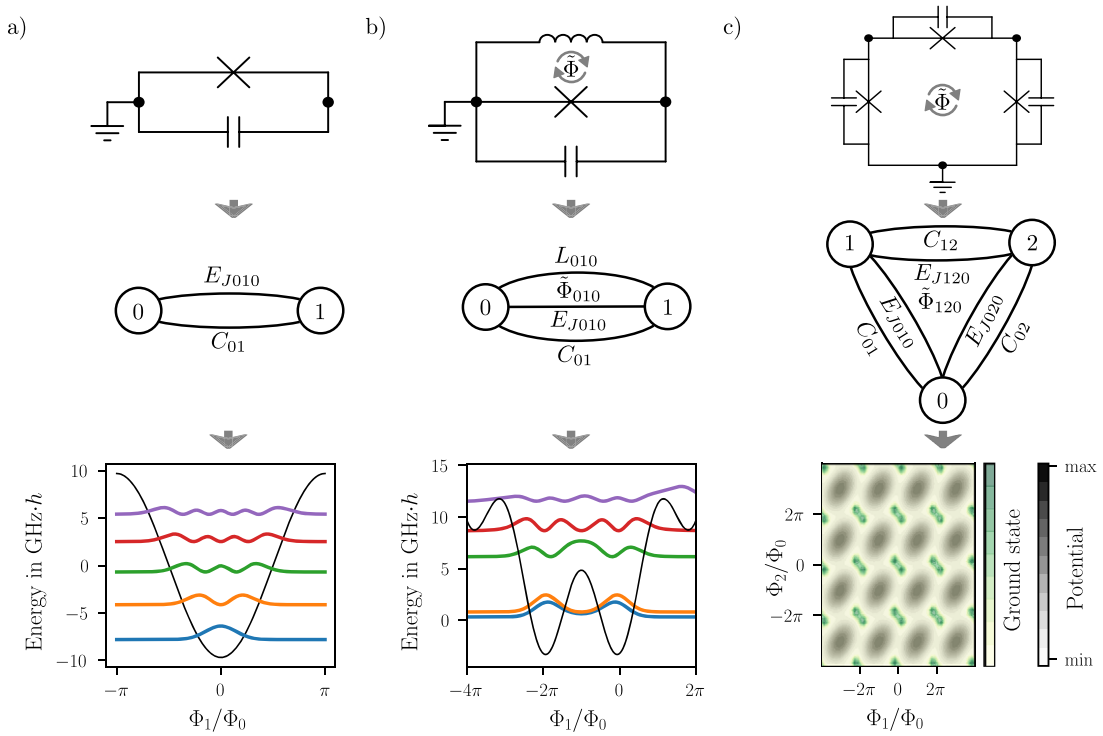

Figure 1. Overview of the functionality of CircuitQ for three example circuits, as discussed in section 4. The general procedure to analyze a superconducting circuit (top row) is to interpret it as a graph whose edges correspond to the circuit elements (middle row). The graph serves as the input of the toolbox, and a symbolic and a numerical Hamiltonian are constructed from it. The potential and eigenspectrum of the Hamiltonian are shown in the bottom row. The symbolic Hamiltonians and parameter values can be found in section 4, and the code to generate these instances is listed in appendix B. (a) Transmon circuit. The last row depicts the scaled absolute value squared of the wavefunctions of the lowest five eigenstates as a function of the flux variable Φ_1, which is the node flux variable connected to node 1 of the respective graph in the middle row, while the scaling factor Φ_0 is the flux quantum. Each wave function is offset by its corresponding eigenvalues. The potential of the Hamiltonian is shown as a black line. (b) Fluxonium circuit. The scaled wavefunctions of the eigenstates and potential energy are shown in the same fashion as in (a). (c) Persistent-current flux qubit circuit. Due to the additional node, the problem becomes two-dimensional. Similar to before, Φ_1 and Φ_2 are the node variables of node 1 and 2 in the circuit graph. We show an overlap of two contour plots, where one depicts the scaled absolute value squared of the wavefunction of the lowest eigenstate and the other visualizes the potential.

circuits, determines a symbolic and numerical Hamiltonian, provides an automated choice of implementation basis and includes a measure for several T_1 contributions.

In this work, we provide a structured review of the superconducting circuit quantization procedure implemented in the software toolbox CircuitQ. This provides an insight into the software implementation, but also serves as a more general review of the quantization process of superconducting circuits by constructing the Hamiltonian. CircuitQ is written in Python and can be used to analyze superconducting circuits that are a user-defined combination of Josephson junctions, linear inductances, and capacitances. It takes the circuit and optionally the circuit component parameters as an input and returns the quantum physical properties of the circuit, particularly the corresponding Hamiltonian in symbolic and numerical form. Figure 1 provides an overview of this conceptual procedure for three example circuits. Depending on the shape of the inductive potential, CircuitQ can dynamically perform the numerical implementation in the charge basis, the flux basis, or in a mixture of both. Therefore, it provides a toolbox for circuits comprising different parameter regimes. As detailed in section 4, this implementation works well for few-node circuits, where the direct implementation of the node variables with the flux and charge basis is a natural choice, whereas the limits of this implementations are reached for more complex circuits. The interface offers the possibility for a general circuit input. A variety of features and degrees of freedom can be adjusted by the user, such as the circuit composition, the component parameters, ground nodes, offset charges, and loop fluxes. In order to analyze the circuit in view of noisy environments, we implemented an estimation of the T_1 lifetime of an eigenstate by considering three common relaxation mechanisms. Finally, we showcase the software by comparing its accuracy to specialized circuit solvers for several prominent qubit circuits. We provide community access to CircuitQ by making the code and documentation publicly available on GitHub [16].

The article is organized as follows. In section 2, we present the procedure to generate the symbolic Hamiltonian from a given input circuit. Subsequently, the numerical implementation of the Hamiltonian

and features of the toolbox are presented in section 3. Lastly, applications of the toolbox to practically relevant circuit examples are provided in section 4.

2. From the circuit to the symbolic Hamiltonian

The first step in the analysis of the quantum properties of a superconducting circuit is the construction of the circuit Hamiltonian. Our software implementation automates the process to derive the symbolic Hamiltonian. A multitude of techniques for circuit quantization has been developed, including several based on the method of nodes [8–10, 17], black box quantization [18, 19] and others [20, 21]. Here we follow the method of nodes-based approach, which is generally applicable to any superconducting circuit that includes capacitances, inductances, and Josephson junctions, given the realistic condition that spurious capacitances exist between all circuit nodes.

The starting point of circuit quantization is a circuit diagram such as the one shown in figure 2, which is a lumped-element representation of a prospective on-chip microfabricated device. A similar version of this circuit example can be found in reference [22]. The circuit diagram can be seen as a graph with the circuit elements on its edges. The nodes are then home to the conjugate charge and flux variables, which represent the charge stored on capacitances connected to the node, and the flux along a specific path from the node to ground, respectively. They are denoted as the node charge q_i and node flux Φ_i, while $\vec{q}$ and $\vec{\Phi}$ are the vectors of all charge and flux variables ordered by node index. One or more ground nodes can be specified by the user when initializing an instance of the `CircuitQ` class. Additionally, all active nodes with only one neighbouring node are added to the ground nodes if the neighbouring node is ungrounded. An active node is a node that is connected to a capacitance as well as an inductive element. If no ground node could be specified, an active node is chosen to be the ground node. Node 0 is the ground node of the circuit depicted in figure 2. Ground nodes do not appear in the constructed Hamiltonian and the associated variables are removed from vector $\vec{q}$ and $\vec{\Phi}$.

In the software, the circuit graph is implemented as a `MultiGraph` instance of the NetworkX package [23]. A code example to initialize the circuit in figure 2 is given in appendix C. We note that a graph is automatically simplified using the common rules for parallel and series capacitors.

The definition of the node fluxes requires the choice of a unique path from each node in the circuit graph to ground. The set of such paths for all nodes is termed the *spanning tree* of the graph, which cannot contain loops. There may be multiple ways to choose it for a given circuit. One such choice is highlighted as a green sub-graph in figure 2. The choice of spanning tree is equivalent to setting a gauge and therefore does not change the physics of a circuit [9]. However, it does affect the circuit quantization procedure and the form of the Hamiltonian. In this work, we follow common practice and route the spanning tree through capacitive circuit elements only. Since each node pair is connected by spurious capacitances, a spanning tree can always be defined in such a way. For the implementation, CircuitQ makes use of the spanning tree functionality of the NetworkX package.

In order to determine the contribution of an inductive circuit element to the Hamiltonian, one needs to evaluate the flux difference across the respective edge. An inductive loop consisting of inductances and junctions can encircle an external flux, and the boundary condition has to be fulfilled that all the fluxes around a loop sum to a multiple of the flux quantum $\Phi_0 = \frac{\hbar}{2e}$. We highlight one such loop in orange in figure 2. The external flux enters the circuit Hamiltonian by being added to the flux difference across an inductive element between two of the nodes that are part of the loop.

In CircuitQ, the automated evaluation of flux differences is started by defining the set of inductive edges $\mathcal{L}$ and splitting it into the subset $\mathcal{S}$ that is in parallel to spanning tree edges and into the remaining edges $\mathcal{B} = \mathcal{L} - \mathcal{S}$. As we allow for multiple parallel inductive elements between two nodes, a restriction needs to be introduced that only the first parallel inductive edge is added to $\mathcal{S}$. The routine then iteratively steps through the edges in $\mathcal{B}$. If the current edge does not close an inductive loop, we add it to a subset $\mathcal{B}_\mathrm{o} \subseteq \mathcal{B}$ and do not assign an external flux to it. In case it does close an inductive loop, we add it to $\mathcal{B}_\mathrm{c} \subseteq \mathcal{B}$ and do assign an external flux to it. Using the fluxoid quantization condition, the directionality of the edges in $\mathcal{S}$ can be chosen such that we obtain the following relation for the edge flux $\Phi_{e_{ijn}}$ of the nth inductive edge e_{ijn} between nodes i and j:

$$\Phi_{e_{ijn}} = \begin{cases} \Phi_j - \Phi_i & \text{for } e_{ijn} \in \mathcal{S} \cup \mathcal{B}_\mathrm{o} \\ \Phi_j - \Phi_i + \tilde{\Phi}_{ijn} & \text{for } e_{ijn} \in \mathcal{B}_\mathrm{c} \end{cases} . \tag{1}$$

For the circuit in figure 2, this procedure identifies the loop flux $\tilde{\Phi}_{120}$ and applies it to the flux difference between nodes 1 and 2.

