Б.Ф. Кирьянов

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ В СРЕДЕ DELPHI

Москва
2012

Кирьянов Б.Ф. Математическое моделирование в среде Delphi: Монография. Изд-во Российской Академии Естествознания. 2012. 150 с.

Книга рассчитана на специалистов, использующих в своей деятельности моделирование на ЭВМ, а также на студентов и аспирантов соответствующих направлений и специальностей. Значительная часть монографии использует материал, полученный автором. Приводятся упражнения, которые читателям предлагается выполнить самостоятельно.

Табл. 4. Ил. 29. Библиогр.: 78 назв.

Рецензенты:

Доктор технических наук, профессор, Заслуженный деятель науки РФ, академик Международной академии наук высшей школы, Председатель Научно-методического совета по математике вузов Северо-запада РФ
В.Г. Дегтярёв.

Доктор технических наук, профессор, Заслуженный деятель науки РФ, Член-корреспондент Академии наук Республики Татарстан
В.А. Песошин.

ISBN

УДК 800.92Delphi:519.711.3

ББК 22.181я73

Оглавление

Введение

Общие сведения о математических моделях и математическом моделировании. Среда Delphi как средство моделирования

1. Понятие системы. Методы исследования систем. Система – это совокупность объектов, функционирующих и взаимодействующих друг с другом для достижения определённой цели. В ряде случаев проводить эксперименты можно на реальных системах. Однако далеко не всегда это возможно, например, при невозможности получения строгого математического описания системы. Поэтому в таких случаях создаются модели реальных систем, которые и исследуются (слово “модель” происходит от греческого слова modulus). Выявленные в процессе исследования характеристики модели приписываются реальной системе, то есть системе-оригиналу. Исследования, проводимее на модели, и называются моделированием [9, 48, 62 и др.].

2. Общие определения модели и моделирования. Отметим: никакое определение не может в полном объёме охватить реально существующую деятельность по моделированию. Однако определения полезны тем, что в них делается попытка выделить наиболее существенные черты моделей и моделирования.

***Свободная международная энциклопедия – википедия*:** Моделирование — исследование объектов познания на их моделях, построение и изучение моделей реально существующих предметов, процессов или явлений с целью получения объяснений этих явлений, а также для их предсказания.

Ляпунов А.А.: Моделирование – это опосредованное практическое или теоретическое исследование объекта, при котором непосредственно изучается не сам интересующий нас объект, а некоторая вспомогательная искусственная или естественная система (модель).

Советов Б.Я. и Яковлев С.А.: Модель – это объект-заместитель объекта-оригинала, обеспечивающий изучение некоторых свойств оригинала [62].

3. Разновидности моделей. Модели принято делить на два класса: физические и математические. Физические модели имеют ту же физическую природу, что и оригинал. Физической моделью является, например, аэродинамическая труба, в которой устанавливается самолёт для исследования поведения его органов управления при продувании трубы воздухом с такой же скоростью, как при реальном полёте. Физическим моделированием являются испытания

опытных образцов различной аппаратуры на тряску, давление и.т.д. Математическая модель – это точное или приближённое математическое описание свойств или закономерностей функционирования оригинала.

4. Определения математической модели и моделирования

Самарский А.А. и Михайлов А.П.: Математическая модель – это «эквивалент объекта, отражающий в математической форме важнейшие его свойства – законы, которым он подчиняется, связи, присущие составляющим его частям». «Математическое моделирование — процесс построения и изучения математических моделей реальных процессов и явлений» [61].

Севостьянов А. Г.: Математической моделью называется совокупность математических соотношений, уравнений, неравенств и т.п., описывающих основные закономерности, присущие изучаемому процессу, объекту или системе.

Скворцова М.: Математическая модель – это приближенное описание какого-либо класса явлений или объектов реального мира на языке математики. Основная задача математического моделирования – не только исследовать эти объекты, но и предсказать результаты будущих наблюдений.

В целом в литературе довольно много похожих определений. Самое короткое и лаконичное определение модели дано в одной из зарубежных публикаций: Математическая модель это *Уравнение, выражающее идею.* Но это определение относится, скорее, к философскому понятию модели.

5. Немного истории. Исследование систем путем их моделирования. Ещё во 2-м веке до н.э. появилась геоцентрическая система (модель) мира Птоломея, затем − гелиоцентрическая система (модель) мира Николая Коперника (16-й век н.э., helio – приставка, означающая солнце). С развитием техники перед изготовлением паровых машин, электрических установок, средств автоматизации вычислений и т.д. в большинстве случаев они предварительно отрабатывались на их моделях. Предварительное моделирование особенно важно проводить при разработке новых перспективных производственных технологий для обеспечения их эффективности, сложной и дорогостоящей техники для обеспечения её надежности (например, надёжности нового космического корабля) и т.д.

В 20-м веке с появлением ЭВМ активизировались разработка математических моделей и методов и технологий математического моделирования на ЭВМ. При этом установилась классификация способов исследования систем с помощью моделей [48, 62 и др.], приведённая на рис. В.1.

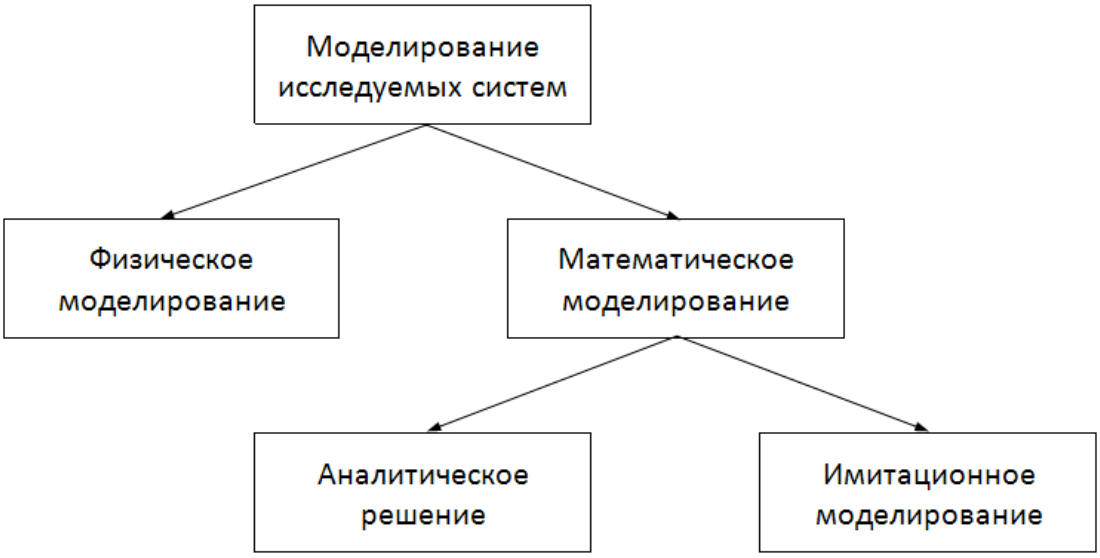


Рис. В.1. Способы исследования систем

В отличие от физического моделирования, предполагающего измерение соответствующих характеристик модели, математическое моделирование предполагает разработку математического описания функционирования исследуемой системы, которое в ряде случаев может иметь аналитическое решение. При имитационном моделировании на основе алгоритмов, описывающих указанное функционирование, модель воспроизводит процессы, происходящие в реальной системе. Частным случаем этого моделирования является эволюционное моделирование, связанное с исследованием эволюции различных систем [57].

6. Понятия функциональной полноты и адекватности модели. Пусть некоторый оригинал (рис. В.2) имеет *k* входов, на которые поступают соответствующие входные величины, задаваемые вектором $\boldsymbol{X} = (X_1, X_2, \ldots X_k)$. Аналогично выходы оригинала задаются вектором $\boldsymbol{Y} = (Y_1, Y_2, \ldots Y_k)$.

Каждая координата Y_i является функцией вектора $\boldsymbol{X}$ и внутреннего состояния оригинала. Последнее в общем случае может изменяться под воздействием как входных, так и выходных величин. Если на вход модели поступает последовательность векторов $\boldsymbol{X}$, то имеют место векторные процессы $\boldsymbol{X}(t)$ и $\boldsymbol{Y}(t)$, которые могут быть детерминированными или случайными.

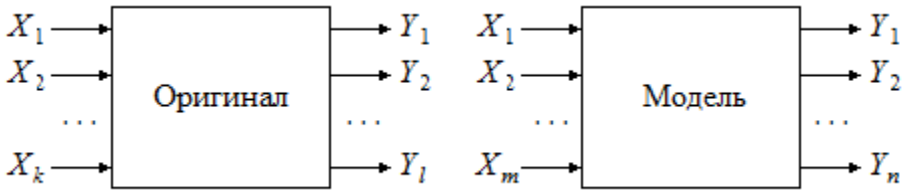


Рис. В.2. Абстрактная система оригинал – модель

Как и оригинал, модель может иметь k входов и l выходов, формируемых аналогично оригиналу. Такая модель называется функционально полной. Если при этом имеет место ещё и полное совпадение выходных процессов модели и её оригинала, соответствующих любому входному процессу, то говорят, что модель адекватна оригиналу. Однако во многих случаях при моделировании нет необходимости воспроизводить все возможности оригинала. В этом случае выбирают $m < k$ и $n < l$ и нередко говорят о степени адекватности построенной модели по отношению к оригиналу. Однако заметим, что понятие «степень адекватности», а также используемые в литературе такие характеристики этой степени как «высокая степень адекватности», «достаточно адекватная» или «относительно адекватная», количественно не определены. Поэтому с математической точки зрения они несостоятельны. Использование же их объясняется тем, что нередко адекватность модели оригиналу просто невозможно проверить (например, ввиду ограниченности технических возможностей и т.д.).

Основные этапы разработки математических моделей:

- Разработка математического описания предварительного варианта модели (уточнение задачи моделирования, определение требований к входным и выходным переменным, построение структурной схемы модели, выбор её параметров, выбор методов получения конечных и промежуточных результатов). Результатом этого этапа является формальное математическое описание блоков модели. При этом как бы исчезает физическая сторона исследуемой задачи, то есть для лиц, не знакомых с решаемой задачей, по математическому описанию модели крайне затруднительно или даже невозможно уяснить сущность этой задачи.
- Разработка машинной программы моделирования, включая алгоритмизацию математического описания, разработку программы в соответствующей среде программирования и проверку правильности её работы (например, по моделированию характерных частных вариантов входных величин, для которых известны или могут быть рассчитаны значения выходных величин).
- Испытания и корректировка модели (моделирование исследуемой системы для известных частных случаев, сравнение получаемых при этом результатов с известными результатами и в случае необходимости – корректировка параметров модели). Данный этап часто называют интерпретацией результатов моделирования.

Разработка модели при отсутствии оригинала. Примером такой модели может служить модель интегральных показателей (ИП) здоровья населения, использующая в качестве параметров показатели здоровья населения. Эти

показатели ежегодно определяются и публикуются Госкомстатом России. Основным вариантом таких моделей является следующий [42]:

$$ИП = К_1ОКР + К_2СППЖ – К_3ОЗО – К_4ПИНВ – К_5ОКС + С,$$

где $К_I$ – весовые коэффициенты, ОКР – общий коэффициент рождаемости (число родившихся живыми младенцев, приходящихся на 1000 человек населения рассматриваемого региона), СППЖ – средняя продолжительность предстоящей жизни (значение прогноза, определяемого для поколения, родившегося в рассматриваемом году), ОЗО – общая заболеваемость по обращениям (число обращений населения в учреждения здравоохранения, приходящихся на 1000 человек населения), ОКС – общий коэффициент смертности (число умерших, приходящихся на 1000 человек населения), С – константа, позволяющая задавать среднее значение ИП для РФ в целом.

В приведённом выражении все ИП изменяются в интервале (0, 1), хотя это и не обязательно.

Факторный анализ на модели. Важной разновидностью исследуемых с помощью моделирования задач является факторный анализ. Его сущность заключается в исследовании влияния соответствующих параметров модели (факторов, входных факторов) на отклики (выходные факторы). В зависимости от числа анализируемых факторов различают однофакторный, двухфакторный, … , многофакторный анализ. При проведении факторного анализа изменяют только рассматриваемый входной фактор, а значения остальных факторов либо не меняются вообще, либо изменяются только от эксперимента к эксперименту. В последнем случае получают семейство таблиц, в каждой строке которых приводятся значения откликов, соответствующих выбранным значениям рассматриваемого фактора. Причем каждая такая таблица составляется для различных значений остальных факторов.

Отметим, что в последние годы исследование математических моделей чаще всего проводят в среде программирования Delphi (см. приложение П1), которая позволяет значительно облегчить работу программистов по разработке программы моделирования и упростить технологию моделирования. Поэтому все рассматриваемые в монографии программные решения задач математического моделирования приводятся для среды Delphi.

В литературе имеется достаточно много публикаций, посвящённых различным вопросам построения математических моделей и непосредственно

моделированию. Большинство их осталось за пределами монографии [3, 8, 19, 20, 30, 33, 39, 43, 47, 50, 54, 59, 60, 63, 69, 75 и др.]. Монография в основном нацелена на реализацию и исследование математических моделей в среде программирования Delphi.

Упражнения[1]

x	4	2	1	3	5
y	2	5	12	9	4

1) В таблице приведены результаты замеров отклика y от значений фактора x неизвестной структуры, на которую воздействуют помехи. Подобрать детерминированную модель зависимости $y = f(x)$ и определить её параметры.

2) Моделями каких оригиналов являются глобус, журнал успеваемости реподавателя, изображение мамонта, нарисованное по его скелету, и выходные данные компьютерной программы Randomize?

3) Какие модели используются в методах Эйлера численного решения дифференциальных уравнений?

4) Привести примеры моделей в самолето- и в кораблестроении, в археологии и в милиции.

Рекомендация. Читателям, не знакомым со средой программирования Delphi, для понимания приводимых в монографии примеров программных решений рекомендуем ознакомиться с этой средой по приложению П1, которое автор старался изложить в простом, доступном широкому кругу читателей виде.

[1]) Здесь и далее в пунктах “Упражнения” приводятся задания и вопросы для читателей, желающих более детально проработать соответствующий материал.

Глава 1. Детерминированные модели

1.1. Основные моделируемые структуры. Цифровые автоматы

В общем случае детерминированная техническая система, реализуемая с помощью работающих в дискретном времени *t* цифровых блоков (устройств), а также сами эти блоки можно представить в виде так называемых цифровых автоматов. Такой автомат имеет внутреннюю память, входную и выходную логические схемы (рис. 1.1). Согласно литературе цифровые автоматы делятся на автоматы Мура и автоматы Мили (по фамилиям соответствующих американских ученых – Moore machine и Mili machine). Аналоговые автоматы рассматриваться не будут (они в основном отжили).

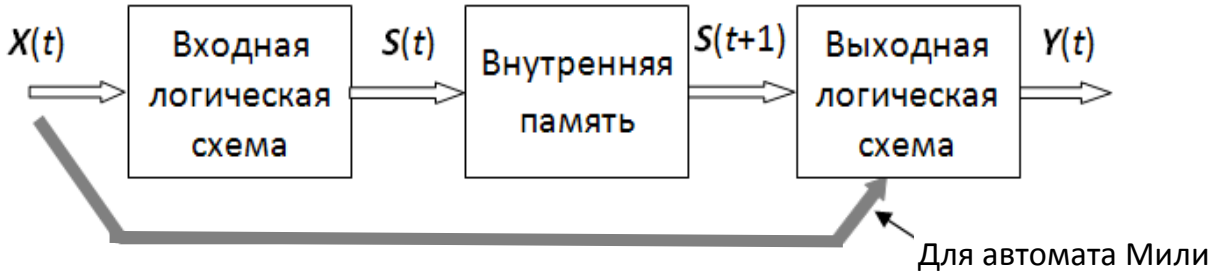


Рис. 1.1. Структурная схема цифровых автоматов Мура и Мили

Будем полагать, что входная и выходная логические схемы автоматов реализуют соответственно функции f и g, а содержимое внутренней памяти и ее входная величина подвергаются некоторому преобразованию h. Тогда поведение автомата Мура можем описать следующей системой уравнений, в которой в общем случае величины $\boldsymbol{S}$, $\boldsymbol{X}$ и $\boldsymbol{Y}$ являются векторными:

$$\begin{cases} \boldsymbol{S}(t+1) = h\big(\boldsymbol{S}(t)\big) = h\big(f\big(\boldsymbol{X}(t)\big), \\ \boldsymbol{Y}(t) = g\big(\boldsymbol{S}(t+1)\big). \end{cases}$$

Для автомата Мили первое уравнение этой остаётся неизменным, а второе принимает следующий вид:

$$\boldsymbol{Y}(t) = g\Big(\boldsymbol{S}(t+1), f\big(\boldsymbol{X}(t)\big)\Big).$$

Приведенные выражения по существу задают разностные уравнения, реализуемые цифровыми автоматами. Такие уравнения можно рассматривать как модели соответствующих дифференциальных уравнений, описывающих процессы в реальном времени.

Отметим, что в автоматах Мили входной вектор $\boldsymbol{X}(t)$ можно подавать на выходную логическую схему и после прохождения им входной логической схемы. В этом случае

$$\boldsymbol{Y}(t) = g\left(\boldsymbol{S}(t), f\left(\boldsymbol{X}(t)\right)\right).$$

На практике обычно приходится моделировать более сложные системы с обратными связями, основой которых являются рассмотренные автоматы.

1.2. Математический аппарат, реализуемый в линейных детерминированных моделях, и основные задачи моделирования на их основе

Поскольку рассмотренные структуры цифровых автоматов являются обобщением большинства логических и арифметических устройств цифровой техники, то остановимся на вопросах их моделирования. Учтем, что обычно функции *f*, *g* и *h* являются линейными.

В указанном случае основным математическим аппаратом для построения моделей являются матричные (в частном случае – векторные) преобразования и линейные разностные уравнения. Рассмотрим их практическую реализацию.

Для выполнения матричных преобразований не требуются тактовые импульсы *t* (импульсы, реализующие временные шаги разностного уравнения, то есть шаги алгоритма). Рассмотрим на примере матричное преобразование входного вектора ***X***, каждая координата которого задается некоторым числом. Пусть, например,

$$\boldsymbol{X} = \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix}, \quad \boldsymbol{A} = \begin{pmatrix} 0 & -2 & 2 \\ 1 & 1 & 3 \\ 2 & 0 & -1 \end{pmatrix}.$$

Тогда изображаемое на схеме преобразование $\boldsymbol{X}(t) \Longrightarrow \boxed{\boldsymbol{A}} \Longrightarrow \boldsymbol{Y}(t)$ при моделировании реализуется как $\boldsymbol{Y}(t) = \boldsymbol{A}\boldsymbol{X}(t)$ в случае представления вектора $\boldsymbol{X}$ вектором-столбцом или как $\boldsymbol{Y}(t) = \boldsymbol{X}(t)\boldsymbol{A}$ в случае представления этого вектора вектором-строкой. При этом результат $\boldsymbol{Y}$ получается на том же шаге, на котором появляется и $\boldsymbol{X}$. Технически он реализуется с помощью логической схемы. В приведенном примере соответственно получаем:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{A} = (5, -5, -2) \text{ или } \boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} = \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix}.$$

При моделировании разностных уравнений соответствующие результаты должны получаться на разных шагах *t.* На схемах задержка на один шаг (такт) часто обозначается буквой *D* (delay). В программах задержка реализуется выбором соответствующей последовательности операторов программ.

Основная задача моделирования систем – исследование их поведения при различных значениях параметров. Центральной проблемой при этом является анализ устойчивости системы, описываемой разностными уравнениями, который в случае сложной правой части этих уравнений выполнить аналитически практически невозможно.

Известно несколько критериев устойчивости линейных цифровых систем, функционирование которых описывается системами разностных уравнений. Наибольшее применение получил, по-видимому, критерий устойчивости Ляпунова. Согласно ему решение системы уравнений размерности *n называется устойчивым по Ляпунову*, если для любого ненулевого конечного начального возмущения ***Y***(0) существует такой вектор ***U*** размерности *n* с ограниченными координатами $U_i > 0$, что при любом $t > 0$ для координат $\overline{Y}_i(t)$ "возмущённого" решения и координат $Y_i(t)$ "невозмущённого" решений выполняется условие $|\overline{Y}_i(t) - Y_i(t)| < U_i$. В противном случае решение *называется неустойчивым*. Устойчивое решение $Y_i(t)$ *называется асимптотически устойчивым*, если вектор, координаты которого суть $|\overline{Y}_i(t) - Y_i(t)|$, при неограниченном увеличении *t* стремится к нуль-вектору.

1.3. Пример подготовки математического описания системы для её моделирования на компьютере

Пусть необходимо провести моделирование некоторой подсистемы, представляющей рассмотренный автомат Мура, в который добавлена обратная связь. Вариант схемы модели этого автомата приведён на рис. 1.2.

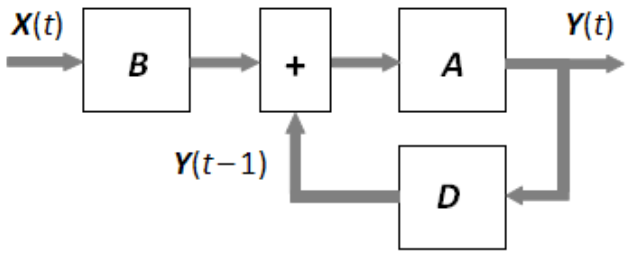


Рис. 1.2. Автомат Мура

Так как матрица ***B*** не является нулевой, то схема на рис. 1.2 описывается неоднородным разностным уравнением:

$$\boldsymbol{Y}(t) = \boldsymbol{A}[\boldsymbol{Y}(t-1) + \boldsymbol{B}\boldsymbol{X}(t)].$$

Если матрица ***B*** – нулевая, что в технике эквивалентно отсутствию входных сигналов, то приведённая модель называется автономной.

Общее решение неоднородного уравнения состоит, как известно, из суммы решений однородной части $\boldsymbol{Y}(t) = \boldsymbol{A}\,\boldsymbol{Y}(t-1)$ этого уравнения и частного решения указанного неоднородного уравнения. Необходимым условием устойчивости получаемого решения является устойчивость решения однородного уравнения. Для устойчивости решения однородного уравнения необходимо и достаточно, чтобы все характеристические числа λ_i этого уравнения (собственные значения матрицы ***A***) находились в единичном круге комплексной плоскости. Если имеются ещё и некратные характеристические числа, лежащие на указанном круге, то решение считается условно устойчивым. Примером такого решения является дискретная синусоида с постоянной амплитудой колебаний.

Для трехмерной матрицы ***A*** значения её λ_i определяются из характеристического уравнения

$$\left|\boldsymbol{A} - \lambda \boldsymbol{E}\right| = \begin{vmatrix} A_{11}-\lambda & A_{12} & A_{13} \\ A_{21} & A_{22}-\lambda & A_{23} \\ A_{31} & A_{32} & A_{33}-\lambda \end{vmatrix},$$

в котором ***E*** – единичная матрица.

Таким образом, если вопрос об устойчивости решения $\boldsymbol{Y}(t)$ рассматриваемого автомата можно решить на прибегая к моделированию, то графики этого векторного решения удобно получать с помощью моделирования.

Читателям рекомендуется составить векторное уравнение и соответствующую ему систему скалярных уравнений, описывающих работу автономной модели, приводимой на рис. 1.3. Матрицы ***A*** и ***B*** имеют размерность 3 x 3.

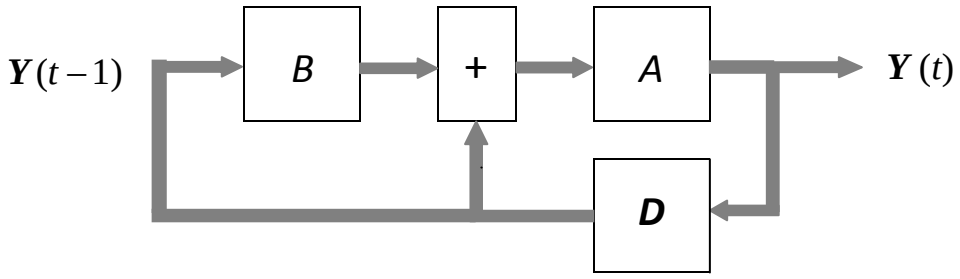


Рис. 1.3. Автономная модель

1.4. Линейные модулярные модели

Рассматриваемые модели часто называют модулярными, так как в них выполняются операции не только по модулю 10, но и по модулю 2.

1.4.1. Элементы, используемые в модулярных структурах

В качестве элементарных элементов в таких структурах используются логические элементы, выполняющие операции НЕ, ИЛИ, И, $\oplus$, D – триггер (задержка на шаг), T-триггер. С математической точки зрения принято говорить, с помощью указанных элементов реализуется алгебра $<M, S>$, в которой носитель $M = \{0, 1\}$, а сигнатура $S = \{\oplus, \otimes, \neg\}$. С учетом частого использования элемента задержки D сигнатуру этой алгебры обычно расширяют и этим элементом.

При программировании указанные логические и модулярные операции могут заменять друг друга: $\neg X = X + 1$ (М2), $X \vee Y = X + Y + XY$ (М2), $X \vee Y \vee Z = X + Y + Z + XY + XZ + YZ + XYZ$ (М2).

