IOPscience iopscience.iop.org

Home Search Collections Journals About Contact us My IOPscience

Elementary gates for cartoon computation

This article has been downloaded from IOPscience. Please scroll down to see the full text article.

2007 J. Phys. A: Math. Theor. 40 F753

(http://iopscience.iop.org/1751-8121/40/31/F01)

View the table of contents for this issue, or go to the journal homepage for more

Download details:
IP Address: 171.66.16.144
The article was downloaded on 03/06/2010 at 06:07

Please note that terms and conditions apply.

IOP PUBLISHING JOURNAL OF PHYSICS A: MATHEMATICAL AND THEORETICAL

J. Phys. A: Math. Theor. **40** (2007) F753–F759 doi:10.1088/1751-8113/40/31/F01

FAST TRACK COMMUNICATION

Elementary gates for cartoon computation

Marek Czachor

Katedra Fizyki Teoretycznej i Informatyki Kwantowej, Politechnika Gdańska, 80-952 Gdańsk, Poland

Received 25 June 2007
Published 19 July 2007
Online at stacks.iop.org/JPhysA/40/F753

Abstract
The basic one-bit gates (X, Y, Z, Hadamard, phase, $\pi/8$) as well as the controlled CNOT and Toffoli gates are reformulated in the language of geometric-algebra quantum-like computation. Thus, all the quantum algorithms can be reformulated in purely geometric terms without any need of tensor products.

PACS numbers: 03.67.Lx, 03.65.Ud

1. Introduction

Cartoon computation [1] is a formalism for quantum-like computation based on geometric operations. One does not need tensor products to speak of entanglement, parallelism, superpositions and interferences. The analysis given in [1] illustrated the basic principle on the Deutsch–Jozsa problem [2]. An analogous construction was recently applied in [3] to the Simon problem [4]. Other oracle problems were mentioned in the context of geometric-algebra computation in [5]. In the present paper I will not work with oracles but concentrate on elementary one-, two- and three-bit gates. This step is essential for both concrete applications and analysis of complexity of algorithms.

I first begin with explaining the link between geometric algebra and binary coding. The idea is essentially the same as in [1], but there are certain technical differences associated with two subsidiary dimensions (here a $(n+2)$-dimensional Euclidean space is used for coding n-bit numbers). Once we know how to code and perform simple operations on bits, we can introduce gates. I start with the basic one-bit gates and then introduce multiply controlled NOTS [6]. Finally I show on a concrete example that the geometric product leads to the same type of 'compression' and parallelism as the tensor product framework of quantum computation. I end this paper with remarks on earlier approaches.

2. Binary parametrization

Take a $(n+2)$-dimensional Euclidean space with the basis $\{b_0, b_1, \ldots, b_n, b_{n+1}\}$. Geometric products of different basis vectors are called *blades*. One-blades (i.e. basis vectors) satisfy the

Clifford algebra

$$b_k b_l + b_l b_k = 2\delta_{kl}.$$

There are 2^{n+2} different blades. The basis vectors b_0 and b_{n+1} play in our formalism a privileged role. *Real* blades are those that do not involve b_0; those including b_0 are termed *imaginary*. We shall see below that this terminology is consistent with a *complex structure* needed for implementation of the one-bit elementary quantum-like gates.

We shall often need in the formulae the blade b_{n+1} so let us shorten the notation by $b_{n+1} = b$. The blades that do not involve b_{n+1} will be termed the *combs*, and are parametrized by n-bit strings according to the following convention [1]: $b_1 = c_{0;10\ldots0}, \ldots, b_n = c_{0;0\ldots1}, b_0b_1 = c_{1;10\ldots0}, \ldots, b_0b_n = c_{1;0\ldots01}, b_1b_2 = c_{0;110\ldots0}, \ldots, b_1b_2\ldots b_n = c_{0;1\ldots1}, b_0b_1b_2\ldots b_n = c_{1;1\ldots1}$. The combs beginning with '0;' or '1;' are real and imaginary, respectively. We supplement the combs by the (real) 0-blade $1 = c_{0;0\ldots0}$. The zeroth bit 'A;', separated by the semicolon from all the other bits $A_1 \ldots A_n$, is not needed for coding binary numbers but only for the complex structure. Therefore, one can skip it if one explicitly works with the complex structure map i introduced below.

