Simulating Schrödinger's Cat for two-choice game moves

Samir Lipovaca, Joe Burchfield

New York Stock Exchange, Inc.
20 Broad Street, 12th floor
New York, N.Y. 10004

slipovaca@nyse.com

Abstract

Although present quantum computers are not robust enough to be effectively used for search, it is possible to use the solution to the Schrödinger's equation of the quantum search problem to simulate quantum probabilistic behavior for two-choice game moves. Exploring the analogy between the state space in AI and the quantum mechanical Hilbert space, we shall arrive at a classical weighted-choice concept. We shall show how to construct a weighted-choice that implements the same math as the solution to the quantum search problem. To illustrate our approach we shall modify a game called Square-the-Diagonal. It is possible to generalize the weighted-choice to simulate quantum probabilistic behavior for many-choice moves.

Introduction

The quantum search algorithm solves the following problem: Given a search space of size N, and no prior knowledge about the structure of the information in it, we want to find an element of that search space satisfying a known property. How long does it take to find an element satisfying that property? Classically, this problem requires approximately N operations, but the quantum search algorithm allows it to be solved using approximately $\sqrt{N}$ operations.

Search is an important aspect of Artificial Inteligence (AI) because in many ways, problem solving in AI is fundamentally a search. An interesting use of search spaces is in games. Also known as game trees, these structures enumerate the possible moves by each player allowing the search algorithm to find an effective strategy for playing and winning the game. The above search algorithm with approximately $\sqrt{N}$ operations is quite a remarkable result. For example, if $N = 10000$ the quantum search algorithm would need approximately only 100 operations. Obviously, AI and game trees would significantly benefit from the quantum search algorithm. Unfortunately, present quantum computers are not yet capable to play a significant role in Artificial Inteligence and games development.

The fact that present quantum computers are not robust enough to be effectively used for the search does not prevent us from using the solution to the Schrödinger's equation of the quantum search problem to simulate quantum probabilistic behavior for two-choice game moves. In this paper we shall show how to construct a corresponding classical weighted-choice that implements the same math as the solution to the quantum search problem. We shall modify a game called Square-the-Diagonal to illustrate our approach.

Quantum search algorithm

We suppose that the algorithm [1] starts with the quantum computer in a state $|\psi>$ at time $t = 0$. The goal of quantum searching is to change $|\psi>$ into $|x>$. Perhaps the simplest Hamiltonian that can do such a job is :

$$H = |x><x| + |\psi><\psi| \tag{1}$$

After a time t, the state of the quantum computer evolving according to the Hamiltonian H and initially in the state $|\psi>$ is given by

$$e^{-iHt}|\psi> \tag{2}$$

where we effectively set *Planck's constant* to 1. We can restrict the analysis to the two-dimensional space spanned by $|x>$ and $|\psi>$. Performing the Gram-Schmidt procedure, we can find $|y>$ such that $|x>, |y>$ forms an orthonormal basis for this space. Thus

$$|\psi> = \alpha|x> + \beta|y> \tag{3}$$

for some α, β such that $\alpha^2 + \beta^2 = 1$, and for convenience we have chosen the phases of $|x>$ and $|y>$ so that α and β are real and non-negative. In the $|x>, |y>$ basis we have

$$H = \begin{bmatrix} 1+\alpha^2 & \alpha\beta \\ \alpha\beta & 1-\alpha^2 \end{bmatrix} = I + \alpha(\beta X + \alpha Z) \tag{4}$$

where X and Z are Pauli matrices. Using the following identity

$$e^{-i\theta \vec{n}\vec{\sigma}/2} = \cos(\frac{\theta}{2})I - i\sin(\frac{\theta}{2})(n_x X + n_y Y + n_z Z)$$

where $\vec{n} = (n_x, n_y, n_y)$ is a real unit vector in three dimensions and $\vec{\sigma}$ denotes the three component vector of Pauli matrices, we have

$$e^{-iHt}|\psi> = e^{-it}[\cos(\alpha t)|\psi> - i\sin(\alpha t)(\beta X + \alpha Z)|\psi>].$$

