УДК 681.3.07

С.Б. Макась, аспирант

ИМИТАЦИОННАЯ МОДЕЛЬ НАДЕЖНОСТИ ДИНАМИЧЕСКОГО ОПЕРАТИВНОГО ЗАПОМИНАЮЩЕГО УСТРОЙСТВА

This article is about creation simulation model of reliability the DRAM memory by means of object-oriented COM technology. Considered aspects program implementation algorithms of statistical simulation for solving problem reliability of devices of the RAM.

Введение.

Программная реализация модели надежности динамического оперативного запоминающего устройства (ДОЗУ) – это имитационная модель, построенная по законам вероятностно-событийного моделирования. Программной реализацией алгоритма событийного моделирования работоспособности элементов ДОЗУ является набор объектов, работающих на основе выработки случайных чисел, статистической обработки результатов моделирования и др.

Реализация модели состоит из нескольких стандартных этапов, где математические модели, алгоритмы и объекты функционирования должны удовлетворять определенным требованиям.

Элементы событийной модели надежности.

Модель надежности ДОЗУ относится к классу стохастических моделей, т.е. моделей, отображающих вероятностные процессы и события. Следовательно, алгоритм - реализующий событийную модель, воспроизводит процесс функционирования системы во времени. Главный принцип здесь – это сохранение логической структуры и последовательности протекания во времени моделируемых элементарных состояний системы, что позволяет получить сведения о состояниях процесса и системы в определенные моменты времени.

Данный метод наиболее точно подходит для решения задач изучения надежности сложных систем, какими, безусловно, являются и ДОЗУ. В реальных условиях на функционирование сложных систем – вычислительных систем, систем оперативного хранения информации, систем передачи информации и т.д. оказывают влияние случайные факторы, которые при математическом моделировании рассматриваются либо как случайные входные (управляющие) сигналы, либо как неконтролируемые воздействия внешней среды. К таким факторам можно отнести случайные интервалы времени между поступлением заданий в вычислительную систему, обращения к системе оперативного хранения и передачи информации, возникновение критической ситуации в процессе функционирования. При построении модели возникает необходимость имитации случайных факторов по заданным законам распределения вероятностей.

Обобщенно характеристики событийной модели можно выразить так:

1) все множество событий можно разбить на небольшое число L типов событий:

$$\{A^{(1)},.....,A^{(N)}\} \subset \{A_j^{(i)}\},\ (j = \overline{1,m}\ , i = \overline{1,N}\ ),\ L << \sum_{i=1}^{N} m_i\ ,$$

где N – количество элементов системы; m – количество значимых действий, приводящих к изменению поведения i-го элемента системы, выбранное из всей последовательности действий;

2) для каждого типа событий определена последовательность действий, приводящая к изменению состояния системы;

3) определены условия перехода от одного события к другому для всех типов событий;

4) интервалы времени между **последовательными наступлениями событий**, сами события и процессы – случайные величины с известными законами распределения вероятностей.

Следовательно, статистическая модель надежности ДОЗУ реализуется посредством принципов и алгоритмов вероятностно-событийного моделирования и представляет собой некоторые множества случайных элементов с известными и получаемыми законами распределения. Следовательно, при построении данных моделей необходимо уделять достаточное внимание "производству" случайных элементов.

Принципы и особенности моделирования случайных элементов

Моделирование на ЭВМ случайного элемента подчиняется двум основным принципам [1]:

1) сходство между случайным элементом оригинала Z^* и его моделью Z состоит в близости вероятностных законов распределения или числовых характеристик;

2) всякий случайный элемент Z определяется как некоторая борелевская функция от простейших случайных элементов, базовых случайных величин (БСВ) [1].

Таким образом, задача моделирования произвольного случайного элемента Z^* разбивается на две подзадачи:

1) моделирование на ЭВМ независимых БСВ $\alpha_1,....\alpha_r$;

2) нахождение функции $f(\cdot)$ такой, чтобы случайный элемент Z обладал требуемым вероятностным законом распределения и числовыми характеристиками.

Поэтому моделирующий алгоритм состоит из двух блоков (рис. 1).