The operation of *reverse* is denoted by * and is defined on blades by $(b_{j_1}\ldots b_{j_k})^* = b_{j_k}\ldots b_{j_1}$. Now let $a = b_0 b$, $a_k = b_k b$, $1 \leqslant k \leqslant n$. Then

$$a_k^* c_{A;A_1\ldots A_k\ldots A_n} a_k = (-1)^{A_k} c_{A;A_1\ldots A_k\ldots A_n}$$

$$b_k c_{A_0;A_1\ldots A_k\ldots A_n} = (-1)^{\sum_{j=0}^{k-1} A_j} c_{A_0;A_1\ldots A_k'\ldots A_n},$$

where the prime denotes negation of a bit, i.e. $0' = 1$, $1' = 0$. Negation of the kth bit can be expressed in algebraic terms

$$n_k c_{A;A_1\ldots A_k\ldots A_n} = b_k a_{k-1}^* \ldots a_1^* a^* c_{A;A_1\ldots A_k\ldots A_n} a a_1 \ldots a_{k-1} = c_{A;A_1\ldots A_k'\ldots A_n}.$$

The complex structure is defined by

$$\mathrm{i} c_{A;A_1\ldots A_n} = (-1)^{A'} c_{A';A_1\ldots A_n}.$$

This definition implies the usual formulae

$$\mathrm{i}^2 c_{A;A_1\ldots A_n} = -c_{A;A_1\ldots A_n}, \qquad \mathrm{e}^{\mathrm{i}\phi} c_{A;A_1\ldots A_n} = (\cos\phi + \mathrm{i}\sin\phi) c_{A;A_1\ldots A_n}.$$

i and n_k commute if $0 < k$.

One has now two options: either work with explicitly real coefficients but having the number of combs doubled (by the presence of the zeroth bit), or allow for 'complex' coefficients explicitly involving the linear map i, and then the zeroth bit can be skipped. I prefer the second option, where the combs are parametrized by n indices $c_{A_1\ldots A_n}$, since it makes the formulae compact and quantum looking, and all the shown bits are used for coding binary numbers. Still, for geometric purposes it is important to bear in mind that the Clifford algebra is real.

3. Elementary gates

A one-bit gate, $1 \leqslant k \leqslant n$, is

$$G_k = \tfrac{1}{2}(\alpha + \beta n_k)(1 + (-1)^{A_k}) + \tfrac{1}{2}(\delta + \gamma n_k)(1 - (-1)^{A_k})$$

where $\alpha = \alpha_1 + \alpha_2\mathrm{i}$, $\beta = \beta_1 + \beta_2\mathrm{i}$, $\gamma = \gamma_1 + \gamma_2\mathrm{i}$, $\delta = \delta_1 + \delta_2\mathrm{i}$; the numbers $\alpha_1, \ldots, \delta_2$ are real. The link to quantum computation is that the matrix of coefficients $\left(\begin{smallmatrix}\alpha & \beta\\ \gamma & \delta\end{smallmatrix}\right)$ should be taken in a form corresponding to an appropriate quantum gate.

Let us check the concrete gates. The three Pauli gates are

$$X_k c_{A_1\ldots A_k\ldots A_n} = n_k c_{A_1\ldots A_k\ldots A_n},$$

$$Y_k c_{A_1\ldots A_k\ldots A_n} = -\mathrm{i} n_k a_k^* c_{A_1\ldots A_k\ldots A_n} a_k,$$