Simple algebra shows that $(\beta X + \alpha Z)|\psi> = |x>$, so the state of the quantum computer after a time

t is

$$|\psi(t)> = \cos(\alpha t)|\psi> - i\sin(\alpha t)|x>. \quad (5)$$

Observation of the quantum computer at time t yields the result $|\psi>$ with probability $|\cos(\alpha t)|^2 = \cos^2(\alpha t)$ and the result $|x>$ with probability $|-i\sin(\alpha t)|^2 = \sin^2(\alpha t)$. Obviously, probabilities sum to one since $\cos^2(\alpha t) + \sin^2(\alpha t) = 1$. A quite remarkable result is observation of the quantum computer at time $t = \frac{\pi}{2\alpha}$. In this case $\sin^2(\alpha t) = 1$ and the quantum computer is in the state $|x>$ with probability one. We have found a solution to the search problem! We choose $|\psi>$ to be the uniform superposition state

$$|\psi> = \frac{\sum_x |x>}{\sqrt{N}}. \quad (6)$$

Such a state can be prepared by doing a Hadamard transform. This choice gives $\alpha = \frac{1}{\sqrt{N}}$ and thus the time of observation $t = \frac{\pi\sqrt{N}}{2}$ does not depend on knowing the value of x.

Square-the-Diagonal game ("Classical" variant)

The game [2] is played by two players as follows. The first player has a choice of 1, 2, or 3 units. When he has made his choice, the second player has a choice of 1 or 2 units. Following this, the first player, knowing how the second player has chosen, again has a choice of 1, 2, or 3 units. Imagine that each choice determines respectively the length, the width, and the depth of a box. Thus the box determined by a play of the game will have dimensions x, y, and z corresponding to the three consecutive choices of the two players. The square of the length of its diagonal will be $d^2 = x^2 + y^2 + z^2$. The payoffs to each of the players will be determined by what d^2 turns out to be. Namely, if for a particular outcome d^2 leaves a reminder of either 0 or 1 upon being divided by 4, then the amount d^2 will be won by the first player. If d^2 leaves a remainder of either 2 or 3 upon being divided by 4, then the amount d^2 will be won by the second player.

The first player choice	The second player choice	The first player choice	The first player payoff
1	1	1	-3
1	1	2	-6
1	1	3	-11
1	2	1	-6
1	2	2	9
1	2	3	-14
2	1	1	-6
2	1	2	9
2	1	3	-14
2	2	1	9
2	2	2	12
2	2	3	17
3	1	1	-11
3	1	2	-14
3	1	3	-19
3	2	1	-14
3	2	2	17
3	2	3	-22

Table 1. The first player's payoffs

The first player's payoffs, including all the choices and the resulting positions are displayed in the Table1. We see that if the first player starts the game by choosing 1 unit, the second player can always be sure of winning something, for the second player can choose 1 unit, which gives the first player a choice among losing 3, losing 6, or losing 11, depending on whether he chooses 1, 2, or 3 units for the depth of the box.

Similarly if the first player starts with 3 units, the second player can again make sure that the first player will lose something, namely by choosing 1 unit, which gives the first player a choice of losing 11, 14, or 19.

The matter looks different if the first player starts by choosing 2 units. For then, if the second player continues with 1 unit, the first player can win 9 (by choosing 2 units on the final move). If the second player continues with 2 units, the first player can win as much as17. It follows that the first player can guarantee himself a win of 9 by starting with 2 units. The second player can do nothing to prevent the first player from winning at least 9.

Weighted Choice Construction

Search algorithms are of interest in AI because many problems can be reduced to simple search

problems in a state space. The state space consists of states (nodes) and operators (edges), allowing the state space to be represented as a graph. Examples range from graphs of physical spaces to massive game trees which are possible with the game of Chess. Each node in the tree is a state in the state space and the operator is simply a legal move between one state and another. The problem of search is to find a sequence of operators that transition from the start to goal state. The sequence of operators is the solution.

