Computer Physics Communications 179 (2008) 647–664

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Simulation of *n*-qubit quantum systems. IV. Parametrizations of quantum states, matrices and probability distributions [☆]

T. Radtke [a,*], S. Fritzsche [b,c]

[a] Institut für Physik, Universität Kassel, Heinrich-Plett-Str. 40, D-34132 Kassel, Germany
[b] Max-Planck-Institut für Kernphysik, D-69029 Heidelberg, Germany
[c] Gesellschaft für Schwerionenforschung (GSI), D-64291 Darmstadt, Germany

ARTICLE INFO

Article history:
Received 28 April 2008
Accepted 5 June 2008
Available online 19 June 2008

PACS:
03.67.Ac
03.67.Mn
03.65.Ud

Keywords:
Qubit
Quantum register
Separable states
Parametrization

ABSTRACT

Entanglement is known today as a key resource in many protocols from quantum computation and quantum information theory. However, despite the successful demonstration of several protocols, such as teleportation or quantum key distribution, there are still many open questions of how entanglement affects the efficiency of quantum algorithms or how it can be protected against noisy environments. The investigation of these and related questions often requires a search or optimization over the set of quantum states and, hence, a parametrization of them and various other objects. To facilitate this kind of studies in quantum information theory, here we present an extension of the Feynman program that was developed during recent years as a toolbox for the simulation and analysis of quantum registers. In particular, we implement parameterizations of hermitian and unitary matrices (of arbitrary order), pure and mixed quantum states as well as separable states. In addition to being a prerequisite for the study of many optimization problems, these parameterizations also provide the necessary basis for heuristic studies which make use of random states, unitary matrices and other objects.

Program summary

Program title: FEYNMAN
Catalogue identifier: ADWE_v4_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/ADWE_v4_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html
No. of lines in distributed program, including test data, etc.: 24 231
No. of bytes in distributed program, including test data, etc.: 1 416 085
Distribution format: tar.gz
Programming language: Maple 11
Computer: Any computer with Maple software installed
Operating system: Any system that supports Maple; program has been tested under Microsoft Windows XP, Linux
Classification: 4.15
Does the new version supersede the previous version?: Yes
Nature of problem: During the last decades, quantum information science has contributed to our understanding of quantum mechanics and has provided also new and efficient protocols, based on the use of entangled quantum states. To determine the behavior and entanglement of *n*-qubit quantum registers, symbolic and numerical simulations need to be applied in order to analyze how these quantum information protocols work and which role the entanglement plays hereby.
Solution method: Using the computer algebra system Maple, we have developed a set of procedures that support the definition, manipulation and analysis of *n*-qubit quantum registers. These procedures also help to deal with (unitary) logic gates and (nonunitary) quantum operations that act upon the quantum registers. With the parameterization of various frequently-applied objects, that are implemented in the present version, the program now facilitates a wider range of symbolic and numerical studies. All

[☆] This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
[*] Corresponding author.
E-mail addresses: t.radtke@physik.uni-kassel.de (T. Radtke), s.fritzsche@gsi.de (S. Fritzsche).

commands can be used interactively in order to simulate and analyze the evolution of n-qubit quantum systems, both in ideal and noisy quantum circuits.
Reasons for new version: In the first version of the FEYNMAN program [1], we implemented the data structures and tools that are necessary to create, manipulate and to analyze the state of quantum registers. Later [2,3], support was added to deal with quantum operations (noisy channels) as an ingredient which is essential for studying the effects of decoherence. With the present extension, we add a number of parametrizations of objects frequently utilized in decoherence and entanglement studies, such that as hermitian and unitary matrices, probability distributions, or various kinds of quantum states. This extension therefore provides the basis, for example, for the optimization of a given function over the set of pure states or the simple generation of random objects.
Running time: Most commands that act upon quantum registers with five or less qubits take $\leqslant 10$ seconds of processor time on a Pentium 4 processor with $\geqslant 2$ GHz or newer, and about 5–20 MB of working memory (in addition to the memory for the Maple environment). Especially when working with symbolic expressions, however, the requirements on CPU time and memory critically depend on the size of the quantum registers, owing to the exponential growth of the dimension of the associated Hilbert space. For example, complex (symbolic) noise models, i.e. with several symbolic Kraus operators, result for multi-qubit systems often in very large expressions that dramatically slow down the evaluation of e.g. distance measures or the final-state entropy, etc. In these cases, Maple's *assume* facility sometimes helps to reduce the complexity of the symbolic expressions, but more often only a numerical evaluation is possible eventually. Since the complexity of the various commands of the FEYNMAN program and the possible usage scenarios can be very different, no general scaling law for CPU time or the memory requirements can be given.
References:
[1] T. Radtke, S. Fritzsche, Comput. Phys. Comm. 173 (2005) 91.
[2] T. Radtke, S. Fritzsche, Comput. Phys. Comm. 175 (2006) 145.
[3] T. Radtke, S. Fritzsche, Comput. Phys. Comm. 176 (2007) 617.

1. Introduction

In the last years, the field of quantum information has become an established branch of research in physics with interesting connections to mathematics and computer science. The interest arises from the fact that, by utilizing entangled, i.e. non-classical states of matter, novel quantum information protocols like teleportation [1], quantum cryptography [2] and efficient quantum computation have opened up new possibilities that have no counterparts in classical information processing [3]. Successful experimental implementations have shown that quantum systems can, in principle, solve certain problems efficiently which are intractable for classical processors, for instance, the factorization of large numbers [4]. Moreover, these capabilities can exist—within certain limits—even in the presence of imperfections and the inevitable noise that is due to the interactions with the environment.

Despite the encouraging result that quantum information processing should be possible also in realistic experimental scenarios, there are still many open questions dealing, for example, with the role of entanglement as the crucial but fragile resource in quantum information and its protection against decoherence in the physical realizations of quantum protocols. In order to investigate those or related problems, the simulation of quantum registers is a useful tool and often even the only way to gain more insight. To facilitate the simulation and analysis of n-qubit quantum systems, we have developed the FEYNMAN program, a quantum simulator package within the framework of the computer algebra system MAPLE. The main motivation for the development has been to provide an easily extendible set of symbolic and numerical utilities, which help shorten some of the typical calculations as they appear frequently in the simulation of quantum registers. In the first two program versions [5,6] we therefore introduced the program's basic data structures along with the tools for creating and manipulating quantum registers, applying quantum gates to them and analyzing their properties with regard to entanglement, entropy, or their distance from other quantum states. The previous program update [7] added extensive support for quantum operations which enables the user to simulate decoherence effects and analyze the properties of quantum channels by making use of the duality (Jamiołkowski isomorphism) between states and channels.

In addition to the large collection of measures that are already implemented in the FEYNMAN program, it also often occurs in the context of quantum information theory that some quantity of interest is defined as an optimization problem, e.g., over the set of all pure states or density matrices or over the set of unitary operators. Often, there are no known analytical expressions for such quantities so that only a numerical evaluation is possible. To perform such a computation, the search space has to be parametrized. In order to facilitate this process, the present program revision provides a variety of parametrizations for frequently used objects like hermitian and unitary matrices, general pure and mixed states and separable pure and mixed states. Apart from optimization or search problems, the availability of parametrizations for the most frequently used objects implies the ability to create random objects which might be of interest in a first heuristic approach to a problem. Therefore, together with the various efficiency improvements that have been made, the present program extension provides an enhanced support for numerical investigations. Consequently, the present extension further increases the flexibility and the range of possible applications of the FEYNMAN program, thus making it an attractive toolbox for research and education in the field of quantum computation and information.

The rest of the work is organized as follows. In the next section, we briefly summarize some theoretical background on the parametrization of hermitian and unitary matrices as well as pure and mixed quantum states. We recall the basic properties of these objects and present the parametrization formulas that are used in the program. Based on this background, Section 3 later describes how these parametrizations are embedded in the FEYNMAN program and gives an overview of the program changes. Section 4 then provides four short examples that illustrate the interactive work with the program where some of the new features are shown in more detail. In Section 5, finally, a brief summary and outlook on the possible further development of the program is given.

2. Theoretical background

In the following, we recall some basic properties of hermitian and unitary matrices as well as the different kinds of quantum states that are needed to understand their parametrization and implementation in the code. The aim is here to summarize all important formulas and major schemes of parametrization, simply for the sake of reference and in order to facilitate the discussion of our examples in Section 4; for all further proofs and derivations, however, we shall refer the reader to the literature.

2.1. Parametrization of hermitian matrices

Let us start with the hermitian (or self-adjoint) matrices whose relevance for quantum mechanics arises from the fact that hermitian operators (matrices) represent the observables of a quantum system, i.e. measurable quantities. Moreover, an important subset of the hermitian matrices also occurs frequently in form of the density matrices which can describe both, pure and mixed states of the system. From the definition of hermitian operator as being self-adjoint, $A = A^\dagger$, it follows immediately that the elements of a complex hermitian matrix obey the relation: $a_{ij} = a^*_{ji}$. Of course, this implies that the diagonal elements are all real and that N^2 parameters are sufficient to characterize a $N \times N$ hermitian matrix. Therefore, a straightforward parametrization is obtained for these matrices by specifying the N real diagonal elements as well as the real and imaginary parts of, say, the upper triangle. In the FEYNMAN program, this *direct* parametrization of hermitian matrices is obtained by calling the command Feynman_parametrize("hermitian","direct", N, [$p_1, \dots, p_{N^2}$]) with N^2 given (real) parameters.

A different but often useful parametrization is obtained if the $N \times N$ hermitian matrices are considered as elements of a vector space over the real numbers, i.e. by making use of the fact that any linear combination of hermitian matrices with real coefficients leads again to a hermitian matrix. As discussed above, the dimension of this vector space must be N^2. Together with the identity matrix I, a suitable basis for this vector space is given by the set of $N^2 - 1$ linearly independent hermitian matrices that are known as generalized Pauli or Gell-Mann matrices from the literature. This set of Gell-Mann matrices, $\{\lambda_i\}$, can be constructed in the following way [8,9]:

1. For every pair (i, j) with $i, j = 1, 2, \dots, N$ and $i < j$, we define two $N \times N$ matrices $\lambda^{\{1\}}(i, j)$ and $\lambda^{\{2\}}(i, j)$ of the form
$$\left[\lambda^{\{1\}}(i,j)\right]_{\mu,\nu} = \delta_{j\mu}\delta_{i\nu} + \delta_{j\nu}\delta_{i\nu}, \qquad \left[\lambda^{\{2\}}(i,j)\right]_{\mu,\nu} = -i(\delta_{i\mu}\delta_{j\nu} - \delta_{i\nu}\delta_{i\nu}). \tag{1}$$
This gives rise to $N(N-1)$ linearly independent matrices.
2. The remaining $N - 1$ matrices have a diagonal shape and they are defined as
$$\lambda_{k^2-1} = \sqrt{\frac{2}{k^2-k}} \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1)_{k-1} & 0 & \dots & 0 \\ 0 & 0 & 0 & -(k-1) & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}_{N \times N} \tag{2}$$
for $k = 2, 3, \dots, N$, i.e. for every k there are $k - 1$ ones on the main diagonal and the kth entry is $-(k - 1)$.