$$Z_k c_{A_1\ldots A_k\ldots A_n} = a_k^* c_{A_1\ldots A_k\ldots A_n} a_k.$$

One verifies on components the usual properties

$$
\begin{aligned}
X_k c_{A_1\ldots 0_k\ldots A_n} &= c_{A_1\ldots 1_k\ldots A_n},\\
X_k c_{A_1\ldots 1_k\ldots A_n} &= c_{A_1\ldots 0_k\ldots A_n},\\
Y_k c_{A_1\ldots 0_k\ldots A_n} &= -\mathrm{i} c_{A_1\ldots 1_k\ldots A_n},\\
Y_k c_{A_1\ldots 1_k\ldots A_n} &= \mathrm{i} c_{A_1\ldots 0_k\ldots A_n},\\
Z_k c_{A_1\ldots 0_k\ldots A_n} &= c_{A_1\ldots 0_k\ldots A_n},\\
Z_k c_{A_1\ldots 1_k\ldots A_n} &= -c_{A_1\ldots 1_k\ldots A_n}.
\end{aligned}
$$

The Hadamard gate:

$$
\begin{aligned}
H_k c_{A_1\ldots A_k\ldots A_n} &= \tfrac{1}{\sqrt{2}} n_k c_{A_1\ldots A_k\ldots A_n} + \tfrac{1}{\sqrt{2}} a_k^* c_{A_1\ldots A_k\ldots A_n} a_k\\
&= \tfrac{1}{\sqrt{2}} (X_k + Z_k) c_{A_1\ldots A_k\ldots A_n}.
\end{aligned}
$$

The phase and $\pi/8$ gates:

$$
\begin{aligned}
S_k c_{A_1\ldots A_k\ldots A_n} &= \tfrac{1}{2}(1+\mathrm{i}) c_{A_1\ldots A_k\ldots A_n} + \tfrac{1}{2}(1-\mathrm{i}) a_k^* c_{A_1\ldots A_k\ldots A_n} a_k\\
T_k c_{A_1\ldots A_k\ldots A_n} &= \tfrac{1}{2}(1+\mathrm{e}^{\mathrm{i}\pi/4}) c_{A_1\ldots A_k\ldots A_n} + \tfrac{1}{2}(1-\mathrm{e}^{\mathrm{i}\pi/4}) a_k^* c_{A_1\ldots A_k\ldots A_n} a_k.
\end{aligned}
$$

Let us check the latter two on components:

$$
\begin{aligned}
S_k c_{A_1\ldots 0_k\ldots A_n} &= c_{A_1\ldots 0_k\ldots A_n}\\
S_k c_{A_1\ldots 1_k\ldots A_n} &= \mathrm{i} c_{A_1\ldots 1_k\ldots A_n}\\
T_k c_{A_1\ldots 0_k\ldots A_n} &= c_{A_1\ldots 0_k\ldots A_n}\\
T_k c_{A_1\ldots 1_k\ldots A_n} &= \mathrm{e}^{\mathrm{i}\pi/4} c_{A_1\ldots 1_k\ldots A_n}.
\end{aligned}
$$

A general controlled two-bit gate is

$$
G_{kl} = G'_k \tfrac{1}{2}(1 + (-1)^{A_l}) + G''_k \tfrac{1}{2}(1 - (-1)^{A_l}),
$$

where G'_k and G''_k are one-bit gates. Control-NOT (CNOT) reads

$$
\mathrm{cn}_{kl} = \tfrac{1}{2}(1 + (-1)^{A_l}) + X_k \tfrac{1}{2}(1 - (-1)^{A_l}).
$$

This can be generalized to arbitrary numbers of controlling bits. An example is given by the three-bit control-CNOT (Toffoli) gate

$$
\mathrm{cn}_{klm} = \tfrac{1}{2}(1 + (-1)^{A_m}) + \mathrm{cn}_{kl} \tfrac{1}{2}(1 - (-1)^{A_m}).
$$

4. Geometric meaning of the gates

The gates such as H_k or cn_{kl} and cn_{klm} consist of *pairs* of operations, a fact suggesting that composition of N gates will require 2^N operations. The problem is, however, more subtle. In order to see the subtlety we have to get used to thinking of all the geometric-algebra operations in geometric terms.