Значительно реже при моделировании возникает необходимость запрограммировать элемент T-триггер (триггер со счетным входом). Этот триггер можно построить с помощью D-триггера и сумматора по модулю 2. Возможен и обратный переход:

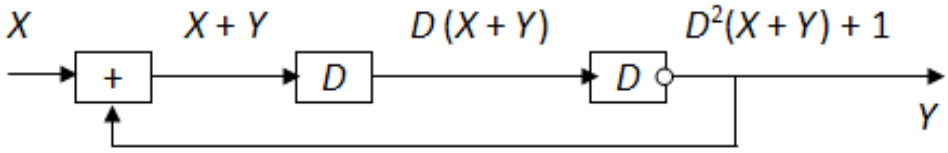
В приведенных примерах все сложения выполняются по mod 2. В этом случае $0 = 2 = 4 = 6 = \ldots$ и $1 = 3 = 5 = \ldots$, а D – оператор задержки на один шаг. Оператор задержки на n шагов обозначается D^n, а $D^0 = I$ – оператор задержки на 0 шагов.

1.4.2. Модулярные структуры и их описание

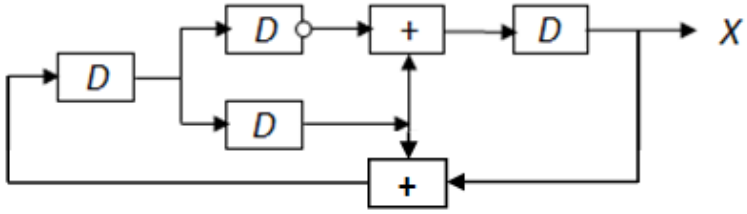
На базе элементов, выполняющих модулярные операции, строятся и сложные системы и устройства, например – контрольно-тестирующие устройства для проверки правильности функционирования различной цифровой аппаратуры, компьютерных программ, моделирующих цифровые системы, программные и аппаратные генераторы последовательностей псевдослучайных кодов (в большинстве компьютеров – программа Randomize) и др. Остановимся на определении зависимости выходных сигналов модулярных систем от входных сигналов и от структуры элементов этих систем. При этом операции сложения по модулю два будем обозначать знаком плюс без кружочка. Если некоторая система не имеет входа, то она называется автономной. В общем случае выход системы может быть векторным, то есть имеющим несколько двоичных выходов.

Рассмотрим два примера. Проанализируем зависимости для выходных сигналов двух схем. Результаты анализа для первого примера приведены на схеме. Для автономных структур многочлены *f*(*D*) в выражениях *X* = *f*(*D*)·X называются производящими.

Пример 1. Неавтономная структура



Пример 2. Автономная структура



Во втором примере для упрощения анализа выходных сигналов целесообразно принять, что на выходе левого *D*-триггера имеет место некоторый операторный сигнал *Z*. Тогда можно быстро придти к выводу, что схема генерирует только одни единицы, то есть является генератором единиц, и ее можно упростить до одного элемента *D* с замыканием его выхода на вход (разумеется, в *D*-триггер предварительно записывается 1).

1.4.3. Периоды состояний автономных структур

Последовательности {*X*}, генерируемые рассматриваемыми автономными структурами, характеризуются различной цикличностью. Их *производящие полиномы имеют вид*: $F(D) = \sum_{i=0}^{n} a_i D^{n-i} \ (mod\ 2)$. Так как $a \in \{0,1\}$, а используемые операции (сигнатура) – сложение и умножение по модулю два, то говорят, что данные полиномы являются полиномами над полем *GF*(2) (Galois field). Их *неприводимость*: Если полином *F*(*D*) или *F*(*x*) не делится ни на один полином степени *k*, где $k = \overline{1,\ n-1}$, то есть *F*(*x*) не раскладывается на множители, то он называется неприводимым. Примеры неприводимых *F*(*x*): $x^2 + x + 1,\ x^3 + x^2 + 1,\ x^5 + x^3 + 1$ и т.д. Неприводимые полиномы (многочлены) не раскладываются на множители, то есть они не приводятся к произведению сомножителей. Если же *F*(*x*) раскладывается на множители, то его называют приводимым. Пример приводимого многочлена: $x^5 + x + 1 = (x^2 + x + 1) \cdot (x^3 + x^2 + 1)$.

На практике большое значение получила частная разновидность неприводимых полиномов: *примитивные* полиномы (поясняются ниже).

Существуют и более общие поля Галуа (1812 – 1832), в которых сложение выполняется по модулю *p*, где *p* – простое число: *GF*(*p*), или даже *GF*(p^m).

Теорема Галуа. *Для любого многочлена F(x) над p-ичным полем, в котором a_0 не является нулевым, существует наименьшее целое положительное число* $\delta \leq p^n - 1$ *такое, что многочлен* $G(x) = x^\delta + 1$ *делится без остатка на F(x).*

Приведённую теорему Галуа доказал и записал в ночь перед дуэлью. Современники-математики важность ряда работ Галуа, в том числе данной теоремы – теоремы существования не понимали. Сейчас теорему Галуа называют "жемчужиной математики". Она является "математическим фундаментом" для ряда важных практических приложений. Два из них будут рассмотрены ниже.

Мы рассматриваем только случай $p = 2$, то есть простые поля Галуа. Для таких полей $\delta \leq 2^n - 1$. ***Если*** $\boldsymbol{\delta = 2^n - 1}$***, то соответствующий многочлен*** $\boldsymbol{F(x)}$ ***порождает последовательности наибольшей длины и называется примитивным***. Указанные последовательности называют *M*-последовательностями (*M* – max), то есть последовательностями максимальной длины. Величина δ называется порядком многочлена. Таким образом, многочлены над полем имеют как степень, так и порядок. Например, примитивный многочлен $x^3 + x + 1$ третьей степени имеет порядок равный 7. Отметим: если многочлен $F(x)$ является примитивным, то многочлен $H(x) = x^n F(1/x)$ тоже примитивный. Данные многочлены называются сопряженными.

Если существует многочлен *G*(x) с порядком, меньшим $2^n - 1$, то *F*(*x*) не является примитивным. Так, полином $x^3 + 1 = (x + 1) \cdot (x^2 + x + 1)$. Поэтому производящий многочлен $D^3 + I$ не порождает М-последовательность (для операторных многочленов вместо 1 используется *I*, то есть D^0).

Ввиду того, что М-последовательности имеют применение в технике связи, вычислительной технике, автоматике, в системах скрытия передачи информации и в некоторых других областях, то в литературе [55 и др.] приводятся таблицы примитивных или неразложимых многочленов. В монографии таблицы часто используемых неприводимых многочленов приведены в приложении.

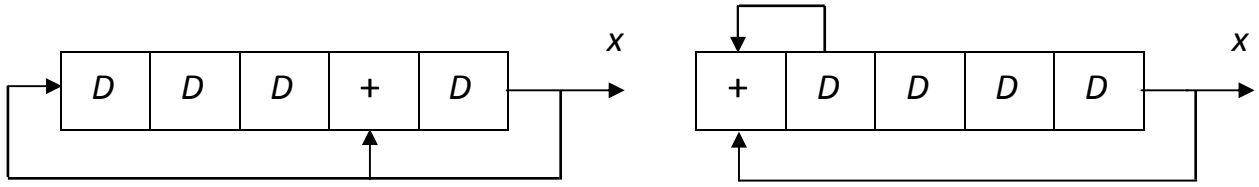
Уравнение генератора символов X на основе многочлена $G(D)$ имеет вид: $G(D)X = 0$. Например: $(D^2 + D + I)X = 0$, то есть $X = (D^2 + D)X$.

Читателям, заинтересованным в освоении приведенного материала, рекомендуем:

1. Проверить, примитивен ли многочлен $x^4 + x^3 + x + 1$ путем деления на него многочлена $x^5 + 1$.

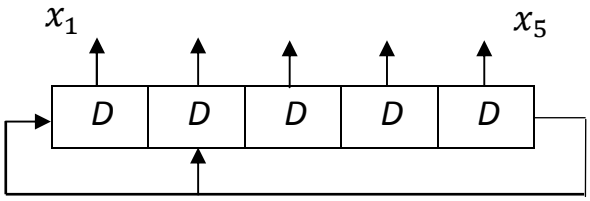
2. Найти результирующий многочлен: $(x^3 + 2x^2 + x + 1) \cdot (2x^2 + x + 2)$ (mod 3).

3. Найти производящие многочлены двух приведённых ниже генераторов, а также последовательностей $\{x\}$, генерируемых ими символов. Какой вывод можно сделать из сравнения этих многочленов?



1.4.4. Матрицы над полем *GF*(2) и их применение для описания работы генераторов кодовых последовательностей

Рассмотрим схему генератора М-последовательности. Его выходами могут быть выходы всех его *D*-триггеров, а не одного. В этом случае с генератора снимается *n*-разрядный код $\{x_1,\ x_2,\ \ldots,\ x_n\}$, который можно рассматривать как вектор ***X*** (рис. справа).



Для таких генераторов имеем: $\boldsymbol{X}(t+1) = \boldsymbol{A}\cdot\boldsymbol{X}(t)$ или $\boldsymbol{X}(t+1) = \boldsymbol{X}(t)\cdot\boldsymbol{B}$, где ***A*** и ***B*** – соответствующие матрицы, а t – дискретное время (номер шага).

В первом выражении вектор ***X*** является вектором-столбцом, а во втором – вектором-строкой. С каждого выхода генерируется М-последовательность (последовательности $\{x_i\}$, сдвинутые относительно “соседних” последовательностей на один шаг). Для приведенной конкретной схемы генератора матрицы ***A*** и ***B*** имеют вид:

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Очевидно, ***A*** совпадает с транспонированной ***B*** и наоборот $\boldsymbol{B} = \boldsymbol{A}^{/}$. При формировании каждого из значений x_i матрица ***A*** “работает” *i*-й строкой, а

матрица $\boldsymbol{B}$ – *i*-м столбцом. Столбец $\boldsymbol{A}_1$ и строка $\boldsymbol{B}_5$ задают производящий многочлен $G(D)$.

Характеристические многочлены этих матриц совпадают. Действительно:

$$H(\lambda)_{\boldsymbol{A}} = \begin{vmatrix} \lambda & 0 & 0 & 0 & 1 \\ 1 & \lambda & 0 & 0 & 1 \\ 0 & 1 & \lambda & 0 & 0 \\ 0 & 0 & 1 & \lambda & 0 \\ 0 & 0 & 0 & 1 & \lambda \end{vmatrix} = \lambda^5 + 1, \quad H(\lambda)_{\boldsymbol{B}} = \begin{vmatrix} \lambda & 1 & 0 & 0 & 0 \\ 0 & \lambda & 1 & 0 & 0 \\ 0 & 0 & \lambda & 1 & 0 \\ 0 & 0 & 0 & \lambda & 1 \\ 1 & 1 & 0 & 0 & \lambda \end{vmatrix} = \lambda^5 + 1.$$

С помощью матричного аппарата удобно анализировать работу рассматриваемых генераторов.

Кольца символов, кодов, матриц. Используя рассмотренные матричные преобразования не сложно получить: $\boldsymbol{X}(t) = \boldsymbol{A}^t \boldsymbol{X}(0)$ или $\boldsymbol{X}(t) = \boldsymbol{X}(0)\boldsymbol{B}^t$, $\boldsymbol{A}^{L+t} = \boldsymbol{A}^L \boldsymbol{A}^t = \boldsymbol{E}\boldsymbol{A}^t = \boldsymbol{A}^t$, где *L* – период.

Рассмотрим *свойства М-последовательностей*.

1) В периоде $L = 2^n - 1$ *М*-последовательности, порождаемой производящим многочленом *n*-й степени, содержится 2^{n-1} единиц и $2^{n-1} - 1$ нулей.
2) Поэлементная сумма по модулю 2 *М*-последовательности с её копией, сдвинутой на τ шагов, где $\tau \neq 0 \pmod{M}$, является той же *М*-последовательностью, сдвинутой относительно исходной последовательности на число шагов, отличное от τ.

   Доказательство: Пусть $\boldsymbol{X}_{\text{осн}}(1)$ – код в генераторе основной последовательности в момент $t = 1$, то есть $\boldsymbol{X}_{\text{осн}}(1) = \boldsymbol{A}\boldsymbol{X}_{осн}(0)$. При появлении в генераторе этого кода с некоторого его выхода выдается символ $x_{\text{осн}}(1.)$ Последовательность символов, выдаваемых с указанного выхода генератора будем считать символами основной последовательности $\{x_{\text{осн}}\}$. Для последовательности $\{x_{\text{сдв}}\}$, сдвинутой на τ шагов вперед (аналогично доказывается и для сдвига назад), в момент $t = 1$ получим: $\boldsymbol{X}_{\text{сдв}}(1) = \boldsymbol{A}\boldsymbol{X}_{\boldsymbol{сдв}}(0) = \boldsymbol{A}\boldsymbol{X}_{\boldsymbol{осн}}(\boldsymbol{\tau}) = \boldsymbol{A}^{\tau+1}\boldsymbol{X}_{\boldsymbol{осн}}(0)$. При сложении последовательностей $\{x_{\text{осн}}\}$ и $\{x_{\text{сдв}}\}$ по mod 2 для суммарной последовательности $\{x_{\text{сум}}\}$ получаем: $\boldsymbol{X}_{\text{сум}}(1) = \boldsymbol{X}_{\text{осн}}(1) + \boldsymbol{X}_{\text{сдв}}(1) = \boldsymbol{A} \cdot \boldsymbol{X}_{\text{осн}}(0) + \boldsymbol{A}^{\tau+1}\boldsymbol{X}_{\text{осн}}(0) = (\boldsymbol{A} + \boldsymbol{A}^{\tau+1})\boldsymbol{X}_{осн}(0)$. Но матрицы $\boldsymbol{A}$, $\boldsymbol{A}^2$, . . . $\boldsymbol{A}^{n-1}$ являются элементами алгебраической структуры “кольцо”. А сумма двух разных элементов кольца даёт элемент того же кольца, отличающийся от слагаемых. Следовательно, $\boldsymbol{A} + \boldsymbol{A}^{\tau+1} = \boldsymbol{A}^{\boldsymbol{k}}$, где *k* отлично от 1 и от τ + 1. Поэтому $\boldsymbol{X}_{\text{сум}}(1)$ на каждом шагу опережает $\boldsymbol{X}_{\text{осн}}(1)$ на *k* шагов, то есть и $\{x_{\text{сум}}\}$ опережает $\{x_{\text{осн}}\}$ на *k* шагов, что и требовалось доказать.

3) Сдвиг последовательности $\{x_{\text{сдв}}\}$ на τ символов относительно основной (исходной) последовательности, формируемой генератором кодов, может быть реализован с помощью дополнительного сумматора, входы которого управляются вектором $\boldsymbol{W}(\tau) = A_1\boldsymbol{A}^{\tau-1}$ (схема на рис. 1.4, [25]).

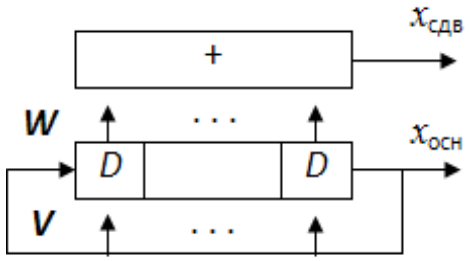


Рис. 1.4. Генератор $\{x_{\text{сдв}}\}$

В формулировке 3-го свойства A_1 – первая строка матрицы $\boldsymbol{A}$. Доказательство этого свойства для поля для *GF*(*p*), где *p* – простое число, приводится в [31].

В литературе [72, 73, 77, 78 и др.] приводится ряд практических вариантов генераторов на рассмотренной теоретической основе, и рассматриваются практические направления их применения.

Глава 2

Стохастические модели

2.1. Общие сведения о стохастических моделях и методах моделирования равновероятных случайных величин

В отличие от детерминированных моделей в структурах стохастических моделей имеются программные или аппаратные датчики случайностей. Эти датчики могут быть параллельными (генерируют сразу все разряды случайных кодов) или последовательными (с каждым тактом генератора синхронизирующих импульсов ЭВМ выдают очередной случайный символ 0 или 1). К указанным датчикам предъявляются требования равновероятности двоичных символов 0 и 1, а также отсутствия корреляции в генерируемых последовательностях. Недостаточное выполнение этих требований может привести к серьёзным погрешностям при решении задач и к неверным выводам.

В последние годы в ЭВМ, как правило, используются программные датчики случайностей: команда random выдает псевдослучайные двоичные символы 0 и 1, которые принимаются за случайные. Многолетняя практика решения задач вероятностными методами (методами Монте-Карло – по названию игорного дома в Швейцарии) показала, что это вполне приемлемо. Имитационное моделирование на основе методов Монте-Карло предполагает воспроизведение исследуемого процесса в ЭВМ очень большое число раз. При этом находятся оценки соответствующих величин, практически совпадающие с искомыми фактическими, которые не удается найти теоретически. Таким образом, стохастические модели позволяют исследовать многие задачи, для анализа которых детерминированные модели непригодны.

Если в качестве датчика случайности используется псевдослучайная последовательность, то за вероятность появления символа 1 принимается величина *p*, значение которой равно:

$$p = \frac{2^{n-1}}{2^n - 1} = 0{,}5 + \frac{0{,}5}{2^n - 1},$$

то есть с увеличением числа разрядов генератора указанной последовательности значение *p* неограниченно близко приближается в 0,5. При этом дисперсия *W* значений символов 0 и 1 псевдослучайной последовательности равна:

$$W = p(1-p) = 0{,}25 \cdot \left[1 - \frac{1}{(2^n - 1)^2}\right].$$

С ростом *n* это значение стремится к дисперсии равновероятных случайных величин 0 и 1, равной 0,25. Нормированная автокорреляционная функция $R(\tau)$ последовательности $\{x\}$, то есть зависимость коэффициента корреляции

случайных символов от числа шагов τ между ними, вычисляемая согласно выражению

$$R(\tau) = \frac{\sum_{i=1}^{L}(x_i - p)(x_{i+\tau} - p)}{L \cdot p(1-p)},$$

где $L = 2^n$-1 (период М-последовательности), принимает следующие значения:

$$R(\tau) = \begin{cases} 1 & \text{при } \tau = 0 \quad (mod\ L), \\ -1/(L-1) & \text{при } \tau \neq 0 \quad (mod\ L). \end{cases}$$

Примечание: согласно литературе для характеристик псевдослучайных последовательностей используются те же названия, что и для случайных.

Из литературы известно несколько методов моделирования равновероятных случайных величин, то есть методов реализации датчиков случайных двоичных символов и двоичных кодов [58 и др.], принимаемых за равновероятные. Рассмотрим предложенную автором ***классификацию*** этих методов [21]:

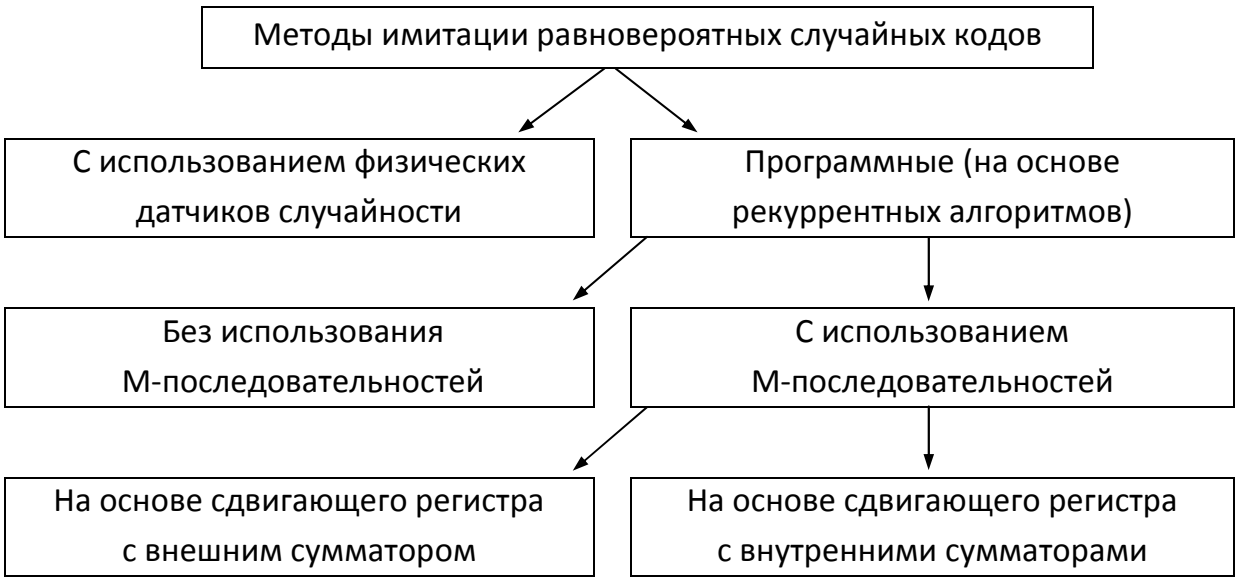


Рис. 2.1. Классификация методов имитации равновероятных случайных кодов

Исторически в качестве ***физических датчиков случайности*** в ЭВМ применялись различные элементы, производящие шумовые сигналы, преобразуемые в последовательности си двоичных символов 0 и 1. Например, шумовой процесс разбивался на две амплитудные зоны, соответствующие 0 и 1. Выдачи символов 0 и 1 осуществлялась по синхронизирующим сигналам тактового генератора ЭВМ.

Недостатком практически всех таких датчиков случайности является нестабильность их характеристик (вероятностей появления символов 0 и 1, автокорреляционной функции и др.), обусловленная зависимостью исходного шумового процесса от стабильности напряжения питания и температуры. Поэтому на практике применение получил лишь метод Неймана, согласно которому определяется число шумовых импульсов за период поступления на датчик синхронизирующих импульсов. Если это число четное, то датчик выдает символ 0, если же оно нечетное, то –1. Нейман показал, что при наличии указанных нестабильностей вероятности появления двоичных символов 0 и 1 практически остаются равными 0,5.

Для сближения вероятностей появления символов 0 и 1 применяют их выравнивание с помощью сумматора по модулю 2, на входы которого подаются генерируемые датчиком $X(t)$ и $X(t-1)$. В этом случае символы 1 будут появляться на выходе сумматора с вероятностью $2p(1-p)$, где p – вероятность их появления на его входе.

В последние годы в вычислительной технике физические датчики случайности используются лишь тогда, когда применение программных методов не допустимо (например, в системах опознавания государственной принадлежности объектов).

К программным методам имитации случайных чисел N, не использующим М-последовательности, относятся несколько методов [9, 10, 48, 58]. Это:

- метод усечения, согласно которому в качестве n-разрядного $N(t)$ принимаются средние n разрядов $2n$-разрядного кода $N^2(t\text{-}1)$;
- метод произведения, при использовании которого $N(t)$ определяется аналогично предыдущему методу по коду $N(t\text{-}1)\cdot N(t\text{-}2)$;
- метод вычетов, использующий преобразование $N(t) = K\cdot N(t\text{-}1)$ по модулю S (например, при $K = 23$ и $S = 10^8 - 1$);
- обобщённый метод вычетов, реализующий зависимость $N(t) = K_0 + K_1\cdot N(t\text{-}1) + \ldots + K_n\cdot N(t\text{-}n) \pmod S$, где сумма весовых коэффициентов K_i равна единице;
- метод перемешивания, формирующий код $N(t)$ путем поразрядного сложения по модулю два кодов ЛК[$N(t\text{-}1)$] и ПК[$N(t\text{-}1)$], где операторы ЛК и ПК сдвигают циклически код $N(t\text{-}1)$ соответственно на К разрядов влево и вправо.

Данные методы нередко используются и в настоящее время, поскольку некоторые специалисты ошибочно полагают, что использование М-последовательностей приводит к значительной автокорреляции в последовательности моделируемых случайных чисел (фактически – в последовательности псевдослучайных чисел, являющейся моделью последовательности случайных чисел). Однако фактически наличие указанной автокорреляции связано, как будет показано далее, с некорректной реализацией этих методов. Поэтому рассмотрим их более подробно.

2.2. Моделирование случайных чисел с использованием М-последовательностей

2.2.1. Генераторы псевдослучайных кодов

На рис. 2.2 приведены три варианта генераторов, моделирующих последовательности 8-разрядных случайных чисел *N*. С каждого их выхода снимаются М-последовательности, которые являются последовательностями разрядов x_i генерируемых чисел. Программно один шаг работы генератора с внешним сумматором реализуется следующими тремя операциями, в которых X_k – значение *k*-го разряда кода:

X[9]:= *X*[1] + *X*[5] + *X*[7] + *X*[8];

for *k*:=1 to 8 do *X*[*k*]:= *X*[*k*+1];

X[8]:=*X*[9];

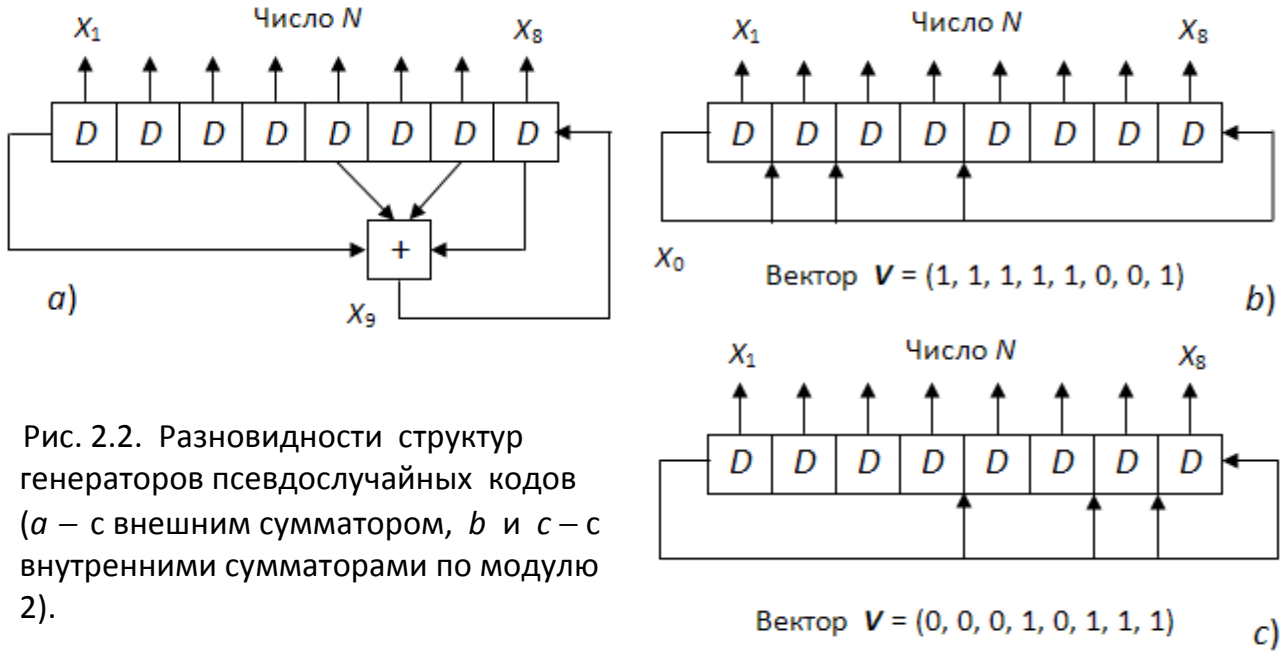


Рис. 2.2. Разновидности структур генераторов псевдослучайных кодов (*a* – с внешним сумматором, *b* и *c* – с внутренними сумматорами по модулю 2).