Within the FEYNMAN program, the set of Gell-Mann matrices is provided by the command Feynman_define("SU generators", N, "symbolic") which, for $N = 2$, yields the well-known Pauli matrices
$$\lambda_1 = \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \lambda_2 = \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad \lambda_3 = \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{3}$$

Note that the set $\{\lambda_i\}$ consists only of traceless matrices and does not include the identity matrix I_N. It is easy to verify that the identity matrix (which might be scaled with some real normalization constant $\sqrt{2/N}$) is hermitian and linearly independent from the $\{\lambda_i\}$ matrices so that we can use it to form a complete N^2-dimensional basis for the vector space of the $N \times N$ hermitian matrices. Using this complete basis, any hermitian matrix A can be written in the form
$$A = p_0\sqrt{2/N}I_N + \sum_{i=1}^{N^2-1} p_i\lambda_i, \tag{4}$$
where $\{p_i\}$ are the N^2 real parameters. The parametrization (4) of the hermitian matrices in terms of the generalized Gell-Mann matrices can be utilized in the FEYNMAN program by Feynman_parametrize("hermitian","standard", N, [$p_1, \dots, p_{N^2}$]).

2.2. Parametrization of unitary matrices

Another important class of matrices are the unitary matrices that appear in many different fields of modern physics. In quantum theory, for example, unitary operators (matrices) are known to describe the time evolution of any closed system as easily seen from the Schrödinger equation. For qubits and quantum registers that are sufficiently isolated from their environment all quantum gates must therefore be given by unitary operations.

A square matrix U of dimension $N \times N$ is called a *unitary* matrix if the adjoint—that is the complex conjugated and transposed—matrix is equal to its inverse
$$U^\dagger = (U^*)^T = U^{-1}, \tag{5}$$

so that

$$U^\dagger U = I, \tag{6}$$

where I is the $N \times N$ identity matrix. From this simple and well-known relation between the adjoint and the inverse matrix of U, it becomes clear why, for closed systems, any sequence of unitary operators and, hence, any quantum circuit is always reversible as long as no measurement is performed. For a unitary matrix, moreover, the modulus of the determinant is always

$$|\det U| = 1, \tag{7}$$

and one can easily show that these matrices are closed under the usual matrix product. Since, in addition, the identity and the inverse of a unitary matrix are also unitary, the $N \times N$ unitary matrices form a group which is often denoted as $U(N)$. From this set of matrices, the subset of matrices with $\det U = +1$ are called *special unitary*, and they form the subgroup $SU(N)$ of the general unitary group $U(N)$.

Due to the importance of the unitary operators in quantum theory, various representations and parametrizations have been presented and discussed in the literature. Although they all describe either the group $U(N)$ or $SU(N)$, they might be more or less favorable for certain applications and, therefore, several of these parametrizations should be easily accessible within the Feynman program.

2.2.1. *Standard parametrization of $SU(N)$*

The canonical way of parametrizing the set $U(N)$ follows from the well-known relation

$$U = e^{iH}, \tag{8}$$

i.e. any unitary matrix can be generated from some hermitian matrix H. In particular, we can apply the $N^2 - 1$ hermitian and traceless Gell-Mann matrices $\{\lambda_i\}$ from the previous section to generate all matrices from the group $SU(N)$. Note that we do not need to consider the identity matrix for generating the special unitary matrices $SU(N)$ since the exponentiation of $i\alpha I_N$ gives rise to just the (complex) phase factor $e^{i\alpha}$ that distinguishes the groups $U(N)$ and $SU(N)$ from each other. Using the Gell-Mann generators (1)–(2) from above, the *standard* parametrization of the group $SU(N)$ is given by

$$U = \exp\left(i \sum_{i=1}^{N^2-1} \theta_i \lambda_i \right) \tag{9}$$

with the $N^2 - 1$ real parameters θ_i. Within Feynman, this parametrization can be accessed by the command Feynman_parametrize("SU", "standard", N, [$\theta_1, \ldots, \theta_{N^2-1}$]).

2.2.2. *Generalized Euler angle parametrization of $SU(N)$*

An alternative and sometimes computationally favorable parametrization of $SU(N)$ is due to Tilma and Sudarshan [8] who generalized the parametrization of $U \in SU(2)$ in terms of the Euler angles,

$$U = e^{i\sigma_z\theta_1} e^{i\sigma_y\theta_2} e^{i\sigma_z\theta_3}, \tag{10}$$

and where σ_y and σ_z are the usual Pauli matrices [cf. Eq. (3)]. For $N > 2$, this parametrization can be generalized as [8]

$$\begin{aligned}
U &= \prod_{N \geqslant m \geqslant 2} \left(\prod_{2 \leqslant k \leqslant m} A\big(k, j(m)\big) \right) e^{i\lambda_3 \theta_{N^2-(N-1)}} \ldots e^{i\lambda_{(N-1)^2-1}\theta_{N^2-2}} e^{i\lambda_{N^2-1}\theta_{N^2-1}}, \\
A\big(k, j(m)\big) &= e^{i\lambda_3 \theta_{(2k-3)+j(m)}} \\
j(m) &= \begin{cases} 0 & m = N, \\ \sum_{0 \leqslant l \leqslant N-m-1} 2(m+l) & m \neq M. \end{cases}
\end{aligned} \tag{11}$$

Here, again, we have $N^2 - 1$ real parameters, θ_i, as necessary for covering all matrices of $SU(N)$. This form can be invoked in the code by Feynman_parametrize("SU", "Euler angles", N, [$\theta_1, \ldots, \theta_{N^2-1}$]).

2.2.3. *Parametrization of $U(N)$*

Since every matrix $U \in U(N)$ can be constructed from some special unitary matrix $U_0 \in SU(N)$ just by multiplying an overall phase

$$U = e^{i\theta_{N^2}} U_0, \tag{12}$$

nothing need to said about the parametrization of the $N \times N$ unitary matrices; in the Feynman program, this parametrization can be utilized by calling either Feynman_parametrize("U", "standard", N, [$\theta_1, \ldots, \theta_{N^2}$]) or Feynman_parametrize("U", "Euler angles", N, [$\theta_1, \ldots, \theta_{N^2}$]), respectively.

2.2.4. *Jarlskog's parametrization*

Recently, moreover, Jarlskog [11] presented a simple and recursive parametrization of the $N \times N$ unitary matrices $U(N)$. In this parametrization, a unitary matrix U is expressed as a product of three unitary matrices

$$U^{(N)} = \Phi^{(N)}(\vec{\alpha}) V^{(N)} \Phi^{(N)}(\vec{\beta}), \tag{13}$$

where the (two) unitary matrices $\Phi^{(N)}$ are diagonal,

$$\Phi^{(N)}(\vec{\alpha}) = \operatorname{diag}\big(e^{i\alpha_1}, e^{i\alpha_2}, \ldots, e^{i\alpha_N}\big) \tag{14}$$

with all the α's being real and similarly for $\Phi^{(N)}(\vec{\beta})$ and its parameters $\beta_1, \ldots, \beta_N$. In Ref. [11], these diagonal matrices Φ are called *external pure phase* matrices. Furthermore, the nontrivial matrix $V^{(N)}$ can be written in the form

$$V^{(N)} = A_{N,2} A_{N,3} \ldots A_{N,N-1} A_{N,N}, \tag{15}$$

where the $A_{N,k}$ are given by

$$A_{N,k} = \begin{pmatrix} \mathbb{A}^{(k)} & 0 \\ 0 & I_{N-k} \end{pmatrix}, \tag{16}$$

and where I_{N-k} denotes the $(N-k) \times (N-k)$ identity matrix (which does not appear for $k = N$). In this representation, moreover, the $\mathbb{A}^{(k)}$ are $k \times k$ unitary matrices,

$$\mathbb{A}^{(k)} = \begin{pmatrix} I_{k-1} - (1 - \cos(\theta_k)) |A^{(k)}\rangle\langle A^{(k)}| & \sin(\theta_k) |A^{(k)}\rangle \\ -\sin(\theta_k) \langle A^{(k)}| & \cos(\theta_k) \end{pmatrix} \tag{17}$$

and $|A^{(k)}\rangle$ complex vectors

$$\left|A^{(k)}\right\rangle = \begin{pmatrix} a_1^{(k)} \\ a_2^{(k)} \\ \cdots \\ \cdots \\ a_{k-1}^{(k)} \end{pmatrix}, \qquad \left\langle A^{(k)} \middle| A^{(k)} \right\rangle = 1, \tag{18}$$

that are normalized and of dimension $(k-1)$. Despite its somewhat sophisticated form, the parametrization (13)–(18) of the $N \times N$ unitary matrices is computationally attractive, especially for large N. In the Feynman program, Jarlskog's parametrization is implemented by the command Feynman_parametrize("U", "Jarlskog", N, $[\theta_1, \ldots, \theta_{N^2+1}]$). For the sake of simplicity, however, this implementation makes use of $N^2 + 1$ parameters, i.e. *one* more than strictly necessary since we use $2N$ parameters for the two diagonal matrices $\Phi^{(N)}(\vec{\alpha})$ and $\Phi^{(N)}(\vec{\beta})$. These parameters are not all independent as only the sums $\alpha_i + \beta_j$ with $i, j = 1, \ldots, N$ contribute in the final unitary matrix (see [11] for a more detailed discussion of the parameter counting).

2.3. Parametrization of classical probability distributions

For the parametrizations of quantum states and density matrices, it is often useful to have a normalized classical probability distribution $\{p_i\}$ available with $\sum_i p_i = 1$. For such distributions, a commonly used trigonometric parametrization is given by [12]

$$p_i = \sin^2 \theta_{i-1} \prod_{j=i}^{N-1} \cos^2 \theta_j \quad \text{and with } \theta_0 = \pi/2, \tag{19}$$

i.e. in terms of the $N-1$ real parameters $\theta_1, \ldots, \theta_{N-1}$. More explicitly, this vector of probabilities can be written as

$$P = \begin{pmatrix} \cos^2(\theta_1) \cdots \cos^2(\theta_{N-1}) \\ \sin^2(\theta_1) \cos^2(\theta_2) \cdots \cos^2(\theta_{N-1}) \\ \vdots \\ \sin^2(\theta_{N-2}) \cos^2(\theta_{N-1}) \\ \sin^2(\theta_{N-1}) \end{pmatrix}. \tag{20}$$

In the Feynman program, this (so-called) *standard* parametrization of a classical probability distribution (CPD) is implemented as Feynman_parametrize(“CPD”, “standard”, N, $[\theta_1, \ldots, \theta_{N-1}]$.