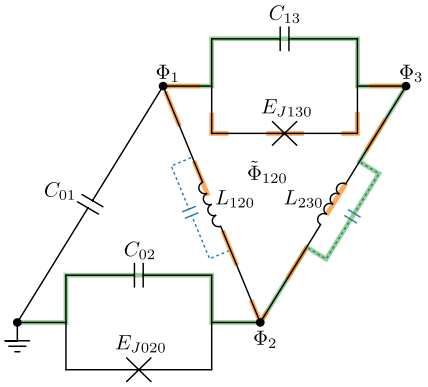

Figure 2. Example circuit to demonstrate the procedure of constructing a Hamiltonian from a superconducting circuit graph. Parasitic capacitances are indicated by the blue elements. A possible spanning tree is indicated in green and a closed inductive loop is indicated by the orange dashed lines. We provide a detailed overview of the circuit quantization steps in the main text.

We note that the loop fluxes are degrees of freedom that can be used to tune the circuit properties. At the same time, they open a path for undesired fluctuations from the environment to couple to the circuit. CircuitQ determines a set of loop fluxes automatically when initializing an instance for a given circuit. These fluxes are treated as conventional circuit parameters whose numerical values can be specified by the user.

Given the relationship between branch and node fluxes, we can explicitly state the inductive potential of the Hamiltonian, which is the sum of linear inductive and Josephson potentials:

$$U_{\text{ind}}\left(\vec{\Phi}\right) = \sum_{\{i,j\}\in\mathcal{I}_{\text{L}}} \sum_{n=0}^{N_{\text{L}}^{ij}-1} \frac{\left(\Phi_{e_{ijn}}\left(\vec{\Phi}\right)\right)^2}{2L_{ijn}} - \sum_{\{i,j\}\in\mathcal{I}_{\text{J}}} \sum_{n=0}^{N_{\text{J}}^{ij}-1} E_{Jijn} \cos\left(\frac{\Phi_{e_{ijn}}\left(\vec{\Phi}\right)}{\Phi_0}\right), \tag{2}$$

where N_{L}^{ij} is the number of linear inductors between node i and j and N_{J}^{ij} is the number of Josephson junctions, while $\mathcal{I}_{\text{L}}$ and $\mathcal{I}_{\text{J}}$ represent the set of node pairs connected by inductors and Josephson junctions. Our formulation of the potential expands upon prior work on general circuit quantization formulations in that it allows for multiple parallel inductive elements per node pair. While this is relevant for Josephson junctions, for example in modelling a frequency tunable transmon, where two Josephson junctions form a SQUID loop, multiple parallel linear inductors are of limited practical relevance for quantum information processing applications because they lead to sensitivity of the circuit energy to static flux offsets.

The kinetic energy T_{cap} of the Hamiltonian is given by the capacitive energy of the circuit. In general, T_{cap} takes the form

$$T_{\text{cap}}\left(\vec{q}\right) = \frac{1}{2}\vec{q}^{\text{T}}\mathbf{C}^{-1}\vec{q}. \tag{3}$$

Here, $\mathbf{C}$ is the node capacitance matrix, which contains the sum of all capacitances connected to a node as the respective diagonal entry and the negative capacitance between two nodes on the off-diagonals. That is, the ith diagonal element of the matrix is given by

$$\mathbf{C}_{i,i} = \sum_n C_{in} \tag{4}$$

by summing over all capacitances C_{in} connected to node i, while the off-diagonal elements are given by

$$\mathbf{C}_{i,j} = -C_{ij} \tag{5}$$

for $i \neq j$ and capacitance C_{ij} linking nodes i and j. The capacitances are invariant under permutation of the indices: $C_{ij} = C_{ji}$. We note that the rows and columns of $\mathbf{C}$ that are associated to ground nodes are removed from the matrix. The form of the kinetic energy arises in the Legendre transformation of the circuit Lagrangian. Charge offsets on a node are taken into account by directly adding the offset to the respective charge operator.

The circuit Hamiltonian in terms of the conjugate coordinates $\vec{\Phi}$ and $\vec{q}$ is given by the sum of the kinetic and potential terms:

$$H\left(\vec{\Phi},\vec{q}\right) = T_{\text{cap}}\left(\vec{q}\right) + U_{\text{ind}}\left(\vec{\Phi}\right). \tag{6}$$

In CircuitQ, it is returned as a symbolic SymPy object [24]. The Hamiltonian that is constructed for the example circuit in figure 2 can be found in appendix E.

For the quantum physical treatment of the system described by the Hamiltonian in equation (6), we perform the usual quantization procedure by promoting the conjugate variables to operators:

$$\Phi_i \to \hat{\Phi}_i \quad \forall\, \Phi_i \in \vec{\Phi} \quad \text{and} \quad q_i \to \hat{q}_i \quad \forall\, q_i \in \vec{q}. \tag{7}$$

Those operators fulfill the canonical commutation relations

$$[\hat{\Phi}_i, \hat{q}_j] = \mathrm{i}\hbar\delta_{i,j}. \tag{8}$$

Generating the Hamiltonian in a symbolic form is the first step towards the automated description of a circuit. The second step is to express the quantized Hamiltonian numerically.

3. From the symbolic Hamiltonian to its numerical implementation

The numerical implementation of the symbolic Hamiltonian is crucial for the analysis of the quantum properties of the input circuit. In this section, we first present important steps of the automated numerical implementation that is part of the toolbox. Subsequently, we describe numerical analysis tools that are implemented in CircuitQ.

3.1. Implementation

Numerical values for the circuit parameters can be specified individually. This includes capacitances, inductances, Josephson energies, external fluxes and charge offsets. If the values are not specified by the user, they are set to a default value. In addition to the numerical parameter values, the charge and flux operators that appear in the Hamiltonian have to be implemented as numerical matrices. These matrices can be either formulated in the flux basis or in the charge basis, which is analogous to choosing a position or momentum space representation. If the potential is periodic along the direction of a node flux variable, the charge basis is the preferred choice for implementing the variables of this node. The connection of a linear inductance to a node leads to a non-periodic harmonic contribution of the potential, which makes a flux basis implementation for the corresponding node variables more desirable. To distinguish these two cases, we label a node as periodic if there is no linear inductance connected to it and if neighbouring nodes that are connected via a Josephson junction are periodic as well. Periodic node variables are automatically implemented in the charge basis and non-periodic variables in the flux basis.

3.1.1. Flux basis

To implement a node variable in the flux basis, we confine the numerical flux values to a finite grid $[-\Phi_{\max}, -\Phi_{\max}+\delta, \ldots, \Phi_{\max}]$ with grid spacing δ. The grid length can be decided by the user or is set to a default value otherwise. Consequently, we can assign a diagonal matrix to the flux variable:

$$\hat{\Phi} \to \begin{pmatrix} -\Phi_{\max} & & & \\ & -\Phi_{\max}+\delta & & \\ & & \ddots & \\ & & & \Phi_{\max} \end{pmatrix}. \tag{9}$$

As the conjugate momentum of the flux, the charge variable can be associated with the derivative with respect to Φ: $q = -\mathrm{i}\hbar\partial_\Phi$. To implement the derivative as a Hermitian operator, we use the finite difference method:

$$\hat{q} \to \frac{-\mathrm{i}\hbar}{2\delta}\begin{pmatrix} 0 & 1 & & \\ -1 & 0 & 1 & \\ & & \ddots & \\ & & -1 & 0 \end{pmatrix}, \tag{10}$$

$$\hat{q}^2 \to \frac{-\hbar^2}{\delta^2}\begin{pmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & & \ddots & \\ & & 1 & -2 \end{pmatrix}. \tag{11}$$

To generate Hermitian matrices, we have chosen a different discretization for the first and second derivative. The cosine terms in equation (2), referring to the energy contribution of Josephson junctions, can be

represented by diagonal matrices in the flux basis, where the diagonal elements are the cosine of the corresponding numerical edge flux value. CircuitQ is capable of working with charge and flux offsets, where the flux offsets $\tilde{\Phi}$ are associated with loop fluxes and charge offsets $\tilde{q}$ with node charges. They are implemented by multiplying them with the identity matrix:

$$\tilde{q} \to \tilde{q} \cdot \mathbb{1}, \qquad \tilde{\Phi} \to \tilde{\Phi} \cdot \mathbb{1}. \tag{12}$$

A Hilbert space is assigned to every node which is not set to ground. To obtain a numerical description of the full Hamiltonian, these subspaces are combined into a composite space using the tensor product by substituting

$$\hat{\Phi}_i \to \mathbb{1} \otimes \ldots \otimes \mathbb{1} \otimes \underset{\substack{\downarrow \\ \text{position of node } i}}{\hat{\Phi}} \otimes \mathbb{1} \otimes \ldots, \tag{13}$$

$$\hat{q}_i \to \mathbb{1} \otimes \ldots \otimes \mathbb{1} \otimes \underset{\substack{\downarrow \\ \text{position of node } i}}{\hat{q}} \otimes \mathbb{1} \otimes \ldots. \tag{14}$$

Here, the variable corresponding to node i is implemented by placing the respective matrix at the position of the composite space which corresponds to node i. We note that node numbering may change due to the elimination of the ground nodes. The sequence of the nodes is deduced by the algorithm and kept consistent throughout the evaluation of an instance.

