В. Ф. Очков,
Национальный исследовательский университет МЭИ, Москва

ТРЕХСТОРОННЯЯ ДУЭЛЬ В МОНТЕ-КАРЛО

Аннотация

В статье рассмотрены некоторые особенности решения на компьютере занимательных задач методом Монте-Карло.

Ключевые слова: трехсторонняя дуэль, метод Монте-Карло, Mathcad.

Контактная информация

Очков Валерий Федорович, доктор тех. наук, профессор, Национальный исследовательский университет МЭИ; *адрес:* 111250, г. Москва, Красноказарменная ул., д. 14; *телефон:* (495) 362-71-71; *e-mail:* ochkov@twt.mpei.ac.ru

V. F. Ochkov,
National Research University MPEI, Moscow

THREE-CORNER DUEL IN MONTE CARLO

Abstract

Some features of the solution some interesting problems by Monte Carlo method are described in the article.

Keywords: three-corner duel, Monte Carlo method, Mathcad.

Есть такой анекдот. Студент просыпается утром, бросает монетку и загадывает: «Выпадет орел — посплю еще на правом боку, выпадет решка — лягу на левый бок, встанет монетка на ребро — пойду в институт, а зависнет в воздухе — займусь курсовой работой».

Шутки шутками, но нам часто приходится подбрасывать монетку — реальную или виртуальную, чтобы случайным образом выбрать одну возможность из двух равновероятных. Футбольный судья, к примеру, перед матчем подбрасывает монетку, чтобы определить, какой команде отдать те или иные ворота. Шахматист перед игрой зажимает в одной руке белую пешку, а в другой — черную и предлагает сопернику выбрать «одну возможность из двух равновероятных» — определить, кто будет играть белыми фигурами, а кто — черными. Тасуя колоду карт или перемешивая костяшки домино, мы опять же отдаем себя на волю случая. А от этого, от воли случая, может зависеть очень многое... Даже сама жизнь человека, если вспомнить название этой статьи. А почему не простая (двухсторонняя), а необычная (трехсторонняя) дуэль, да еще и в Монте-Карло!? Дело в том, что задача о трехсторонней дуэли описана во многих книгах, например в [1, 2], где приводится одно из частных решений этой задачи, полученное методом логических рассуждений. Мы же рассмотрим более полное решение этой задачи на компьютере методом Монте-Карло (методом статистических испытаний): смоделируем одиночную дуэль, проведем ее достаточно большое количество раз и подсчитаем число побед в этих дуэлях каждого участника [3]. Если эти числа поделить на общее число дуэлей, то результатом и будет искомая вероятность побед.

Но начнем мы с простых задач, бескомпьютерное решение которых известно.

Давайте вспомним нашего нерадивого студента из анекдота, с помощью компьютера подбросим монетку много-много раз и подсчитаем, сколько раз выпадет орел, а сколько — решка. На рисунке 1 показана соответствующая программа для компьютера[1] — функция *ОрелИлиРешка* с аргументом *n* (число бросаний монетки), возвращающая вероятность выпадения орла или решки.

```
ОрелИлиРешка(n) :=
    Орел ← 0
    Решка ← 0
    for i ∈ 1..n
        if rnd(1) < 0.5
            Орел ← Орел + 1
        else
            Решка ← Решка + 1
    [Орел; Решка] / n

ОрелИлиРешка(10000000) = [49.95552%; 50.04448%]
```

Рис. 1

[1] Она написана в среде математической программы Mathcad Prime.

Ядро программы, показанной на рис. 1, — встроенная в Mathcad функция *rnd*, возвращающая случайное число[2] в интервале от нуля до значения аргумента функции *rnd,* в нашем случае — до единицы. Если (if) функция *rnd*(1) вернет число, меньшее 0.5, то будем считать, что выпал орел, иначе (else) выпала решка. Остается только в цикле for бросаний монетки подсчитывать, что выпало:

Орел ← Орел + 1

или

Решка ← Решка + 1.