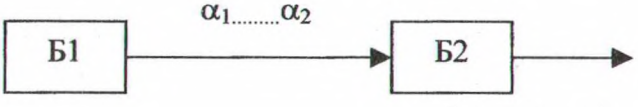

Рис. 1: Б1 – блок моделирования БСВ (общий для всех Z); Б2 – блок функционального преобразования $f(\cdot)$ БСВ для различных законов распределения вероятностей

При реализации на ЭВМ первого блока "производства" случайного элемента используются программные датчики БСВ – алгоритмы, позволяющие по запросу получать реализацию БСВ или несколько независимых реализаций $a_1,...a_r$ БСВ α. Ввиду использования в ЭВМ дискретных чисел датчики позволяют получать случайную квазиравномерную величину [4].

Любая случайная величина, получаемая с помощью датчика, должна как можно точнее соответствовать критерию Вейля [6]: чтобы ряд $\alpha_1,....\alpha_r$ был распределен равномерно в интервале от 0 до 1, необходимо и достаточно, чтобы для любой интегрируемой по Риману функции $f(x)$ было справедливо соотношение $P\{=Mf(x)\}=1$. Это соотношение выражает свойство, называемое эргодичностью и заключающееся в том, что среднее по реализациям псевдослучайных чисел равно среднему по всему их множеству с вероятностью 1. Таким образом, Вейль доказал, что эргодичность гарантирует отсутствие экзотичности в поведении последовательности.

Однако эти результаты далеки от практики получения псевдослучайных рядов чисел. Дело в том, что теорема Якоби относится к действительным числам x и y, которые не могут быть использованы при вычислениях, потому что иррациональные действительные числа требуют для своей записи бесконечного числа знаков. Попытки замены настоящего иррационального числа его приближением на ЭВМ для генерации псевдослучайной последовательности опасны, так как получаемые последовательности оканчиваются циклами с коротким периодом.

В наше время известно несколько алгебраических датчиков БСВ, но построению программной реализации по-прежнему уделяется достаточно внимания ввиду динамичного усовершенствования ЭВМ. Рассмотрим несколько основных алгоритмов построения программных датчиков БСВ и особенности их реализации с применением основных языков программирования.

1. Мультипликативный метод (метод вычетов).

Псевдослучайная величина получается вычислением рекуррентной формулы

$$a_i = a_i^* / M,\ \ a_i^* = (\beta a_{i-1}^*) \bmod M\ (i = 1,2,...),$$

где β, M, a_0^* – параметры датчика, натуральные числа; M – модуль; $a_0^* \in \{1,....M-1\}$ – стартовое значение. Данный алгоритм, начиная с некоторого i_0, образует цикл, который повторяется бесконечное число раз. Период последовательности $T \le M-1$ [1].

В [5] приведены практические рекомендации для реализации метода вычетов на ЭВМ.

$$a_i = A(P) \cdot a_{i-1} \bmod m,\ A(P) = 5^{2P+1}\ и\ m = 2^N\ \ L = 2^{N-2},$$

где m, N – целые числа, а P=1,2,3….. L – период последовательности.

При использовании данного датчика БСВ необходимо подобрать значение параметров и стартовое значение [6], которые однозначно определяют период. Программа, созданная на основе этого алгоритма, показывала следующие результаты.

При задании стартового в виде целого числа из диапазона $a_0^* \in \{1,....M-1\}$, при любых N и P уже на первых нескольких итерациях вызывалась исключительная ситуация – переполнение операционного стека. Данное переполнение вызывает расчет выражения $A(P) \cdot a_{i-1}$ и не зависит от платформы реализации (*C++, Pascal, Basic*). И хотя эти языки программирования имеют отличия в работе со стеком, успешного результата достичь не удалось. В случае реализации алгоритма на *Pascal* или *Basic* стек операции с целыми числами имеет предел 2^{31}, что эквивалентно предельно допустимому значению типа *unsigned long* [2].

При внесении операции вычета и приведения к единице в саму рекуррентную формулу: $a_i = (A(p) * a_{i-1} / m) - (unsigned\ long)(A(p) * a_{i-1} / m)$, и задания начального значения $a_0^* \in [0,....1)$ удалось получить следующие результаты:

- устойчивые периоды "производства" БСВ при подборе значений P и N (когда среднее по реализации примерно совпадает со средним по множеству с погрешностью < 0.01) имели длину 127, 1024, 1225, 8192, 32768 [6][4];
- результаты в некоторой степени совпали с характеристиками программного типового датчика псевдослучайной величины *Rand()*.