4.1. Two bits, gates X_1 and X_2

For two bits the geometric background is provided by a plane spanned by some orthonormal basis $\{e_1, e_2\}$. The blades are $1 = \circ$ (a 'charged' point), $e_1 = \rightarrow, e_2 = \uparrow$ (oriented line segments), $e_{12} = \Box$ (an oriented plane segment). The action of the gates is

$X_1 c_{A_1A_2} = c_{A_1'A_2}$, $X_2 c_{A_1A_2} = c_{A_1A_2'}$. We can forget about the zeroth bit (leading to a third dimension) and visualize as follows:

$$X_1 \begin{pmatrix} \circ \\ \rightarrow \\ \uparrow \\ \square \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \circ \\ \rightarrow \\ \uparrow \\ \square \end{pmatrix}.$$

One recognizes in the above matrix the tensor product $\mathbf{1} \otimes \sigma_1$:

$$X_2 \begin{pmatrix} \circ \\ \rightarrow \\ \uparrow \\ \square \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \circ \\ \rightarrow \\ \uparrow \\ \square \end{pmatrix}.$$

Now the matrix is $\sigma_1 \otimes \mathbf{1}$. Similar representation is found if one takes a multivector $V = V_0 + V_1 e_1 + V_2 e_2 + V_{12} e_{12} = (V_0, V_1, V_2, V_{12})$. Then

$$X_1 V = (V_1, V_0, V_{12}, V_2), \qquad X_2 V = (V_2, V_{12}, V_0, V_1).$$

Let us recall that multivectors are, from a geometrical standpoint, sets containing different shapes, so they have a clear geometric interpretation [1]. Simultaneously, in the context of computation, they play a role of entangled states.

4.2. Two bits, gates Z_1 and Z_2

$Z_1 c_{A_1A_2} = (-1)^{A_1} c_{A_1A_2}$, $Z_2 c_{A_1A_2} = (-1)^{A_2} c_{A_1A_2}$,

$$Z_1 \begin{pmatrix} \circ \\ \rightarrow \\ \uparrow \\ \square \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} \circ \\ \rightarrow \\ \uparrow \\ \square \end{pmatrix},$$

$$Z_2 \begin{pmatrix} \circ \\ \rightarrow \\ \uparrow \\ \square \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} \circ \\ \rightarrow \\ \uparrow \\ \square \end{pmatrix}.$$

4.3. Two bits, gates H_1 and H_2

Since $H_k = (X_k + Z_k)/\sqrt{2}$,

$$H_1 \begin{pmatrix} \circ \\ \rightarrow \\ \uparrow \\ \square \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} \circ \\ \rightarrow \\ \uparrow \\ \square \end{pmatrix},$$

$$H_2 \begin{pmatrix} \circ \\ \uparrow \\ \rightarrow \\ \square \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} \circ \\ \uparrow \\ \rightarrow \\ \square \end{pmatrix}.$$

Note that H_2 is represented with permuted $\rightarrow$ and $\uparrow$.

Let us stress again that although formally one can identify certain tensor products in the above matrices, the space of states does not involve abstract tensoring of qubits, but only geometric operations in Euclidean spaces.

4.4. Two bits, gates cn_{12} *and* cn_{21}

Here $\mathrm{cn}_{12}c_{A_1 0} = c_{A_1 0}$, $\mathrm{cn}_{12}c_{A_1 1} = c_{A_1' 1}$, $\mathrm{cn}_{21}c_{0A_2} = c_{0A_2}$, $\mathrm{cn}_{21}c_{1A_2} = c_{1A_2'}$.

$$\mathrm{cn}_{12}\begin{pmatrix}\circ\\ \rightarrow\\ \uparrow\\ \square\end{pmatrix} = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}\circ\\ \rightarrow\\ \uparrow\\ \square\end{pmatrix},$$

$$\mathrm{cn}_{21}\begin{pmatrix}\circ\\ \uparrow\\ \rightarrow\\ \square\end{pmatrix} = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}\circ\\ \uparrow\\ \rightarrow\\ \square\end{pmatrix}.$$

