*International Journal of Theoretical Physics, Vol. 37, No. 5, 1998*

# Solving the Schrödinger Equation for the Feynman Quantum Computer

**Tino Gramss**[1]

*Received November 4, 1997*

Feynman showed that a closed, locally interacting quantum system is capable of performing deterministic computation. For a finite-size version of such a computer, the Schrödinger equation is solved analytically. The probability that the computer yields a final result upon measurement is derived.

## 1. INTRODUCTION

The concept of quantum computers, first introduced by Benioff (1980a, b), has recently found increasing interest. Usually such computers are discussed and analyzed regardless of whether their Hamiltonian exhibits long-range interactions or not. However, there are two reasons for the interest in locally interacting quantum computers. (1) Nonlocal interactions introduce longer communication pathways, which give rise to reduced computational speed. (2) Quantum computers with nonlocal interactions are probably more difficult to realize because long-range interactions have to be implemented on a submicroscopic level.

This work focuses on a closed, locally interacting quantum computer model, first introduced by Feynman (1985, 1986).

Following Feynman's original idea, the power of quantum computers with only local interactions has been analyzed by various authors. Peres (1985) was the first to obtain some analytical results for such computers. Margolus (1986, 1990), Grössing and Zeilinger (1988), and Biafori (1993) generalized the concepts to local cellular automata. Results concerning computational complexity theory can be found in Gramss (1994) and Lloyd (1996).

[1] Physics Department, Medical Physics Group, University of Oldenburg, D-26111 Oldenburg, Germany. The author passed away on Jan. 12th, 1998.

## 2. THE FEYNMAN COMPUTER

Consider a deterministic classical computer that consists of $k$ gates connected in a serial way, as depicted in Fig. 1. It passes through a number $k$ of computational states until it displays a final result at the output of the last gate. We now want to model the same computation by purely quantum means. Assuming that the operation of gate $n$ can be described by a unitary matrix $\mathbf{D}_n$ and working on the computer's memory,

$$|\psi_{n+1}\rangle = \mathbf{D}_n|\psi_n\rangle \tag{1}$$

where $|\psi_n\rangle$ is the wave function of the computer at time $n$, and $|\psi_{n+1}\rangle$ is the wave function at time $n + 1$. In order to realize a physical device like our computer, we have to know its Hamiltonian. The Hamiltonian $\mathbf{H}_n$ that is effective from time $n$ to time $n + 1$ is then given through

$$\mathbf{H}_n = i \ln \mathbf{D}_n$$

Clearly, only a reversible computational step can be described by a unitary matrix. It was shown first for classical systems (Toffoli, 1977; Fredkin and Toffoli, 1982) and later computing quantum systems (Bennett, 1973, 1982, 1989; Levine and Sherman, 1990) that universal computation can be done by purely reversible means without essential additional expenditure in space and time. Therefore, reversibility is not a restriction for computational power.

However, it should be mentioned that a computer as depicted in Fig. 1 is not a universal one. Methods to construct a universal and locally interacting quantum computer are described in Gramss (1994) and Lloyd (1996).

Applying sequentially the operations $\mathbf{D}_n$ results in a time-dependent Hamiltonian that describes the overall evolution: After one computational step, the Hamiltonian $\mathbf{H}_n$ has to be altered to $\mathbf{H}_{n+1}$, which in turn will then be effective for exactly one time unit.

Since the overall Hamiltonian is obviously time dependent, such a computation cannot be done in a closed system. Feynman found a way to derive a time-independent Hamiltonian that performs the desired computation (Feynman, 1985, 1986). In the following, we will give a short description of the Feynman computer.

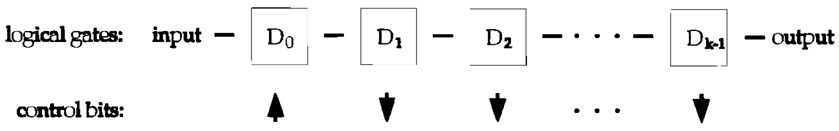

**Fig. 1.** The Feynman computer. A logical gate performs an operation if the corresponding control bit is set to "spin up." The up-spin is shifted to the right after a computational step.

Figure 1 describes the operation of a Feynman computer with $k$ logical gates and $k$ associated "control bits." The role of the control bits will become clear in a moment.

To obtain a computer with a time-invariant Hamiltonian, the first step is to add the control bits to each computational gate $n$. A control bit is set to $|1\rangle$ if the associated computational gate $n$ is active, otherwise it is set to $|0\rangle$. In a tensor product Hilbert space of the computer memory (on which the operations $\mathbf{D}_n$ are performed) and the control bits, the effective unitary operation for each computational step reads

$$\mathbf{F}_n = \mathbf{D}_{n+1}|n + 1\rangle \langle n|$$

[This notation differs from the one used by Feynman and in Gramss (1997). It was introduced in Peres (1985).]