Известно, что датчик *Rand()* в *C++*, *Rnd(N) и Randomize(M)* в *Pascal* и *Basic* работает по принципу вычетов, где для инициализации используется текущее время в миллисекундах или явно заданное значение. Эти датчики генерируют псевдослучайные последовательности целых двухбайтных чисел с периодом, определяемым разрядностью процессорного слова (два или четыре байта). В таблице приведены значения коэффициентов мультипликативного датчика, используемого языками программирования *Pascal, Basic* и *C++*.

Таблица

Q	M	β	α^*_0	T
16	215=32768	28 + 3 = 259	259	M/4 = 8192
32	231=2147483648	216+3 = 65539	65539	M/4 = 536870912

Из таблицы видно, что если при моделировании случайного элемента в качестве датчика БСВ выступит типовой программный датчик, то качество блока Б1, а соответственно, и Б2 (см. рис. 1) будет определяться следующими факторами:

- Период последовательности сильно зависит от разрядности целого числа (16 или 32 бита), используемого в системе.
- Несмотря на то, что уже довольно давно используются ЭВМ с 32-битным процессором, многие системы разработки, компиляторы и интерпретаторы для совместимости "назад" продолжают использовать только младшие 2 байта (16 бит) слова.

2. Метод линейных смешанных формул.

В этом случае псевдослучайная последовательность вычисляется по общей формуле

$$a_i = a_i^* / M, \quad a_i^* = (\beta_1 a_{i-1}^* + ... \beta_p a_{i-p}^* + c) \bmod M \quad (i = 1,2,....).$$

Параметры датчика: p – порядок; $a_{-p+1}^*, a_{-p+2}^*, ..., a_{-1}^*, a_0^*$ – начальные значения; $\beta_1, ..., \beta_p$ – множители; c – приращение; M – модуль. Период датчика $T \le M^p - 1$ растет с увеличением M и p.

Практическая реализация метода линейных смешанных формул представляет собой линейные сопряженные генераторы (ЛСГ) порядка s=1,2,…[4]. В этом случае программа реализует следующий алгоритм:

$$s = 1.\ a_i^* = k_i a_{k-1}^* (\bmod p),$$

$$s = 2.\ a_i^* = k_1 a_{i-1}^* + k_2 a_{i-2}^* (\bmod p).$$

Период неповторяющейся последовательности: $T=p^s-1$, k_i и p – найдены экспериментально и имеют определенные значения для каждого порядка.

В своем классическом виде реализация данных алгоритмов при помощи *Pascal* и *Basic* вызывает следующие затруднения: переполнение в стеке операций при вычислении части выражения $k_i a_{k-1}^*$ или $k_1 a_{i-1}^* + k_2 a_{i-2}^*$; выравнивание результата стандартной функцией вычисления остатка по границе целого числа, и как результат слева от операции вычисления остатка всегда будет число, меньшее 36768. Это означает, что из всего предполагаемо доступного диапазона случайных целых чисел от 1 до p-1 [3] реально

будут использоваться только первые 36768. Особенности компилятора *Pascal* и *Basic* при вычислении арифметических выражений заключаются в четком контроле типа операнда, а также преобразование типа результата к типу результата операции. Это, в свою очередь, приводит к выравниванию в стеке операций для целых чисел по границе типа *unsigned long int* и генерации исключительной ситуации – *Stack overflow* [2].

Применение для реализации алгоритма ЛСГ компилятора *C* и *C++* дает другие результаты. Так, компилятор *C* при выполнении унарных операций тип результата операции приводит к типу операндов по правилам приведения с *type*. Другими словами, при реализации ЛСГ 1, 2 и далее порядка в целых числах, в отличие от *Pascal* и *Basic*, получим скрытую логическую ошибку, заключающуюся в неявном приведении типа результата операции $k_i a^*_{k-1}$ или $k_1 a^*_{i-1} + k_2 a^*_{i-2}$ к *unsigned long*. Исходя из этого, можно предложить три способа реализации алгоритма.

1) Для исключения ошибки переполнения стека в компиляторах *Pascal* и *Basic* можно данный алгоритм смоделировать с помощью циклического алгоритма на вещественных числах, где условие выхода будет $\frac{k_i * a^*_{i-1}}{p} < p$. Данный способ приводит к необоснованным затратам аппаратных ресурсов.

2) В компиляторе *C* и *C++* есть поддержка вышеописанных операций для чисел вещественных типов *float* и *double*. Использование этих функций позволяет реализовать данный алгоритм в «чистом виде», без каких-либо преобразований и допущений для особенностей платформы реализации.