В генераторах с внутренними сумматорами, которые находятся между соседними элементами задержки и на схемах обычно не рисуются, обратная

связь задается вектором ***V***. Программно один шаг этих генераторов выполняется операциями:

for *k*:=1 to 8 do *X*[*k*-1]:= *X*[*k*]; *X*[8]:=0;

if $X_0 = 1$ then for *k*:=1 to 8 do *X*[*k*]:= *X*[*k*] + *V*[*k*];

Задание обратной связи рассмотренных генераторов вектором *V* удобно при проведении исследований, когда необходимо оперативно изменять различные варианты обратной связи. Поэтому в данном случае вектор *V* можно называть вектором обратной связи [61].

Отметим, что М-последовательности, генерируемые схемами с внешним и с внутренними сумматорами, могут задаваться одинаковыми производящими многочленами. Так, генераторы на рис. 2.2*a* и рис. 2.2*b* формируют М-последо-вательности на основе производящего многочлена $G(D) = D^8 + D^4 + D^2 + D + I$. Следовательно, они будут генерировать одинаковые М-последовательности, которые, однако, по-разному сдвинуты друг относительно друга. Поэтому в указанном случае генераторы не будут эквивалентными относительно генерируемых последовательностей кодов [50].

Читателям рекомендуется на примере любой неособенной матрицы 3-го порядка проверить, что $H(\lambda) = \lambda^3 G(1/\lambda)$ и $G(D) = D^3 H(1/D)$ – свойство сопряженности многочленов $H(\lambda)$ и $G(D)$.

Значения генерируемых чисел *N* зависят от местоположения запятой. Так, если принять, что запятая фиксируется перед первым разрядом находящегося в генераторе кода ***X*** = [$X_1, X_2, \ldots X_8$], то получаем: $N = \sum_{i=1}^{8} 0{,}5^i X_i$. В случае целых чисел $N = \sum_{i=1}^{8} 2^{8-i} X_i$.

Найдём характеристики последовательности чисел *N* на полном периоде, полагая, что значения этих чисел определяются приведёнными выражениями и находятся в (0, 1). Математическое ожидание:

$$M(N) = \sum_{i=1}^{n} 0{,}5^i \, M(N_i) = \frac{2^{n-1}}{2^n - 1} \sum_{i=1}^{n} 0{,}5^i = 0{,}5.$$

Среднее квадратическое отклонение [10]:

$$\sigma(N) = \sqrt{\frac{1}{2^n - 1} \sum_{i=1}^{2^n - 1} \left(\frac{i}{2^n} - 0{,}5 \right)^2} = \sqrt{\frac{2^{n-1} - 1}{3 * 2^{n+1}}} \approx \frac{1}{2 \cdot 3^{0{,}5}} \quad \text{при больших } n.$$

Важной характеристикой последовательности генерируемых чисел *N* является и её автокорреляционная функция. Наличие корреляции в этой последовательности может привести к неверным выводам при решении задач. Поэтому стараются уменьшить корреляцию символов на близких шагах. Обычно пользуются нормированной корреляционной функцией *r*(τ), представляющей последовательность значений коэффициентов корреляции для дискретных значений τ.

Для последовательностей чисел *N*, получаемых на основе М-последовательностей, значения коэффициентов корреляции на полном периоде проще всего найти путем моделирования работы генераторов этих последовательностей. Результаты моделирования начальных участков выходных последовательностей псевдослучайных чисел *N* трех генераторов, приведённых на рис. 2.2.*a*, *b* и *c*, представлены в табл. 2.1.

Таблица 2.1. Значения коэффициентов корреляции $r(\tau)$ для начальных участков полного периода последовательностей $N(t)$, генерируемых чисел *N*

Значения τ	0	1	2	3	4	5	6	7
Для схемы *a*	1	0,494	0,241	0,115	0,051	0,019	0,000	-0,059
Для схемы *b*	1	0,003	-0,002	-0,003	-0,002	-0,002	-0,001	-0,001
Для схемы *c*	1	0,108	0,049	0,023	0,011	0,004	-0,001	-0,004
Значения τ	8	9	10	11	12	13	14	15
Для схемы *a*	-0,012	-0,012	-0,012	-0,012	-0,012	-0,012	-0,012	-0,012
Для схемы *b*	-0,002	-0,002	-0,001	-0,001	-0,002	-0,001	-0,001	-0,000
Для схемы *c*	-0,004	-0,003	-0,003	-0,002	-0,001	-0,001	-0,002	-0,001

Результаты, приведенные в табл. 2.1 для схемы 2.2.*a*, можно получить и расчетным путём с помощью следующей зависимости, вывод которой приводится в [30]:

$$r(\tau) = \begin{cases} \frac{2^n(2^{n-\tau}-3)-2^\tau+3}{(2^n-1)(2^n-2)} & \text{при } |\tau| \le n \pmod n, \\ -3/(2^n-2) & \text{в противном случае} \\ & \text{(фоновое значение } r(\tau)). \end{cases} \qquad (2.1)$$

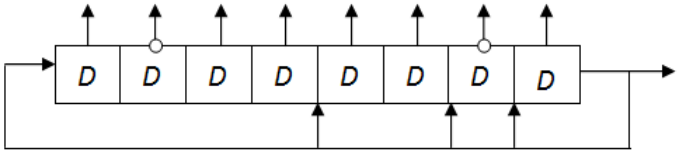
Рассмотренные функции являются периодическими (период $L = 2^n - 1$) и симметричными относительно значения τ, равного нулю. При этом *сумма значений функции на полном периоде равна нулю*:

$$r(-\tau) = r(\tau), \qquad \sum_{\tau=0}^{L-1} r(\tau) = \sum_{\tau=1}^{L} r(\tau) = 0. \qquad (2.2)$$

Отметим, что эти свойства справедливы не только для циклических последовательностей псевдослучайных чисел на основе М-последовательностей, но и для циклических последовательностей псевдослучайных чисел любого алгоритма их формирования.

Упражнения. 1) Доказать, что для произвольной циклической последовательности чисел X_t, принимаемых за случайные, сумма всех коэффициентов корреляции $r(X_t, X_{t+\tau})$ на полном периоде *L* (τ = 1, 2, …, *L*) равна нулю.

2) Вопрос: Какой код не генерирует схема, приведённая на рисунке?



2.2.2. Основные методы улучшения характеристик генераторов псевдослучайных кодов (ГПСК)

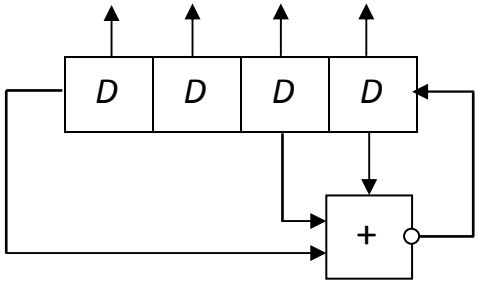
Метод 1. Выравнивание вероятностей *P*(1) и *P*(0) путём использования М-1-последовательностей. При небольшом периоде генерируемых кодов, когда важно, чтобы вероятности появления символов 0 и 1 были бы равны 0,5, вместо М-последовательностей можно использовать М-1-последовательности, длина периода которых равна $2^n - 2$, то есть на один шаг меньше, чем у М-последовательности.

Производящий многочлен рассматриваемого *n*-разрядного генератора образуется согласно выражению $G(D) = (D + I)G_{\text{пр}}(D)$, где $G_{\text{пр}}(D)$ – $G_{\text{пр}}(D)$ – производящий многочлен *n*-1-разрядного генератора М-последовательности. Причем если М-последовательность формируется согласно выражению $G_{\text{пр}}(D)X = 0$, то для М-1-последовательности формирующее её выражение имеет вид: $G(D)X = 1$. Введение единицы в это выражение означает инверсию.

Рассмотрим пример. Пусть выбран $G_{\text{пр}}(D) = D^3 + D^2 + I$. Тогда $G(D) = (D + I)(D^3 + D^2 + I) = D^4 + D^2 + D + I$. Поэтому выражение, формирующее М-1-последовательность, имеет вид: $(D^4 + D^2 + D + I)X = 1$. Следовательно:

$$X = D^4X + D^2X + DX + 1.$$

Вариант схемы, реализующей это выражение, приведен на рисунке. Инвертирование двоичных символов можно осуществлять не на выходе сумматора, а на выходе или на входе одного из D-триггеров.



При сложении по используемому простому модулю М- и М-1-последовательностей получается последовательность, период которой равен произведению их периодов.

Метод 2. Уменьшение фоновых значений *r*(τ). При решении некоторых задач желательно, чтобы фоновые значения *r*(τ) были бы очень маленькими. Это можно сделать путем инвертирования некоторых выходов генератора кодов [40]. Действительно, инверсия x_i дает символ $1 - x_i$. Но это значение можно представить как $1 + (-1) * x_i$. Следовательно, при изменении знака одной из случайных величин коэффициент корреляции этих случайных величин изменяет знак. Поэтому в общем выражении для коэффициента корреляции псевдослучайных чисел

$$r[X(t), X(t+\tau)] = \sum_{I=1}^{n} \sum_{j-1}^{n} 0{,}5^i \, 0{,}5^j \, r(x_i(t) x_j(t+\tau))$$

соответствующие значения коэффициентов корреляции двоичных символов поменяют знак. Исследования показывают [7д], что оптимальным вариантом, когда абсолютные значения $r(X(t), X(t+\tau))$ снижаются максимально, является либо инвертирование только старшего разряда кода числа, либо всех разрядов, кроме первого. В этом случае для фонового значения $r_{\text{фон}}[X(t), X(t+\tau$ получим следующее выражение[30]:

$$r_{\text{фон}}[X(t), X(t+\tau)] = {}^{-3}/_{(2^{3n} - 2^{2n} - 3^n - 2),}$$

что существенно меньше фонового значения $r_{\text{фон}}$(τ) в выражении (2.1).

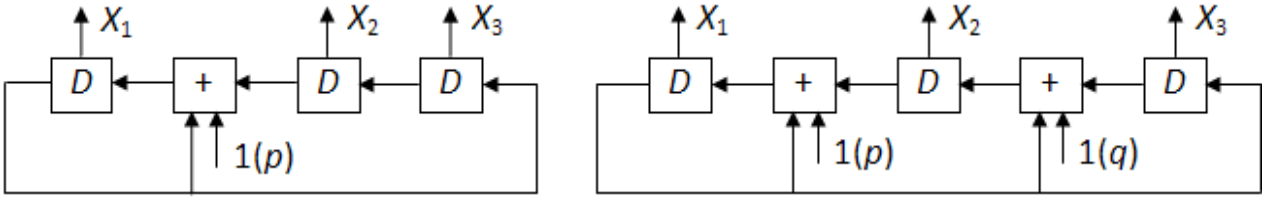
2.2.3. Построение высококачественных генераторов случайных чисел

Рассматриваемые генераторы, формирующие не псевдослучайные, а случайные числа, можно создавать путем встраивания в структуры генераторов псевдослучайных кодов физических датчиков случайных символов 0 и 1. Но

они, как указывалось в п. 2.1, не позволяют добиваться практически равновероятного появления указанных символов. Однако рассматриваемый метод построения генераторов позволяет решить данную проблему.

Рассмотрим принципы построения и функционирования указанных генераторов, называемых часто комбинированными, так как они объединяют генераторы псевдослучайных и случайных кодов, формируя случайные коды.

На рис. 2.3 приведены схемы двух комбинированных трехразрядных генераторов случайных кодов (ГСК) $\boldsymbol{X} = (X_1, X_2, X_3)$. Датчики случайных символов посылают в эти схемы символы 1 с вероятностями p и q. Для упрощения математических выводов указанные вероятности будем считать постоянными.



a) *b*)

Рис. 2.3. Комбинированные ГСК

Проанализируем на случайность последовательности кодов $\boldsymbol{X}$ и их разрядов. Для этого от генерируемых кодов удобнее перейти к состояниям рассматриваемых схем. Так как на каждом шаге t в каждой схеме может быть записан один из восьми кодов (двоичных чисел), то введем следующее взаимно однозначное отображение (соответствие) 3-разрядных кодов $\boldsymbol{X}$ и координат P_i 8-координатного вектора $\boldsymbol{P}$ вероятностей нахождения генераторов в соответствующих состояниях:

000 – 00000001, 001 – 00000010, 010 – 00000100, 011 – 00001000,

100 – 00010000, 101 – 00100000, 110 – 01000000, 111 – 10000000.

Если процесс перехода генераторов или какой-либо другой цифровой системы из одного состояния в другое является случайным, то координаты векторов состояний можно использовать для задания вероятностей нахождения генераторов в соответствующих состояниях. В этом случае координаты векторов состояний могут принимать любые положительные значения от нуля до единицы. В этом случае векторы состояний будем обозначать буквой $\boldsymbol{P}$.

Итак, запись $\boldsymbol{P}(t) = (P(7), P(6), \ldots, P(0)) = (0,\ 0,\ 0{,}1,\ 0{,}1,\ 0{,}2,\ 0{,}3,\ 0{,}2,\ 0{,}1)$ означает, что на шаге t с вероятностью 0,3 генератор находится в состоянии

00000100, то есть в нем находится код 010, а коды 110 и 111 на шаге t появиться не могут. Аналогичный смысл имеют и другие координаты вектора $\boldsymbol{P}(t)$. Сумма элементов векторов $\boldsymbol{P}$ равна, очевидно, нулю.

При анализе работы рассматриваемых генераторов удобно пользоваться матричным аппаратом, а именно – матрицами переходов $\boldsymbol{A}^t$, элементы которых задают вероятности переходов генератора из каждого состояния в любое из его состояний за произвольное число шагов t. В этом случае векторы $\boldsymbol{P}(t)$ определяются согласно уравнением Колмогорова[2] – Чепмена, широко используемым в теории марковских процессов и в различных технических и других приложениях:

$$\boldsymbol{P}(t) = \boldsymbol{A}^t \boldsymbol{P}(0), \qquad (2.3)$$

где $\boldsymbol{P}(0)$ – вектор, задающий начальное состояние рассматриваемой системы.

Матрица $\boldsymbol{A}$ для генератора на рис. 2.5*a* (матрица справа) содержит только вероятности p и $1–p$, так как в этом генераторе есть только один источник случайности. Ниже приведена матрица $\boldsymbol{A}$ для генератора на рис. 2.5.*b*. Она содержит уже 4 вероятности, соответствующие двум входящим в генератор источникам случайности.

$$\boldsymbol{A} = \begin{pmatrix} 1\text{-}p & 0 & 0 & 0 & p & 0 & 0 & 0 \\ 0 & 0 & 1\text{-}p & 0 & 0 & 0 & p & 0 \\ p & 0 & 0 & 0 & 1\text{-}p & 0 & 0 & 0 \\ 0 & 0 & p & 0 & 0 & 0 & 1\text{-}p & 0 \\ 0 & p & 0 & 0 & 0 & 1\text{-}p & 0 & 0 \\ 0 & 0 & 0 & p & 0 & 0 & 0 & 1\text{-}p \\ 0 & 1\text{-}p & 0 & 0 & 0 & p & 0 & 0 \\ 0 & 0 & 0 & 1\text{-}p & 0 & 0 & 0 & p \end{pmatrix}$$

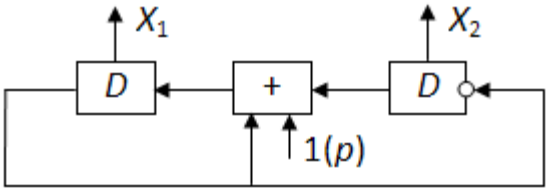
$$\boldsymbol{A} = \begin{pmatrix} (1\text{-}p)(1\text{-}q) & 0 & (1\text{-}p)q & 0 & p(1\text{-}q) & 0 & pq & 0 \\ (1\text{-}p)q & 0 & (1\text{-}p)(1\text{-}q) & 0 & pq & 0 & p(1\text{-}q) & 0 \\ p(1\text{-}q) & 0 & pq & 0 & (1\text{-}p)(1\text{-}q) & 0 & (1\text{-}p)q & 0 \\ pq & 0 & p(1\text{-}q) & 0 & (1\text{-}p)q & 0 & (1\text{-}p)(1\text{-}q) & 0 \\ 0 & p(1\text{-}q) & 0 & pq & 0 & (1\text{-}p)(1\text{-}q) & 0 & (1\text{-}p)q \\ 0 & pq & 0 & p(1\text{-}q) & 0 & (1\text{-}p)q & 0 & (1\text{-}p)(1\text{-}q) \\ 0 & (1\text{-}p)(1\text{-}q) & 0 & (1\text{-}p)q & 0 & p(1\text{-}q) & 0 & pq \\ 0 & (1\text{-}p)q & 0 & (1\text{-}p)(1\text{-}q) & 0 & pq & 0 & p(1\text{-}q) \end{pmatrix}$$

[2] Колмогоров А.Н. (1903 – 1987) – академик АН СССР, профессор МГУ. Чепмен – британский учёный.

Обе матрицы являются дважды стохастическими (сумма элементов каждой их строки и сумма элементов каждого их столбца равны единице). Поэтому при $t \to \infty$ выражение 2.3 имеетсвоим пределом финальный вектор $\boldsymbol{P}$ = (1/8, 1/8, . . . , 1/8), то есть с увеличением числа шагов генераторов их состояния стремятся к равновероятным. Однако число шагов (период), через которое с генераторов можно снимать “практически равновероятные Коды зависит от параметров схемы генератора и соответственно матрицы $\boldsymbol{A}$, входящей в выражение 2.3. Указанная характеристика обычно является предметом исследований.

Результаты исследований последовательностей кодов формируемых различными ГПСК приведены в [7, 23 – 25];

Упражнения. 1) Построить матрицу переходов, задающую работу двухразрядного ГПСП (на рисунке справа) и матрицу его двухшаговой работы.



2) Построить схему выравнивания вероятностей появления символов 0 и 1 случайной последовательности на основе сумматора по модулю два и элемента задержки на один шаг. Найти зависимость вероятности появления символа “1” в выходной последовательности от вероятности его появления на входе схемы.

2.2.4. Использование генераторов М-последовательностей и псевдослучайных кодов для формирование и обработки сложных сигналов

Рассматриваемые генераторы используются в системах связи, управления, контроля, радиолокации, скрытой передачи информации и т.д.

В ряде специальных систем связи, в первую очередь в системах обнаружения и опознавания наземных, воздушных, космических, надводных и подводных объектов для повышения надежности обнаружения и опознавания используются так называемые сложные сигналы. Такие сигналы представляют собой высокочастотные синусоидальные колебания, промодулированные некоторой последовательностью, в частности, М-последовательностью. Они посылаются по обозреваемому направлению и в случае наличия на этом направлении обнаруживаемого объекта отражаются от него и принимаются запрашивающим объектом (обычно – радиолокатором). Обработка принятых сложных сигналов с наложившимися на них помехами позволяет выделять одиночный импульс, свидетельствующий о наличии определенного объекта (самолета,

подводной лодки и др.) на соответствующем направлении и на определенной дальности даже в том случае, когда уровень естественных и специально создаваемых помех в 100 и более раз превышает полезный сложный сигнал [22, 26, 28, 29]. Поэтому указанные сложные сигналы и получили широкое применение в системах передачи информации с высоким уровнем помех.

Рассмотрим упрощенный вариант схемы формирования излучаемого сложного сигнала, цифровой фильтрации принимаемого сигнала и получения одиночного импульса, указывающего на наличие объекта (в частности, цели) в зоне обзора (рис. 2.4).

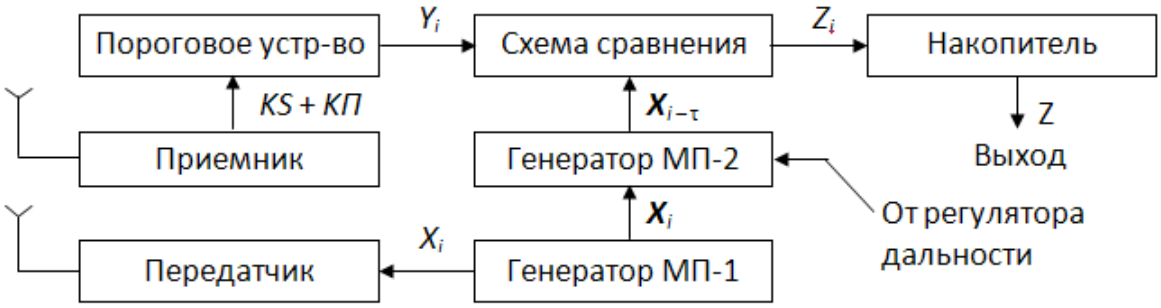


Рис. 2.4. Схема формирования и обработки сложных сигналов РЛС с использованием М-последовательностей

Генератор МП-1 используется для модуляции передаваемых высокочастотных колебаний, которые излучаются в выбранном направлении антенной передатчика. Принятая смесь отраженного от находящегося в “просматриваемой” зоне объекта сигнала *S* (или от нескольких таких объектов) и помех *П* усиливается и демодулируется приемником, а затем поступает на пороговое устройство. В зависимости от знака входного сигнала пороговое устройство выдаёт положительные или отрицательные импульсы Y_i, соответствующие двоичным символам 1 и 0.

Схема сравнения сравнивает импульсы Y_i с импульсами $X_{i-\tau}$ от генератора МП-2, которые по сравнению с импульсами X_i задержаны на τ шагов соответственно времени прохождения сигнала от радиолокатора до обнаруженного объекта и обратно. Указанное сравнение производится на полном периоде М-последовательности, то есть на $2^n - 1$ шагах. Если полярности импульсов $X_{i-\tau}$ и Y_i совпадают, то схема сравнения выдает на накопитель импульсов положительный импульс (двоичный символ) $Z_i = 1$; в противном случае на накопитель выдается отрицательный импульс (двоичный символ $Z_i = -1$). На полном периоде М-последовательности на накопитель поступает $2^n - 1$ таких символов).

Накопитель складывает входные символы 1 и -1 на полном периоде сравнения $X_{i-\tau}$ и Y_i, то есть на полном периоде М-последовательности. Если, например, на периоде 1023 шага будет зарегистрировано 713 совпадений $X_{i-\tau}$ и Y_i, и соответственно 310 их несовпадений, то суммарный сигнал Z, выдаваемый на экран локатора, составит 403 условные единицы. На практике для устранения колебаний величины этого сигнала, неудобных для наблюдения оператором, обычно его величину делают стандартной.

Направление обзора обычно регулируется поворотом антенн с помощью соответствующей электромеханической системы или путём соответствующей электронной настройки их диаграмм направленности. На летательных аппаратах используется только второй метод.

Рекомендуем читателям самим разобраться в следующих вопросах:

1. Почему нестандартизированное значение сигнала Z на экране радиолокатора колеблется?
2. Чему равно среднее значение нестандартизированного сигнала Z, если период М-последовательности составляет 511 шагов, значения сигналов Z_i равны 1 и -1, а вероятность неправильного приема символа М-последовательности равна 0,2?
3. Через какой интервал времени после начала излучения высокочастотного сигнала, промодулированного М-последовательностью с периодом 1023 шага, этот сигнал после отражения от цели будет полностью принят приёмником радиолокатора, если частота формирования сигналов X_i равна 5 МГц?
4. Что общего и в чём состоит отличие двоичного и бинарного сигналов?
5. Почему на летательных аппаратах для определения направлений на обнаруженный объект не применяются системы с поворотом антенн?

2.3. Моделирование дискретных случайных величин и случайных событий

Пусть необходимо построить алгоритм моделирования дискретной случайной величины (ДСВ) $X \in \{x_1, x_2, \dots, x_n\}$. Будем вначале полагать, что указанные значения ДСВ независимые и должны появляться с вероятностями $P(x_i)$. В этом случае функция распределения данной ДСВ определяется выражением

$$F(x) = \begin{cases} 0 & \text{при } x < x_1, \\ \sum\limits_{\forall x_i \le x} P(x_i) & \text{при } x \in [x_1, x_n], \\ 1 & \text{при } x > x_n. \end{cases}$$

Данное выражение является общей формой задания произвольных распределений ДСВ.