2.4. Classification of quantum states

Before we shall continue with describing the parametrization of quantum states, let us first recall the basic classification of quantum states with respect to pure and mixed states on the one hand and the subclassification into entangled and separable states on the other hand.

In the usual computational basis, an N-dimensional pure state is represented by a complex normalized N-vector $|\psi\rangle$. Each complex vector component $c_i = |c_i| e^{i\theta_i}$ can be expressed by its modulus $|c_i|$ and a phase factor $e^{i\theta_i}$. As the overall phase of the quantum state is not observable and therefore physically irrelevant it is common to neglect one (usually the first) phase factor and interpret the remaining $N-1$ phase factors as relative phases. Moreover, it is well known that the squares of the moduli describe the probabilities $p_i = |c_i|^2$ of finding the quantum system in the corresponding basis states when a measurement in that basis is performed. As discussed in Section 2.3, the description of the N normalized probabilities p_i requires $N-1$ real parameters. Together with the $N-1$ relative phases, the total number of parameters for an N-dimensional state vector amounts to $2N-2$.

Mixed quantum states are described by density matrices ρ, i.e. hermitian, positive semidefinite and trace-normalized operators. The positive semidefiniteness implies that all eigenvalues of ρ are nonnegative which is consistent with their interpretation as weights or probabilities in an ensemble of pure states, i.e. due to the spectral theorem, any mixed state can be written as

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|. \tag{21}$$

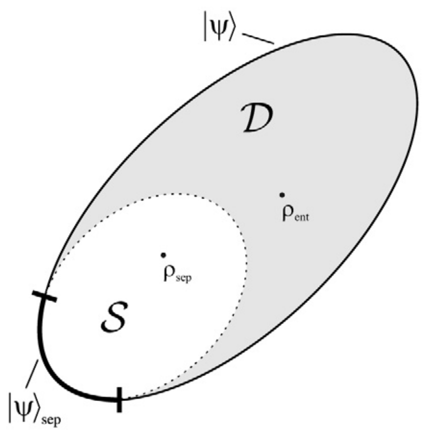

Fig. 1. Schematic visualization of the convex set $\mathcal{D}$ of all states and the convex subset $\mathcal{S}$ of all separable states. Note that, formally, the closure of $\mathcal{S}$ corresponds to the *pure* product states $|\psi\rangle_{\text{sep}}$ which are however a subset of the closure of $\mathcal{D}$. Therefore, strictly speaking, the dotted line cannot belong to the closure of $\mathcal{S}$ because these points represent mixed states.

Moreover, we recall that any statistical 'mixture' of density matrices (i.e. a convex combination) of the form

$$\rho = \sum_i p_i \rho_i \tag{22}$$

with $p_i \geqslant 0$ and $\sum_i p_i = 1$ also yields a valid density matrix. Hence, the set of density matrices forms a convex set [14] that is often denoted $\mathcal{D}$.[1] Together with the pure-state decomposition (21), one can see that the pure states correspond to the closure of $\mathcal{D}$ since they cannot be written as a convex combination of two density matrices. In other words, the pure states are the extreme points of the set of all states (see also Fig. 1). In order to parametrize a general $N \times N$ density matrix $N^2 - 1$ parameters are required, the same number as required for $SU(N)$ as will be discussed in more detail in Section 2.5.2.

Apart from the fundamental distinction between pure and mixed states, in quantum information theory, it is also important to distinguish between entangled and separable states.

A general multipartite pure state is said to be separable if it can be written as a product state of the form

$$|\psi\rangle_{\text{sep}} = |\psi\rangle_A \otimes |\psi\rangle_B \otimes \cdots \otimes |\psi\rangle_Z. \tag{23}$$

One can easily see that the number of required parameters for a $d_A \cdot d_B \cdot \ldots \cdot d_Z$ dimensional product state is given by the sum of the parameter numbers of its constituents $|\psi\rangle_i$, i.e. $\sum_i 2d_i - 2$ $(i = A \ldots Z)$.

In the case of a (multipartite) mixed state, i.e. a statistical ensemble of (multipartite) pure states, the situation is more involved. Such a state is separable if it can be written in the form

$$\rho_{\text{sep}} = \sum_{i=1}^{r^2} p_i \rho_A^i \otimes \rho_B^i \otimes \cdots \otimes \rho_Z^i \quad \text{with } r = \text{Rank}(\rho_{\text{sep}}), \tag{24}$$

where $p_i \geqslant 0$ and $\sum_i p_i = 1$ and $\rho_k^i = |\psi_k^i\rangle\langle\psi_k^i|$, $k = A \ldots Z$. The fact that (at most) r^2 ensemble members are needed to represent a general mixed state is due to Caratheodory's theorem (see, e.g., [12]). It states that any element of a d-dimensional compact convex set can be represented by a convex combination (i.e. a mixture) of at most $d+1$ points. As we know from the discussion above, the set $\mathcal{D}$ of density matrices is such a compact and convex set. We also know that for the common example of a mixed two-qubit system, the general state space is of dimension $d = N^2 - 1 = (2 \times 2)^2 - 1 = 15$. Therefore, at most $d + 1 = 16$ pure states are required to represent a two-qubit mixed state. In order to represent the convex subset $\mathcal{S}$ of *separable* two-qubit mixed states, only 2×2 pure *product* states may enter the pure-state decomposition of the density matrix where each of the product states requires 4 real parameters. Together with the 15 parameters for the 16 weights p_i, we have then in total (at most) $15 + 16 \cdot 4 = 79$ real parameters that are needed to represent the set of separable two-qubit states.

Fig. 1 shows an attempt to schematically visualize the relations between general pure and mixed states and product states and separable states. Note, however, that this visualization has its subtleties since $\mathcal{D}$ is a convex set with *all* pure states as its extreme points, and at the same time $\mathcal{S}$ is a convex subset of $\mathcal{D}$ with the *pure product* states as extreme points. In other words, being a true subset of $\mathcal{D}$, $\mathcal{S}$ must be included in $\mathcal{D}$ but the boundary of $\mathcal{S}$ (i.e. the pure product states) cannot lie in the interior of $\mathcal{D}$. However, these two facts are hard to visualize at the same time in a simple and intuitive way. Therefore, we recall that in Fig. 1 the dotted part of the border of $\mathcal{S}$ should actually not be interpreted as pure states because the dotted parts are in the interior of $\mathcal{D}$ and, thus, must correspond to mixed states. Instead, only the bold part of the boundary of $\mathcal{S}$ represents the pure product states.

Finally, let us remark that the distinction between entangled and separable states is of central importance in quantum information since entanglement is the crucial resource in many quantum protocols. It is, however, in general not easy to determine if a given state can be written in the form (23) or (24) and, hence, decide whether it is entangled or not. This problem has been addressed by a variety of separability criteria in the literature some of which have been implemented within FEYNMAN (together with several entanglement measures) and have been discussed in a previous version [6].

[1] Note that, in the literature, $\mathcal{D}$ is sometimes also used for the set of *disentangled* (i.e. separable) states.

2.5. Parametrization of quantum states

In the Feynman program, the (n-qubit) quantum states are typically stored and manipulated by means of the data structure qregister(). These quantum registers contain either a normalized 2^n-dimensional state vector, if the state is pure, or a $2^n \times 2^n$ trace normalized density matrix for all mixed states. In the following, however, we shall not restrict the parametrization of quantum states to (multi-)qubit systems but allow for any dimension of states, including those that are not supported by the qregister() data structure.

2.5.1. Pure states

From our discussion in Section 2.4 and by ignoring an irrelevant global phase, an N-dimensional *pure* state can be described by (the square roots of) a probability distribution [see Eq. (19)] as well as $N-1$ relative phases

$$|\psi\rangle = \begin{pmatrix} \sqrt{p_0} \\ \sqrt{p_1} e^{i\theta_1} \\ \vdots \\ \sqrt{p_{N-1}} e^{i\theta_{N-1}} \end{pmatrix}, \tag{25}$$

i.e. in terms of $2N-2$ real parameters. We refer to this as the 'standard' parametrization of pure states which is implemented in the Feynman as Feynman_parametrize("pure state","standard",N,$[p_0, \ldots, p_{N-1}, \theta_1, \ldots, \theta_{N-1}]$).

2.5.2. Mixed state parametrization

In the previous section, we have mentioned that the parametrization of a general $N \times N$ density matrix requires $N(N-1)$ real parameters. Below, these matrices are parametrized in two ways: (i) Directly by specifying all matrix elements, which leads to an over-parametrization, and (ii) based on the generalized Euler angle parametrization of $SU(N)$ from Section 2.2.2.

Direct parametrization. Following the work of Grondalski and coworkers [13], a simple (over-)parametrization of an $N \times N$ density matrix is obtained by using a complex matrix T whose elements $T_{nm} = t_{\mathrm{Re}} + it_{\mathrm{Im}}$ are chosen by altogether $2N^2$ real parameters in the interval $[0, 1]$. From such a matrix, a valid density matrix is then constructed by

$$\rho = \frac{TT^\dagger}{\mathrm{Tr}(TT^\dagger)}. \tag{26}$$

Although this scheme applies $N^2 + N$ parameters more than strictly necessary, its simplicity and fast implementation has been found attractive for various applications.

Generalized Euler angle parametrization. The second parametrization of density matrices makes use of the well-known fact that any density matrix can be expressed in the form

$$\rho = U \rho_d U^\dagger, \tag{27}$$

where ρ_d is the diagonal form of the density matrix and U a unitary matrix. Since ρ is hermitian and positive semi-definite, the diagonal elements of ρ_d must be real and nonnegative and they can be interpreted as a probability distribution. Adopting the notation from [8], we need $N-1$ real parameters to parametrize ρ_d which then takes the form

$$\rho_d = \begin{pmatrix} \sin^2(\theta_1)\cdots\sin^2(\theta_{N-1}) & 0 & \cdots & 0 \\ 0 & \cos^2(\theta_1)\sin^2(\theta_2)\cdots\sin^2(\theta_{N-1}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \cos^2(\theta_{N-1}) \end{pmatrix}. \tag{28}$$

Moreover, by applying Eq. (4), the matrix ρ_d can be expanded in the hermitian basis that consists of the generalized Gell-Mann matrices $\{\lambda_i\}$ and the identity matrix $I_N = \lambda_0$.