4.5. Three bits, gates cn_{123}, cn_{312} *and* cn_{231}

Here the only nontrivial actions are $\mathrm{cn}_{123}c_{A_1 11} = c_{A_1' 11}$, $\mathrm{cn}_{312}c_{11A_3} = c_{11A_3'}$, $\mathrm{cn}_{231}c_{1A_2 1} = c_{1A_2' 1}$. The Euclidean space is three dimensional. The blades involve a point 1, three edges b_1, b_2, b_3, three walls b_{12}, b_{23}, b_{13}, and the cube b_{123}.

$$\begin{aligned}
\mathrm{cn}_{123}c_{011} &= \mathrm{cn}_{123}b_{23} = c_{111} = b_{123},\\
\mathrm{cn}_{123}c_{111} &= \mathrm{cn}_{123}b_{123} = c_{011} = b_{23},\\
\mathrm{cn}_{312}c_{110} &= \mathrm{cn}_{312}b_{12} = c_{111} = b_{123},\\
\mathrm{cn}_{312}c_{111} &= \mathrm{cn}_{312}e_{123} = c_{110} = b_{12},\\
\mathrm{cn}_{231}c_{101} &= \mathrm{cn}_{231}b_{13} = c_{111} = b_{123},\\
\mathrm{cn}_{231}c_{111} &= \mathrm{cn}_{231}b_{123} = c_{101} = b_{13}.
\end{aligned}$$

Geometrically in 3D the Toffoli gate means squashing a cube into a square (one of its walls), or the other way around—reconstructing a cube from a wall. Composition of two different Toffoli gates exchanges walls of the cube, e.g. $\mathrm{cn}_{312}\mathrm{cn}_{123}b_{23} = \mathrm{cn}_{312}b_{123} = b_{12}$.

5. Example

As an example we take the simple but impressive application of 'quantum parallelism', where applying n Hadamard gates (i.e., performing n algorithmic steps) one generates a superposition of 2^n n-bit numbers. In quantum computation the operation looks as follows:

$$\begin{aligned}
H^{\otimes n}|0_1\ldots 0_n\rangle &= \frac{1}{\sqrt{2^n}}(|0_1\rangle + |1_1\rangle)\ldots(|0_n\rangle + |1_n\rangle)\\
&= \frac{1}{\sqrt{2^n}}\sum_{A_1\ldots A_n}|A_1\ldots A_n\rangle.
\end{aligned}$$

Quantum speedup comes from the fact that most of the operations have not to be performed by the computer itself but are taken care of by properties of the tensor product.

So let us take a look at an analogous calculation performed in the geometric-algebra framework:

$$\begin{aligned}
H_n c_{0_1\ldots 0_n} &= \frac{1}{\sqrt{2}}\left(n_n c_{0_1\ldots 0_n} + a_n^* c_{0_1\ldots 0_n} a_n\right)\\
&= \frac{1}{\sqrt{2}}\left(c_{0_1\ldots 1_n} + c_{0_1\ldots 0_n}\right) = \frac{1 + b_n}{\sqrt{2}},
\end{aligned}$$

$$H_{n-1}H_n c_{0_1\ldots 0_n} = \frac{n_{n-1}(1+b_n) + a^*_{n-1}(1+b_n)a_{n-1}}{\sqrt{2^2}}$$
$$= \frac{b_{n-1}(1+b_n) + 1 + b_n}{\sqrt{2^2}} = \frac{(1+b_{n-1})(1+b_n)}{\sqrt{2^2}}.$$

Let us note that the multivector $1 + b_n$ is treated by n_{n-1} as a whole. From a Clifford-algebra point of view this is simply a single multivector. It makes no sense to treat $1 + b_n$ as a combination of just *two* blades, since a change of basis will map it into a combination of another number of blades. There exists a single geometric object represented by $1 + b_n$. This general observation applies to all the universal gates introduced above, and shows how to geometrically interpret the number of steps of an algorithm.