Therefore, if we have the computer in a state $|\psi_n\rangle|n\rangle$ where $|\psi_n\rangle$ is the state of the memory and $|n\rangle$ the control bit state, we get

$$\begin{aligned} \mathbf{F}|\psi_n\rangle|n\rangle &= |\psi_{n+1}\rangle|n + 1\rangle \\ \mathbf{F}^\dagger|\psi_n\rangle|n\rangle &= |\psi_{n-1}\rangle|n - 1\rangle \end{aligned} \tag{2}$$

$\mathbf{F}$ describes one step of a forward computation, $\mathbf{F}^\dagger$ describes the reverse, backward computation. By $\mathbf{F}$, the "cursor" (the control bit set to $|1\rangle$) is shifted right, by $\mathbf{F}^\dagger$ it is shifted left. Thus, $|n\rangle$ simply codes the position of the cursor. For a finite-size computer, additional care has to be taken for the cases $n = 0$ and $n = k - 1$, as will be done below.

Feynman defined a time-independent Hamiltonian to be

$$\mathbf{H} = \mathbf{F} + \mathbf{F}^\dagger \tag{3}$$

where

$$\mathbf{F} = \sum_{n=0}^{k-1} \mathbf{F}_n \tag{4}$$

$\mathbf{H}$ is indeed a Hamiltonian since it is a Hermitian matrix. Also, it consists of a sum of locally acting Hamiltonians $\mathbf{H}_n = \mathbf{F}_n + F_n^\dagger$.

According to the solution of the Schrödinger equation, the dynamical evolution of the computer is described by the unitary matrix

$$\mathbf{U} = \exp(-i\mathbf{H}t)$$

By expanding the exponential we get—aside from constant factors—terms like $\mathbf{F}_0\mathbf{F}_1^\dagger\mathbf{F}_3$ or $\mathbf{F}_1^\dagger\mathbf{F}_1\mathbf{F}_0$ or $\mathbf{F}_3\mathbf{F}_2\mathbf{F}_1\mathbf{F}_0$. Applied to an initial computational state with control bits all set to zero, the first operation will result in a zero vector, since on the subspace of the control bits $|3\rangle\langle 2|0\rangle = \mathbf{0}$. By this example, the relevance of the control bits becomes clear: They are necessary to avoid noncomputational states, i.e., states that do not correspond to the state of the classical version of a Feynman computer. The second example, $\mathbf{F}_1^\dagger\mathbf{F}_1\mathbf{F}_0$, will result in a computational state that has just performed the first operation $\mathbf{F}_0$ since $\mathbf{F}_1^\dagger$ reverses the operation $\mathbf{F}_1$. On the subspace of the control bits $|0\rangle\langle 1|1\rangle\langle 0|0\rangle = |0\rangle$, the cursor is at position 0 again. The third operation will result in a computer that has done four steps of computation in correct order.

Considering these examples, we note two things:

First: Only a superposition of computational states will arise if **U** operates on the initial computational state or any other computational state or superposition of such states. In other words, the possible states of the computer are confined to the subspace of computational states.

Second: On performing a proper measurement, we will find a computational state with a certain probability larger than zero and smaller than one.

In the following sections, the Hamiltonian (3) will be analyzed in detail.

## 3. COMPUTATIONAL CLOCKS AND THE COMPUTATIONAL TIME OPERATOR

First we consider the dynamical evolution of the control bit states which may serve as a quantum clock. It can be described by the operator **C** defined by

$$\begin{aligned}\mathbf{C}|n\rangle &= |n+1\rangle \\ \mathbf{C}^\dagger|n\rangle &= |n-1\rangle\end{aligned} \tag{5}$$

Thus $\mathbf{C}\dagger$ is the clock which "runs backward." In the case of a finite clock, incrementing and decrementing $n$ has to be done modulo the maximum and minimum time values

$$\begin{aligned}\mathbf{C}|N-1\rangle &= |0\rangle \\ \mathbf{C}^\dagger|0\rangle &= |N-1\rangle\end{aligned} \tag{6}$$

We now introduce a simple operator that measures computational time.

Say the time operator is **N** and the nondegenerate orthogonal eigenstates read $|n\rangle$, where $n$ is the corresponding eigenvalue. Then we can measure the "time" $n$ by using the time operator

$$\mathbf{N}|n\rangle = n|n\rangle \tag{7}$$

In the case of a finite-size computer, it is not difficult to give simple matrices that represent the overall shift operator that works on the control bit state and shifts the control bits from left to right in Fig. 1

We choose a representation in which the $i$th of the $N$ orthogonal clock states reads $[0, 0, \ldots, 0, 1, 0, \ldots, 0]$, where the 1 occurs at the $i$th position.

The matrix representation for the clock **C** is

$$\mathbf{C} = \begin{bmatrix} 0 & & & & & & 1 \\ 1 & 0 & & & & & \\ & 1 & 0 & & & \mathbf{0} & \\ & & & \cdot & \cdot & & \\ & \mathbf{0} & & & \cdot & \cdot & \\ & & & & & \cdot & \cdot \\ & & & & & 1 & 0 \end{bmatrix} \tag{8}$$

The matrix representation of **N** reads

$$\mathbf{N} = \begin{bmatrix} 0 & & & & & & \\ & 1 & & & & & \\ & & 2 & & \mathbf{0} & & \\ & & & \cdot & & & \\ & \mathbf{0} & & & \cdot & & \\ & & & & & \cdot & \\ & & & & & & N-1 \end{bmatrix} \tag{9}$$

That is, the control bit state, which reads in vector notation as [1, 0, 0, . . . , 0] (the 1 at position 0) is transformed to [0, 1, 0, . . . , 0] (the 1 at position 1) and so on, [0, 0, 0, . . . , 1] is transformed to [1, 0, 0, . . . , 0]. The states are simply rotated.