Especially for two-qubits (i.e. $N = 4$), Eq. (28) takes the form [10]

$$\begin{aligned} \rho_d &= \begin{pmatrix} \sin^2(\theta_1)\sin^2(\theta_2)\sin^2(\theta_3) & 0 & 0 & 0 \\ 0 & \cos^2(\theta_1)\sin^2(\theta_2)\sin^2(\theta_3) & 0 & 0 \\ 0 & 0 & \cos^2(\theta_2)\sin^2(\theta_3) & 0 \\ 0 & 0 & 0 & \cos^2(\theta_3) \end{pmatrix} \\ &= \frac{1}{4} I_4 + \frac{1}{2}(2w^2 - 1)x^2 y^2 \cdot \lambda_3 + \frac{1}{2\sqrt{3}}(3x^2 - 2)y^2 \cdot \lambda_8 + \frac{1}{2\sqrt{6}}(4y^2 - 3) \cdot \lambda_{15}, \end{aligned} \tag{29}$$

where $w^2 = \sin^2(\theta_1)$, $x^2 = \sin^2(\theta_2)$ and $y^2 = \sin^2(\theta_3)$. Moreover, using Eq. (11) with $N = 4$, the generalized Euler parametrization of U and $U^\dagger$, evaluates to

$$\begin{aligned} U &= e^{i\alpha_1\lambda_3} e^{i\alpha_2\lambda_2} e^{i\alpha_3\lambda_3} e^{i\alpha_4\lambda_5} e^{i\alpha_5\lambda_3} e^{i\alpha_6\lambda_{10}} e^{i\alpha_7\lambda_3} e^{i\alpha_8\lambda_2} e^{i\alpha_9\lambda_3} \\ &\quad \times e^{i\alpha_{10}\lambda_5} e^{i\alpha_{11}\lambda_3} e^{i\alpha_{12}\lambda_2} e^{i\alpha_{13}\lambda_3} e^{i\alpha_{14}\lambda_8} e^{i\alpha_{15}\lambda_{15}}, \end{aligned} \tag{30}$$

while its adjoint matrix becomes

$$U^\dagger = e^{-i\alpha_{15}\lambda_{15}} e^{-i\alpha_{14}\lambda_8} e^{-i\alpha_{13}\lambda_3} e^{-i\alpha_{12}\lambda_2} e^{-i\alpha_{11}\lambda_3} e^{-i\alpha_{10}\lambda_5} e^{-i\alpha_9\lambda_3} e^{-i\alpha_8\lambda_2} e^{-i\alpha_7\lambda_3} \\ \times e^{-i\alpha_6\lambda_{10}} e^{-i\alpha_5\lambda_3} e^{-i\alpha_4\lambda_5} e^{-i\alpha_3\lambda_3} e^{-i\alpha_2\lambda_2} e^{-i\alpha_1\lambda_3}. \tag{31}$$

In this representation of U and $U^\dagger$, we denote the free parameters (angles) by α_i, $i = 1, \dots, 15$, in order to distinguish them from the θ_i in Eq. (28). If we substitute Eqs. (29), (30) and (31) into Eq. (27) we obtain

$$\rho = \cdots e^{i\alpha_{13}\lambda_3} e^{i\alpha_{14}\lambda_8} e^{i\alpha_{15}\lambda_{15}} \\ \times \left(\frac{1}{4} I_4 + \frac{1}{2}(2w^2 - 1)x^2 y^2 \cdot \lambda_3 + \frac{1}{2\sqrt{3}}(3x^2 - 2)y^2 \cdot \lambda_8 + \frac{1}{2\sqrt{6}}(4y^2 - 3) \cdot \lambda_{15} \right) \\ \times e^{-i\alpha_{15}\lambda_{15}} e^{-i\alpha_{14}\lambda_8} e^{-i\alpha_{13}\lambda_3} \cdots. \tag{32}$$

In Eq. (32) the first line shows only the last three factors of U as given by Eq. (30) and, similarly, the last line shows only the first three factors of $U^\dagger$ as given by Eq. (31). We note that all three lines of Eq. (32) involve only I_4, λ_3, λ_8, λ_{15} which all commute with each other. This allows for a further simplification since the last three terms of U and the first three terms $U^\dagger$ can now cancel out. Therefore, a two-qubit density matrix can always be written in the form

$$\rho = e^{i\alpha_1\lambda_3} e^{i\alpha_2\lambda_2} e^{i\alpha_3\lambda_3} e^{i\alpha_4\lambda_5} e^{i\alpha_5\lambda_3} e^{i\alpha_6\lambda_{10}} e^{i\alpha_7\lambda_3} e^{i\alpha_8\lambda_2} e^{i\alpha_9\lambda_3} e^{i\alpha_{10}\lambda_5} e^{i\alpha_{11}\lambda_3} e^{i\alpha_{12}\lambda_2} \\ \times \left(\frac{1}{4} I_4 + \frac{1}{2}(2w^2 - 1)x^2 y^2 \cdot \lambda_3 + \frac{1}{2\sqrt{3}}(3x^2 - 2)y^2 \cdot \lambda_8 + \frac{1}{2\sqrt{6}}(4y^2 - 3) \cdot \lambda_{15} \right) \\ \times e^{-i\alpha_{12}\lambda_2} e^{-i\alpha_{11}\lambda_3} e^{-i\alpha_{10}\lambda_5} e^{-i\alpha_9\lambda_3} e^{-i\alpha_8\lambda_2} e^{-i\alpha_7\lambda_3} e^{-i\alpha_6\lambda_{10}} e^{-i\alpha_5\lambda_3} e^{-i\alpha_4\lambda_5} \\ \times e^{-i\alpha_3\lambda_3} e^{-i\alpha_2\lambda_2} e^{-i\alpha_1\lambda_3}. \tag{33}$$

The last simplification effectively reduces the number of required parameters for U by three, which is exactly the number of parameters that are needed for ρ_d.

This scheme, that was indicated above for $N = 4$, i.e. for any mixed two-qubit state, can be generalized also for mixed states of any higher dimension. It requires $N^2 - 1$ parameters in order to parametrize an $N \times N$ density matrix, well in line with the parametrization of $SU(N)$. Note that the reduced parametrization of $SU(N)$ from above is known in the literature as the subgroup $SU(N)/\mathbb{Z}_N$ [10,14].

In the Feynman program, the direct (over-)parametrization [cf. Eq. (26)] of $N \times N$ density matrices can be utilized by the command Feynman_parametrize("mixed state", "direct", N, $[p_1, \dots, p_{2N^2}]$), while the generalized Euler representation (28) is implemented by Feynman_parametrize("mixed state", "Euler angles", N, $[\theta_1, \dots, \theta_{N-1}, \alpha_1, \dots, \alpha_{N^2-1}]$). In addition, the above mentioned subgroup $SU(N)/\mathbb{Z}_N$ is implemented as Feynman_parametrize("SU over Z", N, $[\alpha_1, \dots, \alpha_{N(N-1)}]$).

2.5.3. *Parametrization of separable states*

Using Eqs. (23) and (24) together with the 'standard' parametrization of arbitrary state vectors [cf. Eq. (25)] yields the 'standard' parametrizations of pure product states and mixed separable states. In the Feynman program, the pure product state of a k-partite quantum systems can be accessed by Feynman_parametrize("product state", "standard", $[d_A, d_B, \dots, d_Z]$, $[p_1, \dots, p_n]$) where the d_k define the dimensions of the individual subsystems $A, B, \dots, Z$. For this implementation, a list of $n = \sum_{i=1}^{k} 2d_i - 2$ parameters is required. For a k-qubit system where $d_i = 2$ for all subsystems, this means that $2k$ parameters are expected. Similarly, a k-partite separable mixed state can be parametrized with the command Feynman_parametrize("separable state", "standard", $[d_A, d_B, \dots, d_Z]$,$[p_1, \dots, p_n]$). For this parametrization, the parameter counting is slightly more complicated and has been discussed in detail for the case of a two-qubit system in Section 2.4. However, for a given list of subsystem dimensions d_i, a very simple way to determine the number of required parameters is the command nops(Feynman_parameters("separable state", "standard", $[d_A, d_B, \dots, d_Z]$)) which will be explained in the following section.

3. Implementation and further extensions to the Feynman program

3.1. *Overview*

The Feynman program has been developed during the last few years as a toolbox for the simulation of n-qubit quantum systems (quantum registers) within the framework of Maple. Our approach does not emphasize any particular experimental realization scheme, such as ion traps, photon based implementations or others. Instead our aim is to implement the general concepts that are common to all relevant implementations within a flexible and extendible framework. Moreover, by using a computer algebra system as the underlying framework, the Feynman package supports both symbolic *and/or* numerical computations which further extends the range of possible applications. With the exception of some Mathematica projects [16], the support for symbolic computations is in contrast to the majority of other quantum simulators, such as Qlib [17] or Qubit4Matlab [18] which are based on Matlab and also most of the C++ programs and libraries where emphasis is placed on numerical studies. A rather exhaustive list of quantum computer simulators and similar quantum information software tools can be found in [19].

So far, the Feynman program has been developed and published in several steps. In the first version [5], we have restricted ourselves essentially to the introduction of the basic data structures for the simulation of n-qubit quantum systems referring, in particular, to the three data types qbit(), qregister() and qoperator(). These structures (auxiliary procedures) are used for keeping together related information and to facilitate the data handling *with* and *within* the program. Moreover, a number of commands were introduced in that initial program version that enable the user to act upon these data structures, e.g., by applying some pre-defined or user-defined quantum gates or for performing standard operations, such as the tensor product of two or more quantum registers or the computation of reduced density operators and so on. In addition, some simple visualization tools were provided in order to 'display' the state of quantum registers by using probability plots or the Bloch vector representation of single qubits.

In a second program version [6], later, we focused mainly on the entanglement properties of quantum states by implementing several separability criteria and measures of entanglement. Most of these measures can be accessed through the command Feynman_measures(), together with a number of other frequently applied quantities like the distance between two quantum states or their entropy.

More recently [7], we have implemented support for quantum operations, i.e. completely positive and trace-preserving maps (CPT maps) that can be used to describe general (unitary or nonunitary) state changes as it is needed to simulate decoherence effects. Additionally, we provided several commands that make use of the duality between quantum states and quantum operations (also known as the Jamiołkowski isomorphism). Using this method, for example, several (distance) measures and separability criteria for quantum states can be transferred to quantum operations.

In the present program extension, we now concentrate on the parametrization of some of the most frequently used objects within the context of quantum computation and quantum information, especially pure and mixed quantum states, pure product states and separable (mixed) states as well as hermitian and unitary matrices.

Technically, the FEYNMAN program is organized as a *module* (package) for MAPLE, containing a hierarchy of currently about 90 (sub)procedures at different hierarchy levels. Apart from various subprocedures, which remain *hidden* to the user, the *main* commands can be used for interactive work and as language elements in order to build-up new commands at some higher level of the hierarchy. Given the large number of functions that are already implemented in the FEYNMAN package and the simple access to MAPLE's built-in mathematical functions, it is particularly easy for the user to extend and adapt the program on a higher level.