Similarly, in quantum mechanics, associated to any isolated physical system is a complex vector space with inner product (Hilbert space) known as the state space of the system. The system is completely described by its state vector, which is a unit vector in the system's state space. The time evolution of the state of the system is described by the *Schrödinger* equation

$$i\hbar \frac{d|\psi>}{dt} = H|\psi> \tag{7}$$

where $\hbar$ is a physical constant known as *Planck's constant.* It is common to absorb the factor $\hbar$ into H, effectively setting $\hbar = 1$. H is a fixed Hermitian operator known as the *Hamiltonian* of the closed system. The state $|\psi>$ of the system at time t_1 is related to the state $|\psi'>$ of the system at time t_2 by a unitary operator U which depends only on times t_1 and t_2,

$$|\psi'> = U|\psi>. \tag{8}$$

If we set $t_1 = 0$ and $t_2 = t$, it is easy to show that the solution to the *Schrödinger's* equation is

$$|\psi(t)> = e^{\frac{-iHt}{\hbar}}|\psi>, \tag{9}$$

where $U = e^{\frac{-iHt}{\hbar}}$. If we set $\hbar = 1$, the equation (9) is equivalent to the equation (2).

Let us call the state of the system at time $t = 0$, the start state, and the state of the system at time t, the goal state. Then, the operator U is simply the solution to the search problem. It transitions from the start to the goal state. This will become obvious if we write U as a sequence of operators. Namely, if we break up the time t into divisions of length $\varepsilon (n\varepsilon = t)$, U can be developed incrementally as follows:

$$U = e^{-\frac{iH\varepsilon}{\hbar}} e^{-\frac{iH\varepsilon}{\hbar}} \ldots e^{-\frac{iH\varepsilon}{\hbar}} (n \text{ factors}),$$

where each factor corresponds to a move between two states separated in time by ε.

Let us now look at the solution (5) to the quantum search problem in the light of this analogy

between the state space in AI and the quantum mechanical Hilbert space. The state $|\psi>$ shall be regarded as the state of the classical computer when the choice 1 has been made. Similarly, the state $|x>$ shall be regarded as the state of the classical computer when the choice 2 has been made. Then, a weighted-choice WCH is defined as

$$WCH(t) = \cos^2(\alpha t)CH1 + \sin^2(\alpha t)CH2, \qquad (10)$$

where $CH1, CH2$ represent choices 1, 2 respectively. WCH is a classical analog of the quantum mechanical solution (5). It cannot be interpreted as an ordinary mathematical equation with two variables $CH1$ and $CH2$. Instead, the equation (10) is a symbolic notation that should be interpreted using random numbers.

Let $p_1 = \cos^2(\alpha t)$ denote the probability that the choice 1 has been made. Similarly, let $p_2 = \sin^2(\alpha t)$ denote probability that the choice 2 has been made. Clearly, $p_1 + p_2 = 1$. Assuming p_1, p_2 are defined up to 2 decimal places, $100\,p_1$ is the number of outcomes leading to the occurrence of the choice 1. $100\,p_2$ is the number of outcomes leading to the occurrence of the choice 2. Outcomes leading to the choice one can be simulated, for example, by an array that has $100\,p_1$ elements equal to 1. Similarly, outcomes for the choice 2 can be simulated by an array that has $100\,p_2$ elements equal to 2. Combining all outcomes together, let WCH denote an array with 100 elements, where the first $100\,p_1$ elements are equal to 1 and remaining $100\,p_2$ elements are equal to 2. Then, the equation (10) is interpreted as follows:

$$WCH(t) = WCH[i^*], \qquad (11)$$

where a randomized sequence of the WCH array indices is created at time t and i^* is the first element of that sequence. For example, if the sequence at time t is $9\ 2\ 76\ 12\ 98 \ldots 5$, then $WCH(t) = WCH[9]$. The APL programming language provides an elegant implementation of the equation (11) as follows:

$$\uparrow WCH[100\,?100],$$

where $100\,?100$ creates a randomized sequence of the first 100 consecutive integers and $\uparrow$ returns the first WCH array element of the randomized sequence. Implicit here is that the above APL statement is executed at time t.