3) Для реализации алгоритма линейных смешанных формул на базе компилятора *Pascal* и *Basic* можно применить, как и в случае метода вычетов, использование в алгоритме вещественных чисел из диапазона $a^*_0 \in [0,....1)$ и непосредственное приведение результата к 1. Результат получался как дробная часть от деления $k_i a^*_{k-1}$ или $k_1 a^*_{i-1} + k_2 a^*_{i-2}$ на модуль *p*. При таких преобразованиях полный цикл последовательности имеет длину $T < 5000$ [6]. Как было получено из результатов вычислительного эксперимента, последовательность псевдослучайных чисел всегда оканчивается циклической бесконечной зависимостью. Изменение коэффициентов в большую сторону с целью увеличения периода приводит к вырождению последовательности [8].

Применение объектно-ориентированной технологии в построении событийных моделей надежности.

В теоретическом варианте решения задачи моделирования надежности, «производство» БСВ и ее дальнейшее борелевское преобразование в выходную величину с заданным законом распределения должны быть полностью независимыми.

Однако при моделировании на ЭВМ приходится сталкиваться с проблемой невозможности выполнения полностью параллельных вычислений для каждого элемента модели. На данном этапе существует три основных метода решения проблемы: 1) введение понятия дискретного «модельного времени» [3]; 2) выполнение вычислений для одних элементов во время отсутствия активности других элементов (*Idle* - технология); 3) использование объектно-ориентированной и многопоточной технологии;

Наиболее рациональное и универсальное решение данной задачи состоит в применении концепции *COM*[1] технологии. Ввиду достаточно узкой специализации решаемой задачи можно воспользоваться упрощенной моделью, т.н. узкоспециализированной псевдо-*COM* технологией, с сохранением всех законов и принципов организации *COM*. Это позволит получить набор независимых объектов-сервисов, каждый из которых реализует собственный вычислительный процесс, и объектов-интерфейсов, которые осуществляют обмен данными и синхронизируют работу объектов-сервисов.

Рассмотрим общую принципиальную схему объектно-ориентированной реализации модели надежности. Здесь весь процесс моделирования разбивается на отдельные "независимые" задачи, каждую из которых реализует объект-сервис. Накопление информации, полученной в процессе моделирования, обработка результатов моделирования и начальная инициализация осуществляется классом объектов-концентраторов. Обмен данными между сервисами, сервисами и концентраторами, синхронизацию работы объектов-сервисов осуществляют объекты-интерфейсы.

Согласно концепции *COM*, объекты-сервисы, а также объекты-концентраторы не "знают" о существовании друг друга и не "общаются" непосредственно. Обмен данными происходит при помощи объектов-интерфейсов, которые поставляют свои сервисы обмена данными клиентам. В данной задаче всех клиентов можно условно разделить на две категории: объекты серверы и объекты клиенты, которые обмениваются данными посредством сервисов объектов-интерфейсов.

Исходя из принципов псевдо-*COM* модели, решение задачи моделирования случайных дискретных величин с заданными вероятностными законами распределения и числовыми характеристиками состоит в следующем. Сначала создается и инициализируется начальными значениями объект-концентратор. В качестве начального значения в данном случае выступает P_i, дискретный ряд чисел диапазона [0..1] с суммой, равной 1, где каждое число характеризует примерную вероятность события A_i, приводящего к изменению состояния моделируемой системы. Вторым этапом создается и инициализируется объект-сервис, задача которого состоит в "производстве" БСВ одним из описанных в предыдущих пунктах способом. Для связки объекта сервиса с объектом-концентратором необходимо создать из набора классов интерфейсов экземпляр объекта-интерфейса, который и будет осуществлять обмен данными и управление вычислительным процессом. Схематически это можно представить следующим образом (рис. 2).

Здесь интерфейс данных представляет собой коллекцию ссылок на общие области данных в глобальной зоне, доступ к которым всем потокам процесса равнозначный и одинаковый по приоритету доступа. Командный интерфейс в данном случае состоит из набора объектов «событий», которые могут находиться в «свободном» и «занятом» состоянии. Объект интерфейса, согласно внешним сигналам, может изменять состояние объектов «событий» и тем самым управлять потоками вычислений в объектах клиента и сервиса.

---

[1] COM – Component Object Model – стандартный механизм, включающий интерфейсы, с помощью которых одни объекты предоставляют свои сервисы другим объектам [7].