3.2. Implementation of the parametrizations

At the higher levels of the FEYNMAN program, the qregister() and qoperator() data structures are typically used to describe all n-qubit quantum states (pure or mixed) and the unitary operations or quantum gates that act on them. However, for the parametrization of quantum states, hermitian and unitary matrices, we decided to implement these new functions at a lower level of the hierarchy. On the one hand, the circumvention of the qregister() and qoperator() data structures increases efficiency for those cases where the parametrization routines are called repeatedly. On the other hand, it provides a more flexible approach as it allows the user to generate, for example, 6×6 density matrices which do not correspond to a multi-qubit system but to a qubit–qutrit (i.e. 2×3) system.

In order to make the parametrizations accessible at the user level, only two new *main* commands have been added that are directly related to this functionality, namely Feynman_parameters() and Feynman_parametrize(). The first mentioned returns a list of parameter ranges and, hence, implicitly the number of parameters for a given object to be parametrized. However, we should note that in most of the currently implemented (often trigonometric) parametrizations, the parameters are actually periodic so that the ranges can be chosen arbitrarily. This freedom in the parameter ranges can be helpful, for example, when a parametrization is used in connection with an unconstrained optimization procedure. On the other hand, in order to support possible future extensions or user modifications, explicit parameter ranges might be necessary so that this feature has been implemented from the beginning.

After determining the required number of parameters via Feynman_parameters(), the second new main command Feynman_parametrize() is used to actually generate the selected object. Both commands expect a keystring as the first argument to specify the object to be parametrized, followed by another keystring to set the parametrization method, and finally a dimension parameter. The Feynman_parametrize() command expects, of course, also a list of the actual (typically numerical) parameters to generate the selected object. Alternatively, instead of the parameter list, the keyword `"random"` can be used return a random object where the parameter values are chosen from a uniform distribution. Note however that this does not necessarily imply that the generated objects also cover the according set uniformly. In order to further support the work with random objects we have also added the alternative command Feynman_random() which can be used to generate uniformly distributed state vectors, density matrices and unitary matrices. These implementations are independent of the parametrizations in Feynman_parametrize(). A summary of the currently implemented syntax options of Feynman_parameters() and Feynman_parametrize() is given in Table 1.

3.3. Further changes to the program

Apart from adding the new main commands Feynman_parameters() and Feynman_parametrize() for generating quantum states and matrices, we have taken the opportunity to reorganize some of the main commands of the FEYNMAN package. The primary aim of this reorganization is the reduction of the total number of commands by collecting similar functionality within one larger main command. One example for this is the new main command Feynman_evaluate() that covers many basic operations like inner and outer products of matrices or vectors, Kronecker products of different objects, (partial) traces and other functions that were previously distributed over several main commands. Another example is the new main command Feynman_type() which combines virtually all checks for predefined properties of matrices, quantum registers and quantum operations within one command. Apart from reducing the number of main commands the reorganization will also make it easier to implement new functions without the need to introduce a new main command.

Due to the discussed reorganization of the main commands we provide a brief summary of *all* main commands of the FEYNMAN program in Table 2 while, for syntax options and all further details, we refer the reader to the manual `Feynman-commands.pdf` which is distributed together with the program.

In addition to the command reorganization, further changes and improvements have been made to the code of many main and auxiliary commands in the FEYNMAN program. Most of these changes are motivated by the fact that the newly introduced parametrizations are typically used in numerical investigations. As MAPLE is an interpreted language, numerical computations and frequent function calling are often less efficient than in compiled code. To take this into account, the FEYNMAN code has been optimized in many places in order to make use of hardware floating evaluation, for example, by calling the compiled NAG routines that come with MAPLE. In this way, the original approach of providing an easily extendible and essentially platform independent program is not abandoned. However, the significant speedup that can be achieved with this method is limited to those cases where MAPLE's computational precision is explicitly set to machine precision using the command `Digits := trunc(evalhf(Digits));` which is now the recommended default setting when loading the FEYNMAN package. This change also affects the global variable `Feynman_precision` that has been introduced in the previous version to define an acceptable error threshold, e.g., when checking the equality of two floating point numbers. By default, this

Table 1
Syntax options of the commands Feynman_parameters() and Feynman_parametrize(). For Feynman_parameters(), the type of the object to be parametrized, the method as well as the dimension must be specified. A list of parameter ranges is returned. In Feynman_parametrize(), a list of parameters must be supplied in addition to the type, method and the dimension of the selected object. Alternatively, the keyword "random" can be used to generate the given object but for a random set of parameters

Object type	Method	Dimension	Description
`"SU"`	`"standard"` `"Euler angles"`	N	$N \times N$ special unitary matrix
`"U"`	`"standard"` `"Euler angles"` `"Jarlskog"`	N	$N \times N$ unitary matrix
`"SU over Z"`	`"Euler angles"`	N	$N \times N$ unitary matrix from the subgroup $SU(N)/\mathbb{Z}_N$
`"hermitian"`	`"direct"` `"standard"`	N	$N \times N$ hermitian matrix
`"CPD"`	`"standard"`	N	N-dimensional normalized classical probability distribution $[p1, \ldots, p_N]$
`"pure state"`	`"standard"`	N	N-dimensional normalized pure state
`"product state"`	`"standard"`	$[\mathrm{d}_A, \mathrm{d}_B, \ldots, \mathrm{d}_Z]$	$d_A \cdot d_B \cdot \ldots \cdot d_Z$ dimensional separable pure state $\lvert\psi\rangle = \lvert\psi_A\rangle \otimes \lvert\psi_B\rangle \otimes \cdots \otimes \lvert\psi_Z\rangle$
`"mixed state"`	`"direct"` `"Euler angles"`	N	$N \times N$ normalized density matrix
`"separable state"`	`"standard"`	$[\mathrm{d}_A, \mathrm{d}_B, \ldots, \mathrm{d}_Z]$	$d_A \cdot d_B \cdot \ldots \cdot d_Z$ dimensional separable mixed state $\rho = \sum p_i \rho_A^i \otimes \rho_B^i \otimes \cdots \otimes \rho_Z^i$

Table 2
Main commands of the Feynman program as accessible by the user. New and significantly modified commands are marked with an $*$. A detailed description of all commands is provided in the manual `Feynman-commands.pdf` which is distributed together with the code

Command	Short explanation
Feynman_adjoint()	Computes the adjoint of a given matrix, vector and dual (='adjoint') quantum operations.
Feynman_apply()	Applies qoperator() and qoperation() structures to a qregister().
Feynman_decompose()	Computes the spectral decomposition of a matrix and several other matrix decompositions as well as the Schmidt decomposition of a pure state.
$*$ Feynman_define()	Returns predefined objects like the generalized Gell-Mann matrices or the generators of the Pauli group.
Feynman_eigenvectors()	Computes eigenvectors and eigenvalues with additional support for some symbolic cases that are not originally supported by Maple.
$*$ Feynman_evaluate()	Carries out a large number of basic operations like inner and outer products, Kronecker products, (partial) traces, commutators, vector and matrix norms, etc.
Feynman_equal()	Checks the equality of two scalars, vectors or matrices within some numerical noise limits.
Feynman_measures()	Computes a variety of different measures (entanglement, fidelity, etc.) New options: "singlet fraction", "Fubiny-Study distance", "Bell violation", new implementation of the average (gate) fidelity of a quantum operation [15].
Feynman_normalize()	Normalizes a given vector or qregister().
$*$ Feynman_parameters()	Returns a list of parameter ranges for the implemented parametrizations.
$*$ Feynman_parametrize()	Computes different objects (state vectors, density matrices, unitary matrices, etc.) from a set of given parameters or randomly.
Feynman_plot()	Returns plots of the Bloch vector representation of a qubit, probability plots of a qregister() and the process matrix of a qoperation().
Feynman_print()	Returns a 'pretty print' of a qregister() (and other objects) using Dirac notation in the computational basis.
Feynman_qoperation_representation()	Computes several representations of a quantum operation, such as its dual state, the superoperator matrix and others.
Feynman_quantum_operation()	Provides the explicit list of Kraus operators for predefined quantum operations like the 'amplitude damping' and 'depolarization' channel and others.
Feynman_quantum_operator()	Provides the explicit matrix representation for predefined quantum gates like the 'controlled-not' and the 'Hadamard' gate and others.
$*$ Feynman_random()	Computes random state vectors, density matrices and unitary matrices. The generated objects are distributed uniformly with respect to the Fubiny-Study metric, Hilbert–Schmidt norm and the Haar measure, respectively.
Feynman_set_qregister()	Provides a large collection of predefined quantum states that appear frequently in the literature.
Feynman_transform()	Carries out basis changes (e.g., qubit permutations) of qregister() and qoperator() structures and performs other notation changes.
Feynman_transpose()	Applies the (partial) transposition operation or a so-called 'general transposition' to a given matrix.
$*$ Feynman_type()	Checks for a large number of properties of matrices, qregister() and qoperation() structures.

variable is now set to `Feynman_precision := evalf(10^(-trunc(0.8*Digits)));` which evaluates to 10^{-11} for `Digits=14` (on 32-bit Windows) and 10^{-12} for `Digits = 15` (on 32-bit Linux). Of course, these default settings can be changed at any time by the user.

Apart from these code optimizations no further efforts have been made to take advantage of Maple's new multiprocessor capabilities that have been introduced in version 11. Our preliminary tests show that the current implementation of the SMP kernel is often significantly slower than the optimized and more mature single processor kernel. Although this can be expected to change in future versions of Maple we currently do not recommend to enable the SMP support (which is disabled by default in version 11).

3.4. Program distribution

As with the previous versions, the FEYNMAN program is distributed as a compressed archive file that contains the FEYNMAN library files, the source code as well as the manual file `Feynman-commands.pdf`. In addition, this archive includes several example worksheets (including those from previous versions and the examples in the following section) and a `Read.me` file to briefly explain the installation of the library. After the proper installation, the program can be loaded like any other module of MAPLE by using the command `with(Feynman)`.

4. Interactive work using the FEYNMAN procedures: Examples

To illustrate the interactive use of the FEYNMAN procedures, we shall discuss four short examples below. These examples demonstrate how the parametrization of matrices and quantum states can be applied in order to derive, for example, a parametrized expression for the fidelity of two arbitrary pure states or for generating randomly distributed states. For the sake of convenience, these examples are included also as MAPLE worksheets in the distribution archive. Of course, these procedures can be utilized also as building blocks to define commands of higher complexity as it might be needed, for instance, for a so-called convex-roof optimization where a given function ('measure'), which is defined for pure states, is extended to mixed states by minimizing or maximizing that function over all possible pure-state decompositions of a given density matrix.