3.1.2. Charge basis

Similar to the flux basis, we restrict the charge variables to a finite grid when using the charge basis. The charge is truncated at a cutoff number of Cooper pairs n_{cutoff}, which leads to the charge grid $2e[-n_{\text{cutoff}}, \ldots, n_{\text{cutoff}}] = [-q_{\max}, \ldots, q_{\max}]$. In this setting, we can express the charge variable as a diagonal matrix:

$$\hat{q} \to 2e \begin{pmatrix} -n_{\text{cutoff}} & & & \\ & -n_{\text{cutoff}}+1 & & \\ & & \ddots & \\ & & & n_{\text{cutoff}} \end{pmatrix}. \tag{15}$$

Flux variables that correspond to periodic nodes appear exclusively in the arguments of the cosine terms in the Hamiltonian. The cosine acts as a hopping operator in the Cooper pair number basis [25]:

$$\cos\left(\frac{\hat{\Phi}}{\Phi_0}\right) \to \frac{1}{2}\sum_n |n\rangle\langle n+1| + |n+1\rangle\langle n|, \tag{16}$$

with $|n\rangle$ being the charge state corresponding to n Cooper pairs. We can then use the decomposition of the cosine into complex exponentials,

$$\cos\left(\frac{\hat{\Phi}}{\Phi_0}\right) = \frac{1}{2}\left(\mathrm{e}^{\mathrm{i}\frac{\hat{\Phi}}{\Phi_0}} + \mathrm{e}^{-\mathrm{i}\frac{\hat{\Phi}}{\Phi_0}}\right), \tag{17}$$

to represent these terms numerically. This procedure is used in scQubits [11]. The exponential of a flux operator describes the tunneling process of a Cooper pair through a Josephson junction, and it can be described as a jump operator in the charge basis [25]:

$$\mathrm{e}^{\mathrm{i}\frac{\hat{\Phi}}{\Phi_0}} \to \sum_n |n\rangle\langle n+1| \to \begin{pmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix}. \tag{18}$$

The composite space has to be considered if there are multiple flux variables in the argument of the cosine. In this case, we again make use of the tensor product:

$$\mathrm{e}^{\mathrm{i}\frac{\hat{\Phi}_i - \hat{\Phi}_j}{\Phi_0}} \to \mathbb{1} \otimes \ldots \otimes \underset{\substack{\downarrow \\ \text{position of node } i}}{\mathrm{e}^{\mathrm{i}\frac{\hat{\Phi}}{\Phi_0}}} \otimes \ldots \otimes \underset{\substack{\downarrow \\ \text{position of node } j}}{\left(\mathrm{e}^{\mathrm{i}\frac{\hat{\Phi}}{\Phi_0}}\right)^{\dagger}} \otimes \mathbb{1} \otimes \ldots. \tag{19}$$

The full cosine function can thus be implemented as

$$\cos\left(\frac{\hat{\Phi}_i - \hat{\Phi}_j}{\Phi_0}\right) \to \frac{1}{2}\left(\mathrm{e}^{\mathrm{i}\frac{\hat{\Phi}_i-\hat{\Phi}_j}{\Phi_0}} + \left(\mathrm{e}^{\mathrm{i}\frac{\hat{\Phi}_i-\hat{\Phi}_j}{\Phi_0}}\right)^{\dagger}\right). \tag{20}$$

To account for a flux offset $\tilde{\Phi}$, the exponential function in equation (19) can be multiplied by the complex scalar $\mathrm{e}^{-\mathrm{i}\tilde{\Phi}}$.

The choice of an appropriate numerical value for the discretization parameters $\Phi_{\max}$, δ and n_{cutoff}, which determine the numerical representation of the flux and charge variables, depend on the particular circuit. An appropriate regime can be found by increasing (for the case of $\Phi_{\max}$ and n_{cutoff}) or decreasing (for the case of δ) the numerical value of the parameter until convergence is reached, such that the resulting spectrum of the circuit becomes almost invariant under a slight modification of those values.

The numerical grid for the numerical Hamiltonian is generated using the `lambdify` function of SymPy [24] with the parameters and matrices that have been described in this section as inputs. The final implementation is returned as a sparse matrix in SciPy format [26].

For some analyses, it may be helpful to visualize the eigenstates as a function of the flux variable even when an implementation in the charge basis has been used. For this purpose, CircuitQ provides a method which transforms the eigenvectors from the charge to the flux basis. To transform a state vector, given in the charge basis $\mathcal{B}_q = \{|q_i\rangle\}_i$, to a representation in the flux basis $\mathcal{B}_\Phi = \{|\Phi_i\rangle\}_i$, the transformation matrix T can be defined, which maps the state vector from the charge to the flux basis. The coefficients of this matrix read:

$$T_{i,j} = \langle q_i|\Phi_j\rangle = \frac{1}{\sqrt{d}}\,\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}q_i\Phi_j}, \tag{21}$$

with d being the number of basis vectors. We follow the same procedure in our numerical implementation, however we use a modified transformation matrix which respects the construction of the composite Hilbert space, which, in general, consists of subspaces that are either implemented in the charge or the flux basis. We note that depending on the size of the numerical matrices, this transformation can be numerically demanding and consequently may lead to a bottleneck in computation time.

3.2. Features for circuit analysis

Since CircuitQ constructs a numerical implementation of the circuit Hamiltonian, it can be used as a tool for the analysis of the quantum properties of superconducting circuits. This includes the energy spectrum of the Hamiltonian and relaxation times of the energy eigenstates.

3.2.1. Spectrum

To calculate the energy spectrum of the numerical Hamiltonian, the toolbox provides a method that returns the lowest eigenstates and eigenenergies of the numerical Hamiltonian matrix. We use the SciPy library for the (partial) diagonalization, which in turn makes use of efficient ARPACK routines [33]. With this functionality, the toolbox can be used to investigate how a change of parameter values—for example an external flux—or a change in the circuit composition affects the energy spectrum and eigenstates.

For the description of the superconducting circuit as a qubit, we associate the lowest two energy levels that have a nonvanishing energy difference with the qubit states $|0\rangle$ and $|1\rangle$ by default. However, it is possible to declare a different state as the excited qubit state manually. The corresponding energy levels should not be degenerate. To operate a circuit as a qubit, a high degree of anharmonicity of its spectrum is desired. CircuitQ provides a method which gives an estimate of the harmonicity of the spectrum in a quantified form.

3.2.2. Relaxation time T_1

In order to determine the performance of the circuit as a qubit, it is crucial to study its sensitivity to various noise sources. The qubit can decay to its ground state as a result of its interactions with the environment. The sensitivity to this relaxation process is quantified by the T_1 time. We note that an undesired excitation of the qubit state may also result from such interactions. Table 1 provides an overview of the noise contributions that can be estimated with CircuitQ. We included relaxation due to quasiparticle tunneling, dielectric loss and flux noise.

3.2.2.1. Quasiparticle tunneling In experiments, significant non-vanishing densities of unpaired electrons could be observed, which are referred to as quasiparticles in this context [34]. Tunneling of such quasiparticles through the junction barrier can lead to a relaxation of the qubit. This effect is separated into two contributions as given in equation (22). The first contribution concerns the junctions in the circuit (sum over j), while the second contribution is associated with every linear inductance in the circuit

Table 1. Overview of the noise contributions to T_1 which are implemented in CircuitQ. The second column lists the formulas used as the basis of the noise estimation. Here, $|g\rangle$ and $|e\rangle$ label the qubit ground and excited state. The angular frequency of the qubit is labeled by ω_q. Other symbols are described in the third column.

Contribution	Formula	
Quasiparticle tunneling [27]	$T_{1_{\mathrm{qp}}} = \left(\frac{S_{\mathrm{qp}}(\omega_q)}{\hbar^2}\left(\sum_j E_{J,j}\left\|\langle g\|\sin\left(\frac{\hat{\Phi}_{e_j}}{2\Phi_0}\right)|e\rangle\right\|^2 + \sum_l E_{L,l}\left\|\langle g|\frac{\hat{\Phi}_{e_l}}{2\Phi_0}|e\rangle\right\|^2\right)\right)^{-1}$ (22)	$S_{\mathrm{qp}}(\omega_q) = \hbar x_{\mathrm{qp}}\frac{8}{\pi}\sqrt{\frac{2\Delta}{\hbar\omega_q}}$: noise spectral density [6, 27] $\hat{\Phi}_{e_{j/l}}$: edge flux operator describing the flux of the inductive edge corresponding to the jth Josephson junction or lth linear inductance (see equation (1)) $E_{\mathrm{J},j}$: Josephson energy of the jth junction $E_{\mathrm{L},l} = \frac{\Phi_0^2}{L_l}$: inductive energy of the lth linear inductance $x_{\mathrm{qp}} = 10^{-8}$: density of quasiparticles which is scaled by the density of Cooper-pairs [6] $\Delta = 1.76 \cdot k_{\mathrm{B}} T_{\mathrm{c}}$: superconducting gap [28] $T_{\mathrm{c}} = 1.2$ K: critical temperature of aluminum [29]
Dielectric loss [6]	$T_{1_{\mathrm{diel}}} = \left(\sum_i \frac{S_{Q_i}(\omega_q)}{\hbar^2}\left\|\langle g|\hat{q}_{e_i}|e\rangle\right\|^2\right)^{-1}$ (23)	$S_{Q_i}(\omega_q) = \frac{\hbar}{Q_{\mathrm{cap}}(\omega_q)C_i}\left(1 + \coth\frac{\hbar\omega_q}{2k_{\mathrm{B}}T}\right)$: noise spectral density [6, 30] $\hat{q}_{e_i} = \hat{q}_{i_2} - \hat{q}_{i_1}$: charge operator of the capacitive branch e_i linking nodes i_2 and i_1 with corresponding capacitance C_i $Q_{\mathrm{cap}}(\omega_q) = 3 \times 10^6\left(\frac{2\pi\times 6\ \mathrm{GHz}}{\omega_q}\right)^{0.7}$: dielectric quality factor [30, 31] $T = 15$ mK: assumed temperature of a sample
Flux noise [6, 32]	$T_{1_{\mathrm{flux}}} = \left(\sum_i \frac{S_\Phi(\omega_q)}{\hbar^2}\left\|\langle g|\hat{I}_i|e\rangle\right\|^2\right)^{-1}$ (24)	$S_\Phi(\omega_q) = \frac{\hbar^2}{(2e)^2\Phi_0^2}2\pi\frac{A^2}{\omega}$: noise spectral density [6] $A = 2\pi 10^{-6}\Phi_0$: noise amplitude [6] $\hat{I}_i$: current operator, which includes all $\frac{\Phi_k - \Phi_l}{L_{kl}}$ and $I_{\mathrm{C},kl}\sin\frac{\Phi_k - \Phi_l}{\Phi_0}$ terms in the circuit that correspond to a circuit graph edge $i \in \mathcal{B}_{\mathrm{c}}$ connecting nodes k and l $I_{\mathrm{C},kl} = \frac{2e}{\hbar}E_{\mathrm{J},kl}$: critical current $E_{\mathrm{J},kl}$: Josephson energy linking node k and l