Square-the-Diagonal ("Quantum" variant)

In this variant of the Square-the-Diagonal game, the second player is replaced with the weighted-choice from the previous section. The game starts at time $t = 0$. Again, the first player has a choice of 1, 2, or 3 units. When he has made his choice, the weighted-choice (11) is executed and only probabilities

p_1, p_2 are revealed to the first player. Following this, the first player again has a choice of 1, 2, or 3 units. If $\sin(\alpha t) = 1$, for example at $t = \frac{\pi}{2\alpha}$, $p_2 = 1$ and the first player knowing that the choice of 2 units has been made could make his choice as in the "Classical" variant of the ordinary Square-the-Diagonal game. Similarly, if $\cos(\alpha t) = 1$, $p_1 = 1$ and the first player knowing that the choice of 1 unit has been made could make his choice as in the "Classical" variant of the game. For all other times t, knowing only probabilities p_1, p_2, the first player could only guess what was the weighted-choice (11) and based on the guess make a choice of 1, 2, or 3 units. When he has made his choice, his payoff is revealed and the game is concluded.

APL weighted-choice implementation

The APL code (IBM APL2 Version 2.0) that implements the equation (11) is displayed in the Appendix. There are 4 functions: PLAYG, Q, GAME1, and GAME2Q. PLAYG function is a top level function to execute GAME1 or GAME2Q function N times. The function Q calculates a weighted-choice. The input parameter N (a search space size) for this function is a large integer (for example 10000).

GAME1 function implements the "Classical" variant of the game with a constraint that the first player cannot make a choice of 2 units on his first move. Without this constraint the "Classical" variant would be too trivial since the second player could do nothing to prevent the first player from winning at least 9. On his first move the first player randomly make a choice of 1 or 3 units. Following this, both players make choices in order to maximize their payoffs. Looking at the Table1 it is easy to see that in this variant of the game the first player always wins 9 or 17. Displayed below is a sample output when the GAME1 was executed 10 times:

```
      10 PLAYG 1
GAME1 FIRST PLAYER PAYOFF:      1 2 2 9
GAME1 FIRST PLAYER PAYOFF:      3 2 2 17
GAME1 FIRST PLAYER PAYOFF:      1 2 2 9
GAME1 FIRST PLAYER PAYOFF:      1 2 2 9
GAME1 FIRST PLAYER PAYOFF:      3 2 2 17
GAME1 FIRST PLAYER PAYOFF:      3 2 2 17
GAME1 FIRST PLAYER PAYOFF:      1 2 2 9
GAME1 FIRST PLAYER PAYOFF:      3 2 2 17
GAME1 FIRST PLAYER PAYOFF:      3 2 2 17
GAME1 FIRST PLAYER PAYOFF:      3 2 2 17.
```

GAME2Q function implements the "Quantum" variant of the game. Again, the first player cannot make a choice of 2 units on his first move, since the second player could do nothing to prevent the first player from winning at least 9. The first move of the first player is a random choice of 1 or 3 units. Following this, the weighted-choice (11) is executed (the second player) and only probabilities p_1, p_2 are revealed to the first player. Knowing only probabilities p_1, p_2, the first player will guess what was the weighted-choice and based on the guess he will make a choice of 1, 2, or 3 units for the final move in order to maximize his payoff. When he has made the final choice, his payoff is revealed and the game is concluded. Displayed below is a sample output when the GAME2Q was executed 10 times:

```
        10 PLAYG 2
GAME2Q FIRST PLAYER PAYOFF:       1  2  1  -6
GAME2Q FIRST PLAYER PAYOFF:       3  1  1  -11
GAME2Q FIRST PLAYER PAYOFF:       3  1  1  -11
GAME2Q FIRST PLAYER PAYOFF:       3  1  1  -11
GAME2Q FIRST PLAYER PAYOFF:       3  1  1  -11
GAME2Q FIRST PLAYER PAYOFF:       1  2  2  9
GAME2Q FIRST PLAYER PAYOFF:       1  2  2  9
GAME2Q FIRST PLAYER PAYOFF:       1  2  2  9
GAME2Q FIRST PLAYER PAYOFF:       1  2  2  9
GAME2Q FIRST PLAYER PAYOFF:       1  2  2  9
```

As we can see from this sample output, the second player (a weighted-choice) has a chance to win.