In contrast to Margolus (1990), Biafori (1993), and Gramss (1997), we will restrict ourselves to finite-size quantum computers.

## 4. EIGENSYSTEM OF THE HAMILTONIAN

Now we will calculate the eigensystem of the Hamiltonian $\mathbf{H} = \mathbf{F} + \mathbf{F}^\dagger$.

Consider equation (2) and Fig. 1. If upon measurement the clock is found in state $|n\rangle$ we can be certain that the computer has done $n$ steps of computation and is in state $|\psi_n\rangle|n\rangle$. In other words, during computation the clock states are rotated in exactly the same way as the states $|\psi_n\rangle$ of the computer's memory. This is why it is sufficient to find the eigensystem of $\mathbf{H}_{\mathrm{sub}} = \mathbf{C} + \mathbf{C}^\dagger$ with **C** from (8).

We first diagonalize the clock matrix **C** from (8). It is a cyclic matrix, which makes this task simple. (A cyclic matrix is a square matrix where a row can be obtained by simply shifting all the entries in the upper row by one. Therefore, a matrix **A** with entries $a_{nm}$ is cyclic if $a_{nm}$, $a_{n,m+1} = a_{nm}$ or $a_{n,m-1} = a_{nm}$.)

First we recall a theorem about cyclic matrices, which is central for the following discussion (Strang, 1992):

All cyclic $N \times N$ matrices have the same eigenvectors, i.e., they can be diagonalized by the same unitary matrix $\Gamma$, where

$$\gamma_{nm} = \frac{1}{\sqrt{N}} \zeta^{nm} \qquad \text{with } \zeta = \exp\left(\frac{2\pi i}{N}\right)$$

Counting for the indices starts with zero. In matrix notation

$$\Gamma = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \zeta & \zeta^2 & \cdots & \zeta^{N-1} \\ 1 & \zeta^2 & \zeta^4 & \cdots & \zeta^{2(N-1)} \\ & & \vdots & & \\ 1 & \zeta^{N-1} & \zeta^{2(N-1)} & \cdots & \zeta^{(N-1)^2} \end{bmatrix} \tag{10}$$

Therefore, the normalized eigenvectors of every cyclic matrix are the rows of $\Gamma$.

$\Gamma$ from (10) diagonalizes $\mathbf{C}$ and $\mathbf{C}^\dagger$ since both matrices are cyclic. Thus, $\Gamma$ also diagonalizes $\mathbf{H}_{\text{sub}}$:

$$\Lambda = \Gamma^\dagger \mathbf{H}_{\text{sub}} \Gamma \tag{11}$$

Since $\mathbf{H}_{\text{sub}}$ is Hermitian, $\Lambda$ is a diagonal matrix which contains the eigenvalues $\lambda_k$. After some calculations, we get from (8), (10), and $\mathbf{H}_{\text{sub}} = \mathbf{C} + \mathbf{C}^\dagger$

$$\lambda_k = 2 \cos\left(\frac{2\pi k}{N}\right) \tag{12}$$

For the eigenvector $\mathbf{v}_k$ the entries $v_{k,l} = (1/\sqrt{N})\zeta^{kl}$. This is the $k$th column of (10)

## 5. SOLUTION OF THE SCHRÖDINGER EQUATION

Having diagonalized the Hamiltonian, we are now also able to compute the time evolution of the Feynman computer. This can be done in a standard way.

Quite generally, the solution to the Schrödinger equation reads

$$|\psi(t)\rangle = \exp(-i\mathbf{H}_{\text{sub}} t)|\psi(0)\rangle$$

With (11) it follows that

$$|\psi(t)\rangle = \Gamma \exp(-i\Lambda t)\Gamma^\dagger |\psi(0)\rangle \tag{13}$$

Say, the $n$th entry of the vector $|\psi\rangle$ is $\psi_n$, and the elements of a matrix $\mathbf{A}$ are denoted by $(\mathbf{A})_{jk}$. Then

$$\begin{aligned}
\psi_j(t) &= \sum_k (\Gamma)_{jk} \exp(-i(\Lambda)_{kk}t) \sum_l (\Gamma^\dagger)_{kl}\psi_l(0) \\
&= \frac{1}{N}\sum_k \zeta^{jk} \exp(-i\lambda_k t) \sum_l \zeta^{-kl}\ \psi_l(0) \\
&= \frac{1}{N}\sum_k \sum_l \zeta^{k(j-l)}\ \zeta^{\lambda'_k t}\psi_l(0) \\
&= \frac{1}{N}\sum_l \sum_k \zeta^{kl}\ \zeta^{\lambda'_k t}\psi_{j-l}(0) \\
&= \frac{1}{N}\sum_l \psi_{j-l}(0) \sum_k \zeta^{kl}\zeta^{\lambda'_k t} \\
&= \frac{1}{N}\sum_l \ \psi_{j-t}(0)\mathcal{F}_l(\zeta^{\lambda'_k t})
\end{aligned}$$

where the abbreviation $\lambda'_k = -(N/2\pi)\lambda_k$ has been used, and $\mathcal{F}_l$ denotes the Fourier transform

$$\mathcal{F}_l(x_k) = \sum_k x_k e^{2\pi i k l/N}$$

Therefore, the solution to the Schrödinger equation is the convolution of the initial state with the Fourier transform of $\zeta^{\lambda_k t}$:

$$|\psi(t)\rangle = \frac{1}{N}|\psi(0)\rangle * \mathcal{F}(\zeta^{\lambda'_k t}) \tag{14}$$