Можно, конечно, спросить, почему в программе на рисунке 1 стоит *rnd*(1) < 0.5, а не *rnd*(1) ≤ 0.5. Замена оператора «меньше» на оператор «меньше или равно» ничего не изменит в расчете, так как выполнение условия *rnd*(1) = 0.5 равносильно падению монетки... на ребро. Вероятность генерации функцией *rnd*(1) величины 0.5 практически равна нулю. Это можно проверить, заменив оператор «меньше» на оператор «равно» в программе, представленной на рисунке 1, и подсчитав, сколько раз выпадет такой «орел» при достаточно большом числе бросаний монетки. Равенство же функции *rnd*(1) нулю или единице вообще (теоретически) невозможно («зависание монетки в воздухе»), так как функция *rnd*(1) генерирует случайные числа в *интервале* 0−1, а не на *отрезке* 0−1.

Если же нужно, чтобы выпадали целые случайные числа не в двух вариантах 0 — 1 (орел — решка, чет — нечет, да — нет и т. д.), а например, в шести вариантах, то в руки берется не монетка, а игральная кость — кубик, с пронумерованными шестью гранями. Целые случайные числа от 1 до 36 генерирует рулетка[3], а числа от 1 до 90 — мешок с «бочонками» для игры в лото в начале игры.

На рисунке 2 показана программа, с помощью которой методом Монте-Карло вычисляется вероятность выпадения чисел 2, 3, 4...12 при бросании двух игральных костей.

Центральным элементом программы, показанной на рисунке 2, также является функция *rnd*, которая возвращает случайные вещественные (нецелые) числа в интервале от нуля до 6 (аргумент функции *rnd*). Другая встроенная в Mathcad функция — функция *ceil*[4] округляет эти числа до наибольшего целого: было 0.3 — стало 1, было 5.7 — стало 6, и т. д. Вложение функции *rnd* в функцию *ceil* (*ceil*(*rnd*(6)) позволяет нам смоделировать бросание одной игральной кости — генерацию целых случайных чисел из множества 1, 2, 3, 4, 5 и 6.

В программе на рисунке 2 десять миллионов раз (как и в задаче о монетке) бросаются две игральные кости и подсчитывается, сколько раз выпало 2, 3, 4,

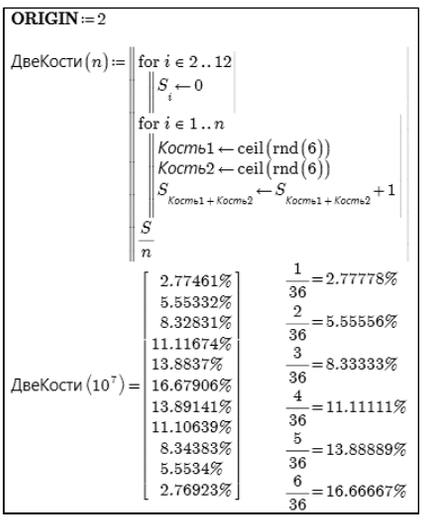

Рис. 2

5... или 12. Эти выпадения суммируются в векторе с именем *s*, первый элемент которого имеет нумерацию 2 (это определяется оператором ORIGIN:=2), а последний — 12.

На рисунке 2 показаны вероятности выпадения двойки (2.77...%), тройки (5.55...%) и т. д. после десяти миллионов бросаний двух костей. Эти вероятности можно подсчитать и без компьютера: одна кость имеет 6 граней, а две кости — 36 (6^2) «граней».

Двойка может выпасть только в одной комбинации: 1 + 1;

тройка — в двух комбинациях: 1 + 2, 2 + 1;

четверка — в трех: 1 + 3, 2 + 2, 3 + 1;

пятерка — в четырех: 1 + 4, 2 + 3, 3 + 2, 4 + 1;

шестерка — в пяти: 1 + 5, 2 + 4, 3 + 3, 4 + 2, 5 + + 1;

семерка — в шести: 1 + 6, 2 + 5, 3 + 4, 4 + 3, 5 + + 2, 6 + 1;

восьмерка — в пяти: 2 + 6, 3 + 5, 4 + 4, 5 + 3, 6 + + 2;

девятка — в четырех: 3 + 6, 4 + 5, 5 + 4, 6 + 3;

десятка — в трех: 4 + 6, 5 + 5, 6 + 4;

одиннадцать — в двух: 5 + 6, 6 + 5;

и, наконец, двенадцать (как и двойка) — в одной комбинации: 6 + 6.