Repeating the above procedure n times we obtain

$$H_1 \ldots H_n c_{0_1\ldots 0_n} = \frac{(1+b_1)\ldots(1+b_n)}{\sqrt{2^n}} \tag{1}$$
$$= \frac{1}{\sqrt{2^n}} \sum_{A_1\ldots A_n} c_{A_1\ldots A_n}. \tag{2}$$

Equation (1) shows that the n-fold Hadamard gate involves $n - 1$ Clifford-algebra multiplications. Even counting the additions in the braces as operations performed by the algorithm we arrive at $2n - 1$ steps needed for producing a linear combination of 2^n binary numbers.

It is therefore clear that the geometric product performs the same type of 'compression' as the tensor product. Multivectors of the form (2) can be acted upon with further gates, and in each single step one processes the entire set of 2^n numbers.

6. Remarks on earlier approaches

Links between qubits, spinors, entangled states and geometric algebra were, of course, noticed a long time ago, much earlier than in [1]. One should mention, first of all, the pioneering works of Hestenes [7] on relations between geometric algebra and relativity, and spinors in particular. In the context of quantum information theory the most important earlier papers are those by Havel, Doran and their collaborators, cf [8–13].

However, it seems that the very way of coding, that is, linking bits with multivectors, was in those works much less straightforward than the convention I work with in the present paper, and which was introduced in [1]. In my opinion the 'old' approach can be reduced to replacing two-component complex vectors by 2×2 matrices whose second column is empty. Such 'spinors' are matrices and thus can be written as linear combinations of the Pauli matrices, simultaneously maintaining the essential properties of the usual spinors or qubits. The Pauli matrices, on the other hand, can be regarded as a representation of geometric algebra of two- or three-dimensional Euclidean spaces. Multiparticle systems are introduced by replacing a three-dimensional space with a configuration space and one arrives at a multiparticle geometric algebra. The tensor product is then constructed by means of bivectors (appropriate bivectors may commute with one another).

The approach used in [1] and in the present paper is so different from those based on multiparticle geometric algebras that it is even difficult to find similarities. Here tensor products are not employed at any stage (of course sometimes some matrices are of a tensor product form, as we have seen in the case of X_k, say, but this is irrelevant for the construction) and even the 'i' I use is different. So the approach I advocate is clearly an alternative to the earlier works that, at least in my opinion, have a status of a standard theory reformulated in a different language.

Acknowledgments

I am indebted to Krzysztof Giaro, Marcin Pawłowski, Tomasz Magulski and Łukasz Orłowski for discussions.

References

[1] Aerts D and Czachor M 2007 *J. Phys. A: Math. Theor.* **40** F259
Aerts D and Czachor M 2006 *Preprint* quant-ph/0610187
Aerts D and Czachor M 2006 *Preprint* quant-ph/0611279
[2] Deutsch D and Jozsa R 1992 *Proc. R. Soc.* A **439** 553
[3] Magulski T and Orłowski Ł 2007 *Preprint* quant-ph/0705.4289
[4] Simon D R 1997 *SIAM J. Comput.* **26** 1474
[5] Pawłowski M 2006 *Preprint* quant-ph/0611051
[6] Nielsen M A and Chuang I L 2000 *Quantum Computation and Quantum Information* (Cambridge: Cambridge University Press)
[7] Hestenes D 1966 *Space-Time Algebra* (New York: Gordon and Breach)
[8] Havel T F and Doran C J L 2001 *Preprint* quant-ph/0106063
[9] Somaroo S S, Cory D G and Havel T F 1998 *Phys. Lett.* A **240** 1
[10] Parker R and Doran C 2001 *Preprint* quant-ph/0106055
[11] Doran C J L, Lasenby A N, Gull S F, Somaroo S and Challinor A D 1996 *Adv. Imaging. Electron. Phys.* **95** 271
[12] Havel T F, Doran C and Furuta S *Proc R. Soc. Lond.* at press
[13] Sharf Y, Cory D G, Somaroo S S, Havel T F, Knill E and Laflamme R 2000 *Mol. Phys.* **98** 1347