In the Appendix, a power series for the Fourier transform in (14) is derived. The Fourier transform of $\zeta^{\lambda'_k t}$ can be written symbolically as

$$\mathcal{F}_n(\zeta^{\lambda'_k t}) = N\exp\{-it[\mu(n, -1) + \mu(n, +1)]\}$$

where $\mu(n, k)\mu(n, l) = \mu(n, k + l)$ and $\mu(n, m) = \tilde{\delta}_{n,m} = \delta_{n \bmod N, m \bmod N}$. This notation is explained in the Appendix.

It is possible to get a solution in closed form for special initial conditions, as will now be shown.

We ask for the time evolution if we start with a single computational state, i.e.,

$$\psi_j(0) = \delta_{j,0}$$

If we solve the Schrödinger equation for this case, but start in a different computational state, the solution can be easily obtained because of the cyclic symmetry of the computer.

The solution to the Schrödinger equation is

$$
\begin{aligned}
\psi_n(t) &= \delta_{j,0} * (\exp\{-it[\mu(j, -1) + \mu(j, +1)]\}) \\
&= \sum \delta_{n-j,0} \exp\{-it[\mu(j, -1) + \mu\ (j, +1)]\} \\
&= \exp\{-it[\mu(n, -1) + \mu(n, +1)]\} \qquad (15)
\end{aligned}
$$

From Figs. 2 and 3 it can be seen that the probability $|\psi_0(t)|^2$ to measure the initial condition first decays in a smooth and oscillatory way before its time evolution becomes irregular. This can be explained by expanding (15):

$$
\begin{aligned}
\psi_n(t) &= \sum_{m=0}^{\infty} \frac{(-it)^m}{m!} (\mu(n, -1) + \mu(n, +1))^m \\
&= \sum_{m=0}^{\infty} \frac{(-it)^m}{m!} \sum_{k=0}^{m} \binom{m}{k} \mu(n, -1)^{m-k} \mu(n, +1)^k \\
&= \sum_{m=0}^{\infty} \frac{(-it)^m}{m!} \sum_{k=0}^{m} \binom{m}{k} \mu(n, -m + k)\mu(n, k) \\
&= \sum_{m=0}^{\infty} \frac{(-it)^m}{m!} \sum_{k=0}^{m} \binom{m}{k} \mu(n, -m + 2k) \\
&= \sum_{m=0}^{\infty} \frac{(-it)^m}{m!} \sum_{k=0}^{m} \binom{m}{k} \delta_{n,(-m+2k)\bmod N}
\end{aligned}
$$

For the sake of simplicity, we will restrict ourselves in the following to the case of even $N$. Here $n$ labels the $N$ computational states. It is convenient to assume that $n$ takes the values $-N/2 + 1 \ldots N/2$. The term $\binom{m}{k}$ in the second sum only survives the Kronecker delta if

$$
n \bmod N = (-m + 2k) \bmod N \qquad (16)
$$

Since the modulo operation does not change the parity if $N$ is even, it follows that $n + m$ must be even. The term $-m + 2k$ takes the values $-m \ldots +m$. There are no terms if $m < |n|$, and the condition (16) is fulfilled exactly for a single $k = (|n| + m)/2$ if $|n| \leq m < N/2$. A little more thinking yields that, for even $n + m$, the condition is fulfilled exactly $\kappa_m$ times for the same term $\binom{m}{(|n|+m)/2}$, where

$$
\kappa_m \frac{N}{4} \leq m < \kappa_m \frac{N}{2}, \qquad \kappa_m \geq 2
$$

We now introduce the new running variable $l$ so that $m = |n| + 2l$. Now, $n + m$ takes only even values and $m \geq |n|$. With this setting, we have

$$\psi_n(t) = \sum_{l=0}^{\infty} \kappa_{|n|+2l} \frac{(-it)^{|n|+2l}}{(|n| + 2l)!} \binom{|n| + 2l}{|n| + l}$$
$$= \sum_{l=0}^{\infty} \kappa_{|n|+2l} \frac{(-it)^{|n|+2l}}{l!(|n| + l)!} \tag{17}$$

Consider now the terms with small $|n| + 2l$, which are dominant if $t$ is small. If $|n| \leq m < N/2$, $\kappa_{|n|+2l} = 1$, and the expansion of (17) reads approximately

$$\psi_n(t) \approx \sum_{l=0}^{\infty} \frac{(-it)^{|n|+2l}}{l!(|n| + l)!} = \begin{cases} \mathrm{J}_{|n|}(2t) & \text{for even } n \\ -i\mathrm{J}_{|n|}(2t) & \text{for odd } n \end{cases}$$

where $\mathrm{J}_\nu(x)$ is the Bessel function of first kind and order $\nu$. The first $N/2 - |n|$ are exact. Therefore the approximation is the better, the smaller is $|n|$.