(sum over l) [27]. The implementation of the $\sin\left(\frac{\hat{\varphi}}{2}\right) = \sin\left(\frac{\hat{\Phi}}{2\Phi_0}\right)$ operator in equation (22) is straightforward in the flux basis, where it is represented by a diagonal matrix with the values of the sine function on the diagonal. However, a more elaborate implementation is needed for the charge basis [35], as the $\sin\left(\frac{\hat{\varphi}}{2}\right)$ operator describes a tunneling process in the basis of single elementary charges, which cannot be represented in the conventional charge basis of Cooper pairs:

$$\sin\left(\frac{\hat{\Phi}}{2\Phi_0}\right) \to \frac{1}{2\mathrm{i}}\left(\sum_{\tilde{n}} |\tilde{n}-1\rangle\langle\tilde{n}| - |\tilde{n}\rangle\langle\tilde{n}-1|\right), \tag{25}$$

with $|\tilde{n}\rangle$ being the basis state of the single electron charge basis. The implementation of this operator in the single charge basis follows the same procedure as for the cosine operator in the Cooper pair basis outlined in equations (16)–(20). However, the operator implemented in the single charge basis has dimension $2d - 1$, where d is the number of states in the Cooper pair charge basis. To calculate the transition element defined as

$$M_{eg} := \langle g|\sin\left(\frac{\hat{\Phi}_{e_j}}{2\Phi_0}\right)|e\rangle, \tag{26}$$

we have to transform the ground and excited states $|g\rangle$ and $|e\rangle$ from the Cooper pair basis to the single charge basis. We can distinguish two different configurations of the state vectors in the single charge basis: one with only even numbered entries of the single charge basis non-zero, and the other with only the odd numbered entries occupied. To transform from the Cooper pair basis to either the even or odd

configuration, we define the respective $(2d-1)\times d$ dimensional transformation matrices $\hat{T}_{\mathrm{ce}}$ and $\hat{T}_{\mathrm{co}}$ with

$$\hat{T}_{\mathrm{ce}} = \begin{pmatrix} 1 & 0 & 0 & \\ 0 & 0 & 0 & \\ 0 & 1 & 0 & \\ 0 & 0 & 0 & \cdots \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & \\ & \vdots & & \end{pmatrix}, \qquad \hat{T}_{\mathrm{co}} = \begin{pmatrix} 0 & 0 & 0 & \\ 1 & 0 & 0 & \\ 0 & 0 & 0 & \\ 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & \\ 0 & 0 & 1 & \\ & \vdots & & \end{pmatrix}. \tag{27}$$

Finally, to evaluate the transition element M_{eg} in equation (26) in the charge basis, we compute

$$M_{eg} \to \langle g|\hat{T}_{\mathrm{ce}}^{\dagger} \sin\left(\frac{\hat{\Phi}_{e_j}}{2\Phi_0}\right)\hat{T}_{\mathrm{co}}|e\rangle. \tag{28}$$

3.2.2.2. Dielectric loss Another noise channel present in superconducting qubits is relaxation due to the fact that the electrical field, which stores capacitive energy, couples to charged fluctuators [36]. The resulting effect on the T_1 time is calculated with equation (23). Here we sum over all capacitors in the circuit.

3.2.2.3. Flux noise The fluctuation of spins on the superconducting material are suspected to be the origin of flux noise [36]. Those fluctuations perturb the magnetic field, which stores inductive energy, effectively leading to fluctuations of the electrical current of the inductive elements. We estimate the corresponding contribution to the relaxation time with equation (24). As described in section 2, we assign an external flux to a subset of edges $\mathcal{B}_{\mathrm{c}}$ of the circuit graph. We therefore sum over all such edges to estimate the flux noise. If the associated element is a Josephson junction, the current operator is given by $\hat{I} = I_{\mathrm{C}} \sin\frac{\hat{\Phi}}{\Phi_0}$, while the expression $\hat{I} = \frac{\hat{\Phi}}{L}$ is used for linear inductors. We included the possibility to obtain a lower bound on the T_1 estimate by summing not only over the edges in $\mathcal{B}_{\mathrm{c}}$ but including all inductive edges.

The focus of this toolbox is on pure qubit design without considering qubit control such as state preparation. Therefore, we do not consider noise due to the Purcell effect for now. We also did not include pure dephasing mechanisms explicitly, which can be attributed to the fluctuation of the qubit frequency due to various noise channels. Estimating those dephasing processes would entail the calculation of the derivative of the qubit frequency with respect to the particular noise source. This derivative could be either calculated numerically or even symbolically, depending on the efficiency of those approaches. Adding dephasing processes to the analysis is an important part of our outlook, as it is an essential part of an extensive and general study of superconducting circuits.

In comparison to the related open-source software toolbox scQubits [11], we follow a similar strategy by estimating the coherence times using Fermi's golden rule combined with the specific noise spectral densities from literature. However, our expressions for the noise spectral densities differ in the case of noise due to quasiparticle tunneling and flux noise, where we follow Nguyen *et al*'s study of the fluxonium qubit [6]. We also add the contribution due to linear inductances to calculate noise due to quasiparticle tunneling.

4. Demonstration and benchmark

To demonstrate the capabilities of CircuitQ, we use three well known circuits from the literature, i.e., the fixed-frequency transmon [37], the fluxonium [38] and the persistent-current flux qubit [39]. The examples are initialized with the corresponding input graph, from which CircuitQ computes the symbolic and numerical Hamiltonian. The latter can be diagonalized to analyze the spectrum and eigenstates of the system. Figure 1 gives an overview of this procedure while figure 3 depicts the T_1 contributions that are estimated by the toolbox for these circuits. For the noise estimates, we considered all three depolarization channels introduced in section 3.2. For the fixed frequency transmon circuit, we consider two noise channels: quasiparticle tunneling and dielectric loss. For the chosen circuit parameters, the second contribution is observed to be the limiting factor. The lifetime of planar 2D transmon fabrications are reported to be limited by dielectric loss [40]. As we simulate the fluxonium at the sweet spot $\tilde{\Phi}_{\mathrm{ext}} = \pi\cdot\Phi_0$ here, the quasiparticle noise of the small junction (first term in equation (22)) is suppressed and we can ascribe the T_1 decay mostly to the linear inductance (second term in equation (22)). Similar to the transmon, the flux qubit is limited by the dielectric loss when comparing the three relaxation processes. The parameter values used for estimating the lifetime of a flux qubit refer to *qubit B* in reference [32], where the lifetime at the sweet spot $\tilde{\Phi}_{\mathrm{ext}} = \pi\cdot\Phi_0$ seems to be limited by flux noise. Our findings indicate that our

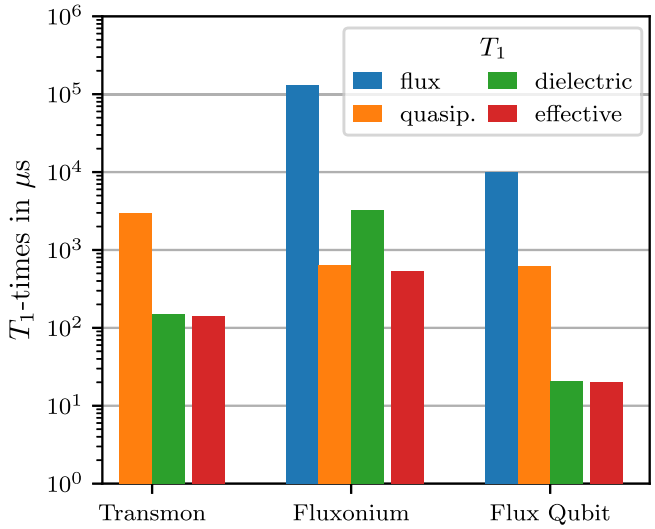

Figure 3. Summary of the estimated T_1 times for the three circuits shown in figure 1. The effective T_1 time is the result of the noise contributions due to quasiparticle tunnling, dielectric loss and flux noise (see table 1). The chosen parameter values for calculating the estimations are given in table 2 in appendix A. We note that these values differ from those chosen for figure 1. There is no flux noise contribution to the transmon as we consider the fixed-frequency transmon circuit here.