Отсюда видно, что при бросании двух игральных костей наибольшая вероятность выпадения — у семерки (одна шестая), а наименьшая — у двойки и двенадцати (одна тридцать шестая).

Запуская на компьютере программы, показанные на рисунках 1 и 2, мы фактически не моделировали бросание монетки (рис. 1) или двух игральных костей (рис. 2), а... проверяли качество генератора случайных чисел программы Mathcad — добротность

---

[2] Вернее, псевдослучайное число. Дело в том, что функция *rnd* при каждом новом ее вызове возвращает один и тот же ряд случайных чисел. При бросании реальной монетки так никогда не бывает, но при бросании виртуальной монетки на компьютере так специально запрограммировано, чтобы можно было отлаживать программы. Но в языках программирования есть специальные инструменты для получения и истинно случайных чисел.

[3] Европейская рулетка. Другие варианты этого «колеса фортуны» могут иметь другое число делений. Европейскую рулетку иногда называют «чертовым колесом» — из-за того, что сумма всех чисел на ней равняется 666. Но скорее потому, что эта азартная игра разорила многих людей, не знающих меры в игре. Недаром говорят: «Хочешь выиграть в казино — купи его!».

[4] Ceil по-английски — это потолок. У нецелых чисел есть и «пол». Функция floor (пол, «напарница» функции ceil) округляет числа до наименьшего целого — было 1.2, стало 1, было минус 4.5 — стало минус 5 и т. д.

функции *rnd*[5]. Это «потребительское качество» генератора случайных чисел можно оценить и визуально. На рисунке 3 показано определение в среде Mathcad Prime методом Монте-Карло значения числа π (отношения длины окружности к ее диаметру). Задача решается так — берется квадрат, в который случайным образом «бросаются» точки и подсчитывается, сколько точек попало в круг, вписанный в данный квадрат. Отношение числа точек, попавших в круг, к общему числу точек при достаточно большом числе бросании должно стремиться к отношению площади круга (πd^2) к площади квадрата (d^2)[6].