For the case where $N/2 \leq m < N$, which will become important below, we have $\kappa_{|n|+2l} = 2$. We then obtain

$$\psi_n(t) \approx \begin{cases} 2\mathrm{J}_{|n|}(2t) & \text{for even } n \\ -2i\mathrm{J}_{|n|}(2t) & \text{for odd } n \end{cases} \tag{18}$$

This approximation is good for $|n| \geq N/2$.

In Fig. 2 the probability $|\psi_0(t)|^2$ is shown for small $t$, if we start in $\psi_j(0) = \delta_{j,0}$. It decays according to the squared Bessel function. For larger $t$, the probability varies in an irregular way, as shown in Fig. 3.

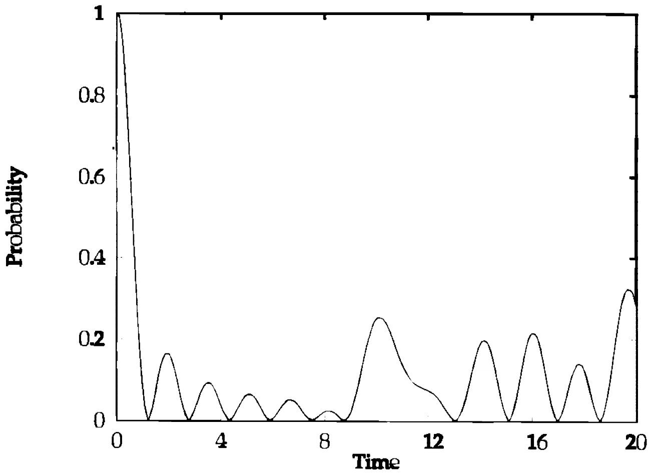

**Fig. 2.** The probability to get the initial state upon measurement for a Feynman computer with 20 gates. For small $t$, the probability decays according to a squared Bessel function.

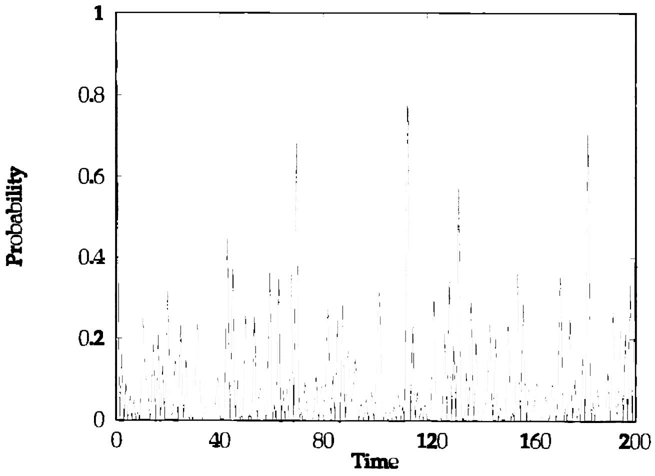

**Fig. 3.** The probability to get the initial state upon measurement for a Feynman computer with 20 gates. For larger $t$, the probability varies in a quasiperiodic and irregular way.

## 6. THE PROBABILITY TO GET A RESULT UPON MEASUREMENT

As already stated in Section 2, the probability to measure the final result is smaller than one. However, if the measurement did not yield the final result, it is possible to simply perform subsequent measurements until we know the result of the computation. In principle, it is not necessary to prepare the initial states of the computer after an unsuccessful measurement, because a proper measurement always yields a computational state, where a proper measurement is a measurement according to a Hermitian matrix with computational states as eigenvectors. After measurement, the computation continues from this state on.

However, to minimize the number of measurements, it is helpful to know the probability to measure the final result at a certain time $t$. One should then try to perform measurements if this probability is high. Since we have solved the Schrödinger equation for the computers, it is possible to calculate the probabilities.

Here, we will assume that the final state is the one after $N/2$ computational steps for a cyclic computer with $N$ states in a cycle. It is of course senseless to assume that the final state is reached after $N-1$ states: In this case, the corresponding reversible classical computer only would have to perform one backward step to yield a result. Also, it is easily possible to construct a Feynman computer with an even number of $k$ gates that works in a cycle with $N = k$ states and yields the result after $N/2$ computational steps: If gate $i$ performs a unitary transformation $\mathbf{D}_i$, then the Feynman computer applies $\mathbf{D}_0$ to the input state, then $\mathbf{D}_1$, and so on until, say, gate $\mathbf{D}_{k/2}$ produces the final state. Then we simply add gates that perform the inverse operations $\mathbf{D}^\dagger_{k/2}, \mathbf{D}^\dagger_{k/2-1}, \ldots$. After $k = N$ computational steps, the initial state is reached. The last gate is then coupled to the first, so that the entire computation is done in a cyclic way.

Of particular interest is again a computation that starts in a single computational state that codes the input to the program. In this case, the entries for the Hilbert vector $|\psi(0)\rangle$ read $\psi_k(0) = \delta_{k0}$. In Figs. 4–6, the probabilities $|\psi_{N/2}|^2$ [from (18)] to measure the result in this case are shown for a computer with a total of 20 or 100 gates. The maximal probability in the interval between $t = 0$ and $t = 100$ is 0.73 for the small computer and 0.14 for the large computer.