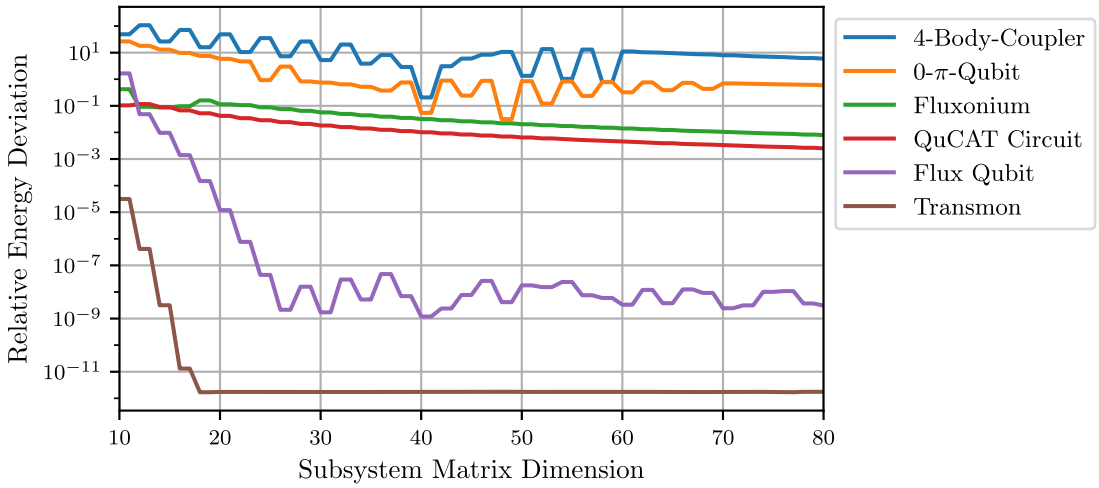

Figure 4. Benchmark tests for six different circuits. We calculated the spectrum for those instances with reference implementations (see main text) and compared the transition energy between the ground state and excited state to the corresponding output of CircuitQ. The deviation is scaled by the qubit energy of the respective reference implementation. The test has been repeated for various dimensions of the subsystem matrices, which are the numerical representation of the node fluxes and charges. We note that the total dimension scales exponentially with the number of ungrounded nodes, which leads to longer evaluation times for larger circuits.

specifications chosen for the flux and charge noise estimates do not resemble this particular experimental set-up accurately. However, our estimate for the effective T_1 time lies in the same order of magnitude compared to the findings in this reference. The effective T_1 time for the transmon and fluxonium range in between 10^2–10^3 μs, while this value is reduced by one order of magnitude for the flux qubit. These numbers are in accordance with values from literature [3]. We note that the exact estimate of the T_1 time depends on parameters like the density of quasiparticles or the dielectric quality factor. Those parameters depend on the specific realization of the circuits and will vary from experiment to experiment. Although we have chosen representative values, the computed T_1 times should not be understood to be exact for a specific circuit layout but should serve as estimates to classify the sensitivity of a circuit to certain noise channels.

In appendix B, code samples are provided to initialize the instances for the example circuits. The symbolic Hamiltonian that is generated for the transmon by the toolbox is

$$H = -E_{\mathrm{J010}} \cos\left(\frac{\Phi_1}{\Phi_0}\right) + \frac{\left(q_1 + \tilde{q}_1\right)^2}{2C_{01}}. \tag{29}$$

Here, Φ_1 and q_1 are the flux and charge variables of node 1, and $\tilde{q}_1$ is a charge offset that can be introduced upon initialization (see appendix B). We associate $E_{\mathrm{J}010}$ with the Josephson energy of the 0th junction between node 0 and 1, which is shunted by the capacitance C_{01}. This notation, which assigns circuit elements like a Josephson junction to Hamiltonian parameters like a Josephson energy by providing the corresponding edge nodes in the index of the symbols, is kept consistent. For better readability, we do not define all the symbols in the following Hamiltonians individually. The flux quantum Φ_0 will be displayed as Φ_o in the toolbox, to distinguish it from the node flux of node 0. The numerical values for the corresponding spectrum plot in figure 1, which shows the lowest eigenstates of the weakly anharmonic cosine-potential, are the default values, i.e. $C_{01} = 100$ fF and $E_{\mathrm{J}010} \approx 9.69$ GHz $\cdot h$.

For the fluxonium qubit, the toolbox determines the symbolic Hamiltonian

$$H = -E_{\mathrm{J}010}\,\cos\left(\frac{\Phi_1}{\Phi_0}\right) + \frac{\left(\Phi_1 + \tilde{\Phi}_{010}\right)^2}{2L_{010}} + \frac{q_1^2}{2C_{01}}, \tag{30}$$

where $\tilde{\Phi}_{010}$ labels the offset flux of the inductive loop. An excerpt of the spectrum of this Hamiltonian is shown in figure 1 with $C_{01} = 10$ fF, $L_{010} = 0.5\ \mu$H, $E_{\mathrm{J}010} \approx 48.43$ GHz $\cdot h$ and $\tilde{\Phi}_{010} = \pi\Phi_0$. It shows the typical low-lying 0 and 1 states that are localized in the wells, with higher plasma states several GHz above.

Finally, the persistent-current flux qubit Hamiltonian constructed by the toolbox can be written as

$$\begin{aligned} H = &-E_{\mathrm{J}010}\,\cos\left(\frac{\Phi_1}{\Phi_0}\right) - E_{\mathrm{J}020}\,\cos\left(\frac{\Phi_2}{\Phi_0}\right) - E_{\mathrm{J}120}\,\cos\left(\frac{\Phi_2 - \Phi_1 + \tilde{\Phi}_{120}}{\Phi_0}\right) \\ &+ \frac{q_1^2(C_{02} + C_{12}) + 2q_1q_2C_{12} + q_2^2(C_{01} + C_{12})}{2(C_{01}C_{02} + C_{01}C_{12} + C_{02}C_{12})}. \end{aligned} \tag{31}$$

We depict the ground state of this Hamiltonian in figure 1 for $\alpha = 0.7$, $C_{01} = C_{02} = \frac{C_{12}}{\alpha} = 50$ fF, $E_{\mathrm{J}010} = E_{\mathrm{J}020} = \frac{E_{\mathrm{J}020}}{\alpha} \approx 9.69$ GHz $\cdot h$ and $\tilde{\Phi}_{020} = \pi\Phi_0$. The ground state is localized in the double well potential, which is repeated periodically throughout the chosen flux grid. As for the transmon circuit, due to the periodicity of the potential, the Hamiltonian is implemented in the charge basis. For figure 1, we use the transformation method of the toolbox to visualize the eigenstates in the flux basis.

In order to test the software and to check the accuracy of our numerical implementation, we perform benchmark tests for a variety of circuits. In addition to the three example circuits that have been discussed in this section, we complete the benchmark by adding the 0-π-qubit [41], the four-body-coupler which is referred to as *circuit C* in reference [42] and the circuit of a transmon that is capacitively coupled to a resonator to the list of test circuits. The latter circuit is called the QuCAT circuit here, as a similar version is discussed in the corresponding reference [12]. The code to construct the CircuitQ instances can be found in appendix D. We use existing software implementations to calculate the spectrum of the test circuits as a benchmark and compare the results of CircuitQ to it. As a reference, we used the toolbox scQubits [11] for the transmon, fluxonium, 0-π-qubit and persistent-current flux qubit circuit. The QuCAT circuit has been compared to its implementation in the QuCAT toolbox [12]. The four-body-coupler has been tested against a direct and individualized software implementation. As the test outcome depends on the size of the numerical matrices which represent the charge and flux variables, we vary the dimension of those matrices. The result is shown in figure 4. CircuitQ automatically implements the transmon and flux qubit in the charge basis, and the fluxonium, 0-π-qubit and four-body-coupler in the flux basis. The QuCAT circuit is implemented in a mixture of both bases. For the transmon, fluxonium and flux qubit as well as for the QuCAT circuit, we find a good agreement between CircuitQ and the benchmark implementation. As detailed in appendix F, it is still possible to observe numerical limitations on less complex circuits like the fluxonium qubit. We observe larger deviations for the four-body-coupler and the 0-π-qubit, which are more complex circuits, even for large numerical matrices. For some circuits, a quantization of the node variables is not the most natural choice, as characteristic modes of the system might be a combination of several node variables. To find a more natural quantization, a coordinate transformation of the node variables can be performed prior to quantization. Therefore, the deviation for the four-body-coupler and the 0-π-qubit can be attributed to the lack of an appropriate coordinate transformation.

5. Conclusion

We presented the core functionalities of CircuitQ. With the ability to derive a symbolic and numerical Hamiltonian from a superconducting circuit in an automated way, CircuitQ can serve as a toolbox for the community to analyze superconducting circuits. The input circuit can be a general superconducting circuit

that combines Josephson junctions, linear inductances and capacitances. An automated procedure to analyze superconducting circuits is a beneficial tool for the study of superconducting circuits within the context of quantum information. Apart from the application to computing, superconducting circuits can be also used as a platform in other areas of application like sensing [43] or studying thermodynamics [44].

While the toolbox is currently limited to the computation of few-node circuits, future work should address the optimization of speed and scalability. As an example, on a conventional personal computer, it took below 1 s to initialize an instance of the transmon circuit and calculate the lowest 10 eigenstates and eigenvalues of the numerical Hamiltonian for subsystem matrix dimensions 40 and 80, while for the 0-π-qubit, it took around 72 s for a subsystem dimension 40 and around 105 s for a subsystem dimension 80, still with lacking accuracy as described in the section 4. A key feature of CircuitQ is its dynamic implementation in the charge and flux basis. At the moment, the variables that are quantized are always the node variables of the circuit graph. For some circuits, it is crucial to perform a variable transformation prior to quantization. Adding a suitable transformation would represent an important development step towards the goal of increasing the calculation speed. Another improvement can be made by implementing hierarchical diagonalization such as discussed in reference [10]. In addition, the toolbox is written in a modular fashion that allows for extensions towards time-dependent simulations.

Thanks to the general functionality, the possibility to include charge and flux offsets, as well as the incorporation of noise estimates, CircuitQ can serve as a versatile tool for the design of superconducting qubits. Adding more noise channels, especially estimates for the dephasing time, would be an important future addition to the software. Moreover, adding the possibility to incorporate external impedances as circuit elements would allow to estimate noise from first principles [8].