Для моделирования ДСВ можно использовать общий метод, который является универсальным. Но в ряде случаем для моделирования проще применить соответствующий частный метод.

2.3.1. Общий метод моделирования

Алгоритм моделирования очередной независимой ДСВ $X \in \{x_1, x_2, \dots, x_n\}$ с произвольным распределением состоит из следующих этапов:

– Отрезок [0, 1] разбивается на *n* непересекающихся участков, длины которых равны $P(x_i)$ или кратко – P_i . При этом устанавливается взаимно-однозначное соответствие множества указанных участков и множества генерируемых ДСВ X, при этом участку длины P_i сответствует значение x_i.
– Моделируется НСВ *Y* с равномерным распределением в интервале [0, 1]. Она попадает на один из указанных участков. Значение x_i, соответствующее этому участку, принимается за реализацию *X.*

При реализации первого этапа алгоритма исходят из того, что вероятность $P(X = x_i) = P_i$ попадания случайной величины, равномерно распределённой в [0, 1], на некоторый участок этого отрезка равна длине данного участка. При этом указанные участки не должны пересекаться, но расположение их на отрезке [0, 1] может быть произвольным. Так, при моделировании ДСВ, принимающей три значения с вероятностями $P_1 = 0{,}5$, $P_2 = 0{,}2$ и $P_3 = 0{,}3$ возможны 6 вариантов их расположения:

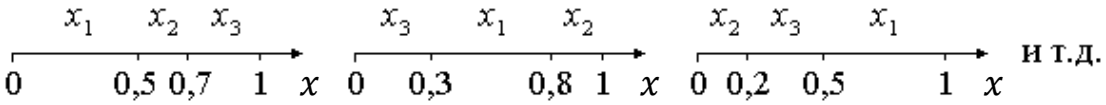
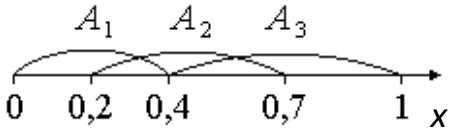


Рис. 2.5. Примеры реализации первой части алгоритма моделирования.

Аналогично моделируются и несовместные случайные события. При моделировании совместных событий участки разбиения отрезка [0, 1] пересекаются. Так, для моделирования совместных случайных событий A_1, A_2 и A_3 при $P(A_1) = 0{,}4$, $P(A_2) = 0{,}5$ и $P(A_3) = 0{,}6$ можно воспользоваться вариантом моделирования, поясняемым на рис. 2.6.

Рис. 2.6. Вариант моделирования совместных событий



Для несовместных случайных событий справедливо следующее выражение, приводимое практически во всех учебниках по теории вероятностей:

$$P(A_1 + A_2 + A_3) = P(A_1) + P(A_2) + P(A_3) - P(A_1 \cdot A_2) - P(A_1 \cdot A_3) - P(A_2 \cdot A_3) + P(A_1 \cdot A_2 \cdot A_3) = 1.$$

Рекомендуем читателям написать по аналогии выражение для вероятности суммы четырёх несовместных случайных событий.

Если необходимо промоделировать последовательность зависимых ДСВ или зависимых случайных событий, то на каждом шаге алгоритма моделирования нужно учитывать результаты последнего шага или (редко) нескольких последних его шагов. В последнем случае число указанных шагов определяется так называемым “интервалом зависимости” или “интервалом корреляции”, то есть интервалом τ, по достижению которого случайные события $A(t)$ и $A(t+\tau)$ или случайные величины $X(t)$ и $X(t+\tau)$ “можно принять” соответственно за независимые или за некоррелированные.

Пример. Моделирование простой цепи Маркова, на каждом шаге которой может появиться одно из трех несовместных событий: *A*, *B* или *D*. Вероятностная связь зависимости этих событий определяется стохастической матрицей переходов *S*, приведенной справа.

$$S = \begin{array}{c} \begin{matrix} A & B & D \end{matrix} \\ \begin{pmatrix} 0{,}2 & 0{,}5 & 0{,}3 \\ 0{,}5 & 0{,}1 & 0{,}4 \\ 0{,}3 & 0{,}3 & 0{,}4 \end{pmatrix} \end{array} \begin{matrix} A \\ B \\ D \end{matrix}$$

Выберем варианты разбиения отрезка [0, 1] изменения случайной величины X, имеющей на этом отрезке равномерный закон распределения, для каждой строки матрицы *S*. Учитывая значения вероятностей переходов одного события (состояния) к другому, приведенные в заданной для реализации матрице переходов, для получаемого значения *X* принимаем:

1-я строка: [0, 0,5) для перехода от *A* к *B*; [0,5, 07) для перехода от *A* к *A*; [0,7, 1] для перехода от *A* к *D*.

2-я строка: [0, 0,5) для перехода от *B* к *A*; [0,5, 0,6) для перехода от *B* к *B*; [0,6, 1] для перехода от *B* к *D*.

3-я строка: [0, 0,3) для перехода от *D* к *A*; [0,3, 0,7) для перехода от *D* к *D*; [0,7, 1] для перехода от *D* к *B*.

Теперь программная реализация моделирования на *k*-м шаге очередного случайного события, в которой появление некоторого события представляется приданием идентификатору этого события значения единица, может выглядеть следующим образом:

```
X:=random;
if A=1 then if X<0.5 then begin B:=1; A:=0; end else if X> 0.7 then begin D:=1; A:=0; end;
if B=1 then if X<0.5 then begin A:=1; B:=0; end else if X>0.6 then begin D:=1; B:=0; end;
if D=1 then if X<0.3 then begin A:=1; D:=0; end else if X>0.7 then begin B:=1; D:=0; end;
```

Упражнения. 1) Дополнить приведённую программу определением оценок значений безусловных вероятностей *P*(*A*=1), *P*(*B*=1) и *P*(*D*=1).

2) Определить теоретические значения указанных вероятностей.

При моделировании ДСВ с заданным распределением ввиду некоторого отличия распределения генерируемой ЭВМ исходной случайной величины *C* от равномерного или ввиду конечности числа реализаций ДСВ характеристики статистического распределения обычно отличаются от одноимённых характеристик моделируемого распределения. Характеристики статистического распределения, как известно, называют оценками характеристик моделируемого распределения. При обозначении оценок над соответствующими идентификаторами ставится черта. Обычно определяются значения оценок начальных ($\overline{\alpha}$) и центральных ($\overline{\mu}$) моментов *k*-го порядка и среднее квадратическое отклонение оценки математического ожидания ДСВ *X* при *N* реализациях *n* её значений:

$$\overline{\alpha}_k = \sum_{i=1}^{n} \overline{P}_i \, x_i^k, \quad \overline{\mu}_k = \sum_{i=1}^{n} \overline{P}_i \left(x_i - \overline{M}_X\right)^k, \quad \sigma\left(\overline{M}_X\right) = \overline{\sigma}_X / \sqrt{N}.$$

Кроме приведённых характеристик для ДСВ иногда используется коэффициент асимметрии или скошенности (skew – косой) распределения: $\overline{As}(X) = \bar{\mu}_3 / \bar{\sigma}_x^3$. Если имеет место скошенность влево, то асимметрия положительная, если вправо – отрицательная. Делением на куб среднего квадратического отклонения достигается безразмерность этого коэффициента. Аналогично определяется и коэффициент эксцесса (остроты распределения):

$$\overline{Ex}(X) = \bar{\mu}_4 / \bar{\sigma}_x^4 \, .$$

В ряде случаев из этого отношения вычитают еще число 3, что делается для сравнения остроты рассматриваемого распределения с остротой гауссового распределения, которая в этом случае равна нулю. Тогда если некоторое распределение является более острым, чем гауссово, то его эксцесс больше нуля, если же оно менее острое, то его эксцесс меньше нуля.

2.3.2. Частные методы

Для реализации ДСВ, подчиняющихся некоторым распределениям, проще воспользоваться частными методами. В качестве таких примеров рассмотрим случаи моделирования ДСВ с биномиальным, геометрическим и треугольным распределением Пирсона.

Биномиальное распределение. Это распределение обозначатся *P*(*m*, *n*, *p*), где *m*, *n*, и *p* – его параметры. Случайная величина X, имеющая биномиальное распределению, принимает значения из $\{x_0, x_1, \dots, x_m, \dots, x_n\}$ с вероятностями $P(m,n,p) = C_n^m p^m (1-p)^{n-m}$.

Биномиальное распределение может быть общего и частного вида. В первом случае разность $x_{i+1} - x_i$ значений ДСВ *X* может быть произвольной. При этом сами значения *x* могут быть не только положительными, но и отрицательными. В распределениях частного вида эта разность равна единице, причем $x_i = i$, а $M(X) = np$.

Обычно используется следующий метод моделирования:

- Для получения одной ДСВ *X* моделируются *n* вспомогательных ДСВ *Y*,- принадлежащих (0, 1). При этом $P(Y=1)=p$, $P(Y=0)=(1-p)$.
- Если реализуется частный, широко распространённый вид биномиального распределения, при котором $x_0 = 0$, $x_1 = 1, \dots x_n = n$, то ДСВ *X* находится как сумма *n* указанных ДСВ *Y*.
- Если реализуется биномиальное распределение общего вида, то вначале находится значение s_i суммы *S* *n* ДСВ *Y*, по которой затем согласно графику соответствия множества значений *S* и множества значений *X* определяется моделируемое значение x_i. В простейшем случае это может

  быть линейная зависимость вида $x_i = k \cdot s_i$, где *k* – некоторое число. Однако график соответствия может быть и произвольным.

Геометрическое распределение: $P_k = p(1-p)^{k-1}$. Этому распределению подчиняется, например, количество изделий, прошедших контрольную

проверку, до фиксирования первого изделия, не соответствующего техническим условиям. Оно часто используется в различных приложениях, например, при моделировании помех в каналах связи.

Так как СВ *k*, подчиняющаяся геометрическому распределению, принимает значения из натурального ряда, то получаем: $F(k) = 1 - (1-p)^k$.

Рассмотрим примеры двух программ, моделирующих k_i значений ДСВ, подчиняющихся геометрическому распределению:

Первая программа

```
k:=0; x:=0;
while x=0 do
begin
  k:= k+1;
  y:= random;
  if y < p then x:=1;
end;
```

Вторая программа

```
k:=0; x:=0;
repeat
   k:= k+1;
   y:= random;
   if y < p then x:=1;
until x=0;
```

Приведенные программы отличаются методом реализации циклов. Получаемые числа *k* имеют геометрическое распределение.

Примечание: геометрическое распределение можно рассматривать и в виде $P_k = (1-p)p^{k-1}$. В этом случае $F(k) = 1 - p^k$.

Треугольное распределение Пирсона: $P_i = \frac{1}{n} - \frac{|i|}{n^2}$, где $|i| \le n$, то есть $i \in [-n, \dots, n]$. При этом $x_i = i$, $M(x) = 0$, $\sigma(x) = \sqrt{\frac{n^2-1}{6}}$.

Данное распределение возникает, например, при сложении двух ДСВ, значения которых изменяются с постоянным шагом и равновероятны. Справа приведен график дискретного распределения Пирсона для случая изменения значений моделируемой ДСВ *X* с шагом 1 при *n* = 4. Математическое ожидание этого распределения можно изменять, добавляя к значениям x соответствующие числа. Простая программа моделирования очередного значения ДСВ *X* при чётном значении *n* может быть такой:

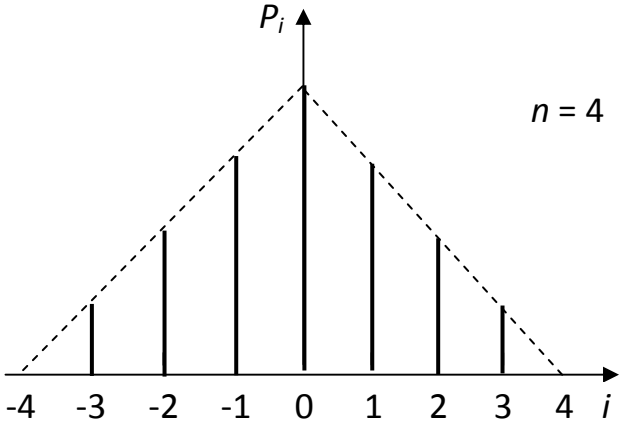


Рис.2.7. График распределения Пирсона

y:= random (round(n/2));

z:= random (round(n/2));

x:= y + z – round(n/2);

Предлагаем читателям самим разобраться в назначении каждого оператора этой программы.

Упражнения

1) Моделируется ДСВ *X*, принимающая значения из $\{x_1,\ x_2,\ x_3\}$ соответственно с вероятностями 0,5, 0,2 и 0,3. Однако используемая для этого случайная величина *C* распределена на [0, 1] не равномерно: функция плотности этой случайной величины равна 0,8 на участке от 0 до 0, 2, 1 на участке от 0,2 до 0,8 и 1,2 на участке от 0,8 до 1. С какими вероятностями появляются значения $x_1,\ x_2,\ x_3$?

2) Построить алгоритм моделирования дискретной СВ $X \in \{x_1,\ x_2,\ x_3,\ x_4,\ x_5\}$, функция распределения $F(x)$ которой приведена на рис. 2.8. Нарисовать гистограмму этого распределения.

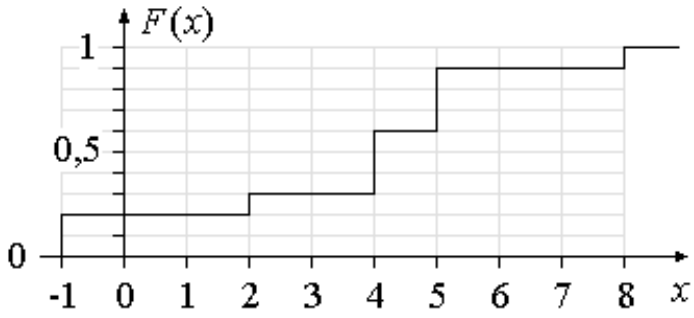


Рис. 2.8. График функции *F*(*x*)

3) Составить алгоритм моделирования ДСВ $X \in \{-4, 1, 3, 9\}$, подчиняющейся биномиальному распределению *P*(*m*, 3, 0,4), и определить вероятности появления указанных значений x_i а также значения математического ожидания *M*(*X*) и среднего квадратического отклонения $\sigma(X)$.

4) Предложить простой алгоритм моделирования ДСВ *X*, принимающей значения $n = 1, 2, 3, \ldots$ из натурального ряда и подчиняющейся геометрическому распределению: $P(n) = 0{,}5^n$.

5) Доказать, что для геометрического распределения ДСВ *X* справедливо: $M(x) = {}^1\!/_p$, $D(x) = (1-p)\cdot p^{-2}$. Подготовить программу моделирования этого распределения.

6) Предложить алгоритм моделирования ДСВ *X* с треугольным распределением Пирсона при нечетном значении *n*.

2.4. Моделирование непрерывных случайных величин

Для моделирования многих непрерывных случайных величин (НСВ) используются общие методы, которые позволяют моделировать НСВ с большинством встречающихся на практике распределений. Однако для моделирования некоторых НСВ они неудобны, например, для моделирования НСВ с нормальным законом распределения. В таких случаях используют частные методы, позволяющие значительно проще моделировать указанные НСВ .

2.4.1. Общие методы моделирования

Рассмотрим три общих (универсальных) метода моделирования.

Метод обратной (квантильной) функции. В данном случае речь идет о функции, значения которой являются квантилями функции распределения моделируемой НСВ, то есть аргументами функции распределения при $F(x) \in [0, 1]$. Аналитически функцию, обратную функции распределения $y = F(x)$ записывают следующим образом: $x = F_x^{-1}(y)$. При этом -1 означает не возведение в отрицательную степень, а определение квантиля от значения y.

При моделировании очередной НСВ согласно данному методу сначала с помощью команды random моделируют значение НСВ y, равномерно распределенной в (0, 1), а затем вычисляют $x = F_x^{-1}(y)$.

Реализация рассматриваемого метода пояснена на рис. 2.9. Из рисунка не следует, что метод обратной функции пригоден только для моделирования НСВ X, изменяющихся на конечном отрезке. Например, при $Y = F(x) = 1 - e^{-x}$ обратная функция имеет вид: $x = -\ln(1 - y)$ или $x = -\ln(y)$, так как НСВ $1 - X$ и X имеют одинаковые законы распределения при $x \in [0, \infty)$.

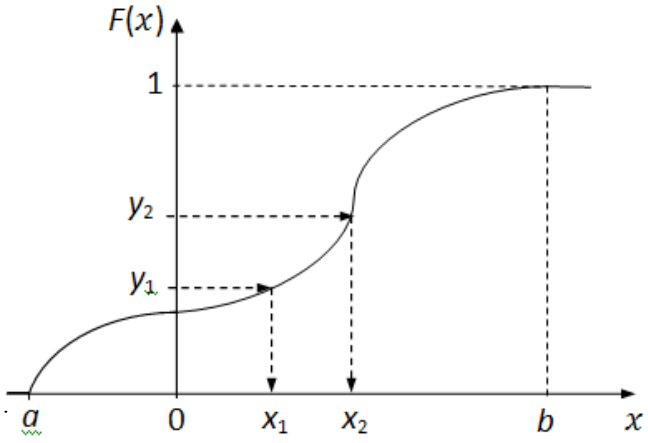


Рис. 2.9. Реализация метода обратной функции

Несложно убедиться, что получаемая согласно данному методу ДСВ X имеет заданную функцию распределения $F(x)$. Действительно, вероятность

попадания значений НСВ *X* на произвольный отрезок $[x_1, x_2]$, принадлежащий области определения функции *F*(*x*), равна вероятности попадания моделируемой НСВ *Y* на отрезок [y_1, y_2], а эта вероятность при равномерном распределении *Y* в (0, 1) равна $F(x_2) - F(x_1)$. Последнее свидетельствует о том, что НСВ *Y* подчиняется распределению с заданной $F(x)$.

Пример: $x \in [1,\ 2]$, $F(x) = \log_2 x = y$. Находим обратную функцию: $x = 2^y$. Преобразуем её к виду, удобному для программирования: ln(x) = y∗ ln(2), то есть $x = \exp(y \cdot \ln(2))$. Моделирование выполнятся с помощью двух операторов: $y := random$; $x := \exp(y * \ln(2))$.

Если найти аналитическое выражение для обратной функции невозможно, то её можно аппроксимировать. При этом погрешность аппроксимации приводит и к соответствующей погрешности в реализации обратной функции. В случае кусочно-линейной аппроксимации каждый участок $[x_i, x_{i+1}]$ обратной функции задается линейным выражениям $x = x_i + (x_{i+1} - x_i)/(y_{i+1} - y_i) * (y - y_i)$, согласно которому по полученному по полученному значению *y* вычисляется *x*.

При моделировании НСВ с чётной функцией плотности, то есть с нечетной функцией $F(x) - 0{,}5$, можно моделировать только положительные значения *x*, получать, например, ДСВ z:= random(2), и воспроизводить отрицательные значения *x* с помощью оператора if *z*=0 then *x*:=−*x*.

Упражнения. 1) Моделируется НСВ *X* с чётной функцией плотности. При этом вначале каждый раз получают положительное значение x, а затем, моделируя НСВ z:= random, соответствующим значениям *x* приписывают знак минус. С помощью какого оператора можно выполнить последнее преобразование x?

2) Подготовить программу или алгоритм моделирования НСВ X с функцией распределения $F(x) = \mathrm{tg}\big((x^2 - 1)\pi/12\big)$ при $x \in [1,\ 2]$.

3) Для моделирования НСВ *X* методом обратной функции была найдена $F(x)^{-1} = \exp(y \cdot \ln 2)$. Найти выражение для $F(x)^{-1} = \exp(y \cdot \ln 2)$ и определить интервал, в котором будут изменяться моделируемые значения x.

Метод суперпозиции (композиции) используется совместно с методом обратной функции в тех случаях, когда обратную функцию найти невозможно, но заданную *F*(*x*) можно представить в виде линейной суммы

$$F(x) = \sum_{i=1}^{n} k_i\, F_i(x), \qquad (2.4)$$

в которой $F_i(x)$ – функции, монотонно возрастающие в области определения $F(x)$ от 0 до 1 и поэтому рассматриваемые как функции распределения, а k_i – весовые коэффициенты (положительные числа), сумма которых равна единице. При этом предполагается, что для каждой $F_i(x)$ можно определить обратную функцию.

При моделировании вначале разыгрывается номер функции $F_i(x)$, которая будет использоваться для определения её квантиля. Иными словами, указанный розыгрыш заключается в выборе события, заключающегося в приравнивании единице одного из *n* коэффициентов k_i и в приравнивании нулю всех остальных k_i (в алгебре *n* таких событий образуют полную группу). Затем моделируется НСВ *Y* и определяется $x = F_i^{-1}(y)$. Таким образом, получаем:

$$P(x \in [x_1,\ x_2]) = \sum_{i=1}^{n}\{P(k_i = 1) \cdot [F_i(x_2) - F_i(x_1)]\}.$$

Несложно доказать [2, 7] , что полученная сумма равна $F(x_2) - F(x_1)$. Следовательно, моделируемая НСВ *X* действительно имеет заданное распределение.

Пример. Заданная для моделирования НСВ должна иметь $F(x) = 0{,}05x + 0{,}1x^2 + 0{,}5\sin(0{,}25\pi x)$ *при* $x \in [0, 2]$. Преобразуем её к виду (2.4), в котором все $F_i(x) \in [0, 1]$. Получаем: $F(x) = 0{,}1 \cdot 0{,}5x + 0{,}4 \cdot 0{,}25x^2 + 0{,}5 \cdot \cdot\sin(0{,}25\pi x)$. В этой формуле в каждом слагаемом выражение, записанное после знака умножения, задаёт соответствующую функцию распределения. Находим обратные функции $F_x^{-1}(y)_i$: $F_x^{-1}(y)_1$: $x = 2y$; $F_x^{-1}(y)_2$: $x = 2y^{0{,}5}$; $F_x^{-1}(y)_3$: $x = 4\pi^{-1} \cdot \arcsin(2y)$.

Ниже приведен вариант программы, моделирующей очередное значение НСВ x.

```
Y:=random;  z:=random;
if z<0.1 then x:=2*y else
if z<0.5 then x:=2*sqrt(y) else
x:=4/pi*arcsin(2*y);
```

В случае необходимости функции $F_i(x)$ или функции $f_i(x)$, а также и $F_x^{-1}(y)$ можно аппроксимировать. В качестве примера рассмотрим моделирование НСВ, подчиняющихся распределению Максвелла, используемого при моделировании электромагнитных полей. Аналитические выражения функции

и плотности распределения таких НСВ *X* имеют следующий вид (*s* – параметр распределения):

$$F(x)=\begin{cases}0 & \text{при } x\le 0,\\ \dfrac{\sqrt{2/\pi}}{s}\displaystyle\int_0^x e^{-\frac{t^2}{s^2}}dt-\dfrac{x}{\sqrt{2\pi}s}e^{-\frac{x^2}{s^2}} & \text{при } x>0;\end{cases}$$

$$f(x)=\begin{cases}0 & \text{при } x\le 0,\\ \dfrac{x^2}{s^3}\sqrt{2/\pi}\,e^{-\frac{x^2}{s^2}} & \text{при } x>0.\end{cases}$$

На рис. 2.10 приведены графики, демонстрирующие вариант аппроксимации функций $f(x)$ и $F(x)$ рассматриваемого распределения, а именно – разложения этих функций на три составляющие.