In the following, we assume that the FEYNMAN program has been 'loaded' via the command `with(Feynman)` at the beginning of the session and that the MAPLE prompt is available for interactive work. Moreover, since several procedures result in a rather large MAPLE output, a colon (instead of a semicolon) is used to terminate some of the input lines in order to suppress the corresponding printout to screen.

4.1. Fidelity of a symbolically parametrized pure state

In our first example, let us show how the fidelity (i.e. the overlap) of two qubits, A and B, changes as a function of the qubit state parameters of the first qubit. For this, we can utilize the symbolic parametrization for the pure state of a single qubit

$$|\psi\rangle = \cos(p_1)|0\rangle + e^{ip_2}\sin(p_1) \tag{34}$$

which is based on the two parameters, p_1 and p_2 (up to some global phase that is irrelevant here). As the parameters correspond to angles we can assume them to be nonnegative here. In the FEYNMAN program, this parametrization is obtained if we enter

```
> assume(p::list(nonnegative));
> psi := Feynman_parametrize("pure state","standard",2,[seq(p[i], i=1..2)]);

                [       cos(p~[1])        ]
         psi := [                         ]
                [exp(p~[2] I) sin(p~[1])]
```

Since the Feynman_parametrize() command returns for a pure state only a (state) vector instead of a complete qregister() structure, we still have to define our first single-qubit qregister(), A, for later use. At the same time, we can generate our second quantum register, B, for example in the (fixed) state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. If needed, the actual state vector of B, say $|\phi\rangle$, can then be extracted as the third operand of B.

```
> A   := qregister(id, 1, psi):
> B   := Feynman_set_qregister("+"):
  phi := op(3, B);
                [ 1/2]
                [2   ]
                [----]
                [ 2  ]
         phi := [    ]
                [ 1/2]
                [2   ]
                [----]
                [ 2  ]
```

After that, we have available the bare state vectors $|\psi\rangle$ and $|\phi\rangle$ as well the corresponding qregisters A and B. To analyze the 'distance' between the states, we can first consider their (squared) overlap $|\langle\psi|\phi\rangle|^2$,

```
> abs(Feynman_adjoint(psi).phi)^2;
                1/2     _ _ _ _ _          1/2 _ _ _ _ _ _ _ _ _ _ _ _ _ _   2
         | 1/2 2    cos((p~[1])) + 1/2 2    (exp(p~[2] I) sin(p~[1])) |
```

which is known in quantum information theory also as the fidelity of the two states

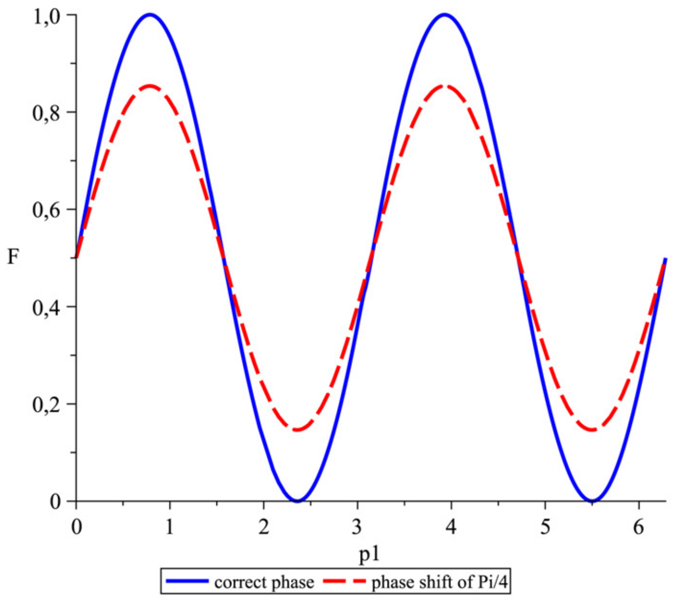

Fig. 2. Fidelity of the state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ with regard to a general single qubit state that is parametrized as $|\psi\rangle = \cos(p_1)|0\rangle + e^{ip_2}\sin(p_1)$. The curves show the value of the fidelity as a function of p_1 for a correct phase $p_2 = 0$ (solid line) and non-optimal phase of $p_2 = \pi/4$.

```
> F := Feynman_measures("fidelity", A, B);
                        1/2      _ _ _ _          1/2 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
          F := | 1/2 2    cos((p~[1])) + 1/2 2    (exp(p~[2] I) sin(p~[1])) |^2
```

To explore now how the relative phase p_2 of the state (34) affects its fidelity with regard to $|\phi\rangle$, we may plot $F(p_1)$ for two fixed values, say, $p_2 = 0$ and $p_2 = \pi/4$. In MAPLE, this is easily achieved by

```
> plot([eval(F, p[2]=0), eval(F, p[2]=Pi/4)], p[1]=0..2*Pi, labels=["p1", "F"],
  thickness=[3,3], linestyle=[solid, dash],
  legend=["correct phase", "phase shift of Pi/4"]);
```

and gives rise to Fig. 2. From this figure, we see immediately the (negative) influence of a phase mismatch (dashed line). While for the correct phase, i.e. $p_2 = 0$, the fidelity reaches periodically the maximum value 1, a perfect overlap cannot be obtained for the phase $p_2 = \pi/4$. Despite its simplicity, this example therefore nicely illustrates the effect of some unwanted interaction with the environment which may influence the relative phase information and, hence, degrade the fidelity, e.g., in a quantum memory.

4.2. Random state generation

In our next example, we visualize and analyze randomly generated density matrices with respect to their distribution over the state space. For this, we make use of the command Feynman_random() that provides pure and mixed states by applying Toth's method [18]. This method distributes the states uniformly over the space of all corresponding (pure or mixed) quantum states. We can convince ourselves of this uniform distribution in the case of single-qubit states by generating a large number of these random states and by displaying them by means of a three-dimensional Bloch sphere.

While, for pure states, the Bloch vector has always length 'one' and thus points somewhere *on* the sphere, these vectors populate the interior of the Bloch sphere in the case of mixed states. For a single-qubit state ρ, of course, the three (real) components of the Bloch vector v are obtained as the expectation values of its product with the Pauli sigma matrices

$$v_i = \mathrm{Tr}(\rho\sigma_i). \tag{35}$$

Therefore, let us start this example by defining the Pauli matrices

```
> X := Matrix(Feynman_quantum_operator("X"), datatype=complex[8]):
  Y := Matrix(Feynman_quantum_operator("Y"), datatype=complex[8]):
  Z := Matrix(Feynman_quantum_operator("Z"), datatype=complex[8]):
```

and where the second argument, `datatype=complex[8]` allows MAPLE to use the more efficient hardware floating point routines for the evaluation of the following commands. Having the Pauli matrices, we can generate a list of the (x, y, z) components of the Bloch vectors for pure states by

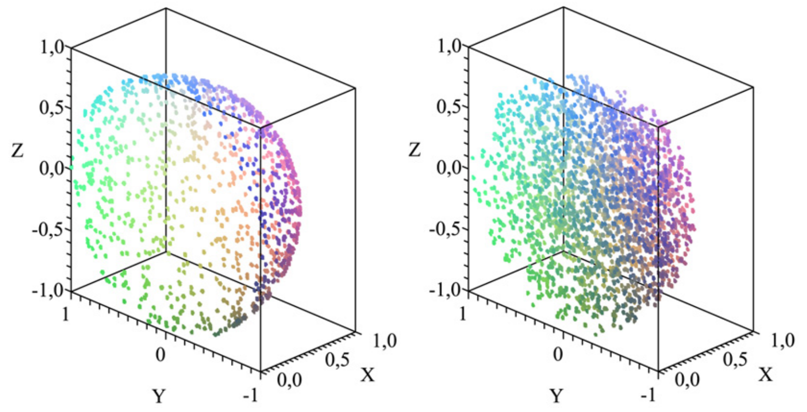

Fig. 3. Bloch sphere visualization of the random pure single qubit states generated by Feynman_random("pure state", 2) (left) and mixed ones from Feynman_random("mixed state", 1) (right). Only one hemisphere is shown to verify that, for pure states, the sphere is empty while mixed states occupy the complete Bloch ball.

```
> with(LinearAlgebra):
> data_pure := table:
  for i to 4000 do
     temp := Feynman_transform("ket to rho", Feynman_random("pure state", 2));
     data_pure[i] := map(Re, [Trace(X.temp), Trace(Y.temp), Trace(Z.temp)]);
  end do:
  data_pure := convert(data_pure, list):
```

and similarly for a list of mixed states

```
> data_mixed := table:
  for i to 7000 do
     temp := Feynman_random_rho(1);
     data_mixed[i] := map(Re, [Trace(X.temp), Trace(Y.temp), Trace(Z.temp)]);
  end do:
  data_mixed := convert(data_mixed, list):
```

Owing to MAPLE's graphic interface, we can easily plot and even rotate these data sets *interactively* in order to verify that the pure states are *uniformly* distributed on the unit sphere, while the mixed states fill its interior.

```
> plots[pointplot3d](data_pure, labels=["X", "Y", "Z"], view=[0..1, -1..1, -1..1],
  axes=boxed, scaling=constrained, font=[TIMES,ROMAN,12], symbol=solidsphere,
  symbolsize=8, orientation=[-135, 65]);

> plots[pointplot3d](data_mixed, labels=["X", "Y", "Z"], view=[0..1, -1..1, -1..1],
  axes=boxed, scaling=constrained, font=[TIMES,ROMAN,12], symbol=solidsphere,
  symbolsize=8, orientation=[-135, 65]);
```

The resulting pictures are shown in Fig. 3.

In order to analyze the distribution of the random (quantum) states from the procedure Feynman_random() also *quantitatively* (but now for arbitrary dimension), we may consider the distribution of their pairwise distance. For density matrices, the natural distance measure is given by the Hilbert–Schmidt distance

$$D_{\text{HS}}(\rho,\sigma) = \|\rho-\sigma\|_{\text{HS}} = \sqrt{\text{Tr}[(\rho-\sigma)^\dagger(\rho-\sigma)]} \tag{36}$$

and can be utilized to examine the statistical distribution of the distance between random two-qubit states

```
> HS_data := [seq(Feynman_evaluate("HS norm",
  Feynman_random("mixed state",2) - Feynman_random("mixed state",2)),
  i=1..5000)]:
> Statistics[Histogram](HS_data, bincount=40, labels=["HS distance","count"]);
```

The distribution for these 5000 randomly chosen two-qubit states is shown in Fig. 4. If other methods are applied for generating random states (e.g., by using the keyword "random" with the command Feynman_parametrize()), then a less balanced distribution is typically obtained.