Acknowledgments

We are thankful for fruitful discussions with Jens Koch and his group as well as with Kyle Serniak and Andrew J Kerman. We also thank three anonymous reviewers for their helpful suggestions. PA thanks Glen Bigan Mbeng, Kilian Ender and Benoît Vermersch for helpful discussions and Martin Lanthaler for designing the logo. This work was supported by the Austrian Science Fund (FWF) through a START Grant under Project No. Y1067-N27 and the SFB BeyondC Project No. F7108-N38, the Hauser-Raspe foundation, and the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 817482. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0068. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. TM acknowledges funding by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) under Air Force Contract No. FA8721-05-C-0002. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the US Government. The US Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Appendix A. Parameter values for T_1 estimates

Figure 3 in the main text provides an overview of the T_1 estimates provided by CircuitQ for the example circuits studied in this article, i.e. transmon, fluxonium and flux qubit. In table 2, we list the parameter values chosen for the purpose of this illustration. The numbers for the fluxonium correspond to *qubit A* from reference [6], while the values for the flux qubit are associated to *qubit B* from reference [32].

Appendix B. Code samples for illustrative circuits in figure 1

In the following subsections, we present the code that generates the three instances which are displayed in figure 1 and which are discussed in section 4.

B.1. Transmon

```
import circuitq as cq
import networkx as nx

graph = nx.MultiGraph()
graph.add_edge(0,1, element = 'C')
graph.add_edge(0,1, element = 'J')
circuit = cq.CircuitQ(graph, offset_nodes=[1])

# Numerical implementation and diagonalisation
h_num = circuit.get_numerical_hamiltonian(400,
                grid_length=np.pi*circuit.phi_0)
eigv, eigs = circuit.get_eigensystem()

# Conversion for the plot
circuit.transform_charge_to_flux()
eigs = circuit.estates_in_phi_basis
```

B.2. Fluxonium

```
import circuitq as cq
import networkx as nx

graph = nx.MultiGraph()
graph.add_edge(0,1, element = 'C')
graph.add_edge(0,1, element = 'J')
graph.add_edge(0,1, element = 'L')
circuit = cq.CircuitQ(graph)

# Numerical implementation and diagonalisation
EJ = circuit.c_v["E"]*.5
L = circuit.c_v["L"]*5
C = circuit.c_v["C"]*0.1
phi_ext =np.pi*circuit.phi_0
h_num = circuit.get_numerical_hamiltonian(400,
        parameter_values=[C, EJ, L, phi_ext])
eigv, eigs = circuit.get_eigensystem()
```

B.3. Persistent-current flux qubit

```
import circuitq as cq
import networkx as nx

graph = nx.MultiGraph()
graph.add_edge(0,1, element = 'C')
graph.add_edge(0,1, element = 'J')
graph.add_edge(1,2, element = 'C')
graph.add_edge(1,2, element = 'J')
graph.add_edge(0,2, element = 'C')
graph.add_edge(0,2, element = 'J')
circuit = cq.CircuitQ(graph)

# Numerical implementation and diagonalisation
dim = 51
EJ = 1*circuit.c_v["E"]
alpha = 0.7
C = circuit.c_v["C"]*0.5
phi_ext = np.pi*circuit.phi_0
h_num = circuit.get_numerical_hamiltonian(dim,
parameter_values=[C,C,alpha*C,
                  EJ,EJ,alpha*EJ,phi_ext])
eigv, eigs = circuit.get_eigensystem()

# Conversion for the plot
circuit.transform_charge_to_flux()
eigs = circuit.estates_in_phi_basis
```

Appendix C. Code sample for example circuit in figure 2

The following code demonstrates the initialization of the circuit in figure 2.

Table 2. Overview of the parameter values used to calculate the T_1 times shown in figure 3 for three example circuits. The energies given for the flux qubit relate to the large junctions and the values have to be scaled by α to deduce the corresponding numbers for the small junction. Both the fluxonium and flux qubit are evaluated at the sweet spot. The energy values are given in frequencies and have to be scaled by h to obtain units of energy. If a value is calculated, it is displayed as rounded to two decimal places.

Circuit	Parameters			
	E_J	E_C		
Transmon	10 GHz	0.24 GHz		
	E_J	E_C	E_L	$\tilde{\Phi}_\mathrm{ext}$
Fluxonium	3 GHz	0.8 GHz	1 GHz	$\pi \cdot \Phi_0$
	E_J	E_C	α	$\tilde{\Phi}_\mathrm{ext}$
Flux qubit	86.19 GHz	0.15 GHz	0.42	$\pi \cdot \Phi_0$

```
import circuitq as cq
import networkx as nx

graph = nx.MultiGraph()
graph.add_edge(0,1, element = 'C')
graph.add_edge(0,2, element = 'J')
graph.add_edge(0,2, element = 'C')
graph.add_edge(1,2, element = 'L')
graph.add_edge(1,2, element = 'C')
graph.add_edge(1,3, element = 'C')
graph.add_edge(1,3, element = 'L')
graph.add_edge(2,3, element = 'L')
graph.add_edge(2,3, element = 'C')

circuit = cq.CircuitQ(graph)
```

Appendix D. Code sample for additional benchmark circuits

In the following, we present the generation of the numerical Hamiltonian for the additional benchmark circuits in figure 4, which have not been given in appendix B yet, for an arbitrary subsystem matrix dimension d. We note that for this benchmark task, contrary to the parameter values given in the code samples in appendix B, we choose the values given in table 2 for the listed circuits.

D.1. 0-π−qubit

```
import circuitq as cq
import networkx as nx

graph = nx.MultiGraph()
graph.add_edge(1,2, element = 'C')
graph.add_edge(1,2, element = 'J')
graph.add_edge(2,3, element = 'L')
graph.add_edge(3,4, element = 'J')
graph.add_edge(3,4, element = 'C')
graph.add_edge(4,1, element = 'L')
graph.add_edge(1,3, element = 'C')
graph.add_edge(2,4, element = 'C')

circuit = cq.CircuitQ(graph, ground_nodes=[1])
h_num = circuit.get_numerical_hamiltonian(d,
    parameter_values=   [False,
                         100 * circuit.c_v['C'],
                         100 * circuit.c_v['C'],
                         False, False,
                         False, False,
                         False, False,
                         False, False]    )
```

D.2. 4-body-coupler

```
import circuitq as cq
import networkx as nx
import numpy as np

graph = nx.MultiGraph()
graph.add_edge(0,2, element = 'C')
graph.add_edge(0,2, element = 'J')
graph.add_edge(2,3, element = 'L')
graph.add_edge(2,3, element = 'C')
graph.add_edge(0,3, element = 'C')
graph.add_edge(0,3, element = 'J')
graph.add_edge(2,1, element = 'C')
graph.add_edge(2,1, element = 'J')
graph.add_edge(1,3, element = 'L')
graph.add_edge(1,3, element = 'C')

circuit = cq.CircuitQ(graph, ground_nodes=[0])
L13 = 289.395  # in pH
L23 = 120.416  # in pH
lj12 = 3.75498  # in um
lj22 = 0.395517  # in um
lj33 = 0.373288  # in um
Jc = 5e-6  # critical current density in A/um^2
wJ = 0.2  # junction width in um
Sc = 60e-15  # specific capacitance in F/um^2
Phi0 = 2.06783385 * 10 ** (-15)  # flux quantum
L13 = L13 * 1e-12  # scale pH -> H
L23 = L23 * 1e-12  # scale pH -> H
Ej12 = Phi0 / (2 * np.pi) * Jc * wJ * lj12
Ej22 = Phi0 / (2 * np.pi) * Jc * wJ * lj22
Ej33 = Phi0 / (2 * np.pi) * Jc * wJ * lj33
circuit.get_numerical_hamiltonian(d,
parameter_values=
        [4.455 * 1e-15 + Sc * wJ * lj22,
        70.556 * 1e-15 + Sc * wJ * lj33,
        Sc * wJ * lj12,
        16.832 * 1e-15,
        85.677 * 1e-15,
        Ej22, Ej33, Ej12, L23, L13,
        0.5 * circuit.phi_0,
        0.02 * circuit.phi_0]     )
```

D.3. QuCAT circuit

```
import circuitq as cq
import networkx as nx
import numpy as np

graph = nx.MultiGraph()
graph.add_edge(0,1, element = 'C')
graph.add_edge(0,1, element = 'J')
graph.add_edge(0,2, element = 'C')
graph.add_edge(0,2, element = 'L')
graph.add_edge(1,2, element = 'C')

circuit = cq.CircuitQ(graph, ground_nodes=[0])
circuit.get_numerical_hamiltonian(d,
            grid_length=np.pi*circuit.phi_0,
            parameter_values=[100e-15,100e-15,
                1e-15,(circuit.phi_0**2)/8e-9,
                10e-9 ]          )
```

Appendix E. Symbolic Hamiltonian for circuit in figure 2

CircuitQ provides the symbolic Hamiltonian given in equation (E1) for the example circuit in figure 2. The parasitic capacitances within the kinetic part of the Hamiltonian are labelled with a p in the index.

$$H = \big(C_{01}C_{02}C_{13} + C_{01}C_{02}Cp_{23} + C_{01}C_{13}Cp_{12} + C_{01}C_{13}Cp_{23} + C_{01}Cp_{12}Cp_{23} + C_{02}C_{13}Cp_{12} + C_{02}C_{13}Cp_{23} + C_{02}Cp_{12}Cp_{23}\big)^{-1}.$$