## 7. SUMMARY AND DISCUSSION

Upon measurement, the Feynman computer only yields a solution with a probability that is smaller than one. Other measurement results correspond to other computational states. However, a solution can always easily be identified. For example, if the control bit of the last gate of a Feynman computer is set, the state holds the solution. Thus, the probability $p(t)$ to find the solution at a certain time $t$ is of particular interest. If the "quantum programmer" knows the function $p(t)$, he will perform measurements at times where $p(t)$ is large.

It was possible to solve the Schrödinger equations and to find $p(t)$ for the finite-size Feynman computer. It is not difficult to get an intuitive

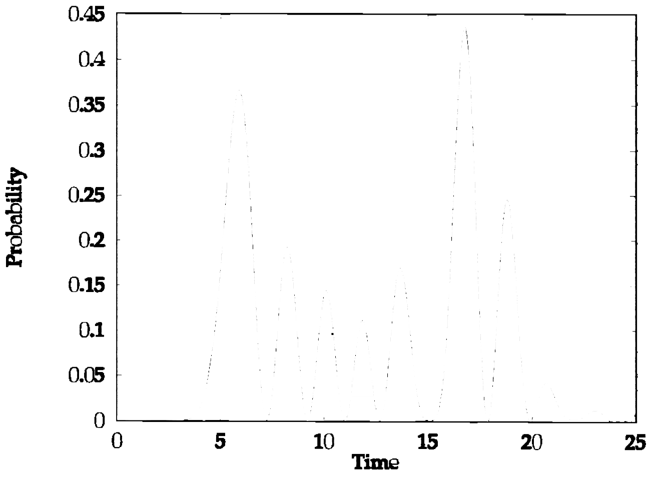

**Fig. 4.** The probability to get a result upon measurement for a Feynman computer with 20 gates and small times.

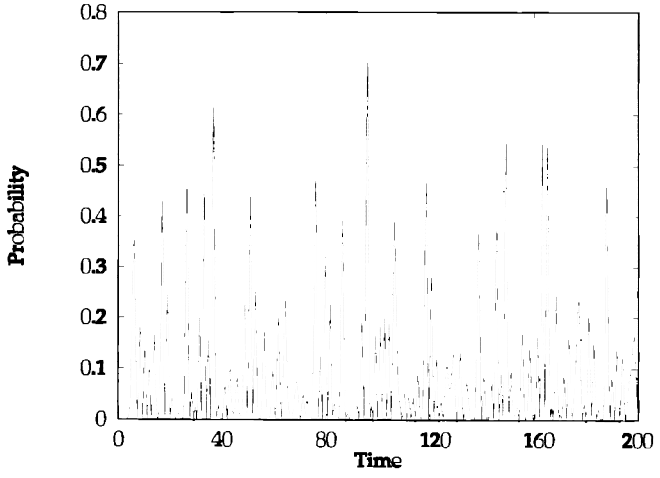

**Fig. 5.** The probability to get a result upon measurement for a Feynman computer with 20 gates and larger times.

understanding of the behavior of $p(t)$ as depicted in Fig. 6. $p(t)$ slowly increases to a maximum value as one would expect from considering the classical analogy. Then it becomes oscillatory due to the cyclic architecture of the computer. For larger times, forward- and backward-spreading wavefunctions representing final states overlap in an irregular way. At predictable times, a constructive interference of the wavefunctions may even give rise to probabilities of finding result that are close to one.

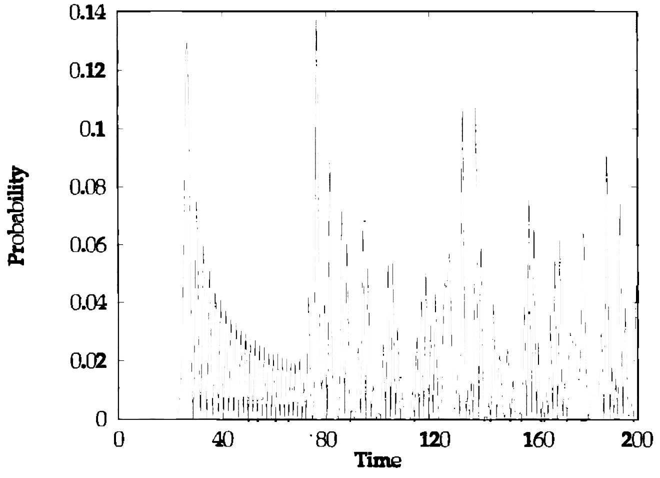

**Fig. 6.** The probability to get a result upon measurement for a Feynman computer with 100 gates.

The extent to which the findings are applicable to predicting the behavior of cellular automata or quantum Turing machines with local Hamiltonians is a subject of current research.

## APPENDIX

In this Appendix, the Taylor expansion of the discrete Fourier transform of $\zeta^{t\lambda'_k}$ is derived, where $\zeta = e^{2i\pi/N}$ and $\lambda'_k = -(N/\pi)\cos(2\pi k/N)$. This is necessary to find the solution for the Schrödinger equation for the cyclic Feynman computer.