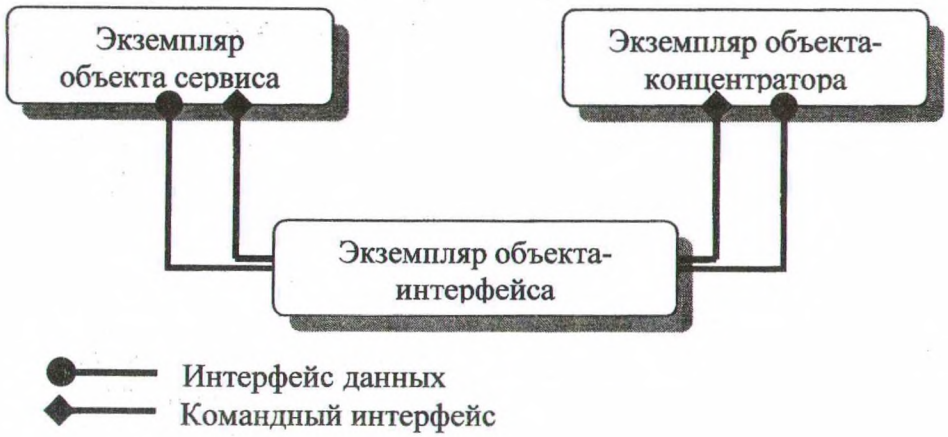

Рис. 2

Практическая проверка действительности объектного метода моделирования состояний системы заключается в том, чтобы событие A_i произошло с моделируемой вероятностью P_i^*. Реализация события A_i должна выполняться с надежностью 95% и удовлетворять условию $P \in (P^* - \varepsilon, P^* + \varepsilon)$, т.е. модель воспроизводит состояния системы с 95%-ной вероятностью и погрешностью ε.

Моделирование случайных дискретных величин с заданными вероятностными законами распределения и числовыми характеристиками.

В практическом применении данная программа использовалась для моделирования двух интервалов вероятностей редких событий P_i^1 и P_i^2, которые имеют следующий вид: $P_i^1 = \{0.1, 0.1, 0.15, 0.25, 0.4\}$ и $P_i^2 = \{0.4, 0.1, 0.15, 0.1, 0.05, 0.1, 0.05\}$.

Параметры моделирования: моделируемый ряд событий P_i^1; алгоритм "производства" БСВ – метод линейных смешанных формул 1-го и 2-го порядка; количество экспериментов 1900; циклов накопления – 2000. При выборе в объекте-сервисе в качестве алгоритма "производства" БСВ метода линейных смешанных формул 2-го порядка результат носил аналогичный характер. При моделировании последовательности событий, представленной рядом P_i^2, при прочих равных условиях, полученные результаты были в достаточно большей степени идентичны результатам, полученным при моделировании последовательности P_i^1.

В результате в двух различных вариантах моделируемой последовательности модельная величина P_i^* имеет довольно большую разбежку (в среднем ±0.15) и доверительный интервал (в среднем ±0.04), что свидетельствует о достаточно большой погрешности воспроизведения реальной последовательности событий.

Наличие в результатах значительной погрешности обуславливается особенностями распределения процессорного времени между объектом-сервисом и объектом-

клиентом, а также соотношения приоритетов потока-клиента и потока-сервиса. Конкретно это проявляется в следующем. Объект-клиент (в нашем случае объект класса *Application* – приложение) начинает процесс моделирования. Далее, используя объект-сервис в качестве "производителя" БСВ, получает посредством объекта-концентратора модельное распределение вероятностей событий (состояний) моделируемой системы.

При условии запуска процесса с условиями приоритетов по умолчанию происходит следующее: поток-клиент, имея равный приоритет со всеми остальными потоками, но имея атрибут основного потока процесса, почти монопольно захватывает процессорное время, отведенное для всего процесса. В результате объект-концентратор для анализа получает последовательность чисел, состоящую из 5–17 последовательно идущих друг за другом одинаковых чисел. Это соответствует тому, что на 5–17 тактов с запросами потока-клиента поток объекта-сервиса делал всего лишь один такт моделирования БСВ.