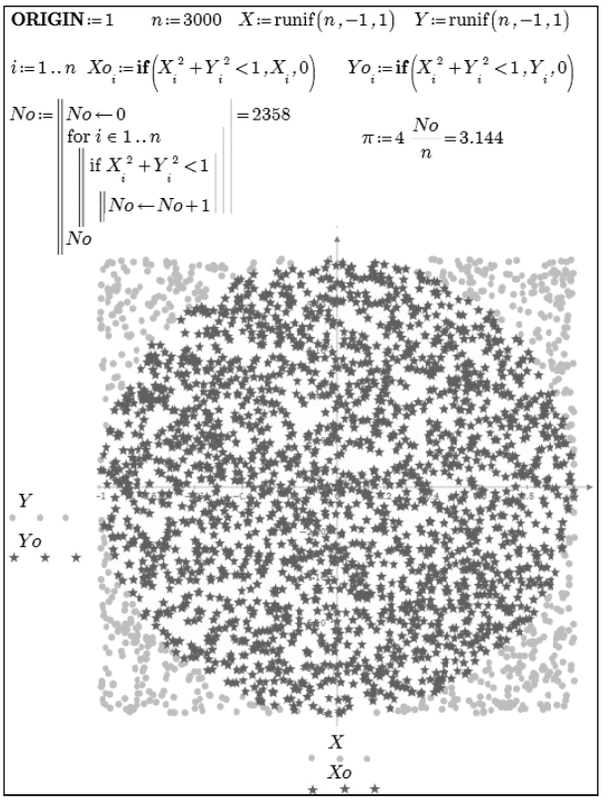

Рис. 3

На рисунке 3 показан Mathcad-расчет числа π методом Монте-Карло. Здесь задействована функция *runif* — модификация функции *rnd*. Функция *runif* возвращает вектор с *n* элементами, содержащий случайные числа в интервале, заданном вторым и третьим аргументами функции *runif*. При решении задачи о приближенном значении числа π мы сгенерировали два вектора *X* и *Y* — координаты *n* точек, брошенных случайным образом в квадрат со стороной 2. Остается только подсчитать число точек (*No*), попавших в круг, выделить координаты этих точек в вектора *Xo* и *Yo* и нарисовать их на графике. Из графика видно, что точки равномерно «размазаны» по квадрату, что свидетельствует о хорошем качестве генератора случайных чисел, встроенного в Mathcad.

Функция *rnd* также поможет нам решить на компьютере **задачу о трехсторонней дуэли**. Вот ее описание из книги [1]:

«Сэм, Билл и Джон договорились сразиться на дуэли втроем по следующим правилам:

- ***жеребьевка определяет, кто стреляет первым, вторым и третьим;***
- ***дуэлянты располагаются на одинаковых расстояниях друг от друга (по углам равностороннего треугольника);***
- ***обмениваются выстрелами по очереди, определенной жребием, пока двое не будут убиты;***
- ***очередной стреляющий может стрелять в любого из живых.***

Известно, что Сэм — снайпер и никогда не промахивается с данной дистанции, Билл поражает мишень в 80 % случаев, а Джон — в 50 %. Какова наилучшая стратегия для каждого из участников и каковы вероятности их выживания, если они следуют оптимальным стратегиям?»

В этой дуэли у Сэма (снайпер) и Билла (хороший стрелок) могут быть две тактики поведения: случайная, когда стреляющий ничего не знает о меткости соперников и целит в первого подвернувшегося — загадывает цель и подбрасывает монетку (см. рис. 1). Вторая тактика («бей в меткого»), когда дуэлянту известно о том, кто как стреляет, и он метит в соперника с наивысшими стрелковыми качествами в надежде остаться tete-a-tete с наихудшим стрелком.

Джон может следовать еще одной, третьей, «хитрой» тактике. Чтобы получить наивысшие шансы выйти победителем из дуэли, он должен намеренно стрелять в воздух, пока двое его соперников живы. Ведь очередной стреляющий, если он будет придерживается тактики «бей в меткого», будет бить не в Джона, а в другого противника. После того как Сэм или Билл будет убит, Джон должен показать все, на что он способен. В такой ситуации его шансы выжить составляют 50 %, если он остался наедине с Сэмом, и более 50 % s если с Биллом.

Давайте смоделируем эту трехстороннюю дуэль на компьютере.

Участник дуэли, прежде чем выстрелить, должен, во-первых, зафиксировать самого меткого соперника, в которого нужно стрелять, руководствуясь тактикой «бей в меткого». Для этого предназначена функция *СамыйМеткий* (рис. 4), возвращающая номер противника с наивысшими стрелковыми качествами. В ней перебором всех участников дуэли (цикл for) учитываются, естественно, только живые противники ($\text{Статус}_{\text{Цель}}$ = “жив”) и не сам стреляющий (Цель ≠ Стрелок).

Участник дуэли, придерживающийся хитрой тактики, перед выстрелом должен определить, сколько противников стреляют лучше его. Эту работу выполняет функция *Меткие* (рис. 5). В нее заложен такой же алгоритм перебора противников, как и в функции *СамыйМеткий*.

---

[5] В среде Mathcad есть другие функции, генерирующие случайные числа на заданном интервале не только равномерно, но и по другим законам распределения — например, по закону нормального распределения, график которого обычно рисуют в виде колокола или холма: в центре интервала вероятность наступления некоего события самая высокая, а на краях интервала — низкая.