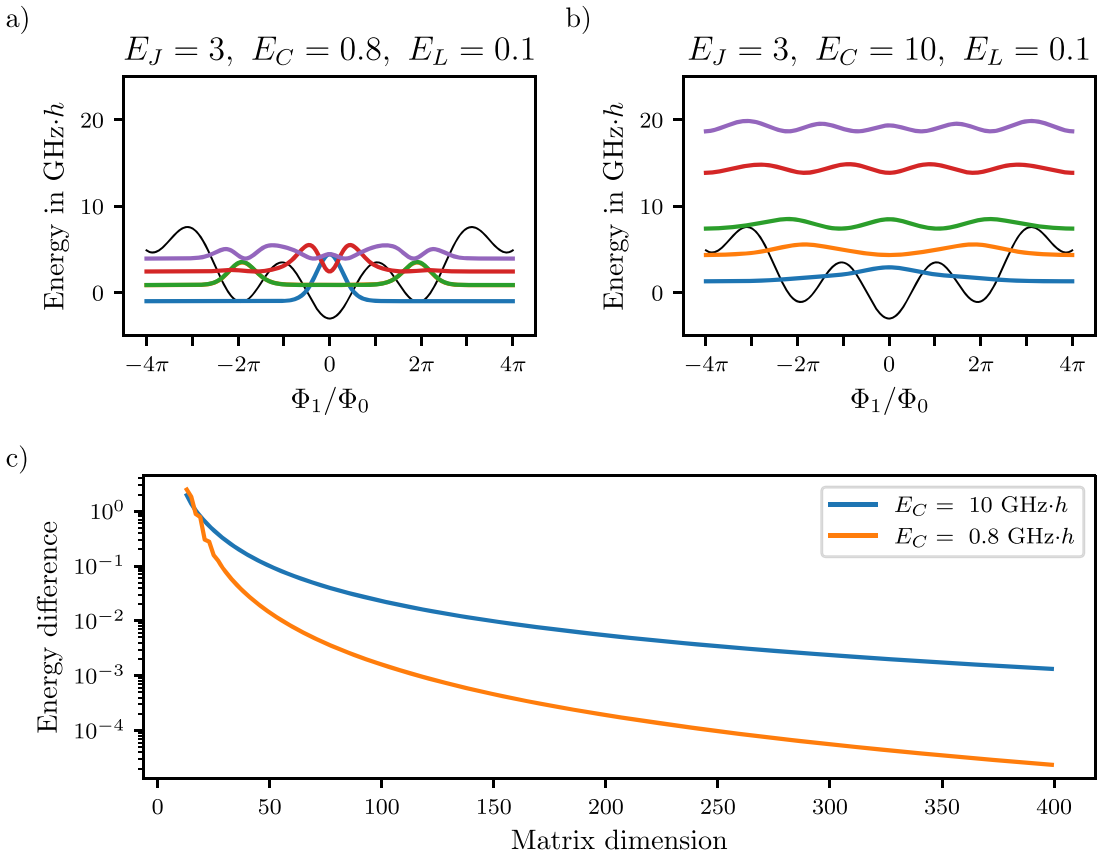

Figure 5. Numerical limitations tested on the fluxonium circuit. (a) and (b) Depiction of the lower energy spectrum of the fluxonium circuit for low and high E_C in the same fashion as in the last row of figure 1. The numerical parameter values chosen are given in the title of the subfigures, while $\tilde{\Phi}_{ext} = 0$ for both subfigures. The black line indicates the potential energy. (c) Difference of the eigenenergies corresponding to the instances introduced in (a) and (b) as a function of the dimension of the numerical matrix representing the numerical Hamiltonian. To calculate the energy difference for a given matrix dimension we sum over the differences between the lowest 5 eigenvalues and the eigenvalues calculated for the previous matrix dimension. The energy difference is in units of GHz ·h and scaled logarithmically. Here, we solely use odd numbers as matrix dimension, as even numbers for the dimension will be converted to odd numbers by CircuitQ.

$$
\begin{aligned}
&\cdot \Big(\frac{q_1}{2}\Big(q_1\big(C_{02}C_{13} + C_{02}Cp_{23} + C_{13}Cp_{12} + C_{13}Cp_{23} + Cp_{12}Cp_{23}\big)\\
&+ q_2\big(C_{13}Cp_{12} + C_{13}Cp_{23} + Cp_{12}Cp_{23}\big)\\
&+ q_3\big(C_{02}C_{13} + C_{13}Cp_{12} + C_{13}Cp_{23} + Cp_{12}Cp_{23}\big)\Big)\\
&+ \frac{q_2}{2}\Big(q_1\big(C_{13}Cp_{12} + C_{13}Cp_{23} + Cp_{12}Cp_{23}\big)\\
&+ q_2\big(C_{01}C_{13} + C_{01}Cp_{23} + C_{13}Cp_{12} + C_{13}Cp_{23} + Cp_{12}Cp_{23}\big)\\
&+ q_3\big(C_{01}Cp_{23} + C_{13}Cp_{12} + C_{13}Cp_{23} + Cp_{12}Cp_{23}\big)\Big)\\
&+ \frac{q_3}{2}\Big(q_1\big(C_{02}C_{13} + C_{13}Cp_{12} + C_{13}Cp_{23} + Cp_{12}Cp_{23}\big)\\
&+ q_2\big(C_{01}Cp_{23} + C_{13}Cp_{12} + C_{13}Cp_{23} + Cp_{12}Cp_{23}\big)\\
&+ q_3\big(C_{01}C_{02} + C_{01}Cp_{12} + C_{01}Cp_{23} + C_{02}C_{13} + C_{02}Cp_{12}\\
&+ C_{13}Cp_{12} + C_{13}Cp_{23} + Cp_{12}Cp_{23}\big)\Big)\Big)\\
&- E_{J020}\cos\left(\frac{\Phi_2}{\Phi_0}\right) - E_{J130}\cos\left(\frac{\Phi_1 - \Phi_3}{\Phi_0}\right) + \frac{\left(\Phi_3 - \Phi_2 + \tilde{\Phi}_{230}\right)^2}{2L_{230}} + \frac{(\Phi_2 - \Phi_1)^2}{2L_{120}}
\end{aligned}
\tag{E1}
$$

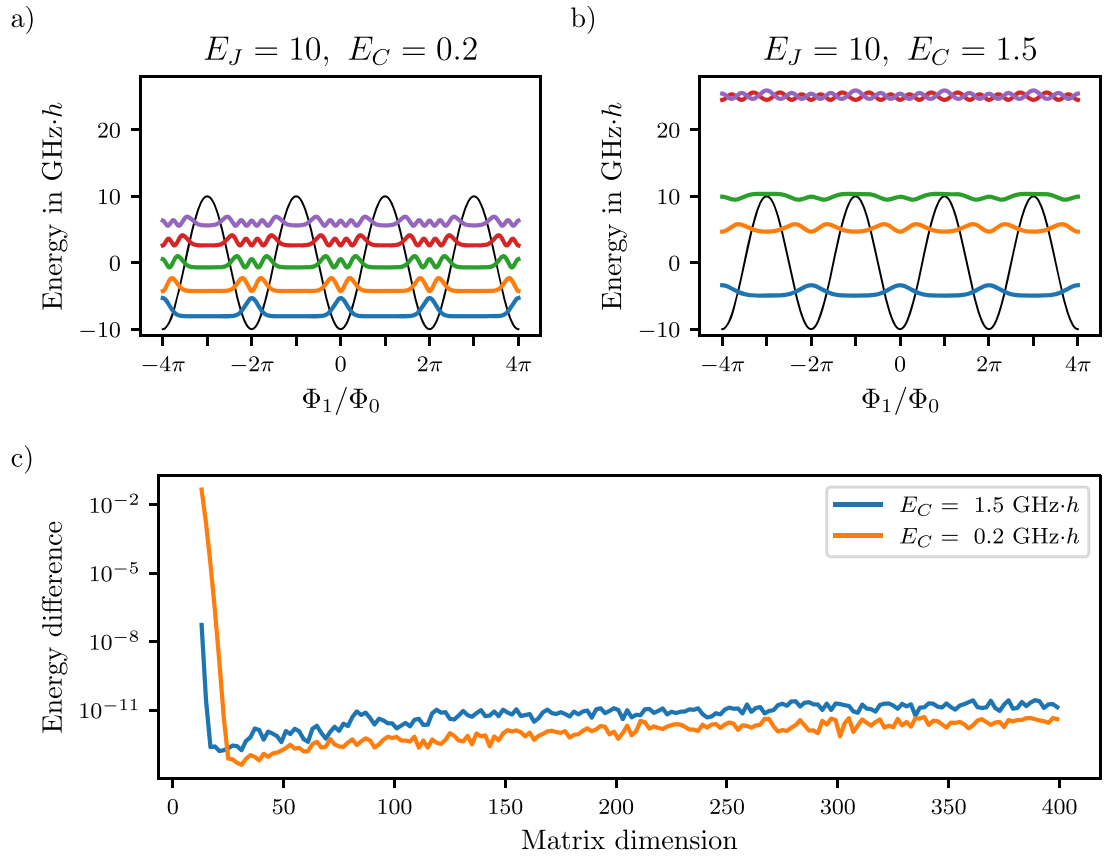

Figure 6. Numerical limitations tested on the transmon circuit. (a) and (b) Lower energy spectrum of the transmon circuit for low and higher E_C depicted in the same fashion as in figures 5(a) and (b). Due to the charge basis implementation, to illustrate the spectrum as a function of flux, the eigenstates are transformed to the flux basis by the basis transformation outlined in section 3.1.2. (c) Difference of the eigenenergies corresponding to the instances introduced in (a) and (b) depicted in the same fashion as in figure 5(c).