Discussion

Exploring the analogy between the state space in AI and the quantum mechanical Hilbert space, we arrived at a weighted-choice concept using the solution to the quantum search problem. The weighted-choice concept is a classical analog of the equation (5). The equation (5) is a superposition of the states $|\psi>$ and $|x>$ in which is not possible to say that the system is definitely in the state $|\psi>$, or definitely in the state $|x>$. We do not experience such a superposition in the classical world - the world of our everyday's experience. Therefore, the equation (10) is interpreted using random numbers. As we showed in the Weighted Choice Construction section, a randomized sequence of the WCH array indices is created at time t and $WCH(t)$ is interpreted as the WCH array element whose respective index is the first element of the randomized sequence. This is equivalent to an observation of the quantum computer at time t which yields the result $|\psi>$ with probability $\cos^2(\alpha t)$ and the result $|x>$ with probability $\sin^2(\alpha t)$.

An obvious function of the weighted-choice concept is to simulate quantum probabilistic behavior for two-choice moves. We designed a "Quantum" variant of the Square-the-Diagonal game in which the second player was replaced by the weighted choice. The "Classical" variant of the Square-the-Diagonal game is a game of perfect information, in which each player knows exactly the position reached in the game so far. The "Quantum" variant of the game is a game of perfect information for the first player only when $\sin(\alpha t) = 1$ or $\cos(\alpha t) = 1$. For all other times t, the game is not a game of perfect information for the first player, since knowing only probabilities p_1, p_2, the first player could only guess what was the weighted-choice (11) and based on the guess make a choice of 1, 2, or 3 units. Guessing makes this variant of the game more challenging than the "Classical" variant. Also in this variant of the game, the first player can still guarantee himself a win of 9 by starting with 2 units.

It is possible to generalize the weighted-choice to simulate quantum probabilistic behavior for many-choice moves. Such a weighted-choice is defined as follows:

$$WCH(t) = p_1 CH1 + p_2 CH2 + \ldots + p_n CHn \qquad (12)$$

where $CH1, CH2, \ldots, CHn$ represent choices $1, 2, \ldots, n$ respectively and $p_1, p_2, \ldots, p_n$ are corresponding probabilities of a solution to the $Schrödinger's$ equation which is, similar to (5), a

superposition of n basis states. Obviously, $\sum_{i=1}^{n} p_i = 1$. The WCH array in this case has the first $100p_1$ elements equal to 1, the following $100p_2$ elements equal to 2, and etc. The equation (12) still has an interpretation given by (11).

Conclusions

Although present quantum computers are not robust enough to be effectively used for the search, it is possible to use the solution to the quantum search problem to simulate quantum probabilistic behavior for two-choice game moves. We showed how to construct a corresponding classical weighted-choice that implements the same math as the solution to the $Schrödinger's$ equation of the quantum search problem . To illustrate our approach we modified a game called Square-the-Diagonal. Most of times this "Quantum" variant of the game is not a game of perfect information for the first player and this makes the game more challenging than the "Classical" variant. It is possible to generalize the weighted-choice to simulate quantum probabilistic behavior for many-choice moves.