[6] Так иногда оценивают площадь облаков на фотографии участка земли, сделанной из космоса: тыкают случайным образом иголкой в картинку и подсчитывают число попаданий в облако.

```
СамыйМеткий(Меткость, Статус, Стрелок) :=
    "Определение самого меткого стрелка"
    ВысшаяМеткость ← 0
    for Цель ∈ 0..2
        if Статус_Цель = "жив"
            if Цель ≠ Стрелок
                if Меткость_Цель > ВысшаяМеткость
                    СамыйМеткий ← Цель
                    ВысшаяМеткость ← Меткость_Цель
    СамыйМеткий
```

Рис. 4

```
Меткие(Меткость, Статус, Стрелок) :=
    "Определение числа противников, стреляющих лучше"
    ЧислоМетких ← 0
    for Цель ∈ 0..2
        if Статус_Цель = "жив"
            if Цель ≠ Стрелок
                if Меткость_Цель > Меткость_Стрелок
                    ЧислоМетких ← ЧислоМетких + 1
    ЧислоМетких
```

Рис. 5

Функции *СамыйМеткий* и *Меткие* в качестве аргументов имеют вектор *Меткость*, вектор *Статус* и скаляр *Стрелок*.

Функция *Победитель* (рис. 6) возвращает номер победителя в одиночной дуэли. При этом учитываются меткость и тактика каждого участника дуэли (два вектора-аргумента функции *Победитель*).

Что происходит в функции *Победитель*?

В начале дуэли все участники живы: все три элемента вектора *Статус* принимают значение "*жив*", а переменная *Убийство* (счетчик числа убийств) обнуляется. Далее проводится жеребьевка: определяются первый стреляющий[7] (переменная *Стрелок*) и направление очередности выстрелов. Если перемен-

```
Победитель(Меткость, Тактика) :=
    "Моделирование трехсторонней дуэли"
    [Статус ← ["жив"; "жив"; "жив"]   Убийство ← 0]
    [Стрелок ← floor(rnd(3))   Очередь ← if(rnd(1) > 0.5, 1, −1)]
    while 1
        if Тактика_Стрелок = "случайная"
            while 1
                Цель ← floor(rnd(3))
                if Цель ≠ Стрелок
                    if Статус_Цель = "жив"
                        break
            if Меткость_Стрелок > rnd(1)
                [Статус_Цель ← "убит"  Убийство ← Убийство + 1  if Убийство = 2
                    return Стрелок]
        if Тактика_Стрелок = "бей в меткого"
            Цель ← СамыйМеткий(Меткость, Статус, Стрелок)
            if Меткость_Стрелок > rnd(1)
                [Статус_Цель ← "убит"  Убийство ← Убийство + 1  if Убийство = 2
                    return Стрелок]
        if Тактика_Стрелок = "хитрая"
            Цель ← СамыйМеткий(Меткость, Статус, Стрелок)
            if Меткие(Меткость, Статус, Стрелок) < 2
                if Меткость_Стрелок > rnd(1)
                    [Статус_Цель ← "убит"  Убийство ← Убийство + 1  if Убийство = 2
                        return Стрелок]
        while 1
            Стрелок ← Стрелок + Очередь
            [if Стрелок > 2   Стрелок ← 0   if Стрелок < 0   Стрелок ← 2   if Статус_Стрелок = "жив"
                break]
```

Рис. 6

[7] Тут бы пригодилась монетка (см. рис 1), но не с двумя, а с тремя сторонами. Такой монетки в природе нет, но мы ее смоделируем оператором *floor*(*rnd*(3)). Кстати, «монетка» с четырьмя сторонами — это тетраэдр, а «монетка» с шестью сторонами-гранями — это игральная кость (см. рис. 2).

ная *Очередь* будет равна единице, то очередность идет в таком направлении ...0→1→2→0→1→2... (...Сэм→ →Билл→Джон→Сэм→Билл→Джон...), если минус единице — в таком ...0→2→1→0→2→1... (...Сэм→ →Джон→Билл→Сэм→Джон→Билл...).