Для устранения данного дефекта программной модели можно поступить следующим образом: 1) изменить приоритеты потоков клиента и сервиса; 2) воссоздать искусственно периоды отсутствия активности потока клиента при равных приоритетах. Применение первого способа приводит к значительно лучшим результатам, но время моделирования и нагрузка на систему возрастают в несколько порядков. Второй способ основан на том, что в моделирующей системе события идут не непрерывным потоком, а через определенные дискретные промежутки времени, что обосновано необходимостью затрат времени на обработку текущего обнаруженного состояния и синхронизацией по времени. Даже незначительное переключение потока-клиента в режим *standby* в этом случае позволяет снять его первичную монополию на процессорное время и получить соотношение 15–40 тактов потока объекта-сервиса на один запрос клиента при 1–10*ms* режиме ожидания. В результате процесс моделирования при прочих равных условиях на выходе получил результат, значительно отличающийся от предыдущего. Результаты моделирования представлены в виде совокупных графиков (рис. 3 и рис. 4).

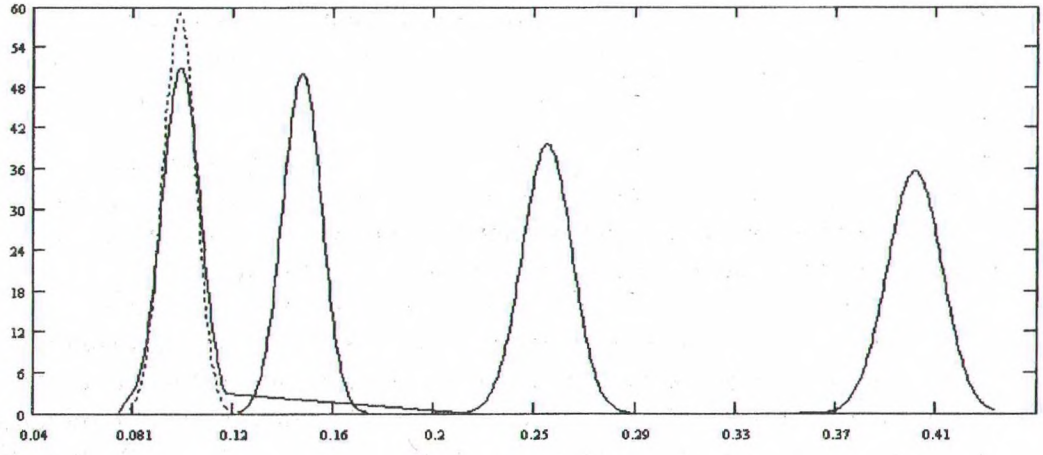

Рис. 3

Параметры моделирования: моделируемый ряд событий P_i^1; алгоритм "производства" БСВ – метод линейных смешанных формул 1-го порядка; количество экспериментов 1000; циклов накопления – 1500. Как видно, в этом присутствует аномальный разброс значений в зоне интервала 0.1 (рис. 3) и,как следствие,отклонение от нормаль-

ного распределения результата. Применение в качестве алгоритма метода линейных смешанных формул 2-го порядка (рис. 4) дает значительно лучшие результаты моделируемой величины P_i^1.

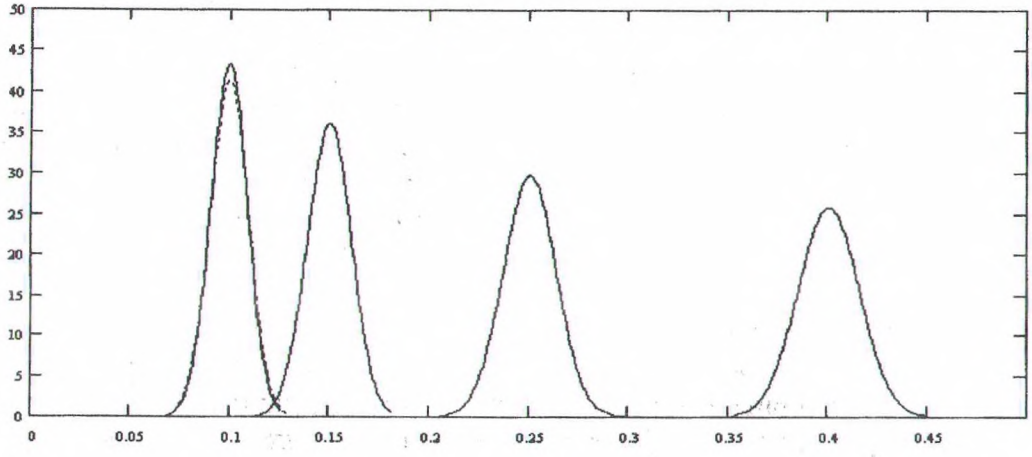

Рис. 4.