Appendix

```
 N PLAYG G;⎕IO;T;V
⍝ TOP FUNCTION TO PLAY N TIMES GAME1 OR GAME2Q
⍝ N IS NUMBER OF TIMES TO PLAY GAME1 OR GAME2Q
⍝ G IS GAME SELECTOR (G=1 PLAY GAME1, G=2 PLAY GAME2Q)
 ⎕IO←1                      ⍝ SET ORIGIN TO 1
 T←G⊃('GAME1' 'GAME2Q')     ⍝ KEEPS WHICH GAME TO PLAY
 V←N/⊂T                     ⍝ VECTOR OF N REPETITIONS OF T
 ⍎¨V                        ⍝ EXECUTE SELECTED GAME N TIMES
```

```
 R←T Q N;⎕IO;ALPHA;PCH1;PCH2;O1;O2;WCH;P2C;⎕RL
⍝ FUNCTION THAT CALCULATES WEIGHTED CHOICE
⍝ N IS AN INTEGER
⍝ T IS TIME IN SECONDS
 ⎕IO←1                  ⍝ SET ORIGIN TO 1
 ⎕RL←+/⎕TS              ⍝ SET RANDOM SEED
 ALPHA←1÷(N*0.5)        ⍝ DEFINE ALPHA
 PCH1←(2○(ALPHA×T))*2   ⍝ PROBABILITY OF CHOICE 1
 PCH2←(1○(ALPHA×T))*2   ⍝ PROBABILITY OF CHOICE 2
 PCH1←2 ROUND PCH1      ⍝ ROUND PROBABILITIES TO 2 DECIMAL PLACES
 PCH2←2 ROUND PCH2      ⍝ ROUND PROBABILITIES TO 2 DECIMAL PLACES
 O1←100×PCH1            ⍝ NUMBER OF OUTCOMES 1
 O2←100×PCH2            ⍝ NUMBER OF OUTCOMES 2
 WCH←(O1/1),(O2/2)      ⍝ WCH ARRAY REFRENCED IN THE PAPER
 P2C←↑WCH[100?100]      ⍝ WEIGHTED CHOICE
 R←PCH1 PCH2 O1 O2 P2C ⍝ RETURN PROBABILITIES, NUMBER OF OUTCOMES AND WEIGHTED CHOICE
```

```
R←GAME1;⎕IO;A;B;T;T2;P;T3;CHOICES;B;M;SP;SC;⎕RL;FP;FC
⍝ CLASSICAL GAME
 ⎕IO←1          ⍝ SET ORIGIN TO 1
 A←1 2          ⍝ SECOND PLAYER CHOICES
 B←(3 1)⍴⍳3     ⍝ FIRST PLAYER CHOICES
⍝ CONSTRUCT FIRST PLAYER'S PAYOFF MATRIX
 T←(6 2)⍴∈A∘.,B      ⍝ COMBINE SECOND PLAYER MOVES WITH FIRST PLAYER SECOND MOVES
 T←(T,[1]T),[1]T     ⍝ APPEND TWICE T TO ITSELF
 T2←,['']6/1 2 3     ⍝ FIRST PLAYER OPENING MOVES
 P←T2,[2]T           ⍝ ASSEMBLE FIRST AND SECOND PLAYER ALL MOVES
 T←4|T3←(+/P×P)      ⍝ DETERMINE REMINDERS WHEN OUTCOME DIVIDED BY 4
 T2←~(0=T)∨(1=T)     ⍝ IDENTIFY WHEN SECOND PLAYER WINS
 T3[T2/⍳⍴T3]←(¯1)×T3[T2/⍳⍴T3] ⍝ MAKE PAYOFFS NEGATIVE WHEN SECOND PLAYER WINS
 P←P,[2]T3           ⍝ FIRST PLAYER'S PAYOFF MATRIX
 CHOICES←0⍴0         ⍝ HOLDS CHOICES FOR BOTH PLAYERS
 ⎕RL←(+/⎕TS)×(+/⎕TS)                    ⍝ SET RANDOM SEED
 M←(1 3)[∈1?2]       ⍝ FIRST PLAYER RANDOM CHOICE (EXCLUDING 2 AS FIRST CHOICE)
 CHOICES←CHOICES,M   ⍝ UPDATE CHOICES
 T←(P[;1]=M)⌿P       ⍝ REDUCE PAYOFF MATRIX TO THE FIRST CHOICE OF THE FIRST PLAYER
 SP←⌊/T[;4]          ⍝ FIND SECOND PLAYER CHOICE THAT MAXIMIZES HIS PAYOFF
 SC←↑(T[;4]=SP)/T[;2]            ⍝ SECOND PLAYER RESPECTIVE CHOICE
 CHOICES←CHOICES,SC              ⍝ UPDATE CHOICES
 B←P[;⍳2]⌷⍨_MM CHOICES           ⍝ FIND WHERE CHOICES IS LOCATED IN P