Математическая модель дуэли опирается на цикл с выходом из середины (while 1... return...): дуэль продолжается до тех, пока не будут сделаны два результативных выстрела — пока переменная *Убийство* не станет равной 2. В тело цикла while вложено три ветви, определяемые тактикой очередного стреляющего: «случайная», «бей в меткого» и «хитрая». В каждой ветви расчета определяется *Цель* — либо случайный, либо самый меткий противник, которого убивают ($Статус_{Цель} \leftarrow$ "убит"), если, во-первых, не промахиваются $Меткость_{Стрелок} > rnd(1)$ и, во-вторых, не стреляют намеренно в воздух. Второе имеет место при хитрой тактике стреляющего ($Тактика_{Стрелок}$ = "хитрая") и если метких противников более одного.

Определение следующего стреляющего ведется также в цикле с выходом из середины (while 1 ... break): цикл прерывается, когда, перебирая очередь, отмеченную выше (...0→1→2→0→1→2... или ...0→2→1→ →0→2→1...), «натыкаются» на живого участника.

Функция *Победитель* возвращает непредсказуемое целочисленное значение 0, 1 или 2, так как в ней вызывается уже рассмотренная нами встроенная в Mathcad функция *rnd*, которая возвращает псевдослучайное число в интервале от нуля до значения аргумента функции *rnd*. Этот аргумент у нас равен либо единице (случайный выбор очередности выстрелов и имитация выстрела с заданной вероятностью попадания, пропорциональной меткости стреляющего), либо трем (случайный выбор первого стреляющего).

Функция *ВероятностьПобеды* (рис. 7) возвращает вектор, элементы которого — это отношение числа побед каждого участника дуэли к общему количеству дуэлей (третий аргумент функции *ВероятностьПобеды*; два первых аргумента-вектора — это параметры дуэлянтов: их меткость и тактика), т. е. вероятность победы.

Теперь, когда все необходимые функции сформированы, можно проводить *статистические испытания* (рис. 7) и фиксировать вероятности побед участников дуэли, исходя из их меткости и тактики. Если увеличивать число дуэлей, то, набравшись терпения, можно получить результат, близкий к теоретическому.

Задача о трехсторонней дуэли, как мы уже отметили, приводится во многих книгах. И что интересно — она там решается неверно, вернее, не совсем верно. Априори считается, что в этой дуэли самый слабый стрелок Джон имеет наихудшие шансы выжить. Но если он немного подумает (хитрая тактика), то вероятность выйти победителем у него становится самой высокой (52.222... %).

Но наше решение (рис. 7) говорит о том, что у Джона и так самые высокие шансы выжить (44—46 %). Начиная хитрить, он мало что выигрывает, но подводит Билла — своего товарища по несчастью стрелять хуже Сэма.

Откуда такая ошибка в постановке задачи? Дело в том, что у дуэлянтов есть еще одна тактика. Если участники дуэли ничего не знают о стрелковых качествах соперников, то они бьют в первого попавшегося. Здесь вероятность побед можно подсчитать сразу без компьютера: Сэм — 43.48 % (1 / (1 + 0.8 + + 0.5)), Билл — 34.78 % (0.8 / (1 + 0.8 + 0.5)) и Джон — 21.74 % (0.5 / (1 + 0.8 + 0.5) или 100 − 43.48 − 34.78). Но мы составили программу и для этого случая.

Задачу подправили, теперь ее можно развить.

Подсчитанная нами вероятность побед относится к ситуации, когда еще не проводилась жеребьевка по очередности выстрелов: переменная *Очередь* у нас случайно равна либо единице, либо минус единице.

Но после жеребьевки шансы Сэма и Билла резко меняются. Дела Билла становятся совсем уж плохи (10—12 %), если Джон после своего намеренного промаха передает право выстрела не ему (*Очередь* = = −1), а Сэму (*Очередь* = 1). И наоборот: Сэм может потерять свои 30 %, если после намеренного промаха Джона Билл будет стрелять в Сэма. У Джона вероятность победы (52.2(2) %) не зависит от очередности выстрелов.