В отличие от первоначальных условий, когда процесс моделирования запускался с параметрами по умолчанию, применение метода обращения потока-клиента к потоку объекта-сервиса через случайные дискретные промежутки времени дало значительно лучшие результаты. Как видно из графиков, максимальный разброс сократился почти вдвое, абсолютное отклонение от математического ожидания (доверительный интервал) составляет порядка $2.5 \cdot 10^{-4}..4.5 \cdot 10^{-5}$, отклонение математического ожидания моделируемой величины P_i^* от первоначально заданного составляет порядка $1 \cdot 10^{-3}..7 \cdot 10^{-4}$.

Полученные данные свидетельствуют о том, что применение данного метода для моделирования последовательности редких событий позволит получать последовательность состояний системы с достаточно малой, порядка $10^{-3}...10^{-4}$, погрешностью.

Заключение

Несмотря на значительные трудности практической реализации объектно-ориентированной технологии, построение программы имитационной модели надежность ДОЗУ в итоге является более перспективным способом по сравнению с классическим процедурно-ориентированным способом создания программ.

Один из самых важных шагов – это использование метода разделения всего процесса моделирования на отдельные вычислительные процессы, реализованные в нескольких категориях объектов: концентраторов, сервисов, интерфейсов и потоков-клиентов. Это позволяет в дальнейшем связывать эти объекты в разных комбинациях, что дает несравнимые преимущества перед линейным процедурно-ориентированным программированием. Одна из главных особенностей заключается в том, что предложенное в этой статье соотношение объектов "один к одному" можно расширить до соотношения "один ко многим" [7]. То есть один объект-сервис может поставлять данные нескольким объектам-концентраторам, или один объект-клиент может содержать несколько объектов-сервисов и равное им или нет количество объектов-концентраторов. Возможность хранить объекты в динамически подключаемых библиотеках позволяет в

дальнейшем без значительного изменения основной программы вносить изменения в сервисные объекты и включать новые, при условии сохранения стандарта интерфейса данных.

Подводя итоги, можно сказать, что использование предложенного математического аппарата и его адаптация к применению на ЭВМ, а также применение объектно-ориентированной псевдо-*COM* технологии позволит создавать имитационные модели надежности, отличающиеся достаточно высокой точностью моделирования и большей мобильностью при конфигурировании программы моделирования для решения различных задач.

ЛИТЕРАТУРА

1. Харин Ю.С. Основы имитационного и статистического моделирования. Минск, 1997.

2. Пригарин С.М. Программирование мультипликативных генераторов псевдослучайных чисел на Турбо-Паскале для IBM PC. Новосибирск, 1992.

3. Ермаков С. М. Математический эксперимент с моделями сложных стохастических систем. СПб., 1993.

4. Макась С.Б. Метод Монте-Карло для статистического моделирования распределения ошибок в динамических оперативных запоминающих устройствах. Минск. 2001.

5. Полляк Ю.Г. Вероятностное моделирование на электронных вычислительных машинах. М.: Сов. радио, 1981.

6. Желников В.М. Криптография от папируса до компьютера. М. 1996.

7. Don Box. Essential COM. Addison-Wesley, 1998.

8. Поллард Дж. Справочник по вычислительным методам статистики. М. Статистика. – 1988.

УДК 517.977

В.И. Горошко, доцент

АНАЛИЗ ЛИНЕЙНЫХ СИСТЕМ ПРИ ВОЗДЕЙСТВИИ ИМПУЛЬСНЫХ ПЕРИОДИЧЕСКИХ СИГНАЛОВ

The alternative interpretation and the calculation method development of the steady-state and transient response in the linear systems with the periodical discontinuous actions are done. The method is based on the idea of the poles separation and makes it possible to find the finite expression for the above-mentioned response of the system. The detailed calculation algorithm for the optional piecewise-smooth discontinuous periodical signals has been developed.

В линейных цепях и системах с периодическими разрывными воздействиями могут рассматриваться два процесса: 1) переходный режим и 2) установившийся режим. Для решения этих задач используют различные подходы, такие, как применение эшелонной функции, введение периодической импульсной реакции цепи, наложение реакций, вызываемых каждым отдельным импульсом периодического сигнала. Для этих методов характерен большой объем вычислений, особенно в слабодемпфированных системах. Другим подходом является матричный метод демпфирующих начальных условий (ДНУ) [1].