 T←B⌿P                           ⍝ REDUCE PAYOFF MATRIX TO CHOICES
 FP←⌈/T[;4]                      ⍝ FIRST PLAYER CHOICE THAT MAXIMIZES HIS PAYOFF
 FC←↑(T[;4]=FP)/T[;3]            ⍝ FIRST PLAYER RESPECTIVE CHOICE
 CHOICES←CHOICES,FC              ⍝ UPDATE CHOICES
 B←P[;⍳3]⌷⍨_MM CHOICES           ⍝ FIND WHERE CHOICES ARE LOCATED IN P
 'GAME1 FIRST PLAYER PAYOFF:     ',,B⌿P ⍝ GAME ENDS


R←GAME2Q;⎕IO;A;B;T;T2;P;T3;CHOICES;B;M;⎕RL;PCH1;PCH2;O1;O2;P2C;FPP;TT;FP2
⍝ QUANTUM VERSION OF THE GAME
 ⎕IO←1          ⍝ SET ORIGIN TO 1
 A←1 2          ⍝ SECOND PLAYER CHOICES
 B←(3 1)⍴⍳3     ⍝ FIRST PLAYER CHOICES
⍝ CONSTRUCT FIRST PLAYER'S PAYOFF MATRIX
 T←(6 2)⍴∈A∘.,B      ⍝ COMBINE SECOND PLAYER MOVES WITH FIRST PLAYER SECOND MOVES
 T←(T,[1]T),[1]T     ⍝ APPEND TWICE T TO ITSELF
 T2←,['']6/1 2 3     ⍝ FIRST PLAYER OPENING MOVES
 P←T2,[2]T           ⍝ ASSEMBLE FIRST AND SECOND PLAYER ALL MOVES
 T←4|T3←(+/P×P)      ⍝ DETERMINE REMINDERS WHEN OUTCOME DIVIDED BY 4
 T2←~(0=T)∨(1=T)     ⍝ IDENTIFY WHEN SECOND PLAYER WINS
 T3[T2/⍳⍴T3]←(¯1)×T3[T2/⍳⍴T3] ⍝ MAKE PAYOFFS NEGATIVE WHEN SECOND PLAYER WINS
 P←P,[2]T3           ⍝ FIRST PLAYER'S PAYOFF MATRIX
 CHOICES←0⍴0         ⍝ HOLDS CHOICES FOR BOTH PLAYERS
 ⎕RL←(+/⎕TS)×(+/⎕TS)                    ⍝ SET RANDOM SEED
 M←(1 3)[∈1?2]       ⍝ FIRST PLAYER RANDOM CHOICE (EXCLUDING 2 AS FIRST CHOICE)
 CHOICES←CHOICES,M   ⍝ UPDATE CHOICES
 T←(P[;1]=M)⌿P       ⍝ REDUCE PAYOFF MATRIX TO THE FIRST CHOICE OF THE FIRST PLAYER
 ⍝ THIS SIMULATES TIME WHEN SECOND PLAYER REVEAL HIS CHOICE PROBABILITIES
 T2←(+/3600 60 1 0.001×3↓⎕TS)×(+/3600 60 1 0.001×3↓⎕TS)
 (PCH1 PCH2 O1 O2 P2C)←T2 Q 10000  ⍝ GET SECOND PLAYER PROBABILITIES AND CHOICE
 FPP←(1+(PCH1≥PCH2))⊃2 1           ⍝ FIRST PLAYER GUESS OF THE SECOND PLAYER CHOICE
 TT←(T[;2]=FPP)⌿T                  ⍝ REDUCE PAYOFF MATRIX TO THE FPP
⍝ SECOND CHOICE OF THE FIRST PLAYER BASED ON HIS GUESS OF THE SECOND PLAYER CHOICE
 FP2←↑(TT[;4]=(⌈/TT[;4]))⌿TT[;3]
⍝ UPDATE CHOICES WITH SECOND PLAYER CHOICE AND FIRST PLAYER GUESS OF THE SECOND ⍝
PLAYER CHOICE
 CHOICES←CHOICES,P2C,FP2
 B←P[;⍳3]⌷⍨_MM CHOICES           ⍝ FIND WHERE CHOICES ARE LOCATED IN P
 'GAME2Q FIRST PLAYER PAYOFF:    ',,B⌿P  ⍝ GAME ENDS
```

References

[1] M. A. Nielsen and I. L. Chuang, “Quantum Computation and Quantum Information”, (Cambridge University Press, Cambridge, **2000**).

[2] A. Rapoport, “Two-Person Game Theory”, (Dover Publications, Inc., Mineola, New York, 1999).