Можно придумать и проанализировать четвертую тактику ведения дуэли: Билл и Джон *сговариваются* целить в Сэма, убить его, раз он такой меткий, а уж потом выяснять отношения между собой. Инициатором такого сговора, как понимает читатель,

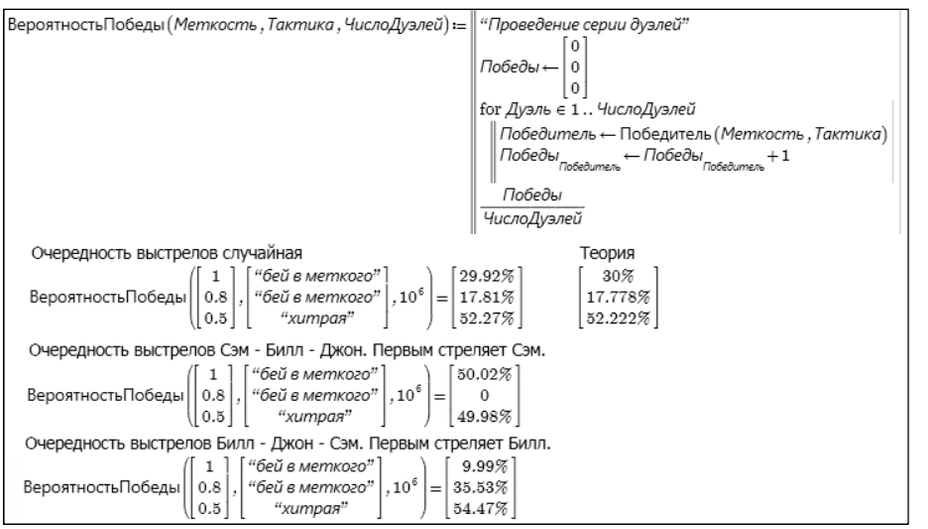

Рис. 7

скорее всего, будет Билл. Джон пойдет на него, если не смоделирует дуэль на компьютере и не узнает, что из этого может получиться.

Еще одно задание читателю: доработать программу имитации дуэли так, чтобы она была пригодна для дуэли с любым числом участников.

Наша модель — не такая уж оторванная от жизни. Дуэли в чистом виде сейчас, к счастью, не проводятся. Но, кстати, можно дуэль проводить в формате игры пейнбол, когда соперники стреляют друг в друга шариками с краской. Какое-то подобие дуэли со сговором участников наблюдается на рынках, включая финансовые. Кровь там не льется, но случаются инфаркты, лопаются компании, банки, разоряются люди и даже целые страны (Греция, Испания, Италия, Кипр, Исландия и т. д., если говорить о современной ситуации).

Теория игр, тактика поведения участников — это не только интересная, но и очень полезная штука. Недаром в 1998 г. лауреатами Нобелевской премии по экономике стали ученые, применившие теорию игр к анализу работы финансовых бирж.

И все-таки сама модель чересчур искусственна. Что такое меткость дуэлянта и как ее определить? Проводить реальные статистические испытания? Но одно дело стрелять по мишеням, а другое — целить в живого человека. На дуэлях, как правило, не убивают наповал, а ранят с различной степенью тяжести. Подстреленный дуэлянт, если хватало сил и злости, стрелял в противника (дуэль Пушкина и Дантеса, например). Попытки «приземлить» задачу о дуэлях неизбежно потребуют привлечения аппарата *теории нечетких множеств* (ТНМ).

Меткость дуэлянта — величина нечеткая, «размытая». Никто и нигде не измеряет ее числами, а только оценивает категориями (лингвистическими константами): «мазила», «хороший стрелок», «снайпер» и т. д. Статус дуэлянта — это никакая не булева переменная. Вспомним «консилиум врачей» у лежащего без чувств Буратино: «Пациент скорее мертв, чем жив», — «Нет, пациент скорее жив, чем мертв»...

Литература

1. *Гудман С., Хидетниеми С.* Введение в разработку и анализ алгоритмов / пер. с англ. М.: Мир, 1975.

2. *Мостеллер Ф.* Пятьдесят занимательных вероятностных задач с решениями: 2-е изд. / пер. с англ. М.: Наука, 1975.

3. *Очков В. Ф.* Mathcad 14 для студентов и инженеров: русская версия. СПб.: БХВ-Петербург, 2009.