Here, the discrete and finite Fourier transform is defined as

$$\mathcal{F}_l(x_k) = \sum_{k=0}^{N-1} x_k e^{2i\pi kl/N}$$

The power series of $\zeta^{t\lambda'_k}$ reads

$$\zeta^{t\lambda'_k} = \sum_{m=0}^{\infty} \frac{(-2it)^m}{m!} \cos^m\left(\frac{2\pi k}{N}\right) = 1 - 2it\cos\left(\frac{2\pi k}{N}\right) + \ldots \tag{A1}$$

It will now be shown by induction that

$$\mathcal{F}_n\left(\cos^m\left(\frac{2\pi k}{N}\right)\right) = \frac{N}{2^m}\sum_{k=0}^{m}\binom{m}{k}\tilde{\delta}_{n,-m+2k}$$

where the tilde denotes that the indices of the Kronecker delta have to be taken modulo $N$:

$$\tilde{\delta}_{i,j} = \delta_{i \bmod N, j \bmod N}$$

This can be written in a symbolic and more intuitive form, which will become helpful below:

$$\mathcal{F}_n\left(\cos^m\left(\frac{2\pi k}{N}\right)\right) = \frac{N}{2^m}(\mu(n,-1) + \mu(n,+1))^m \tag{A2}$$

where

$$\mu(n,k)\mu(n,l) = \mu(n,k+l) \tag{A3}$$

and

$$\mu(n,m) = \tilde{\delta}_{n,m}$$

Note the extent to which this can only be interpreted in a symbolic way. For example,

$$\mu(a, b)\mu(a, c) \neq \tilde{\delta}_{a,b}\tilde{\delta}_{a,c}$$

The operation (A3) has to be performed first:

$$\mu(a, b)\mu(a, c) = \mu(a, b + c) = \tilde{\delta}_{a,b+c}$$

The Fourier transform of the first term in (A1), which is a constant, yields a Kronecker delta:

$$\mathcal{F}_n(1) = \sum_k e^{2i\pi kn/N} = N\tilde{\delta}_{n,0}$$

The Fourier transform of a cosine yields the sum of two Kronecker deltas:

$$\begin{aligned}\mathcal{F}_n\left(\cos\left(\frac{2\pi k}{N}\right)\right) &= \frac{1}{2}\mathcal{F}_n\left(e^{2i\pi k/N} + e^{-2i\pi k/N}\right)\\ &= \frac{1}{2}\left(\sum_k e^{2i\pi k/N}e^{2i\pi kn/N} + \sum_k e^{-2i\pi k/N}\, e^{2i\pi kn/N}\right)\\ &= \frac{1}{2}\left(\sum_k e^{2i\pi k(n+1)/N} + \sum_k e^{-2i\pi k(n-1)/N}\right)\\ &= \frac{N}{2}\left(\tilde{\delta}_{n,-1} + \tilde{\delta}_{n,+1}\right)\end{aligned}$$

With the above symbolic notation, this reads

$$\mathcal{F}_n\left(\cos\left(\frac{2\pi k}{N}\right)\right) = \frac{N}{2}(\mu(n, -1) + \mu(n, +1)) \tag{A4}$$

To obtain the power series of the Fourier transform of (A1), we have to know the Fourier transform of the $m$th power of a cosine. The problem is that the Fourier transform is finite, which makes the evaluation more difficult than in the infinite case.

The Fourier transform of a product is the convolution of two Fourier transforms. With the proper normalization this reads

$$\begin{aligned}\mathcal{F}_n\left(\cos^{m+1}\left(\frac{2\pi k}{N}\right)\right) &= \mathcal{F}_n\left(\cos^m\left(\frac{2\pi k}{N}\right)\cos\left(\frac{2\pi k}{N}\right)\right)\\ &= \mathcal{F}_n\left(\cos^m\left(\frac{2\pi k}{N}\right)\right) * \mathcal{F}_n\left(\cos\left(\frac{2\pi k}{N}\right)\right)\\ &= \frac{1}{N}\sum_l \mathcal{F}_{n-l}\left(\cos^m\left(\frac{2\pi k}{N}\right)\right)\mathcal{F}_l\left(\cos\left(\frac{2\pi k}{N}\right)\right)\end{aligned}$$