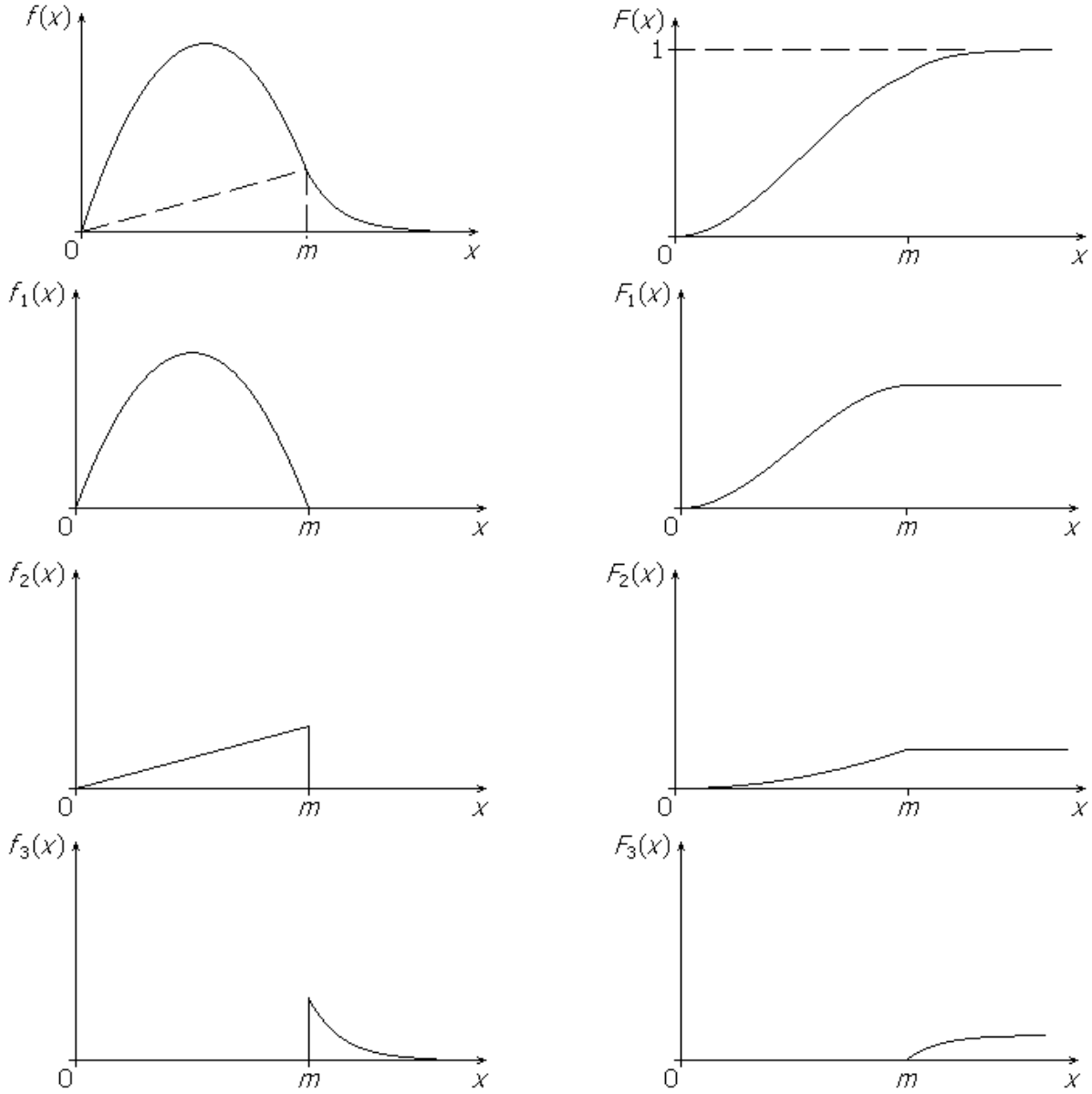


Рис. 2.10. Графики распределения Максвелла и их разложение на составляющие

В рассмотренном примере после получения аналитических выражений для результатов аппроксимации можно использовать для моделирования:

$$f(x) = f_1(x) + f_2(x) + f_3(x) \text{ и } F(x) = F_1(x) + F_2(x) + F_3(x).$$

Однако следует помнить, что полученные составляющие не являются соответственно функциями плотности или функциями распределения. Поэтому они не могут быть использованы для нахождения параметров результирующего распределения с помощью обычно применяемых для этого выражений [9, 50 и др.].

Все приведённые на рисунке графики были получены при одном и том же значении параметра *s*, по которому математическое ожидание M_x и среднее квадратическое отклонение σ_x рассматриваемой НСВ *X* определяются следующим образом:

$$M_x = 2\sqrt{2/\pi}\, s = 1{,}596s; \;\; \sigma_x = \sqrt{3 - 8/\pi}\, s = 0{,}6734s.$$

Упражнения. 1) Составить алгоритм моделирования НСВ *X* с функцией распределения

$$F(x) = \frac{x}{18} + \sin\frac{\pi \cdot \lg(x)}{6} - \frac{1}{18}, \; x \in [1, \; 10].$$

2) Подготовить программу моделирования НСВ *X*, у которой

$$F(x) = 0{,}5 \cdot \log_2(x-1) + \sin\frac{\pi x}{3}, \qquad x \in [2, \; 3].$$

3) Составить алгоритм моделирования непрерывной случайной величины *X*, функция распределения которой имеет вид:

$$F(x) = \begin{cases} x^2 + 0{,}2x & \text{при } x \in [0, \; 0{,}375]; \\ \dfrac{3}{8} + \sqrt{x - \dfrac{3}{8}} & \text{при } x \in [0{,}375, \; 0{,}625\,]; \\ \sin^2\left(\dfrac{\pi x}{2}\right) + \cos^3\left(\dfrac{\pi x}{2}\right) & \text{при } x \in [0{,}675, \; 1]. \end{cases}$$

Метод отбора (метод исключения, метод режекции, метод фон Неймана). Этот метод используется тогда, когда для моделирования удобнее использовать не функцию распределения, а функцию плотности $f(x)$ моделируемых НСВ. Идея метода весьма проста. Пусть в некоторую область *Q*, ограниченную сверху функцией $y(x) \geq 0$ и снизу – осью абсцисс, в которую входит площадь, ограниченная функцией плотности $f(x)$, "бросаются" случайные точки *T*(*x*, *y*) с равномерным распределением (рис. 2.11). При этом вероятность

того, что генерируемая случайная точка окажется в области, ограниченной $f(x)$, равна:

$$P = \frac{\int_a^b f(x)\,dx}{\int_A^B y(x)dx} = \frac{1}{\int_A^B y(x)dx}.$$

Произведение PN, где N – число “бросаемых” в область Q точек, определяет

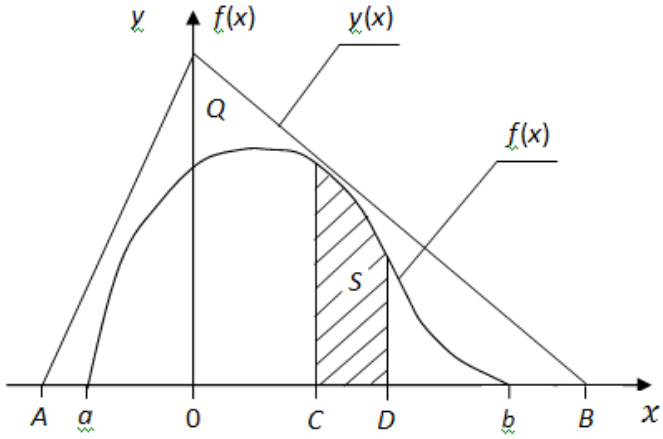


Рис. 2.11. Реализация метода отбора

математическое ожидание $M(n)$ числа n точек T, которые попадут в область, ограниченную сверху функцией $f(x)$. Вероятность того, что некоторая из указанных n точек попадёт в область S, ограниченную сверху функцией $f(x)$, то есть значение координаты x точки T будет принадлежать произвольному отрезку CD области определения функции $f(x)$, численно равна площади S, показанной на рисунке. А это означает, что плотностью распределения координаты x точек T, то есть НСВ x, является $f(x)$.

Таким образом, алгоритм моделирования очередного значения x состоит из следующих двух этапов:

1. Моделирование двумерной НСВ (точки T), равномерно распределённой в выбранной области Q. При в качестве области Q чаще всего выбирают прямоугольник, основание которого совпадает с отрезком ab, а высота равна максимальному значению $f(x)$. Это упрощает моделирование и увеличивает вероятность P попадания точек Т в область, ограниченную заданной $f(x)$.
2. Проверка выполнения критерия $y < f(x)$ попадания $T(x, y)$ в область, ограниченную сверху функцией $f(x)$. Если этот критерий выполняется, то полученное значение x отбирается, то есть принимается за значение НСВ с заданным распределением. В противном случае оно исключается из последовательности моделируемых НСВ, и осуществляется переход к первому этапу.

Пример. Подготовить программу моделирования НСВ X, подчиняющейся гамма-распределению

$$f(x) = \frac{x^{m-1}}{a^m(m-1)!}e^{-x/a}$$

при $a = 0{,}5,\ m = 3\ u\ x \in [0, 2]$. Итак, $f(x) = 4x^2e^{-2x}$. Не сложно убедиться, в том что максимальное значение $f(x)$ имеет место в окрестности точки $x = 1$ и не превышает значения 0,55. Поэтому программа моделирования очередного значения НСВ *X*, подчиняющейся гамма-распределению, может быть такой, как на вкладке справа (наиболее простой вариант выполнения рассмотренных выше циклов). По окончанию этих циклов при выполнении условия $y \le f(x)$ значение *x* может быть передано основной программе.

```
repeat
  x:=randon*2;
  y:=random*0.55;
  f:=4*x*x*exp(-2*x);
until  y > f;
```

Моделирование непрерывно-дискретных случайных величин. При исследовании некоторых систем с помощью математического моделирования на ЭВМ приходится моделировать случайные величины, которые могут принимать как непрерывные, так и дискретные значения (НДСВ).

На рис. 2.12*a* приведен пример переходной характеристики (зависимости величины выходного сигнала S_2 от входного S_1) ограничителя сигналов. В данном случае он пропускает на выход только положительные сигналы. На рис. 2.12*b* приводятся графики функции и плотности распределения шума на выходе ограничителя при равномерном

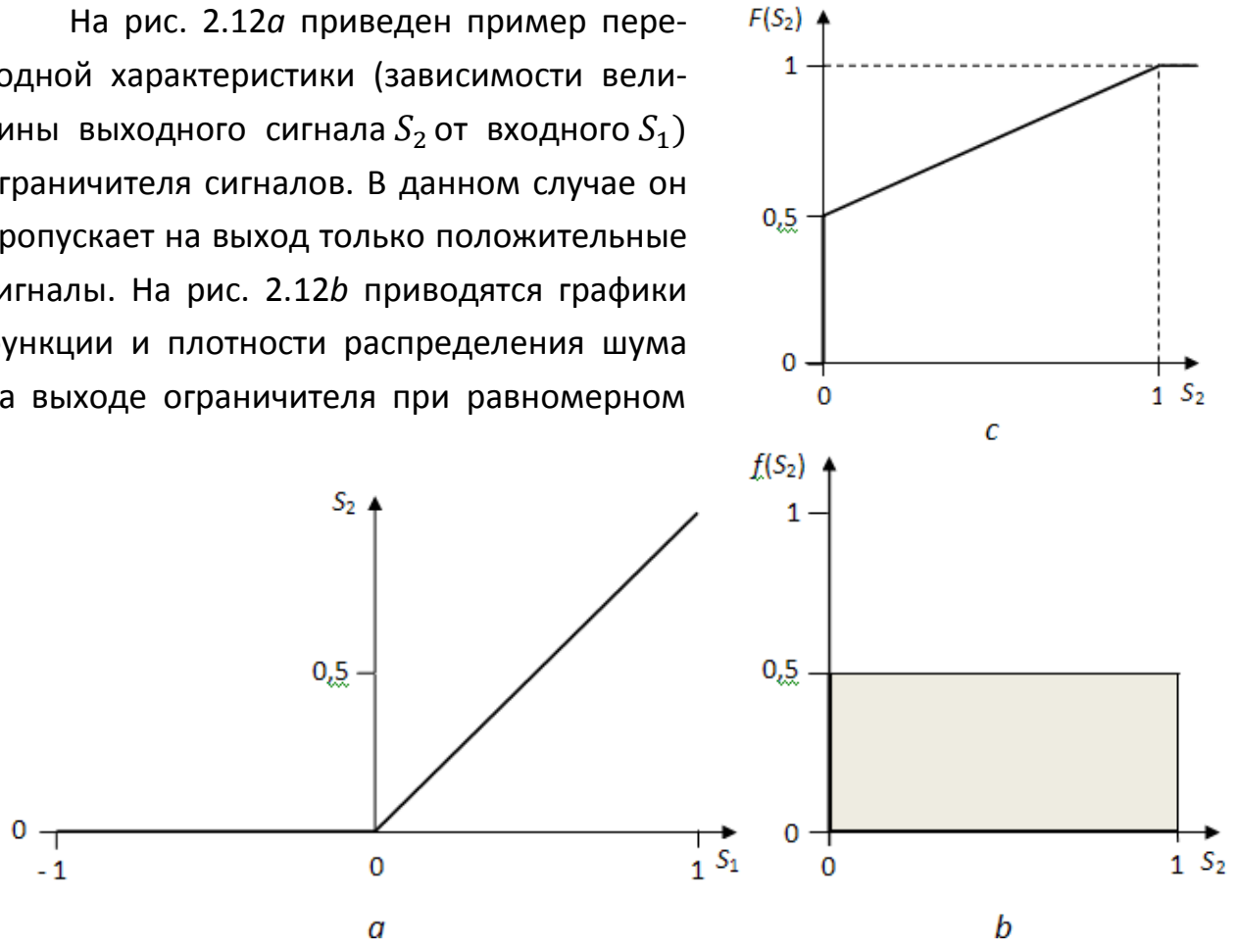


Рис. 2.12. Иллюстрации к примеру

распределении шума на его входе в промежутке [-1, 1]. Эти графики иллюстрируют пример распределения НДСВ S_2, которая может принимать значение 0

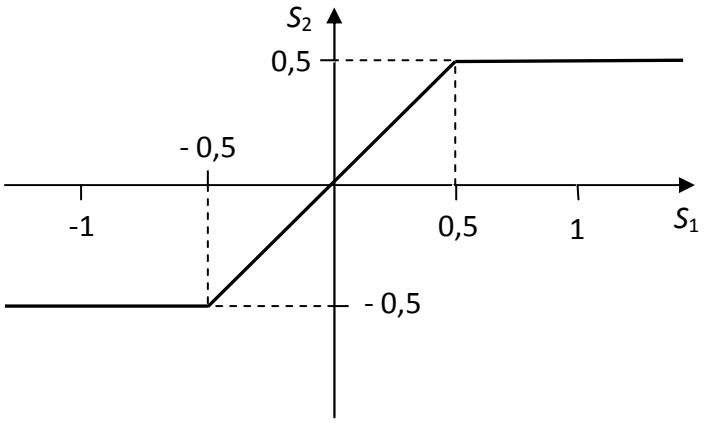
с вероятностью 0,5 и любые другие значения из промежутка (0, 1] с вероятностью появления каждого из них равной нулю – рис. 12.с.

Читателям предлагается разобраться в вопросе: Почему после прохождения шумового сигнала через ограничитель в его распределении появилась дискретная составляющая?

Упражнения. 1). Предложить алгоритм моделирования НДСВ *X*, подчиняющейся на промежутке [0, 2) бета-распределению $f(x) = \frac{4}{27}x^2(3-x)$ и принимающей при $x = 2$ значение *P*(2), которое необходимо определить.

2) Записать простой алгоритм моделирования НДСВ, принимающей в точках 0, 1, 2, . . . , 10 значения 1 с вероятностями 0,05, а в промежутках между этими точками имеющей одинаковые равномерные распределения.

3) Предложить алгоритм моделирования НДСВ S_2 на выходе ограничителя, переходная характеристика которого приведена на рисунке справа. На вход этого ограничителя подаётся шумовой процесс S_1, равномерно распределённый в промежутке [-1, 1].



2.4.2. Моделирование нормально распределённых случайных величин (частные методы)

Рассмотренные общие методы моделирования НСВ неудобны для моделирования нормально распределённых СВ, так как их функция распределения, изменяющаяся в бесконечном интервале, выражается только через интеграл вероятности. Ввиду бесконечности интервала изменения функции плотности не удаётся использовать и метод отбора. Поэтому для моделирования рассматриваемых СВ применяются только частные методы.

Частные методы моделирования нормально распределенных СВ делятся на три группы: методы приближённого представления обратной функции нормального распределения или его функции плотности, методы, использующие

композицию общих методов моделирования НСВ, и методы, основанные на теореме Ляпунова о бесконечном увеличении числа складываемых независимых случайных величин. Сущность большинства этих методов изложена, например, в [9, 10, 58]. Остановимся только на двух наиболее простых и популярных методах – методе кусочно-линейной интерполяции обратной функции распределения и на методах, основанных на указанной теореме Ляпунова.

Использование кусочно-линейной интерполяции и аппроксимации обратной функции распределения. Для реализации этого метода проще всего воспользоваться таблицей функции Лапласа

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-0{,}5t^2} dt,$$

приводимой в литературе по теории вероятностей и математической статистике [56 и др.] или в [45, 58 и др.]. При $x = 0$ значение этой функции равно нулю, а $\Phi(-x) = -\Phi(x)$.

В таблицах значения функции Лапласа приводятся для случая $M(x) = 0$, $\sigma(x) = 1$. При моделировании же чаще всего бывает необходимо получать значения нормально распределённой НСВ с другими значениями этих параметров распределения. В этом случае удобно воспользоваться общим выражением для функции $F(x)$ нормального распределения при $x \geq 0$:

$$y = F(x) = 0{,}5 + \Phi\left(\frac{x-m}{\sigma}\right), \qquad (2.5)$$

где m – математическое ожидание *x*, а σ – его среднее квадратическое отклонение. При этом $F(m) = 0{,}5$, $F(m-x) = 1 - F(m+x)$ и, следовательно, $\Phi^{-1}(0{,}5 - y) = -\Phi^{-1}(0{,}5 + y)$, где $y \in (0,\ 0{,}5)$, то целесообразно каждый раз моделировать только положительное значение *x*, которому, например, при выполнении условия random(2) = 0 придаётся знак минус.

С учетом указанного рассмотрим пример приближённого представления функции $F^{-1}(y)$ при $y \in [0{,}5, 1)$, не касаясь вопросов точности используемых интерполяции и аппроксимации, и сформулируем алгоритм моделирования нормально распределённой НСВ *X* с заданным значением σ. Значение M можно пока принять равным нулю, так как при моделировании его можно прибавлять к получаемым значениям *x*. Итак:

1) Выберем в качестве узлов кусочно-линейной интерполяции точки (0, 0,5),

$(F^{-1}(0{,}6),\ 0{,}6)$, $(F^{-1}(0{,}7),\ 0{,}7)$, $(F^{-1}(0{,}8),\ 0{,}8)$ и $(F^{-1}(0{,}9)\ 0{,}9)$.

2) Определяем значения *Ф* в выбранных точках. В соответствии с выражением (2.5) они соответственно равны: 0, 0,1, 0,2, 0,3 и 0,4.
3) По таблице значений функции Лапласа находим значения её аргумента для значений *Ф* в п. 2. Соответственно получаем: 0, 0,254, 0,525, 0,843, 1,282.
4) Пусть, например, задано: $\sigma = 2$. Тогда при *M* = 0 находим следующие значения x в узлах интерполяции: $x_0 = 0$, $x_1 = 0{,}508$, $x_2 = 1{,}050$, $x_3 = 1{,}686$, $x_4 = 2{,}564$. Значения ординат этих узлов были выбраны в п.1 (0,5, 0,6, 0,7 и т.д.).

Таким образом, алгоритм моделирования очередной НСВ *X* при попадании значения генерируемой НСВ *Y* на отрезок [0,5, 0,9) должен выполнить следующее:

- определить номер *N* участка $[y_i,\ y_{i+1})$, на который попало значение *y*;
- определить значение *x* генерируемой НСВ *X* согласно выражению $x = x_i + (x_{i+1} - x_i) \cdot (y - y_i)/(y_{i+1} - y_i)$.

Для “моделирования хвостов” нормального распределения обычно используется их приближённое представление (аппроксимация) соответствующей монотонно изменяющейся функцией, определённой на бесконечном интервале [9, 58]. Приведём простое выражение, предложенное автором для указанной цели, не претендующее на достаточно высокую точность:

$$x(y) = x(0{,}9) + [x(0{,}95) - x(0{,}9)]\cdot(y - 0{,}9)/(1 - y), \quad y \in (0{,}9,\ 1).$$

Если разыгрываемая НСВ *Y* попадает в указанный интервал, то в алгоритме моделирования *x* можно предусмотреть его получение согласно этому выражению. Каждый цикл шагов рассмотренного алгоритма должен заканчиваться добавлением к *x* значения его математического ожидания.

Упражнение. Подготовить программу моделирования НСВ *X* на основе интерполяции и аппроксимации обратной функции распределения.

2.4.3. Случайные величины с усечёнными распределениями

Необходимость моделирования НСВ с усеченными распределениями возникает при исследовании систем с элементами, ограничивающими значения входных величин, или при использовании распределений с бесконечными “хвостами” в моделях случайных факторов, изменяющихся в ограничен-

ных интервалах. Например, описание распределения размеров банковских вкладов населения страны.

Существуют три способа усечения законов распределения НСВ:

- усечение с переходом к непрерывно-дискретной СВ (рассмотрено в п. 2.4.1);
- одностороннее или двухстороннее усечение функции плотности с соответствующим увеличением её в области, где $f(x) > 0$;
- усечение функции плотности снизу с соответствующим увеличением её в области, где $f(x) > 0$.

В справочной литературе обычно рассматривается двухстороннее усечение функции плотности случайных величин. Однако получаемые при этом случайные величины на практике почти не встречаются.

На практике наиболее часто встречаются колоколообразные распределения СВ, имеющие один экстремум и строго монотонно уменьшающиеся до нуля в обе стороны от экстремума. Автором был предложен способ моделирования таких распределений путём усечения снизу классических одноэкстремальных распределений с бесконечными пределами [42]. Рассмотрим предложенный способ на примере усечения функции плотности $f(x)$ нормального распределения для получения колоколообразной функции плотности $g(x)$ на отрезке $[a, b]$ (рис. 2.13).

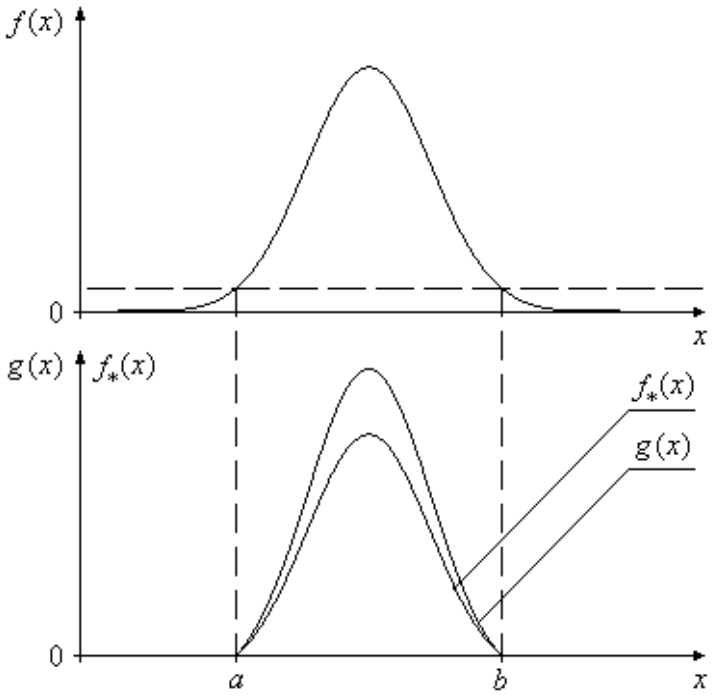


Рис. 2.13. Реализация колоколообразной функции плотности $g(x)$

При усечении $f(x)$ снизу и отбрасывании отсечённой её части получаем функцию $f_*(x)$, которая на (a, b) больше нуля. Площадь S, ограниченная этой функцией и осью абсцисс, меньше единице:

$$S = 1 - 2\Phi\left(\frac{a}{\sigma}\right) - (b-a)\Phi\left(\frac{a}{\sigma}\right).$$

Поэтому, выбирая $g(x) = f_*(x)/S$, получаем функцию $g(x)$, интеграл на [a, b] от которой равен единице. Следовательно, эта функция может быть функцией плотности.

Для моделирования НСВ X с функцией плотности $g(x)$ на каждом шаге моделирования вначале следует получить X с функцией плотности $f(x)$. Затем, если $x \in (a, b)$, то значение НСВ следует отобрать для использования его в основной программе. В противном случае это значение НСВ передавать для использования в основной программе не следует. Поэтому при использовании данного метода шагов моделирования превышает количество используемых в основной программе значений НСВ X.

Заметим, что среднее квадратическое значение σ_g НСВ, подчиняющихся распределению $g(x)$, несколько меньше значения σ исходного нормального распределения:

$$\sigma_g = \sqrt{\int_a^b x^2 g(x)dx} = \frac{1}{S}\sqrt{\int_a^b x^2 f_*(x)dx}.$$

Поэтому для получения желаемого значения σ_g можно рекомендовать соответственно увеличить значение σ НСВ X исходного нормального распределения. Это можно сделать его подбором при моделировании.

Упражнение: Определить закон распределения случайной величины, получаемый при усечении снизу функции плотности экспоненциального распределения $f(x) = \lambda\exp(-\lambda x)$.

Глава 3
Моделирование и анализ случайных процессов

3.1. Общие сведения о разновидностях, задачах и методах моделирования случайных процессов

Реальные случайные процессы $X(t)$ обычно представляются своими реализациями в непрерывном или в дискретном времени *t.* В общем случае различают четыре типа случайных процессов (СП):

- СП общего типа, в которых x и t могут принимать любые значения на соответствующих отрезках;
- дискретные СП (время *t* непрерывно, а значения *X*(*t*) дискретны);
- случайные последовательности общего типа (t дискретно, а x может принимать любые значения на соответствующем отрезке);
- дискретные случайные последовательности (x и *t* дискретны).

При решении задач ЭВМ все воспроизводимые СП фактически представляют собой дискретные случайные последовательности. Однако в современных ЭВМ шаги Δx и Δt настолько ничтожны, что обычно говорят о воспроизведении на ЭВМ непрерывных случайных процессов. На самом же деле ЭВМ воспроизводит только дискретные случайные последовательности, которые являются моделями СП других типов.

В литературе можно встретить идеализированные модели СП. К ним прежде всего относятся *белый шум* и *импульсный* шум. В модели белого шума принимается, что значения $x(t+\Delta t)$ и $x(t)$ не коррелированны при любом значении Δt, отличном от 0. Это означает, что спектр белого шума содержит любые, сколь угодно высокие частоты.

В отличие от белого шума, присутствующего на всём отрезке наблюдения (моделирования), импульсный шум представляет собой сумму большого числа импульсов одинаковой формы. Амплитуда этих импульсов является случайной величиной с конечной дисперсией, а моменты их появления также случайны и статистически независимы.