Appendix F. Limitations of the numerical treatment

As discussed in section 4, the limitations of our numerical implementation become evident for complex circuits like the 0-π-qubit. However, it is possible to investigate the numerical limitations on more simple circuits like the fluxonium qubit. In figures 5(a) and (b), we depict the lowest eigenstates together with the potential energy of the fluxonium qubit for low and high capacitive energy E_C. The inductive energy E_L has been kept small to avoid strong confinement. We observe the lowest eigenstates to be located within the potential wells for the case of low E_C, while the eigenstates for higher E_C tend to become more delocalized. In CircuitQ, the fluxonium circuit will be implemented in the flux basis, which works well for localized states that are trapped in a harmonic potential. To measure the numerical accuracy of the software implementation, figure 5(c) shows the deviation of the eigenenergies for the instances of subfigures 5(a) and (b) as a function of the numerical matrix dimension. In an ideal case, the spectrum is almost invariant under slight changes of the matrix dimension. Such a convergence can be observed for high values of matrix dimension. However, for lower values of matrix dimension, we observe a significant deviation of the energy, which is, besides the regime of very low matrix dimension, drastically higher for the case of high E_C. This indicates that for the case of weakly localized wavefunctions, the flux basis implementation of the toolbox is reaching its numerical limitation. Moreover, another inaccuracy is introduced for higher lying states due to the cut-off of the numerical flux grid from -4π to 4π.

In figure 6, we present a similar study for the fixed-frequency transmon circuit (see figure 1(a)), which will be implemented in the charge basis. Figure 6(a) depicts the spectrum of the transmon circuit for low E_C with $E_J/E_C = 50$. Here, the wavefunctions are periodically localized within the potential wells. In figure 6(b) we show the spectrum for higher E_C with $E_J/E_C \approx 7$, where higher eingestates become less confined. As before, we depict the difference in energy as a function of matrix dimension in figure 6(c). The energy difference drops of fast for both instances and fluctuates due to numerical fluctuations at small values for high matrix dimension. In comparison to the fluxonium study in figure 5(c), here, the depicted values of energy difference are small as the y-axis is scaled to smaller values. This indicates, that, in contrast to the

flux basis, the charge basis is well suited to describe delocalized states. The states lying energetically above the potential can be associated with free particles, which, in the case of conventional mechanics, are efficiently described in the momentum basis. As the charge basis is analogous to the momentum basis, we find a more accurate description for the transmon circuit with high capacitive energy in comparison to the case of the fluxonium circuit.

ORCID iDs

Philipp Aumann https://orcid.org/0000-0001-8685-3693
Tim Menke https://orcid.org/0000-0002-7205-752X
William D Oliver https://orcid.org/0000-0001-8041-0824
Wolfgang Lechner https://orcid.org/0000-0003-3662-1020

References

[1] Arute F *et al* 2019 Quantum supremacy using a programmable superconducting processor *Nature* **574** 505–10
[2] Blais A, Grimsmo A L, Girvin S and Wallraff A 2021 Circuit quantum electrodynamics *Rev. Mod. Phys.* **93** 025005
[3] Kjaergaard M, Schwartz M E, Braumüller J, Krantz P, Wang J I-J, Gustavsson S and Oliver W D 2020 Superconducting qubits: current state of play *Annu. Rev. Condens. Matter Phys.* **11** 369–95
[4] Weiss D K, Li A C Y, Ferguson D G and Koch J 2019 Spectrum and coherence properties of the current-mirror qubit *Phys. Rev.* B **100** 224507
[5] Paolo A D, Grimsmo A L, Groszkowski P, Koch J and Blais A 2019 Control and coherence time enhancement of the 0-π qubit *New J. Phys.* **21** 043002
[6] Nguyen L B *et al* 2019 High-coherence fluxonium qubit *Phys. Rev.* X **9** 041041
[7] Mirrahimi M, Leghtas Z, Albert V V, Touzard S, Schoelkopf R J, Jiang L and Devoret M H 2014 Dynamically protected cat-qubits: a new paradigm for universal quantum computation *New J. Phys.* **16** 045014
[8] Burkard G, Koch R H and DiVincenzo D P 2004 Multilevel quantum description of decoherence in superconducting qubits *Phys. Rev.* B **69** 064503
[9] Vool U and Devoret M 2017 Introduction to quantum electromagnetic circuits *Int. J. Circ. Theor. Appl.* **45** 897–934
[10] Kerman A J 2020 Efficient numerical simulation of complex Josephson quantum circuits (arXiv:2010.14929 [quant-ph])
[11] Groszkowski P and Koch J 2021 Scqubits: a Python package for superconducting qubits *Quantum* **5** 583
[12] Gely M F and Steele G A 2020 QuCAT: quantum circuit analyzer tool in Python *New J. Phys.* **22** 013025
[13] Klots A and Ioffe L B 2018 SuperQuant GitHub https://github.com/andreyklots/SuperQuantPackage
[14] Minev Z *et al* 2021 Qiskit Metal: an open-source framework for quantum device design & analysis Zenodo
[15] Heinsoo Z *et al* 2021 KQCircuits Zenodo https://doi.org/10.5281/zenodo.4944796
[16] *Link to GitHub Repository* https://github.com/PhilippAumann/circuitq
[17] Girvin S M 2014 Circuit QED: superconducting qubits coupled to microwave photons *Quantum Machines: Measurement and Control of Engineered Quantum Systems* (Oxford: Oxford University Press) https://oxford.universitypressscholarship.com/view/10.1093/acprof:oso/9780199681181.001.0001/acprof-9780199681181-chapter-3
[18] Nigg S E, Paik H, Vlastakis B, Kirchmair G, Shankar S, Frunzio L, Devoret M H, Schoelkopf R J and Girvin S M 2012 Black-box superconducting circuit quantization *Phys. Rev. Lett.* **108** 240502
[19] Solgun F, Abraham D W and DiVincenzo D P 2014 Blackbox quantization of superconducting circuits using exact impedance synthesis *Phys. Rev.* B **90** 134504
[20] Minev Z K *et al* 2021 Energy-participation quantization of Josephson circuits *npj Quantum Inf.* **7** 131
[21] Minev Z K, McConkey T G, Takita M, Corcoles A D and Gambetta J M 2021 Circuit quantum electrodynamics (cQED) with modular quasi-lumped models (arXiv:2103.10344 [cond-mat, physics:quant-ph])
[22] Leib M 2015 Many-body physics with circuit quantum electrodynamics PhD Thesis Technische Universität München http://mediatum.ub.tum.de/1241486?id=1241486&change_language=en
[23] Hagberg A A, Schult D A and Swart P J 2008 Exploring network structure, dynamics, and function using networkX *Proc. 7th Python in Science Conf.* 11–15 Pasadena, CA USA ed G Varoquaux, T Vaught and J Millman http://conference.scipy.org/proceedings/SciPy2008/paper_2/
[24] Meurer A *et al* 2017 SymPy: symbolic computing in Python *PeerJ Comput. Sci.* **3** e103 https://peerj.com/articles/cs-103
[25] Langford N K 2013 Circuit QED—lecture notes (arXiv:1310.1897)
[26] Virtanen P *et al* 2020 SciPy 1.0: fundamental algorithms for scientific computing in Python *Nat. Methods* **17** 261–72
[27] Catelani G, Schoelkopf R J, Devoret M H and Glazman L I 2011 Relaxation and frequency shifts induced by quasiparticles in superconducting qubits *Phys. Rev.* B **84** 064517
[28] Fernandes R M 2015 Lecture notes: BCS theory of superconductivity https://portal.ifi.unicamp.br/images/files/graduacao/aulas-on-line/fen-emerg/lecture_notes_BCS.pdf
[29] Cochran J F and Mapother D E 1958 Superconducting transition in aluminum *Phys. Rev.* **111** 132–42
[30] Smith W C, Kou A, Xiao X, Vool U and Devoret M H 2020 Superconducting circuit protected by two-Cooper-pair tunneling *npj Quantum Inf.* **6** 8
[31] Pop I M, Geerlings K, Catelani G, Schoelkopf R J, Glazman L I and Devoret M H 2014 Coherent suppression of electromagnetic dissipation due to superconducting quasiparticles *Nature* **508** 369–72
[32] Yan F *et al* 2016 The flux qubit revisited to enhance coherence and reproducibility *Nat. Commun.* **7** 12964
[33] Lehoucq R B, Sorensen D C and Yang C 1998 *ARPACK Users' Guide. Software, Environments and Tools* (Philadelphia: Society for Industrial and Applied Mathematics)
[34] Martinis J M, Ansmann M and Aumentado J 2009 Energy decay in superconducting Josephson-junction qubits from nonequilibrium quasiparticle excitations *Phys. Rev. Lett.* **103** 097002

[35] Serniak K 2019 Nonequilibrium quasiparticles in superconducting qubits PhD Thesis Yale University https://cpb-us-w2.com/campuspress.yale.edu/dist/2/3627/files/2020/10/kyle_thesis.pdf
[36] Krantz P, Kjaergaard M, Yan F, Orlando T P, Gustavsson S and Oliver W D 2019 A quantum engineer's guide to superconducting qubits *Appl. Phys. Rev.* **6** 021318
[37] Koch J *et al* 2007 Charge-insensitive qubit design derived from the Cooper pair box *Phys. Rev.* A **76** 042319
[38] Manucharyan V E, Koch J, Glazman L I and Devoret M H 2009 Fluxonium: single Cooper-pair circuit free of charge offsets *Science* **326** 113–6
[39] Orlando T P, Mooij J E, Tian L, van der Wal C H, Levitov L S, Lloyd S and Mazo J J 1999 Superconducting persistent-current qubit *Phys. Rev.* B **60** 15398–413
[40] Place A P M *et al* 2021 New material platform for superconducting transmon qubits with coherence times exceeding 0.3 milliseconds *Nat. Commun.* **12** 1779
[41] Brooks P, Kitaev A and Preskill J 2013 Protected gates for superconducting qubits *Phys. Rev.* A **87** 052306
[42] Menke T *et al* 2021 Automated design of superconducting circuits and its application to four-local couplers *npj Quantum Inf.* **7** 1–8
[43] Danilin S and Weides M 2021 Quantum sensing with superconducting circuits (arXiv:2103.11022 [quant-ph])
[44] Kerremans T, Samuelsson P and Potts P 2022 Probabilistically violating the first law of thermodynamics in a quantum heat engine *SciPost Phys.* **12** 168