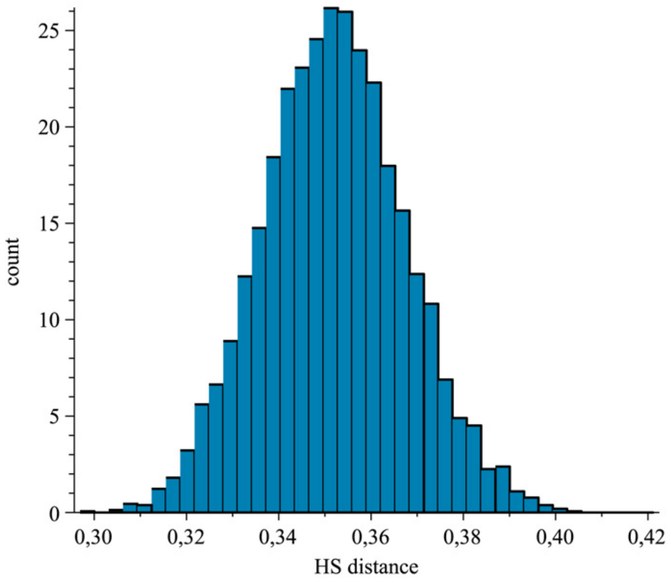

Fig. 4. Distribution of the Hilbert–Schmidt distance between random density matrices generated with Feynman_random().

4.3. Nonlocality versus entanglement

The previous example just showed a way how we can generate random (pure and mixed) states within the FEYNMAN program. Let us next apply these states in order to explore heuristically the relation between the *violation* of the CHSH inequality (Clauser–Horne–Shimony–Holt inequality [20]), as a sign of the nonlocality, and the entanglement of a quantum state, which was discussed in more detail in Ref. [21].

For a two-qubit system, the entanglement is most easily measured by Wootter's concurrence which is given by [22]

$$\mathcal{C}(\rho) = \max\bigl(0, \sqrt{e_1} - \sqrt{e_2} - \sqrt{e_3} - \sqrt{e_4}\bigr), \tag{37}$$

where $\sqrt{e_i}$ are the square roots of the eigenvalues of the matrix

$$\rho\bigl(\sigma_y^{(1)} \otimes \sigma_y^{(2)}\bigr)\rho^*\bigl(\sigma_y^{(1)} \otimes \sigma_y^{(2)}\bigr),$$

taken in descending order. In this matrix, ρ^* denotes the complex conjugate of ρ and $\sigma_y^{(1,2)}$ are the usual Pauli σ_y matrices, acting on the first or second qubit, respectively.

The CHSH inequality as a variant of Bell's inequality [23] has been widely applied for studying the nonlocal properties of quantum systems. This inequality limits the correlations that may occur between the outcomes of two measurements (that are carried out at places sufficiently distant from each other), if one assumes that the measured observables must correspond to *local* statistical variables. A stronger correlation than the CHSH threshold for the outcome of two (distant) measurements is therefore a sign of nonlocality as it has been analyzed in numerous tests, e.g., the famous experiment by Aspect et al. [24].

For an arbitrary two-qubit state ρ, the Bell-CHSH inequality can be written as

$$\bigl|\mathrm{Tr}(\rho\mathcal{B}_{\mathrm{CHSH}})\bigr| \leqslant 2, \tag{38}$$

where $\mathcal{B}_{\mathrm{CHSH}}$ is the (so-called) Bell-CHSH operator given as

$$\mathcal{B}_{\mathrm{CHSH}} = \boldsymbol{a} \cdot \boldsymbol{\sigma}^{(1)} \otimes (\boldsymbol{b} + \boldsymbol{b}') \cdot \boldsymbol{\sigma}^{(2)} + \boldsymbol{a}' \cdot \boldsymbol{\sigma}^{(1)} \otimes (\boldsymbol{b} - \boldsymbol{b}') \cdot \boldsymbol{\sigma}^{(2)}, \tag{39}$$

and $\boldsymbol{a}, \boldsymbol{a}', \boldsymbol{b}, \boldsymbol{b}' \in \mathbb{R}^3$ being unit vectors. In this notation of the CHSH operator, moreover, the $\boldsymbol{\sigma}^{(1,2)} = (\sigma_x^{(1,2)}, \sigma_y^{(1,2)}, \sigma_z^{(1,2)})$ are vectors of the Pauli spin matrices that act on the first or second qubit, respectively. If, for a given density matrix ρ, there exist vectors $\boldsymbol{a}$, $\boldsymbol{a}'$ and $\boldsymbol{b}$, $\boldsymbol{b}'$ so that the criterion (38) is not fulfilled, then the state is said to *violate* the Bell-CHSH inequality, i.e. it features nonlocal correlations that are incompatible with a local realistic model. In order to quantify the (maximal) violation, we first rewrite the two-qubit density matrix in terms of the Pauli basis

$$R_{ij} = \mathrm{Tr}\bigl(\rho\sigma_i^{(1)} \otimes \sigma_j^{(2)}\bigr). \tag{40}$$

Using this notation, the 'Bell violation' measure in the FEYNMAN program is defined as

$$B(\rho) = \sqrt{\max\bigl(0, s_1^2 + s_2^2 - 1\bigr)}, \tag{41}$$

and where $s_{1,2}$ are the two largest singular values of the matrix R_{ij} [21,25]. The definition (41) has the useful property of being equal to the concurrence measure (37) for *pure* states as we will see also below.

In order to visualize the relation between the Bell violation, $B(\rho)$, and the entanglement, $\mathcal{C}(\rho)$, of general two-qubit (pure and mixed) states, we first generate a set of 5000 random states and calculate both measures for each of them.

```
> data1 := table():
  data2 := table():
  for i from 1 to 5000 do
     temp1 := Feynman_set_qregister("random", 2);
     temp2 := 'qregister'(id, 2, Feynman_random("pure state", 4));
     data1[i] := [Feynman_measures("concurrence", temp1),
                  Feynman_measures("Bell violation", temp1)];
     data2[i] := [Feynman_measures("concurrence", temp2),
                  Feynman_measures("Bell violation", temp2)];
  end do:
  data1 := convert(data1, list):
  data2 := convert(data2, list):
```

Here, we recall that we have to define the qregister() structure `temp2` manually because the Feynman_random() command generates only a state vector. From this data we can create a plot structure in which later the grey data points will correspond to random two-qubit *mixed* states while the blue data points (which will actually form the diagonal blue line in the plot) correspond to random two-qubit *pure* states.

```
> scatter_plot := plot([data1, data2], color=[grey,blue],
  labels=["Concurrence","Bell-CHSH violation"], style=point, symbolsize=[5,7]):
```

In addition, we can also create a parametric plot for the one-parameter family of the Werner states [26] and the maximally entangled mixed states (MEMS) [27,28] that are both already *predefined* in the Feynman program

```
> k := 50:  # defines the number of data points for the curves
  Werner_plot := plot([seq(
  [Feynman_measures("concurrence", Feynman_set_qregister("Werner", i/k)),
   Feynman_measures("Bell violation", Feynman_set_qregister("Werner", i/k))],
  i=0..k)], color=red, linestyle=dash, thickness=3):
  MEMS_plot := plot([seq(
  [Feynman_measures("concurrence", Feynman_set_qregister("MEMS", i/k)),
   Feynman_measures("Bell violation", Feynman_set_qregister("MEMS", i/k))],
  i=0..k)], color=black, thickness=2):
> plots[display]([scatter_plot, MEMS_plot, Werner_plot], axes=boxed);
```

The plot finally obtained from the last line combines the previously generated scatter plot and the just computed curves. The plot is shown in Fig. 5 (left) and is very similar to the one shown by Verstraete and Wolf [21], except for a slightly different normalization of the Bell violation measure. While the Bell violation and the entanglement is apparently the same for the pure states (blue diagonal), for mixed states entanglement does not necessarily imply the violation of the Bell inequality. This is seen especially for some of the mixed states that have a concurrence $\mathcal{C}(\rho) \geqslant 0.5$ but show only little or no Bell violation at all. The most famous examples of entangled states that do not violate the CHSH-inequalities are probably the Werner states (dashed line) and the MEMS which have the maximum amount of entanglement for a given degree of mixedness (in terms of their entropy). As a third dimension of the plot, therefore, it might be instructive to add also the linear entropy $S_l(\rho) = \frac{4}{3}(1 - \mathrm{Tr}(\rho^2))$, which can be done very similarly as above (see Fig. 5, right). In particular, by using Maple's ability to rotate such three-dimensional plots interactively, one has a simple and very intuitive way to explore the interplay between the mixedness of a state and its entanglement as well as nonlocality. Moreover, such plots help to understand the role of special families of quantum states, such as the Werner states or MEMS.

4.4. Minimizing the relative entropy for two-qubit states

In our final example, we demonstrate the computation of the so-called 'relative entropy of entanglement' (REE) for a two-qubit system. This entanglement measure has been one of the first concepts in the literature for the quantification of entanglement and can be defined intuitively as the minimal distance of a given state to the set of separable states [12],

$$S_{\mathrm{REE}}(\rho) = \min_{\sigma_{\mathrm{sep}}} S(\rho \| \sigma_{\mathrm{sep}}) = \mathrm{Tr}\,\rho(\log\rho - \log\sigma_{\mathrm{sep}}), \tag{42}$$

where $S(\rho\|\sigma_{\mathrm{sep}})$ represents the quantum relative entropy which serves as a quasi-distance measure and where the minimization has to be performed over all separable two-qubit states σ_{sep}. But despite the simple interpretation of the relative entropy of entanglement, it can usually be determined only numerically. This minimization over the space of all separable states is demonstrated in the following.