Задачи, решаемые на основе моделирования с использованием соответствующих СП в крупном плане по характеру их приложений можно разбить, по крайней мере, на пять групп:

- анализ ошибок в различных системах, обусловленных случайным характером входных воздействий, нестабильностью параметров систем и пр.;
- изучение задач исследования операций, в частности, задач массового обслуживания, поиска экстремальных значений функций и др.;
- разработка и исследование систем, работающих в условиях случайных внешних воздействий (системы связи, системы обнаружения различных объектов и т.д.);
- моделирование сигналов со сложным спектром частот и использующих их систем;
- анализ и прогнозирование временных рядов в самых различных отраслях: экономика, здравоохранение, социология и ряд других.

Все перечисленные задачи используют методы Монте-Карло, так как не решаются обычно аналитическими методами и поэтому для получения качественных решений требуют имитационного моделирования, то есть моделирования случайных процессов функционирования исследуемых систем.

В общем случае для моделирования СП необходимо знать его многомерную функцию или плотность распределения (их многомерность определяется числом шагов t на периоде моделирования). Эти функции дают исчерпывающие сведения о характеристиках СП, включая корреляционные характеристики, размерность которых не превышает числа шагов на периоде моделирования. Если функция и плотность распределения не зависят от времени t, СП считается стационарным в широком смысле. Однако, как правило, указанные функции бывают известны лишь для частных случаев, и при моделировании СП обычно используются только двумерные корреляционные характеристики – ковариации $K_{ij} = = M\left(X_i X_j - M_i M_j\right)$ или коэффициенты корреляции $R_{ij} = K_{ij}/K_{jj}$, где под X_i и M_i понимаются соответственно $X(t_i)$ и $M[X(t_i)]$, а i и j указывают номера временных шагов на периоде моделирования. При независимости M_i и R_{ij} от t СП считается стационарным в узком смысле (в смысле А.Я. Хинчина – см. далее). Число шагов от t_i к t_j обычно обозначают τ. Если машинное значение τ равно фактическому его значению, то говорят, что моделирование выполняется в реальном масштабе времени. В противном случае говорят, что моделирование выполняется соответственно в ускоренном или в замедленном масштабе времени.

Отметим, что если значения СП на указанных выше шагах являются независимыми случайными величинами, то задача моделирования такого СП становится тривиальной: в этом случае каждая случайная величина $X(t)$ моделируется независимо от других случайных величин с помощью методов, рассмотренных в разделе 3.

Технология моделирования перечисленных выше задач предусматривает выполнение соответствующих частных задач, связанных с моделированием СП. Ниже в п. 3.2 – 3.4 рассматриваются методы решения трёх таких задач.

3.2. Моделирование случайных последовательностей с заданными корреляционными функциями

Рассматриваемая задача используется при моделировании сигналов с различными спектральными характеристиками для радиосистем и некоторых систем другого назначения. Дело в том, что корреляционная функция $R(\tau)$ и энергетический спектр $S(\omega)$ стационарного СП связаны между собой парой преобразований Фурье (теорема Хинчина – Винера[3]):

$$R(\boldsymbol{\tau}) = (4\pi)^{-1} \int_{-\infty}^{\infty} S(\omega)\, e^{i\omega\tau}\, d\omega = (2\pi)^{-1} \int_{0}^{\infty} S(\omega) \cos \omega\tau\, d\omega,$$

$$S(\omega) = 2 \int_{-\infty}^{\infty} R(\boldsymbol{\tau})\ e^{-i\omega\tau} d\tau = 4 \int_{0}^{\infty} R(\boldsymbol{\tau}) \cos \omega\tau d\tau.$$

Поэтому по аналитическому выражению для требуемого спектра $S(\omega)$ можно найти соответствующую ему корреляционную функцию $R(\tau)$и воспроизвести (промоделировать) СП с этой $R(\tau)$. Полученный СП и будет являться сигналом с необходимым энергетическим спектром.

Рассмотрим вопросы моделирования и анализа характеристик стационарных случайных последовательностей $Y(t)$. Обе эти задачи обычно решаются на основе алгоритмов авторегрессии, задаваемых разностным уравнением

$$Y(t) = A_1 Y(t-1) + A_2 Y(t-2) + \cdots + A_n Y(t-n) + BX(t), \qquad (3.1)$$

в котором $X(t)$ – последовательность независимых случайных величин (СВ) с заданным одномерным распределением $G(x)$, n – порядок уравнения, а A_i и B – коэффициенты, которые могут быть либо детерминированными, либо случайными ортогональными [21, 27, 37]. В обоих указанных случаях выбором

---

[3] Хинчин А.Я. (1894 – 1959) – выдающийся советский математик, член-корреспонд. АН СССР. Винер Норберт (1894 – 1964) – известный американский математик, “отец” кибернетики.

этих коэффициентов последовательность $X(t)$ независимых СВ преобразуют в последовательность $Y(t)$ с заданной корреляционной функцией $R(\tau)$.

Алгоритмы вида (3.1), в которых значение выходной СВ на шаге *t* зависит от значений этой величины предыдущих шагах, а также от значения входной СВ на шаге *t*, называют авторегрессионными [4, 65 и др.].

Рассмотрим два вида уравнения (3.1).

1) *Коэффициенты уравнения* (3.1) *детерминированные*. В этом случае при устойчивом решении уравнения (3.1) для коэффициентов корреляции $R(\tau)$ последовательности $Y(t)$ справедливо уравнение Юла-Уолкера[4]:

$$R(\tau) = A_1R(\tau - 1) + A_2R(\tau - 2) + \cdots + A_nR(\tau - n), \qquad (3.2)$$

в котором должно выполняться условие $\tau > 0$.

На основе выражения (3.2) можно записать систему *n* уравнений, позволяющую по известным коэффициентам A_i проанализировать корреляционную характеристику моделируемой последовательности, то есть выполнить часть задачи анализа характеристик последовательности *Y*(*t*), или по заданной корреляционной характеристике этой последовательности найти значения всех коэффициентов уравнения (3.2). Например, для уравнения с четырьмя коэффициентами получаем:

$$\begin{cases} R(1) = A_1 + A_2R(1) + A_3R(2) + A_4R(3), \\ R(2) = A_1R(1) + A_2 + A_3R(1) + A_4R(2), \\ R(3) = A_1R(2) + A_2R(1) + A_3 + A_4R(1), \\ R(4) = A_1R(3) + A_2R(2) + A_3R(1) + A_4. \end{cases} \qquad (3.3)$$

В уравнениях (3.3) учтено, что $R(0) = 1$ и $R(-\tau) = R(\tau)$. Определив из этих уравнений значения $R(1) \div R(4)$, можно найти значения $R(\tau)$ для любого интервала τ. Например: $R(5) = A_1R(4) + A_2R(3) + A_3R(2) + A_4R(1)$ и т.д.

При моделировании и анализе рассматриваемых последовательностей, как правило, интересуются значением математического ожидания $M(Y)$, а часто и распределением значений *y*. Из уравнения (3.1) следует:

$$M(Y) = \sum_{i=1}^{n} A_i\, M(Y) + BM(X).$$

Следовательно,

$$M(Y) = \frac{BM(X)}{1 - \sum_{i=1}^{n} A_i}, \qquad (3.4)$$

---

[4] Юл и Уолкер – британские математики 20-го века.

и изменяя коэффициент B, можно изменять значения $M(Y)$ и других характеристик выходной последовательности.

При детерминированных коэффициентах уравнения (3.1) сложной является задача определения аналитического выражения для распределения СВ *Y* в зависимости от распределения СВ *X*. Как известно, линейные преобразования только нормально распределённых СВ не изменяют “нормальности” распределения получаемой с их помощью СВ. Но в общем случае распределение СВ *Y* получается отличным от распределения СВ *X*. В этом заключается сложность использования регрессионных алгоритмов (моделей) получения СП с заданными одномерным распределением выходной СВ и автокорреляционной функцией. Правда они позволяют получить СП $\{Y\}$ с $M(Y) = M(X)$ в случае $B = 1 - \sum_{i=1}^{n} A_i$, что следует из выражения (3.4).

Следует отметить, что при использовании детерминированных коэффициентов выходная последовательность может оказаться неустойчивой, то есть при неограниченном увеличении t значения модулей СВ *Y* могут стремятся к бесконечности, что несколько ограничивает возможности рассмотренного алгоритма. В монографии [21] автором было показано, что в случае, когда все коэффициенты A_i положительные и их сумма меньше единицы, решение однородной части уравнения (3.1) является асимптотически устойчивым. Очевидно, в данном случае при ограниченности значений СВ *X* асимптотически устойчиво и общее решение рассматриваемого уравнения.

2) *Коэффициенты уравнения (3.1) случайные, двоичные, ортогональные* [21, 27], то есть события $A_i = 1$ и $B = 1$ образуют полную группу событий (в такой группе одновременно один из коэффициентов равен 1, а все остальные равны 0).

Примем для рассматриваемого случая следующие обозначения: $M(A_i) = P(A_i = 1)$, $M(B = 1) = P(B = 1) = b$, $M[Y_i(t)] = y(t)$, $M[X(t)] = x(t)$. Тогда уравнения (3.1) ÷ (3.4), в которых случайные величины заменяются их математическими ожиданиями, сохраняют силу. Так, уравнения (3.1) и (3.2) принимают следующий вид [21, 27]:

$$y(t) = a_1 y(t-1) + a_2 y(t-2) + \cdots + a_n y(t-n) + bx(t), \qquad (3.5)$$

$$R(\tau) = a_1 R(\tau - 1) + a_2 R(\tau - 2) + \cdots + a_n R(\tau - n). \qquad (3.6)$$

Выражение (3.6) позволяет по заданным значениям a_i определить все $R_i(\tau)$ при анализе полученной последовательности или, наоборот, найти все a_i

по заданным $R_i(\tau)$ при построении уравнения авторегрессии (3.5), моделирующего СП.

Поскольку события $(A_i = 1)$ и $(B = 1)$ образуют полную группу событий, то, очевидно,

$$\sum_{i=1}^{n} a_i + b = 1. \tag{3.7}$$

При этом в случае получения согласно расчёту отрицательных значений a_i и моделировании СП с чётной функцией плотности $f(y)$ в уравнении (3.5) соответствующий коэффициент a_i можно принять положительным, а знак у СВ $y(t-i)$ изменить на противоположный. Если же в указанном случае функция $f(y)$ не является чётной, то в отличие от первого метода моделирование СП данным методом невозможно.

Для рассматриваемого алгоритма справедлива следующая теорема [21]: характеристическое уравнение

$$H(\lambda) = \lambda^n - \sum_{i=1}^{n} a_i \lambda^{n-i} = 0$$

разностного уравнения (3.5) в области $\lambda > 0$ имеет один и только один действительный корень λ_+, меньший единицы, а модули всех остальных корней не превышают значения λ_+. Следовательно, решение однородной части уравнения (3.5) является асимптотически устойчивым. При ограниченности значений x, что всегда имеет место, общее решение уравнения (3.5) также асимптотически устойчиво. Поэтому при использовании второго алгоритма не нужно заботиться об устойчивости получаемого решения: оно всегда устойчиво.

Другим достоинством второго алгоритма является то, что распределение СВ *Y* также асимптотически стремится к распределению СВ *X*. Докажем это утверждение. Возьмём в уравнении (3.5) вместо СВ *y*(*t*) величину $F(y,t)$, то есть функцию распределения СВ *y* для шага t. Тогда уравнение (3.5) примет вид:

$$F[y(t)] - a_1 F[y(t-1)] - a_2 F[y(t-2)] - \cdots - a_n F[y(t-n)] = bF(x), \tag{3.8}$$

где под $F[y(t)]$ понимается величина, равная вероятности появления СВ $Y(t)$, меньшей $y(t)$. В силу выражения (3.7), все коэффициенты которого принадлежат промежутку (0, 1), сумма коэффициентов a_i уравнения (3.8) меньше единицы. Поэтому согласно приведённой теореме решение однородной части уравнения (3.8) стремится к нулю, а его общее решение асимптотически

устойчиво и стремится к частному решению. Нетрудно убедиться, что при $F[y(t-i)] = F(x)$, где $i \in \{0, 1, \dots n\}$, разностное уравнение (3.8) превращается в тождество. Следовательно, $F(y)$ асимптотически стремится к $F(x)$. Если $F(x)$ со временем изменяется, то $F(y)$ всегда стремится к "текущей" $F(x)$.

Пример построения модели случайной последовательности. Пусть нужно построить авторегрессионную модель стационарной СП 2-го порядка со случайными коэффициентами. Значения СВ этой последовательности должны быть равномерно распределённых в промежутке [-2, 2], а значения коэффициентов корреляции $R(1)$ и $R(2)$ должны быть соответственно равны 0,4 и 0,1. Итак, строим модель

$$Y(t) = A_1 Y(t-1) + A_2 Y(t-2) + BX(t).$$

Основная задача построения этой модели состоит в определения коэффициентов a_1 и a_2, то есть вероятностей $P(A_1 = 1)$ и $P(A_2 = 1)$. На основе выражения (3.6) получаем систему двух уравнений:

$$\begin{cases} R(1) = a_1 R(0) + a_2 R(1), \\ R(2) = a_1 R(1) + a_2 R(0), \end{cases}$$

или

$$\begin{cases} a_1 + a_2 \cdot 0{,}4 = 0{,}4, \\ a_1 \cdot 0{,}4 + a_2 = 0{,}1. \end{cases}$$

Из полученной системы уравнений находим: $a_1 = 3/7$, $a_2 = -1/14$. Следовательно, $b = 1 - a_1 - |a_2| = 0{,}5$. Случайная величина X моделируется оператором: random*4-2.

Таким образом, один шаг моделирования СВ *Y* можно реализовать следующей программой:

```
X:=random*4−2;
Z:=random;     // моделирование СВ, определяющей выбор коэффициента
               // A1 = 1 или A2 = 1 для следующего оператора.
If Z<3/7 then Y0:=Y1 else if Z<3/7+1/14 then Y0:=−Y2 else Y0:=X;
Y2:=Y1; Y1:=Y0;  // сдвиг значений Y на один шаг.
```

В приведённой программе приняты обозначения: $Y(t) \sim Y0$, $Y(t-1) \sim Y1$ *и* $Y(t-2) \sim Y2$.

Вопросы и упражнения. 1) Если рассмотренное в примере задание выполнить с помощью регрессионной модели 2-го порядка с детерминированными коэффициентами, то каковы будут значения этих коэффициентов? Ответить на этот вопрос, не вычисляя значений коэффициентов.

2) Почему в приведенной программе моделирования регрессионной модели случайная величина *Y*2 используется со знаком “минус”?

3) Почему нельзя построить регрессионную модель со случайными коэффициентами согласно приведенному выше заданию, если СВ *Y* должна иметь равномерное распределение в [0, 2]?

4) В чём состоит недостаток регрессионных моделей с детерминированными коэффициентами?

5) Определить в рассмотренном примере значения $R(3)$ и $R(4)$.

6) Определить значения коэффициентов корреляции $R(1)$ и $R(2)$ последовательности $Y(t)$, моделируемой авторегрессионной моделью 2-го порядка с детерминированными коэффициентами $A_1 = 0{,}4$, $A_2 = 0{,}8$ и $B = 1$. СВ *X* равномерно распределена в [0, 1]. Сделать выводы о работоспособности модели.

3.3. Декомпозиция временных рядов

Под временным рядом (ВР) понимают последовательность числовых значений какой-либо случайной величины *X*, обычно изменяющейся во времени *t*: $x(0)$, $x(1)$, . . . , $x(t)$, . . . , то есть значений некоторого протекающего во времени процесса. Однако нередко в роли аргумента величины *X* используется не время, а другой параметр, в частности, − номер измерения. Примером такого ряда может служить последовательность результатов взвешивания новорождённых в Великом Новгороде за некоторый год. Подобные ряды либо называют *числовыми*, либо условно сохраняют за ними название *временные* [4, 49, 65, 71].

Цель декомпозиции временного ряда (ВР) состоит в разложении его на компоненты, что позволяет лучше судить о тенденциях изменения рассматриваемого ВР и прогнозировать возможные значения ВР на несколько шагов вперёд.

К компонентам ВР относятся:

- тренд (плавно изменяющаяся, не циклическая, детерминированная компонента, учитывающая влияние долговременных факторов);
- сезонная компонента, отражающая присущие рассматриваемому ряду повторяющиеся или почти повторяющиеся периоды его элементов в зависимости временного интервала (года, месяца, дня и т.п.);
- циклическая (нерегулярная) компонента, состоящая из длительных периодов относительного подъёма и спада, которые могут меняться по амплитуде и протяжённости;
- случайная составляющая (короткие всплески или провалы значений элементов ряда, обусловленные случайными факторами).

Если значения ВР чередуются с шагом один год, то сезонная компонента в таком ВР как правило отсутствует, так как её периоды “скрываются” внутри шага.

Декомпозицию ВР удобнее начинать с выделения детерминированных компонент. Разделение этих компонент является большим искусством. В рамках данного издания ограничимся рассмотрением примера выполнения декомпозиции ВР с помощью моделирования методом проб и ошибок. Желающих ознакомиться с более совершенной (но и более сложной) методикой декомпозиции ВР можем адресовать к [6]. Декомпозицию и прогнозирование ВР можно также выполнять в среде программирования Mathematica 6.0.

В качестве ВР, на примере которого продемонстрируем вариант решения рассматриваемой задачи, выберем приведённый в Интернете ряд относительных значений числа простудных заболеваний (ПрЗаб) населения Санкт-Петербурга, приходящихся на 1000 человек населения. Статистические данные фиксировались с декабря 2006 года по ноябрь 2007 года с временным шагом 6 дней (рис. 3.1.*а*).

Отметим характерные черты исследуемого ВР, которые всегда следует выявить перед тем как приступить непосредственно декомпозиции ВР. За существенными ошибками в декомпозиции ряда как правило увеличиваются и ошибки в прогнозировании этого ряда. Итак:

1. Имеется сезонная компонента, огибающую которой можно представить синусоидой

$$\text{СК}(t) = a_{\text{с}} \sin(\omega_{\text{с}} t + \varphi_{\text{с}}).$$

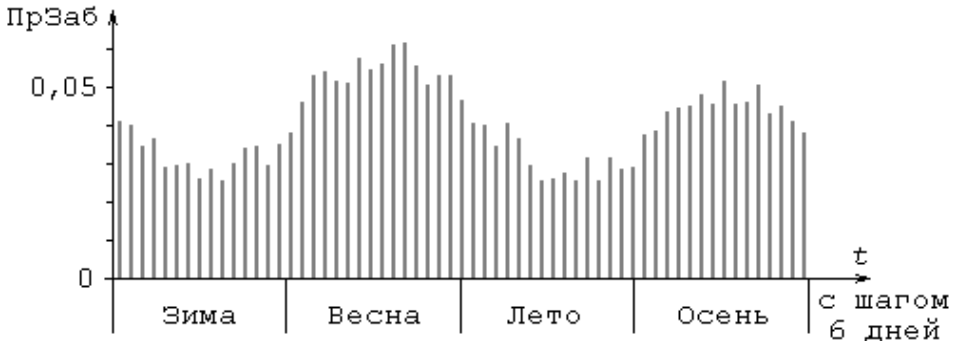
Приблизительные значения амплитуды a_c и круговой частоты ω_c можно определить из рисунка. Предварительное значение фазы φ_c подбирается при моделировании СК(t) и сравнением её графика с графиком на рис. 3.1a.

2. С увеличением времени амплитуда моделируемой синусоиды несколько уменьшается, а сама синусоида несколько наклоняется в сторону роста t. Это свидетельствует о наличии в ряде циклической компоненты (ЦК), по-видимому, тоже типа синусоиды с огибающей

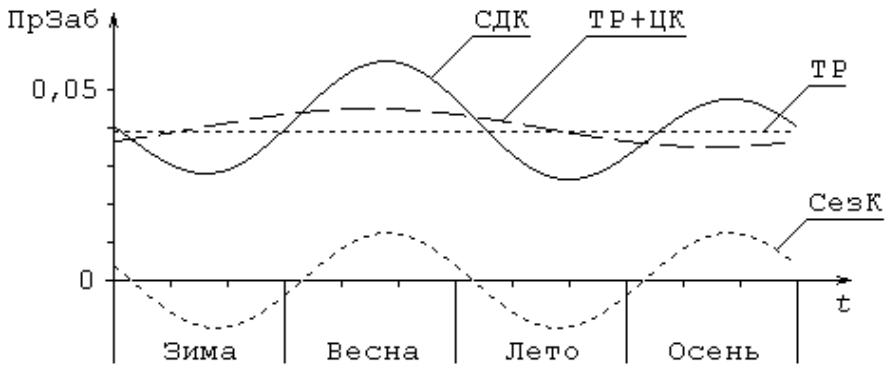
$$\text{ЦК}(t) = a_{\text{ц}} \sin\left(\omega_{\text{ц}} t + \varphi_{\text{ц}}\right),$$

частота $\omega_{\text{ц}}$ которой примерно в два раза меньше ω_c. Поэтому амплитуда a_c одного полупериода синусоиды как бы увеличивается, а другого – уменьшается.

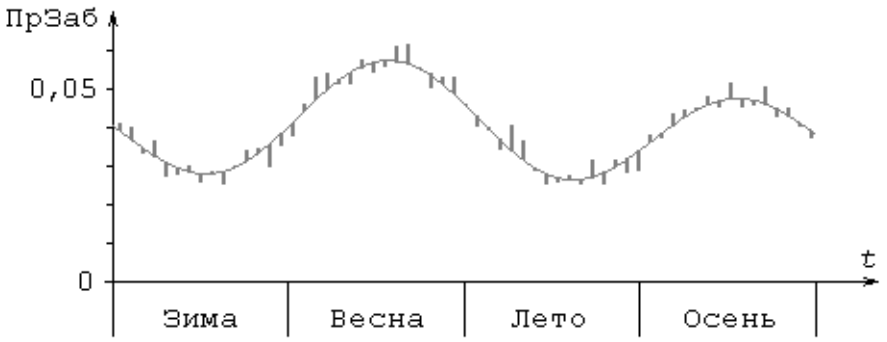
3. По виду ряда, представленному на рис. 3.1a, предварительное можно принять, что тренд (Тр) ряда на рассматриваемом интервале является постоянной величиной, равной среднему значению 61 элементов этого ряда, то есть 0,0386.



a) Временной ряд значений простудной заболеваемости на 1000 человек населения



b) Сумма детерминированных компонент (СДК), тренд (ТР), суммарное значение ТР и циклической компоненты (ТР+ЦК), сезонная компонента (СезК)



c) Сумма детерминированных компонент и случайной компоненты ряда

d) Случайная составляющая временного ряда

Рис. 3.1. Временной ряд и его компоненты

Рекомендуемая методика декомпозиции ВР предусматривает два этапа. На первом этапе по характерным особенностям ВР выбираются предварительная модель для каждой из детерминированных компонент рассматриваемого ряда. В приведённом выше примере все эти компоненты приведены на рис. 3.1b. На этом же рисунке приведена и огибающая суммы детерминированных компонент, которая играет важную роль в оценке приемлемости выбираемого варианта декомпозиции ВР.

На втором этапе рекомендуемой методики осуществляется некоторая коррекция моделей детерминированных компонент ВР. Для этой цели из исходного ряда вычитается огибающая суммы детерминированных компонент (СДК). В результате получается модель случайной компоненты СлК ряда. На рис. 3.1.*c* она представлена так, как будто СлК располагается по огибающей СДК. При этом на каждом шаге *t* при сложении значений СДК(*t*) и СлК(*t*) получается точное значение элемента ВР. Отдельно СлК может быть представлена так как на рис. 3.1*d*. Этот график получается вычитанием графика СДК(*t*) из графика ВР.

Для реализации рассмотренной методики не сложно подготовить программу с запоминанием в ЭВМ значений элементов ряда и выражений для

моделей его детерминированных компонент. Программа должна обеспечить многошаговое приближение к варианту декомпозиции с минимальным значением суммы квадратов отклонений суммы всех компонент ряда от значений элементов ряда на каждом шаге рассматриваемого его отрезка. Эта сумма должна определяться программой на каждом шаге (с запоминанием последнего лучшего варианта параметров моделей компонент ряда). Указанное приближение проще всего организовать по методу Зейделя [1], когда изменение одного из параметров в нужную сторону (шаг по этому параметру) выполняется до тех пор, пока значение целевой функции, то есть значение указанной суммы квадратов, уменьшается. После этого начинается аналогичное изменение другого параметра, затем – следующего и т.д. Процесс приближения продолжается до тех пор, пока ни по одному из параметров модели не будет сделан очередной шаг изменения их значений погрешности приближения. Сумма полученных компонент ряда называется аддитивной моделью этого ряда. Кроме аддитивной модели, иногда используют мультипликативную модель ряда. В этом случае ВР представляется в виде произведения своих компонент [64 и др.].

Таким образом, рассмотренная методика реализует поиск оптимального решения в *n*-мерном пространстве. В данном случае *n* – это число параметров модели всего ряда, равное 7 (по 3 параметра в синусоидальных моделях СК и ЦК и один в выражении для тренда).

3.4. Прогнозирование временных рядов

Вопросы прогнозирования временных рядов рассматриваются во многих литературных источниках [4, 6, 29, 49, 64, 65, 68, 70, 74 и др.]. Кроме того, они рассматриваются и в некоторых учебных дисциплинах. Поэтому ограничимся рассмотрением двух основных методов прогнозирования ВР, каждый из которых основывается на той или иной модели ВР.