If we substitute (A2) and (A4), we get

$$
\begin{aligned}
&\mathcal{F}_n\left(\cos^{m+1}\left(\frac{2\pi k}{N}\right)\right)\\
&\quad= \frac{N}{2^{m+1}} \sum_{l=0}^{N-1} (\mu(n-l,-1)+\mu(n-l,+1))^m(\mu(n,-1)+\mu(n,+1))\\
&\quad= \frac{N}{2^{m+1}} \sum_{l=0}^{N-1} \sum_{k=0}^{m} \binom{m}{k}\tilde{\delta}_{n-l,-m+2k}(\tilde{\delta}_{l,-1}+\tilde{\delta}_{l,+1})\\
&\quad= \frac{N}{2^{m+1}} \sum_{k=0}^{m} \binom{m}{k}\tilde{\delta}_{n+1,-m+2k} + \binom{m}{k}\tilde{\delta}_{n-1,-m+2k}\\
&\quad= \frac{N}{2^{m+1}} \sum_{k=0}^{m} \binom{m}{k}\tilde{\delta}_{n,-m+2k-1} + \binom{m}{k}\tilde{\delta}_{n,-m+2k+1}\\
&\quad= \frac{N}{2^{m+1}} \sum_{k=0}^{m-1} \binom{m}{k+1}\tilde{\delta}_{n,-m+2k+1} + \binom{m}{k}\tilde{\delta}_{n,-m+2k+1}\\
&\qquad + \binom{m}{0}\tilde{\delta}_{n,-m+1} + \binom{m}{m}\tilde{\delta}_{n,-m+2m+1}\\
&\quad= \frac{N}{2^{m+1}} \sum_{k=0}^{m-1} \left(\binom{m}{k+1} + \binom{m}{k}\right)\tilde{\delta}_{n,-m+2k+1}\\
&\qquad + \binom{m}{0}\tilde{\delta}_{n,-m+1} + \binom{m}{m}\tilde{\delta}_{n,m+1}\\
&\quad= \frac{N}{2^{m+1}} \sum_{k=0}^{m-1} \binom{m+1}{k+1}\tilde{\delta}_{n,-m+2k+1}\\
&\qquad + \binom{m}{0}\tilde{\delta}_{n,-m+1} + \binom{m}{m}\tilde{\delta}_{n,m+1}\\
&\quad= \frac{N}{2^{m+1}} \sum_{k=0}^{m} \binom{m+1}{k}\tilde{\delta}_{n,-m+2k-1}\\
&\qquad - \binom{m+1}{0}\tilde{\delta}_{n,-m+1} + \binom{m}{0}\tilde{\delta}_{n,-m+1} + \binom{m}{m}\tilde{\delta}_{n,m+1}\\
&\quad= \frac{N}{2^{m+1}} \sum_{k=0}^{m} \binom{m+1}{k}\tilde{\delta}_{n,-(m+1)+2k} + \binom{m}{m}\tilde{\delta}_{n,\mathrm{m}+1}
\end{aligned}
$$

$$= \frac{N}{2^{m+1}} \sum_{k=0}^{m+1} \binom{m+1}{k} \tilde{\delta}_{n,-(m+1)+2k}$$

$$= \frac{N}{2^{m+1}} (\mu(n, -1) + \mu(n, +1))^{m+1}$$

Comparing with (A2) completes the induction. With (A1) and (A2) we therefore obtain

$$\mathrm{SF}_n(\zeta^{t\lambda'_k}) = N \sum_{m=0}^{\infty} \frac{(-it)^m}{m!} [\mu(n-l, -1) + \mu(n-l, +1)]^m$$
$$= N \exp\{-it[\mu(n-l, -1) + \mu(n-l, +1)]\}$$

## REFERENCES

Benioff, P. A. (1980a). Quantum-mechanical models of Turing-machines that dissipate no energy, *Physical Review Letters*, **48**, 1581–1585.

Benioff, P. A. (1980b). A computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines, *Journal of Statistical Mechanics* **22**, 563–591.

Bennett, C. (1973). Logical reversibility of computation, *IBM Journal of Research and Development*, **17**, 525–532.

Bennett, C. (1982). The thermodynamics of computation— A review, *International Journal of Theoretical Physics*, **21**, 905–940.

Bennett, C. (1989). Time-space trade-offs for reversible computation, *SIAM Journal on Computing*, **18**, 766–776.

Biafori, M. (1993). Few body cellular automata, Thesis, MIT/LCS/TR-597.

Feynman, R. (1985). Quantum mechanical computers, *Optics News* **11**, 11–20.

Feynman, R. (1986). *Foundations of Physics*, **16**, 507–531.

Fredkin, E., and Toffoli, T. (1982). Conservative logic, *International Journal of Theoretical Physics* **21**, 219–253.

Gramss, T. (1994). A quantum Turing machine with a local Hamiltonian, Santa Fe Institute Working Paper Series 94-08-047.

Gramss, T. (1998). The theory of quantum compuation: An introduction, in *Non-Standard Computation: Molecular Computation, Cellular Automata, Evolutionary Algorithms, Quantum Computers*, T. Gramss *et al.*, Wiley-VCH Weinheim.

Grössing, G., and Zeilinger, A. (1988). Structures in quantum cellular automata, *Physica B* **151**, 366–370.

Levine, R. Y., and Sherman, A. T. (1990). A note on Bennett's time-space tradeoff for reversible computation, *SIAM Journal on Computing*, **19**, pp. 673–677.

Lloyd, S. (1996). Universal quantum simulators, *Science* **273**, 1073–1078.

Margolus, N. (1986). Quantum computation, in New Techniques in Quantum Measurement Theory, *Annals of the New York Academy of Sciences*, **480**, 487–497.

Margolus, N. (1990). Parallel quantum computation, in *Complexity, Entropy, and the Physics of Information*, W. H. Zurek, ed., Addison-Wesley, Reading, Massachusetts, pp. 273–287.

Peres, A. (1985). Reversible logic and quantum computation, *Physical Review A*, **32**, 3266–3276.

Strang, G. (1992). *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Wellesley, Massachusetts.

Toffoli, T. (1977). Computation and construction universality of reversible cellular automata, *Journal of Computer and System Sciences*, **15**, 213.