Let us first define two (two-qubit) states, an entangled one and a separable state. For the sake of convenience, we choose two Werner states but there is nothing specific about this choice and we could take also any other pair of an entangled and a separable state.

```
> separable_state := Feynman_set_qregister("Werner", 0.2):
> entangled_state := Feynman_set_qregister("Werner", 1.0):
```

We may convince ourselves of their respective separability properties by using, for example, the Feynman_type() command in connection with the `separable` keyword and the well-known 'positive partial-transpose' (PPT) criterion

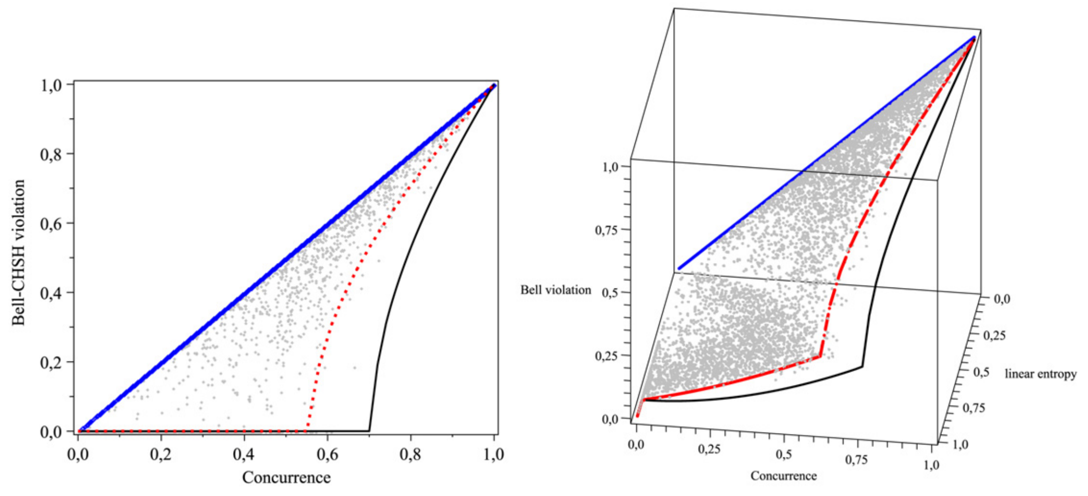

Fig. 5. (Left) The maximal violation of the Bell-CHSH inequality as a function of the concurrence. For pure states (diagonal solid line), both measures coincide and correspond to the upper bound for the Bell violation. The maximally entangled mixed states (black solid line) correspond to the lower bound. The well-known Werner states (dotted line) lie in between. (Right) Same as on the left but with the linear entropy as additional dimension of the plot. (For interpretation of the references to color, the reader is referred to the web version of this article.)

```
> Feynman_type(separable_state, "separable","PPT", [[1],[2]]);
  Feynman_type(entangled_state, "separable, "PPT", [[1],[2]]);

          true
          false
```

In order to perform now the minimization over the whole set of separable two-qubit states, we may first ask the program about the number of the required parameters (see also Section 2.5.3) and then write a small objective function that uses these parameters to compute the relative entropy between the (globally) given state, i.e. the entangled or separable Werner state from above, and the parametrized separable state.

```
> nops(Feynman_parameters("separable state", "standard", [2,2]));

          79

> objective := proc(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13,
  x14, x15, x16, x17, x18, x19, x20, x21, x22, x23, x24, x25, x26, x27, x28,
  x29, x30, x31, x32, x33, x34, x35, x36, x37, x38, x39, x40, x41, x42, x43,
  x44, x45, x46, x47, x48, x49, x50, x51, x52, x53, x54, x55, x56, x57, x58,
  x59, x60, x61, x62, x63, x64, x65, x66, x67, x68, x69, x70, x71, x72, x73,
  x74, x75, x76, x77, x78, x79)::float[8];
   options hfloat;
   global given_state;
   Feynman_measures("relative entropy", given_state, 'qregister'(id, 2,
   Feynman_parametrize("separable state","standard", [2,2], [args])));
  end proc:
```

In the following, we shall exploit the fact that a local minimum of a convex function (such as the relative entropy) over a convex set (the separable states) is automatically a global minimum [12]. Therefore, it appears sufficient to employ a local minimization procedure, e.g., the derivative-free Nelder-Mead or simplex method which are implemented in MAPLE's built-in 'Optimization' package. Indeed, we could provide a random initial point for the minimization. As the required time depends also on this initial point, it can be useful to repeat the random generation several times as to avoid particularly high values (e.g., > 10) of the objective function.

```
> given_state := entangled_state:
> some_point  := seq(RandomTools[Generate](float(range=0..10,
  method=uniform)), i=1..79):
> objective(some_point);
        2.0225158437315
```

Here we have used MAPLE's built-in `RandomTools` package to generate 79 random floating point parameters within the range 0..10. The actual minimization is done using the 'Minimize' command where the desired method, accuracy and a limit on the number of objective function evaluations can be specified. The result is a list containing the lowest found objective function value and the vector of parameters corresponding to that solution.

```
> infolevel[Optimization] := 3;
> Optimization[Minimize](objective, initialpoint=[some_point],
  method=nonlinearsimplex, optimalitytolerance=1.0E-5,
  evaluationlimit=7000);

NLPSolve: calling NLP solver
SolveUnconstrainedNM: using method=nonlinearsimplex
SolveUnconstrainedNM: number of problem variables 79
SolveUnconstrainedNM: trying evalhf mode
SolveUnconstrainedNM: trying evalf mode
Warning, limiting number of function evaluations reached
E04CCA: number of function evaluations 5958

       [1.00014022910700007, Vector(1..79)]
```

The result $S_{\mathrm{REE}} \approx 1.0$ clearly indicates an entangled state, as we have expected from the separability test above. Note that the status reports indicate that the minimization stopped before the maximum number of function evaluations was reached so that the result should be correct within the specified accuracy. To ensure this, it is advisable to enable these status reports by typing `infolevel[Optimization] := 3;` before carrying out the optimization.

For the second state, the procedure is of course very similar except that S_{REE} should vanish because the state *is* separable and, obviously, the closest separable state is the state itself.

```
> given_state := separable_state:
> some_point  := seq(RandomTools[Generate](float(range=0..10,
  method=uniform)), i=1..79):
> objective(some_point);
        1.1044598870200

> Optimization[Minimize](objective, initialpoint=[some_point],
  method=nonlinearsimplex, optimalitytolerance=1.0E-5,
  evaluationlimit=7000);

NLPSolve: calling NLP solver
SolveUnconstrainedNM: using method=nonlinearsimplex
SolveUnconstrainedNM: number of problem variables 79
SolveUnconstrainedNM: trying evalhf mode
SolveUnconstrainedNM: trying evalf mode
E04CCA: number of function evaluations 6541

         [0.0000611719017000000002, Vector(1..79)]
```

Depending on the objective value of the initial point and the number of allowed objective function evaluations, convergence can be different. Still, a relative entropy close to zero has been found as expected.

5. Summary and outlook

We have presented an updated version of the Feynman program which has been developed during the last years as a versatile tool for the simulation and analysis of quantum registers but also to teach and to deal with typical tasks in quantum information theory. While, in the first version, the basic data structures for quantum registers and (unitary) gates were implemented, several program extensions then added support for noisy quantum channels and the analysis of separability and entanglement properties of quantum registers. In particular, the program can connect the two fundamental concepts of states and channels by using the Jamiołkowski isomorphism that enables the user to transfer concepts like distance measures or separability from states to channels. For the two data structures qregister() and qoperation(), a variety of different measures are provided that support also a quantitative analysis of many properties.

With the present revision, we further extend the program capabilities by providing convenient access to the parametrization of frequently used objects like pure and mixed quantum states, hermitian and unitary matrices and classical probability distributions.

This functionality is useful since many problems in the context of quantum information involve the search over all quantum gates (i.e. unitary operators), pure or separable states, etc., as shown in our last example. As an additional benefit, the parameterizations enable a very simple access to random objects which can be useful in heuristic studies. To further support such numerical studies, the program has also been optimized in various places to take advantage of hardware floating-point evaluation. Together, these changes considerably improve Feynman's capability to support numerical studies in the simulation of quantum registers.

For the future development of the package, there are different directions in which the program could be extended. Given the current capability to describe error models as quantum operations, an interesting direction for the further development of the program is certainly an improved support for quantum error correction techniques. This could include tools for the identification of noiseless subsystems as well as established qubit encodings from the literature and the ability to design and optimize new encodings that are adapted to a given noise type. Such an extension might be complemented by the implementation of some time propagation schemes which can facilitate, for example, the study of entanglement dynamics in small and medium-sized systems.

References

[1] C.H. Bennett, G. Brassard, C. Crépeau, R. Josza, A. Peres, W.K. Wootters, Phys. Rev. Lett. 70 (1993) 1895.
[2] A.K. Ekert, Phys. Rev. Lett. 67 (1991) 661.
[3] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge, 2000.
[4] P.W. Shor, SIAM J. Sci. Statist. Comput. 26 (1997) 1484.
[5] T. Radtke, S. Fritzsche, Comput. Phys. Comm. 173 (2005) 91.
[6] T. Radtke, S. Fritzsche, Comput. Phys. Comm. 175 (2006) 145.
[7] T. Radtke, S. Fritzsche, Comput. Phys. Comm. 176 (2007) 617.
[8] T. Tilma, E.C.G. Sudarshan, J. Phys. A 35 (2002) 10467.
[9] W. Greiner, B. Müller, Quantum Mechanics: Symmetries, Springer, Berlin, 1994.
[10] T. Tilma, M. Byrd, E.C.G. Sudarshan, J. Phys. A 35 (2002) 10445.
[11] C. Jarlskog, J. Math. Phys. 46 (2005) 103508;
C. Jarlskog, J. Math. Phys. 47 (2006) 013507.
[12] V. Vedral, M.B. Plenio, Phys. Rev. A 57 (1998) 1619.
[13] J. Grondalski, D.M. Etlinger, D.F.V. James, Phys. Lett. A 300 (2002) 573.
[14] I. Bengtsson, K. Zyczkowski, Geometry of Quantum states, Cambridge University Press, Cambridge, 2006.
[15] L.H. Pedersen, N.M. Møller, K. Mølmer, Phys. Lett. A 367 (2007) 47.
[16] QDENSITY by B. Julia-Diaz, F. Tabakin, http://library.wolfram.com/infocenter/MathSource/5715/, QUCALC by P. Dumais, http://library.wolfram.com/infocenter/MathSource/657/, QMATRIX by T. Felbinger, http://library.wolfram.com/infocenter/MathSource/1893/.
[17] S. Machnes, QLIB, arXiv: 0708.0478.
[18] G. Tóth, QUBIT4MATLAB, arXiv: 0709.0948.
[19] http://www.quantiki.org/wiki/index.php/List_of_QC_simulators.
[20] J.F. Clauser, M.A. Horne, A. Shimony, R.A. Holt, Phys. Rev. Lett. 23 (1969) 880.
[21] F. Verstraete, M.M. Wolf, Phys. Rev. Lett. 89 (2002) 170401.
[22] W.K. Wootters, Phys. Rev. Lett. 80 (1998) 2245.
[23] J.S. Bell, Physics (Long Island City, N.Y.) 1 (1964) 195.
[24] A. Aspect, J. Dalibard, G. Roger, Phys. Rev. Lett. 49 (1982) 1804.
[25] A. Miranowicz, Phys. Lett. A 327 (2004) 272.
[26] R.F. Werner, Phys. Rev. A 40 (1989) 4277.
[27] W.J. Munro, D.F.V. James, A.G. White, P.G. Kwiat, Phys. Rev. A 64 (2001) 030302.
[28] F. Verstraete, K. Audenaert, B. De Moor, Phys. Rev. A 64 (2001) 012316.