Один из указанных методов, который условно назовём ***первым методом прогнозирования ВР***, предусматривает предварительную декомпозицию ряда, если его детерминированные компоненты не известны. В случае отсутствия сезонной компоненты прогнозирование выполняют только по тренду и циклической компоненте (если она есть).

Итак, пусть при моделировании получены аналитическая зависимость для СДК(t). Тогда продолжая эту зависимость на прогнозируемое число шагов, получим значения прогнозов Пр(t+1), Пр(t+2), . . . Пр(t+k). Точность таких прогнозов зависит от следующих четырёх факторов:

- число шагов ВР, на основе которых выполнена декомпозиция ВР;
- точность декомпозиции детерминированных компонент ряда;
- дисперсии случайной составляющей ряда;
- числа шагов, на которые производится прогнозирование.

Характер влияния этих факторов на точность прогнозирования очевиден и поэтому не поясняется. Среднее значение погрешности прогнозирования не может быть меньше среднего значения половины амплитуды выбросов случайной компоненты ВР.

Во многих практических задачах прогнозировать приходится по небольшому отрезку ряда, декомпозицию которого выполнять не имеет смысла, так как в этом случае погрешность декомпозиции, а, следовательно, и прогнозирования, будет слишком большой.

Второй метод прогнозирования ВР основан на предварительном выравнивании (сглаживании) участков ВР. При этом выравнивание всегда выполняется в соответствии с моделью, которая, по мнению исполнителя, наиболее подходит для рассматриваемого участка ряда. Продолжая выровненный ряд согласно выражению, полученному для выбранной модели, получают значения прогнозов Пр(t).

Выравнивание ВР, в том числе различных экспериментальных зависимостей, как правило выполняется по критерию минимума суммы квадратов отклонений значений элементов рассматриваемого ряда от выбранной для него модели или, выражаясь проще, по методу наименьших квадратов [1, 4, 42]. Для применения этого метода нет указанного выше ограничения. Поэтому рассмотрим подробнее алгоритмы выравнивания ВР и его прогнозирования, построенные на основе метода наименьших квадратов.

Итак, пусть на некотором шаге, которому удобно присвоить 0-й номер, ведётся прогнозирование ряда $\{X(t)\}$ на шаги $t = 1,\ 2,\ 3\ldots$ Для прогнозирования используются значения ряда $X(0),\ X(-1),\ X(-2),\ldots,X(1-m)$, обозначаемые далее для простоты как X_0, X_{-1}, X_{-2} и т.д. На 1-м шаге, когда стало уже известно значение X_1, для прогнозирования логично использовать значения $X_1,\ X_0,\ X_{-1},\ldots,X_{-2}$ рассматриваемого ВР и т.д. То есть на каждом шаге

для прогнозирования используются значения *m* последних элементов ряда. Эти элементы принято называть скользящим окном, величину m – шириной этого окна.

Для прогнозирования на каждом шаге производится выравнивание участка ряда, попавшего в скользящее окно. Выравнивающая (аппроксимирующая) функция $\varphi(t, a, b, c, \dots)$ с параметрами $a, b, c, \dots$ рассматривается как модель участка ВР, попавшего в скользящее окно. Продолжая эту функцию на следующие шаги, получают значения Пр для этих шагов.

В принципе для повышения точности прогнозирования на каждом шаге прогнозирования можно выбрать ту или иную модель выравнивания ряда. Однако это усложняет программу прогнозирования. Поэтому обычно на каждом шаге прогнозирования рассчитываются только новые значения параметров модели, выбранной для всех шагов прогнозирования.

Согласно методу наименьших квадратов неизвестные параметры *a*, *b, c*, … аппроксимирующей функции, находятся исходя из условия [1, 4, 42]:

$$\sum_{t=1-m}^{0} [X_i - \varphi(t, a, b, c, \dots)]^2 = \min. \qquad (3.9)$$

Для обеспечения минимального значения выражения (3.9) необходимо, чтобы все частные производные этого выражения по искомым параметрам были равны нулю (можно показать, что в рассматриваемом случае указанное условие является не только необходимым, но и достаточным). Поэтому дифференцируя выражение (3.9) по параметрам *a, b, c,…* и приравнивая получаемые выражения для производных нулю, получают следующую систему *m* уравнений с *m* неизвестными параметрами:

$$\begin{cases} \sum\limits_{t=1-m}^{0} [X_i - \varphi(t, a, b, c, \dots)](\partial\varphi/\partial a)_t = 0, \\ \sum\limits_{t=1-m}^{0} [X_i - \varphi(t, a, b, c, \dots)](\partial\varphi/\partial b)_t = 0, \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \end{cases} \qquad (3.10)$$

Если, например, используется линейное выравнивание (линейная аппроксимация) участка ряда функцией $\varphi(t, a, b) = at + b$, то $\partial\varphi/\partial a = t$ и $\partial\varphi/\partial b = 1$. В этом случае при $m = 3$ система уравнений (3.10) после подстановки в неё значений t и значений частных производных принимает следующий промежуточный вид:

$$\begin{cases} (X_{-2} + 2a - b) \times (-2) + (X_{-1} + a - b) \times (-1) + (X_0 - b) \cdot 0 = 0, \\ (X_{-2} + 2a - b) \times 1 + (X_{-1} + a - b) \times 1 + (X_0 - b) \cdot 1 = 0. \end{cases}$$

После приведения подобных членов получаем:

$$\begin{cases} 5a - 3b = -2X_{-2} - X_{-1}, \\ -3a + 3b = X_{-2} + X_{-1} + X_0. \end{cases}$$

Следовательно, $a = 0{,}5(X_0 - X_{-2})$, $b = (5X_0 + X - X_{-2})/6$. Согласно этим выражениям значения *a* и *b* следует находить на каждом шаге прогнозирования значений *X*(*t*).

Аналогично находятся параметры и других аппроксимирующих функций, являющихся степенными многочленами от аргумента *t*, так как при применении метода наименьших квадратов определение их коэффициентов (параметров) связано с решением системы линейных алгебраических уравнений. Для того, чтобы число линейно независимых уравнений этой системы было бы равно числу определяемых параметров, необходимо, чтобы ширина скользящего окна *m* была бы не менее степени аппроксимирующего многочлена, увеличенной на единицу.

При использовании для аппроксимации других зависимостей получаемая система уравнений для определения параметров этих зависимостей может оказаться нелинейной и не решаемой аналитически.

В табл. 3.1 приведены полиномиальные модели ВР и реализующие их алгоритмы прогнозирования. Полиномиальные алгоритмы являются наиболее распространенными при моделировании и прогнозировании ВР. В частности, они использовались автором для определения прогнозов различных показателей здоровья населения [42].

Таблица 3.1

Полиномиальные алгоритмы прогнозирования временных рядов на Т шагов

Модель ВР	*m*	Алгоритм вычисления прогнозов
1. Скользящего среднего ($X_T = a$)	1	$\text{Пр}_T = X_0$;
	2	$\text{Пр}_T = 0{,}5 \cdot (X_0 + X_{-1})$;
	3	$\text{Пр}_T = (X_0 + X_{-1} + X_{-2})/3$;
	4	$\text{Пр}_T = 0{,}25 \cdot (X_0 + X_{-1} + X_{-2} + X_{-3})$;
	5	$\text{Пр}_T = 0{,}2 \cdot (X_0 + X_{-1} + X_{-2} + X_{-3} + X_{-4})$;

2. Линейная $(X_T = aT + b)$	2	$\text{Пр}_T = (X_0 - X_{-1})\,T + X_0$;
	3	$\text{Пр}_T = 0{,}5\cdot(X_0 - X_{-2})\,T + (5X_0 + 2X_{-1} - X_{-2})/6$;
	4	$\text{Пр}_T = 0{,}1\cdot(3X_0 + X_{-1} - X_{-2} - 3X_{-3})\,T + 0{,}1\cdot(7X_0 + 4X_{-1} + X_{-2} - 2X_{-3})$;
	5	$\text{Пр}_T = 0{,}1\cdot(2X_0 + X_{-1} - X_{-3} - 2X_{-4})\,T + 0{,}2\cdot(3X_0 + 2X_{-1} + X_{-2} - X_{-4})$;
3. Параболическая $(X_T = aT^2 + bT + c)$	3	$\text{Пр}_T = 0{,}5\cdot(X_0 - 2X_{-1} + X_{-2})\,T^2 + 0{,}5\cdot(3X_0 - 4X_{-1} + X_{-2})\,T + X_0$;
	4	$\text{Пр}_T = 0{,}25\cdot(X_0 - X_{-1} - X_{-2} + X_{-3})\,T^2 + 0{,}05\cdot(21X_0 - 13X_{-1} - 17X_{-2} + 9X_{-3})\,T + 0{,}05\cdot(19X_0 + 3X_{-1} - 3X_{-2} + X_{-3})$;
	5	$\text{Пр}_T = (2X_0 - X_{-1} - 2X_{-2} - X_{-3} + 2X_{-4})\,T^2/14 + (54X_0 - 13X_{-1} - 40X_{-2} - 27X_{-3} + 26X_{-4})T/70 + (31X_0 + 9X_{-1} - 3X_{-2} - 5X_{-3} + 3X_{-4})/35$.

Важным вопросом является оценка точности прогнозирования с помощью того или иного алгоритма (при прогнозировании различных рядов предпочтительными могут оказаться разные алгоритмы). Проще всего такая оценка делается для стационарных ВР, имеющих достаточно длинную предысторию, то есть когда до шага прогнозирования было получено достаточно много реализаций X_iэлементов ряда. В этом случае можно получить значения прогнозов на шаги с известными значениями и определить погрешность прогнозирования (обычно абсолютную или среднюю квадратическую).

Увеличить объём выборки для оценки погрешности можно путём моделирования ряда и прогнозирования на шаги с полученными значениями X_i. Такой метод использовался в [10] для выбора лучшего алгоритма при прогнозировании показателей здоровья населения и определения зависимости погрешностей прогнозирования от значения *T* (с увеличением *T* эти погрешности, как правило, увеличиваются).

Ниже приводится программа (процедура Prognoses), моделирующая ВР с суммой детерминированных компонент $X_{\text{дет}}(t) = 1 + a \cdot \sin(0{,}1t + 0{,}1)$ и со случайной составляющей $X_{\text{сл}}(t) = b \cdot (\text{random} + \text{random} - 1)$, имеющей треугольное распределение Пирсона на промежутке (-*b*, *b*) для значений *b*, равных 0, 0,05 и 0,1. Программа выполняет прогнозирование 10000 значений ряда на 1, 2 и 3 шага вперёд. По результатам прогнозирования определяются среднее значение абсолютной погрешности *Delta* полученных прогнозов. Прогнозирование выполняется с помощью каждого из алгоритмов, приведённых в табл. 3.1 с шириной скользящего окна, равной трём.

```
Procedure Prognoses(c: TCanvas);
  var X: array[-2,..3] of real; Pr,Delta: array[1..3] of real;
      I,J,Num: integer; a,b: real;
  begin
    a:=1; b:=0.1; Num:=1;
                                    // Установка начальных условий:
    for J:=1 to 3 do Delta[J]:=0;
    X[-2]:= 1 + a*sin(0.1-0.2) + b*(random+random-1);
    X[-1]:= 1 + b*(random+random-1);
    X[0]:= 1 + a*sin(0.1) + b*(random+random-1);
    X[1]:= 1 + a*sin(0.2) + b*(random+random-1);
    X[2]:= 1 + a*sin(0.3) + b*(random+random-1);
    X[3]:= 1 + a*sin(0.4) + b*(random+random-1);
                                  // Начальные условия установлены
    for I:=1 to 10000 do begin
      for J:=-2 to 2 do X[J]:=X[J+1];
      X[3]:= 0.5 + a*sin(0.1*I+0.4)+ b*(random+random-1);
    // Нашли новое значение ряда и сместили скользящее окно на 1 шаг вправо
     // Начинаем вычислять значения прогнозов Pr[i] на i шагов
     // Для модели скользящего среднего:
      if Num =1 then begin
        Pr[1]:= (X[0] + X[-1] + X[-2])/3;
        Pr[2]:=Pr[1];
        Pr[3]:=Pr[1];
      end;
     // Для линейной модели:
      if Num=2 then begin
        Pr[1]:= 0.5*(X[0] – X[-2]) + (5*X[0] + 2*X[-1] – X[-2])/6;
        Pr[2]:= X[0] – X[-2] + (5*X[0] + 2*X[-1] – X[-2])/6;
        Pr[3]:= 1.5*(X[0] – X[-2]) + (5*X[0] + 2*X[-1] –X[-2])/6;
      end;
     // Для параболической модели:
      if Num=3 then begin
        Pr[1]:= 0.5*(X[0] −2*X[-1] + X[-2]) + 0.5*(3*X[0] −4*X[-1] + X[-2]) + X[0];
        Pr[2]:= 2*(X[0] −2*X[-1] + X[-2]) + 4*X[0] – 4*X[-1]+X[-2];
        Pr[3]:= 4.5*(X[0] – 2*X[-1] + X[-2]) + 1.5*(3*X[0] – 4*X[-1] + X[-2]) + X[0];
      end;                          // Нашли значения прогнозов
```

```
      for J:=1 to 3 do Delta[J]:= Delta[J] + abs(Pr[J] – X[J]);
                  // Добавили погрешности на данном шаге  к их текущим суммам
    end;
  // Вывод полученных результатов в таблицу:
    c.TextOut(25,20,'Num = '+IntToStr(Num));
    c.TextOut(80,20,'a = '+FloatToStr(a));
    c.TextOut(80,40,'b = '+FloatToStr(b));
    c.TextOut(25,40,'Шаг'); c.TextOut(30,60,'1)');
    c.TextOut(30,80,'2)');   c.TextOut(30,100,'3)');
    for I:= 1 to 3 do c.TextOut(50,40+20*I,FloatToStr(0.0001*Delta[I]));
  end;
```

Таблица 3.2

Средние значения абсолютных погрешностей прогнозирования

Алгоритм	Шаг	Значения погрешностей при a = 0,15		
		b = 0	b = 0,1	b = 0,2
1.3	1	0,1269	0,1278	0,1307
	2	0,1900	0,1905	0,1924
	3	0,2525	0,2529	0,2543
2.3	1	0,0106	0,0319	0,0615
	2	0,0265	0,0467	0,0836
	3	0,0487	0,0670	0,1099
3.3	1	0,0008	0,0742	0,1482
	2	0,0028	0,1745	0,3486
	3	0,0069	0,3162	0,6318

В табл. 3.2 приведены результаты, полученные с использованием процедуры Prognoses при a = 1. Из данных таблицы следует, что средняя погрешность прогнозирования растёт с увеличением числа шагов, на которые производится прогнозирование, а также с увеличением дисперсии случайной компоненты ВР.

С помощью моделирования можно определить какой алгоритм прогнозирования является предпочтительным для решаемой задачи определения прогнозов. В рассмотренном выше примере таким алгоритмом оказался алгоритм 2.3 – алгоритм, использующий линейную зависимость, коэффициенты которой определяются по трём последним значениям ряда.

В заключение отметим, что нередко повысить точность прогнозирования можно с помощью суперпозиции алгоритмов прогнозирования, определяемой согласно выражению:

$$\text{Пр}_k(T) = \sum_{k=1}^{n} \alpha_k \text{Пр}_k(T),$$

где α_k– весовой коэффициент *k*-го алгоритма. При этом сумма всех весовых коэффициентов равна единице.

3.5. Моделирование случайных векторов

Необходимость моделирования случайных векторов $(y_1, y_2, \dots, y_n)$ возникает при реализации нескольких, в общем случае коррелированных воздействий на рассматриваемую систему. Если эти воздействия независимы, то задача их моделирования тривиальна: каждая координата вектора моделируется одним из рассмотренных в настоящем пособии методов как случайная величина или как случайная последовательность. В последнем случае получаем векторный случайный процесс с независимыми координатами.

При моделировании случайных векторов с зависимыми координатами необходимо получать *n* случайных величин с заданной функцией плотности *n*-мерного распределения $f(y_1, y_2, \dots, y_n)$. Эту задачу можно было бы свести к последовательной имитации независимых скалярных величин:

$$f(y_1, y_2, \dots, y_n) = f_1(y_1) \cdot f_2\,(y_2/y_1) \cdot f_3(y_3/y_1, y_2) \cdots f_n(y_n/y_1, y_2, \dots\ y_{n-1}).$$

Однако приведённые в этом выражении условные функции плотности распределения координат вектора, как правило, бывают неизвестны. Поэтому в общем случае задача моделирования произвольно распределённых случайных векторов с коррелированными координатами является проблемной.

В связи с указанным остановимся на решении лишь двух частных случаев рассматриваемой задачи.

3.5.1. Случай нормального распределения координат вектора

Задача моделирования случайных векторов с коррелированными координатами обычно решается в следующей постановке: заданы корреляционная (ковариационная) матрица ***K*** и вектор математических ожиданий ***M*** координат вектора, которые должны иметь задаваемые распределения. Вместо

корреляционной матрицы можно задавать матрицу коэффициентов корреляции $\boldsymbol{R}$ координат вектора: $R_{ij} = [M(X_i X_j) - M_i M_j]/\sqrt{K_{ii} K_{jj}}$.

В рассматриваемом случае для моделирования используется следующее преобразование [9, 10, 57]:

$$\boldsymbol{Y} = \boldsymbol{A} \cdot \boldsymbol{X} + \boldsymbol{M}, \tag{3.11}$$

где $\boldsymbol{A}$ – диагональная матрица коэффициентов a_{ij}, которые нужно определить по заданной корреляционной матрице, а $\boldsymbol{X}$ – вектор независимых нормально распределённых случайных величин с нулевыми математическими ожиданиями и единичными дисперсиями.

Пусть, например, $\boldsymbol{K} = \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix}$, $\boldsymbol{A} = \begin{pmatrix} a_{11} & & \\ a_{21} & a_{22} & \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$, $\boldsymbol{X} = \begin{pmatrix} x_1 \\ x_2 \\ x_1 \end{pmatrix}$,

$\boldsymbol{M} = (m_1,\ m_2,\ m_3)$ и $\boldsymbol{Y} = (y_1,\ y_2,\ y_3)$. Для пояснения методики определения значений a_{ij} [1, 6] запишем векторное уравнение (3.11) в виде системы скалярных уравнений:

$$\begin{cases} y_1 = a_{11} x_1, \\ y_2 = a_{21} x_1 + a_{22} x_2, \\ y_3 = a_{31} x_1 + a_{32} x_2 + a_{33} x_3. \end{cases} \tag{3.12}$$

Поскольку линейные преобразования нормально распределённых случайных величин сохраняют “нормальность” случайных величин, получаемых с их помощью [13], то все y_1 имеют нормальный закон распределения. Следовательно, все y_i системы (3.12) распределены по нормальному закону. При этом их математические ожидания равны нулю, так как равны нулю все $M(x_i)$.

Таким образом, необходимо определить такие значения коэффициентов a_{ij}, при которых теоретические значения корреляционных моментов будут соответственно равны их значениям, указанным в задаваемой матрице $\boldsymbol{K}$. Рассмотрим решение этой задачи.

Потребуем, чтобы в системе (3.12) дисперсии СВ y_i были бы равны заданным. Тогда получим:

$$\begin{cases} K_{11} = a_{11}^2, \\ K_{22} = a_{21}^2 + a_{22}^2, \\ K_{33} = a_{31}^2 + a_{32}^2 + a_{33}^2. \end{cases}$$

Из первого уравнения этой системы следует: $a_{11} = \sqrt{K_{11}}$. Для определения a_{21} и a_{22} необходимо получить ещё одно уравнение, связывающее эти коэффициенты. Для этого перемножим первое и второе уравнения системы (3.12). В результате получаем:

$$y_1 y_2 = a_{11} a_{21} x_1^2 + a_{11} a_{22} x_1 x_2.$$

Потребуем равенства математических ожиданий СВ, указанных в левой и правой частях этого равенства. Тогда при нулевых значениях математических ожиданий СВ x_i и y_i и при независимости x_1 и x_2 имеем:

$$K_{21} = \sqrt{K_{11}} a_{21}, \text{ то есть } a_{21} = K_{21}/\sqrt{K_{11}}.$$

Теперь из уравнения для K_{22} находим: $a_{22} = \sqrt{K_{22} - K_{21}^2/K_{11}}$.

Аналогично находятся коэффициенты и третьей строки системы (3.12). Однако в этом случае для получения двух дополнительных уравнений следует перемножать первое уравнение с третьим и второе уравнение также с третьим.

Продолжая увеличивать размерность моделируемого вектора, по индукции получаем [1, 6]:

$$a_{ij} = \frac{K_{ij} - \sum_{k=1}^{j-1} a_{ik} a_{jk}}{\sqrt{K_{jj} - \sum_{k=1}^{j-1} a_{jk}^2}}.$$

Этот алгоритм и используется при моделировании векторов произвольной размерности согласно выражению (3.11).

Упражнение: определить значения коэффициентов a_{ij} для моделирования нормально распределённого четырёхкоординатного вектора с характеристиками $R_{ij} = 0{,}2$ при $i \neq j$ и $K_{ii} = 10$.

3.5.2. Случай одинакового произвольного распределения координат вектора

Рассматриваемую задачу удалось решить, использовав в выражении (3.11) случайные ортогональные коэффициенты [21, 22, 34]. В этом случае моделируется вектор ***X***, все координаты x_i которого независимы и имеют заданное распределение. Как и в предыдущем случае задача матрицы ***A*** заключается в том, чтобы создать необходимую корреляцию координат вектора ***Y***, которые должны иметь то же распределение, что и координаты вектора ***X***. Однако

в данном случае элементы a_{ij} диагональной матрицы ***A*** следует рассматривать как вероятности появления единичных значений A_{ij}, численно равных математическим ожиданиям случайных двоичных величин A_{ij}.

Ортогональность случайных коэффициентов в каждой строке матрицы ***A*** означает, что одновременно один из них равен 1, а все стальные равны 0. В частности коэффициент A_{11} всегда равен единице, то есть всегда $Y_1(t) = X_1(t)$. Значения $Y_i(t)$ всегда равны одному из следующих значений: $X_1(t)$, $X_2(t)$, … $X_i(t)$. Следовательно, координаты вектора ***Y*** имеют то же распределение, что и координаты вектора ***X***.

Для определения значений a_{ij} воспользуемся системой уравнений (3.12), в которой теперь $a_{11} = 1$, а все $M(X_i) = x_i = x$, то есть имеют одинаковые значения. По-прежнему будем считать, что для моделирования задана матрица ковариаций ***K*** или матрица коэффициентов корреляции ***R***.

Рассмотрим *i*-ю строку системы (3.12). Умножим её на 1-ю строку этой системы и найдём математическое ожидание произведения $y_1 y_i$:

$$M(y_1 y_i) = a_{i1} M(x_1^2) + \sum_{j=2}^{i} a_{ij}\, M\left(x_1 x_j\right) = a_{i1}(x^2 + D_x) + x^2 \sum_{j=2}^{i} a_{ij}.$$

Следовательно,

$$y^2 + K_{21} = a_{i1} D_x + x^2 \sum_{j=2}^{i} a_{ij} = a_{i1} D_x + x^2 \sum_{j=1}^{i} a_{ij} = a_{i1} D_x + x^2.$$

Учитывая, что $y^2 = x^2$, окончательно получаем: $a_{i1} = K_{21}/D_x = R_{21}$.

Аналогично, умножая y_j на y_i, где $1 < j < i$, находим [34]:

$$a_{ij} = \left(R_{ij} - \sum_{k=1}^{j-1} a_{ik} a_{jk}\right) \Big/ a_{jj}.$$

Последним, учитывая ортогональность случайных коэффициентов, находится коэффициент a_{ii}: $a_{ii} = 1 - \sum_{j=1}^{i-1} a_{ij}$.

Отметим, что если при нечётной функции плотности моделируемых распределений в результате расчёта будет получено отрицательное значение a_{ij}, то система (3.12) с помощью рассмотренного метода нереализуема. Если же в этом случае указанная функция является чётной, то при моделировании в качестве значения соответствующего коэффициента берётся его модуль, а СВ X_i, умножаемая на данный коэффициент, берётся с противоположным знаком.

При наличии отрицательных коэффициентов в некоторой строке системы (3.12) значение коэффициента a_{ii} для этой строки определяется согласно выражению a_{ii}: $a_{ii} = 1 - \sum_{j=1}^{i-1} mod(a_{ij})$.

Пример. Необходимо моделировать векторы ***Y*** с координатами (y_1, y_2, y_3), равномерно распределёнными в [-3, 3], и с матрицей коэффициентов корреляции $\boldsymbol{R} = \begin{pmatrix} 1 & 0{,}8 & 0{,}2 \\ 0{,}8 & 1 & 0{,}1 \\ 0{,}2 & 0{,}1 & 1 \end{pmatrix}$. Найти значения $a_{ij} = P(A_{ij} = 1)$, подготовить программу моделирования***.***

Решение. В соответствии с полученными выше результатами выбираем:

$$a_{11} = 1;\ \ a_{21} = R_{21} = 0{,}8;\ \ a_{22} = 1 - a_{21} = 0{,}2;\ \ a_{31} = R_{31} = 0{,}2.$$

Вычисляем: $a_{32} = (R_{32} - a_{31}a_{21})/a_{22} = -0{,}3;\ \ a_{33} = 1 - 0{,}2 - 0{,}3 = 0{,}5.$

Составляем процедуру Vectors, моделирующую 1000 векторов ***Y***:

```
Procedure Vectors(c: TCanvas);
  var a: array [1..3,1..3] of real;
      X, Y: array[1..3] of real];
      k, n: integer; z: real;
  begin
    for k:=1 to 1000 do begin
      for n:= 1 to 3 do begin X[n]:= random(6)-1;
      Y[n]:= random(6)-1; end;
      Y[1]:=X[1];
      Z:= random;
      if z<0.8 then Y[2]:=X[1] else Y[2]:=X[2];
      Z:= random;
      if Z<0.2 then Y[3]:=X[1] else if Z<0.5 then Y[2]:= -X[2] else Y[3]:= X[3];
    end;
  end;
```

Изменяя в процедуре Vectors значения констант, написанных жирным шрифтом, можно варьировать значениями элементов матрицы ***R***.

Упражнение. Разобраться с вопросом: почему в процедуре Vectors координате y_2 присваивается значение координаты x_2 с обратным знаком?

Глава 4. Имитационное моделирование систем передачи информации по каналам связи

4.1. Технологии имитационного моделирования

Имитационное моделирование является наиболее распространённым методом исследования операций и многих динамических систем различного назначения. В последние годы реализуемые модели всё более усложняются. Однако возможности их программной реализации на ЭВМ существенно расширились с появлением специальных пакетов прикладных программ и развитых сред программирования, особенно системы Delphi, включающей в себя модули C^{++} Builder, Delphi for Win32, Delphi for Microsoft NET и Enterprise Core Objects for Code Geer RAD Studio.

Имитационное моделирование, как правило, выполняется с использованием ЭВМ. Конечно, на ЭВМ исследуются и модели, алгоритмы анализа которых используют аналитические решения. Однако большинство моделей современных динамических систем настолько сложны, что практически исключается возможность их аналитического решения. В то же время имитационное моделирование является мощным аппаратом статистического исследования различных процессов путём их многократного повторения и определения оценок и их характеристик.

Переменную величину, характеризующую при имитационном моделировании текущее значение модельного времени, часто называют часами модельного времени. При выполнении имитационного моделирования на основе объектно-ориентированного программирования используются два подхода к определению текущего модельного времени:

- определение времени от события к событию (дискретно-событийное время или дискретно-событийное моделирование);
- определение времени с постоянным шагом.

При дискретно-событийном моделировании фиксируются последовательные моменты появления соответствующих событий.

Отношение модельного времени к реальному времени называется масштабом модельного времени. Если этот масштаб меньше, больше или равен единице, то соответственно говорят, что моделирование выполняется в ускоренном, в замедленном или в естественном, то есть в натуральном) масштабе времени.

Технологии моделирования на ЭВМ сложных систем развиваются в двух основных направлениях. Во первых это – построение пакетов прикладных программ (ППП), позволяющих выполнять имитационное моделирование, а во вторых – развитие универсальных и специализированных сред программирования. В большинстве случаев специализация указанных ППП позволяет решать с их помощью соответствующие задачи более просто, чем с применением универсальных сред программирования. Однако универсальные среды программирования, несмотря на некоторые возможные элементы их специализации, позволяют решать очень широкий класс задач, включая задачи имитационного моделирования.

4.1.1 Пакеты прикладных программ для имитационного моделирования

До 21-го века основным ППП для моделирования считался MathCAD [21]. В настоящее время существует очень много визуальных средств моделирования. Многие ППП ориентированы на узкие прикладные области или задачи (медицина, электроника, поиск экстремальных значений функций и др.). Универсальные ППП для исследования структурно-сложных систем условно можно разделить на три группы:[5]

- пакеты «блочного моделирования»;
- пакеты «физического моделирования»;
- пакеты, ориентированные на схему гибридного автомата.

Приведённое деление является условным так как, что эти пакеты имеют много общего: позволяют строить многоуровневые иерархические функциональные схемы, предоставляют сходные возможности визуализации и анимации. Отличия обусловлены тем, какой из аспектов моделируемых сложных динамических системы сочтен наиболее важным.

Пакеты «блочного моделирования» ориентированы на графический язык иерархических блок схем, из которых можно “компоновать” модели различных систем. Элементарные блоки являются либо предопределенными, либо могут конструироваться с помощью некоторого специального вспомогательного языка более низкого уровня. Новый блок можно собрать из имеющихся

[5] exponenta.ru › soft/others/mvs/ds_sim.asp

блоков можно собирать новые блоки с использованием ориентированных связей и параметрической настройки. В число предопределенных элементарных блоков входят чисто непрерывные, чисто дискретные и гибридные блоки.

К достоинствами блочного моделирования следует отнести, прежде всего, чрезвычайную простоту создания не очень сложных моделей даже не слишком подготовленным пользователем. Другим достоинством является эффективность реализации элементарных блоков и простота построения модели системы. В то же время при создании сложных моделей приходится строить довольно громоздкие многоуровневые блок-схемы, не отражающие естественной структуры моделируемой системы. То есть “блочный подход” работает хорошо, когда есть подходящие стандартные блоки.

Наиболее известными пакетами «блочного моделирования» являются:

- подсистема **SIMULINK** пакета MATLAB (MathWorks, Inc.): http://www.mathworks.com);
- **EASY5** (Boeing);
- подсистема **SystemBuild** пакета MATRIX$_X$ (Integrated Systems, Inc.);
- **VisSim** (Visual Solution: http://www.vissim.com).

Пакеты «физического моделирования» позволяют использовать неориентированные и потоковые связи. Пользователь может сам определять новые классы блоков. Непрерывная составляющая поведения элементарного блока задается системой алгебро-дифференциальных уравнений и формул. Дискретная составляющая задается описанием дискретных событий которые задаются логическими условиями или являются периодическими. При возникновении соответствующих событий могут выполняться мгновенные присваивания переменным новых значений. Дискретные события могут распространяться по специальным связям. Изменение структуры уравнений возможно только косвенно через коэффициенты в правых частях (это обусловлено необходимостью символьных преобразований при переходе к эквивалентной системе).

Такой подход к построению пакетов очень удобен и естественен для описания типовых блоков физических систем. Недостатками являются необходимость символьных преобразований, что резко сужает возможности описания гибридного поведения, а также необходимость численного решения большого числа алгебраических уравнений, что значительно усложняет задачу автоматического получения достоверного решения.

К пакетам «физического моделирования» следует отнести:

– **“20-SIM”** (Controllab Products B.V: http://www.rt.el.utwente.nl/20sim/);
– **Dymola** (Dymasim: http://www.dynasim.se);
– **Omola**, **OmSim** (Lund University: http://www.control.lth.se/~cace/omsim.html);

Как обобщение опыта развития ППП этого направления международной группой ученых разработан язык **Modelica** (The Modelica Design Group: http://www.dynasim.se/modelica), предлагаемый в качестве стандарта при обмене описаниями моделей между различными пакетами.

Пакеты, основанные на использовании схемы гибридного автомата, позволяют очень наглядно и естественно описывать гибридные системы со сложной логикой переключений. Необходимость определения эквивалентной системы при каждом переключении заставляет использовать только ориентированные связи. Пользователь может сам определять новые классы блоков. Непрерывная составляющая поведения элементарного блока задается системой алгебро-дифференциальных уравнений и формул. К недостаткам следует также отнести избыточность описания при моделировании чисто непрерывных систем.

К данному направлению относится пакет **Shift** (California PATH: http://www/path.berkeley.edu/shift), а также отечественный пакет **Model Vision Studium** . Пакет Shift в большей степени ориентирован на описание сложных динамических структур, а пакет MVS – на описание сложных поведений.

Заметим, что между вторым и третьим направлениями нет непреодолимой пропасти. Невозможность из совместного использования обусловлена лишь сегодняшними вычислительными возможностями. В то же время, общая идеология построения моделей практически совпадает.

4.1.2 Моделирование в средах программирования

В последние годы активно развиваются среды программирования, использование которых в ЭВМ позволяет значительно облегчить работу программистов при разработке программ соответствующих классов.

В настоящее время наиболее популярной средой программирования, ориентированной на решение широкого класса прикладных задач и, в частности, позволяющей выполнять имитационное моделирование, является среда Delphi [46, 66]. Эта среда поддерживает так называемую быструю разработку прикладных программ, основанную на технологии визуального проектирования и событийного программирования. Суть этой разработки состоит в том, что среда берёт на себя большую часть рутинной работы, оставляя программисту работу по созданию диалоговых окон (визуальное проектирование) и процедур

обработки событий (событийное программирование). Производительность программиста возрастает при этом во много раз.

Статистический опрос программистов США, произведённый в конце прошлого десятилетия, показал, что 44% из них предпочитают работать в среде Delphi, 17% – на языке C^{++} и остальные – в других средах или на других языках.

В связи с указанным все примеры программной реализации рассмотренных в монографии методов моделирования приведены на языке Delphi, а основы программирования в среде Delphi изложены в приложении П1.

4.2 Моделирование каналов передачи цифровой информации

Различные системы связи всё чаще сроятся на цифровой основе, так как аналоговые каналы связи имеют относительно плохую помехозащищённость. При этом на передающем конце канала связи передаваемые биты цифровой информации используются для модуляции высокочастотного сигнала. На приёмном конце канала связи осуществляется демодуляция принимаемых сигналов. Помехи, которые имеют место и в каналах передачи цифровой информации, могут приводить к неверному приёму бит на приёмном конце канала связи, то есть к приёму символа “0” вместо “1” или к приёму “1” вместо “0”. Рассматриваемые каналы связи обычно называют бинарными. Задача моделирования заключается в оценке работоспособности каналов при различных структурах и разной интенсивности помех [63, 64 и др.].

Такая оценка может быть получена на основе статистических данных при проведении имитационного моделирования, для которого необходимо имитировать (моделировать) соответствующие помехи (рис. 4.1). При этом может оцениваться и эффективность используемого метода исправления ошибок в принимаемой информации.

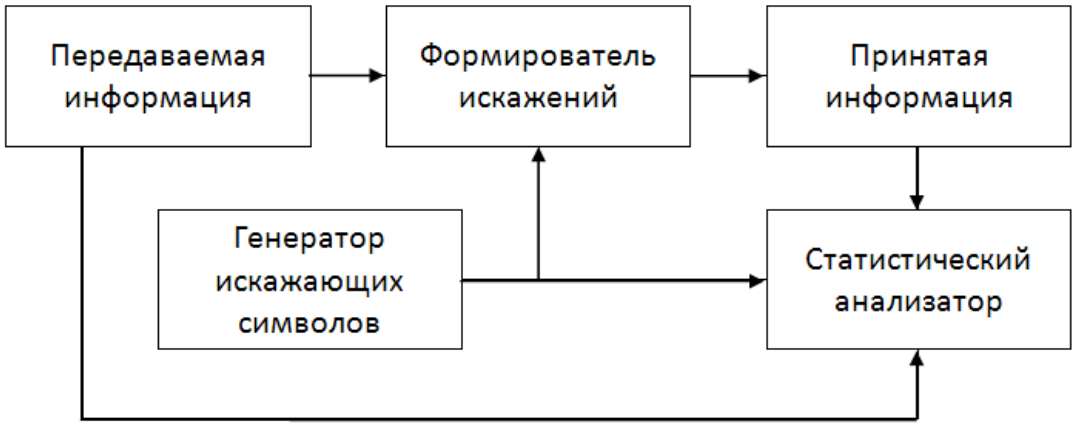


Рис. 4.1. Блок-схема моделирования канала связи

В процессе имитационного моделирования работы канала связи символы передаваемой информации последовательно поступают на формирователь искажений информации, являющийся сумматором по модулю два. На второй его вход поступают символы искажающей последовательности, инвертирующие соответствующие символы передаваемой информации. Принятые из канала символы частично или полностью восстанавливаются, если используются коды с исправлением ошибок. Статистический анализатор формирует статистику исправленных и неисправленных ошибок (двоичных символов или слов).

Искажающие информацию последовательности битовых помех выбираются в соответствии с той или иной моделью образования помех [29, 30 и др.]. Различные модели отражают специфику соответствующих каналов связи и особенности помех, имеющих место в этих каналах.

В качестве примера рассмотрим алгоритм моделирование помех, искажающих передаваемые символы, в соответствии с полигеометрической моделью, описанной в [22, 27]. Сущность этой модели состоит в следующем:

- канал связи может находиться в одном из двух состояний – в “хорошем” или в “плохом”;
- ошибки в передаваемой информации возникают независимо, причём при хорошем состоянии канала вероятность $W_{\text{х}}$ неверного приёма одного двоичного символа *X* очень мала, а при плохом вероятность $W_{\text{п}}$ указанного приёма может быть значительной;
- время $t_{\text{х}}$ (число шагов) пребывания канала в хорошем состоянии является случайной величиной и подчиняется полигеометрическому распределению:

$$P(t_{\text{х}}) = \sum_{i=1}^{m} a_i\,(1-p_i)p_i^{t_{\text{х}}-1},\ \text{где}\ \ \sum_{i=1}^{m} a_i = 1;$$

- время $t_{\text{п}}$ нахождения канала в плохом состоянии также является случайной величиной и тоже подчиняется полигеометрическому распределению:

$$P(t_{\text{п}}) = \sum_{i=1}^{n} b_i\,(1-q_i)q_i^{t_{\text{п}}-1},\ \text{где}\ \ \sum_{i=1}^{n} b_i = 1.$$

В приведённых выражениях a_i и b_i – весовые коэффициенты. Плохое состояние канала отражает тенденцию ошибок, обусловленных помехами в каналах связи, группироваться в так называемые пакеты [29], то есть появляться с более высокой вероятностью на отдельных интервалах времени. Схема алгоритма работы модели с описанным полигеометрическим распределением её характеристик приведена на рис. 4.2.

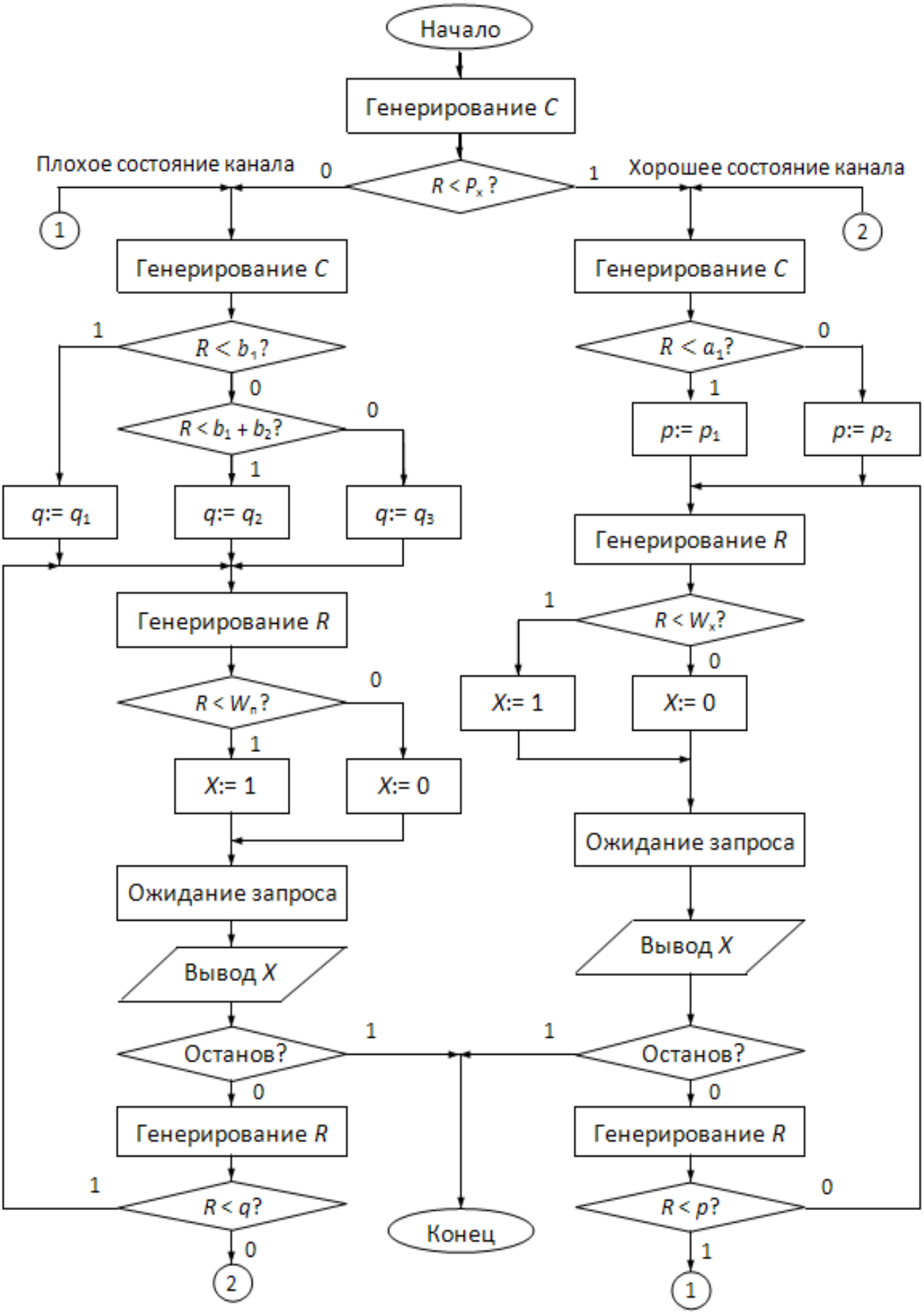


Рис. 4.2. Алгоритм работы модели при $m = 2$ и $n = 3$

Левая часть приведённого алгоритма моделирует работу канала связи в плохом состоянии с выбранными значениями b_1, b_2, $b_3 = 1 - b_1 - b_2$, q_1, q_2, q_3 и $W_{\text{п}}$, а правая – в хорошем состоянии с заданными значениями a_1, $a_2 = 1 - a_1$, p_1, p_2 и $W_{\text{х}}$. Блоки “ожидание запроса” служат для изменения частоты следования импульсов, отображающих символы 0 и 1 (символы *X*), при необходимости визуального наблюдения их динамики на экране монитора или записи их статистики.

В качестве источника случайностей в модели используется программа Randomize. Генерируемые ею случайные величины равномерно распределены в промежутке (0, 1) и обозначаются идентификатором *R* (random).

Вероятность $P_{\text{х}}$ того, что в начале “передачи” канал окажется в хорошем состоянии может быть определена как вероятность попадания случайной величины, равномерно распределённой на интервале длиной $M(t_{\text{х}}) + M(t_{\text{п}})$, на участок длиной $M(t_{\text{х}})$ этого интервала, равна [27]:

$$P_{\text{х}} = \frac{M(t_{\text{х}})}{M(t_{\text{х}}) + M(t_{\text{п}})} = \left[\sum_{i=1}^{m} \frac{a_i}{(1-p_i)^2}\right] \Big/ \left[\sum_{i=1}^{m} \frac{a_i}{(1-p_i)^2} + \sum_{i=1}^{n} \frac{b_i}{(1-q_i)^2}\right].$$

Аналогично можно определить и вероятность $P_{\text{п}}$ того, что в начале “передачи” канал окажется в плохом состоянии:

$$P_{\text{х}} = \frac{M(t_{\text{х}})}{M(t_{\text{х}}) + M(t_{\text{п}})} = \left[\sum_{i=1}^{m} \frac{a_i}{(1-p_i)^2}\right] \Big/ \left[\sum_{i=1}^{m} \frac{a_i}{(1-p_i)^2} + \sum_{i=1}^{n} \frac{b_i}{(1-q_i)^2}\right].$$

При реализации приведённого алгоритма в системе программирования Delphi for Win32 “ветки” алгоритма, реализующие работу канала в хорошем и в плохом состояниях, целесообразно представить в виде отдельных процедур. Переключение указанных веток и отслеживание общего объёма получения и вывода информации удобно поручить процедуре активизации формы.

Ниже приведен сокращённый вариант модуля unit Un_Kanal проекта Pr_Kanal, моделирующего алгоритм работы модели.

```
Unit  Un_Kanal;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, ExtCtrls;
```

```
type
  Tform1 = class(TForm)
   Image1: TImage;
   procedure  FormActivate(Sender: TObject);
  private
   { Private declarations }
  public
   { Public declarations }
  end;
var
  Form1: Tform1;
  R, Px, X: real;
  Stop, BSt, GSt, k, T: integer;                    // T – номер модельного шага
  procedure BadState(c: TCanvas);                   // Плохое состояние канала
  procedure GoodState(c: TCanvas);                  //  Хорошее состояние канала
implementation
{$R *.res}                  // Директива компилятору подключить файл ресурсов
procedure BadState(c: TCanvas);
  var b1, b2, b3, q0, q1, q2, q3, Wn: real;
  begin
    BSt:=1;                              //  Индикатор работы канала
    // . . .                             Операторы работы канала в плохом состоянии
    BSt:=0;                              // Конец плохого состояния
  end;
procedure GoodState(c: TCanvas);
  var a1, a2, p0, p1, p2, Wx: real;
  begin
    GSt:=1;                              //  Индикатор работы канала
    // . . .                             Операторы работы канала в хорошем состоянии
    GSt:=0;                              // Конец хорошего состояния
  end;
```

```
procedure Tform1.FormActivate(Sender: Tobject);
 begin
  randomize;
  Px:=0.7; T:=0;                          // Установка Px и начального значения T
  R:=random;
  if R<Px then GoodState(Image1.Canvas)
   else BadState(Image1.Canvas);        // Выбор и запуск 1-го состояния
  for k:=1 to 50 do begin                    // Выдача 50 последовательностей X
   if BSt=0 then GoodState(Image1.Canvas);          //  с каждой ветки алгоритма
   if GSt=0 then BadState(Image1.Canvas);
  end;
 end;
end.
```

Упражнения и вопросы. 1) В приведённом варианте программы не приведены операторы, реализующие работу канала в хорошем и в плохом состояниях (операторы двух веток алгоритма). Подготовить эти программы в среде программирования Delphi.

2) Добавить в программу два оператора, выполняющие вывод информации в буфер компьютера.

3) Почему в программе в списках идентификаторов ни один их них не обозначен буквой *C*?

4.3 Исследование характеристик систем со скрытой передачей информации

Ещё в 1963 г. автором был предложен способ установки синхронизма генераторов псевдослучайных кодов (ГПСК), расположенных на удалённых друг от друга объектах. В открытом варианте он был опубликован в [21, 22]. В режиме синхронизма указанные генераторы строго одновременно выдают одинаковые коды и могут использоваться как высокоточные часы, по сигналам которых объекты, на которых они расположены, могут одновременно выполнять какие-либо действия, например, обмениваться информацией.

4.3.1 Системы связи с конфиденциальной информацией. Принцип скрытия передаваемой информации

В ряде систем различного назначения осуществляется обмен информацией, имеющей конфиденциальный характер. В общем случае в таких системах

каждый её объект может связываться по каналу связи с каждым другим объектом этой системы (рис. 4.3). В системах с управлением от одного объекта функционирует связь между этим объектом и каждым другим объектом системы.

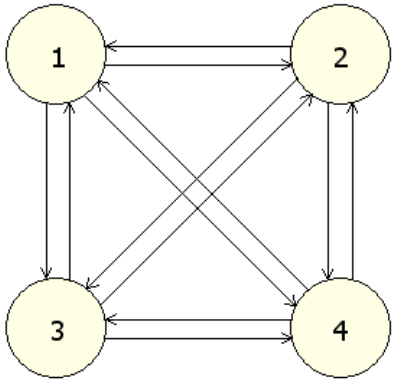


Рис. 4.3. Система связи четырёх объектов

Примером систем с конфиденциальной информацией являются системы, создаваемые в России под названием под названием “Электронные правительства”. Они создаются в соответствии с требованиями федерального закона РФ от 9 февраля 2009 г. №8-ФЗ «Об обеспечении доступа к информации о деятельности государственных органов и местного самоуправления». В сети Интернет уже имеются официальные сайты различных органов исполнительной власти, с которыми граждане связываются по электронной почте. Информация, поступающая в любой государственный орган от каждого гражданина недоступна другим гражданам, то есть имеет конфиденциальный характер. Разумеется, эта информация должна быть недоступна и потенциальным “взломщикам” сети при передаче её по каналам связи.

На основе предложенного способа установления синхронизма ГПСК автором была предложена модель (методика) построения алгоритмов передачи информации в подобных системах, обеспечивающая защиту информации в каналах их связи. Это модель получила название “скрытой передачи информации” [22]. Согласно предложенной методике запрашивающий связь объект посылал запрашиваемому объекту несколько символов работающего в нём ГПСК, по которым ГПСК запрашиваемого объекта надёжно входил в режим синхронизма с ГПСК запрашивающего объекта. После этого по параллельному каналу начинала передаваться последовательность случайных двоичных символов, задаваемых физическим генератором шума, то есть цифровой шум. По определённым кодам в последовательности символов ГПСК, являющимися паролем и действующим, например, в течение суток, вместо нескольких символов цифрового шума запрашиваемому объекту должны передаваться символы полезной информации для отбора их этим объектом из последовательности цифрового шума [15 – 18, 22].

Средняя частота передачи символов полезной информации зависит от числа разрядов указанного пароля и, естественно, от частоты Ω передачи кодов ГПСК. Если используемые псевдослучайные коды генерируются *n*-разрядными ГПСК, а код пароля состоит из *m* разрядов, где $M < n$, то, используя свойства М-последовательностей [50, 65 и др.], несложно установить, что за один $2^n - 1$ – шаговый период генерирования этих кодов в разрядах пароля 2^{n-m} раз появятся каждый из *m*-разрядных кодов, не состоящих из одних нулей, и $2^{n-m} - 1$ раз появится код, состоящий из одних нулей. Следовательно, при использовании в качестве пароля кода, в котором хотя бы один разряд равен 1, за каждый период генерирования кодов ГПСК по каналу передачи полезной информации будет передано 2^{n-m} символов этой информации. При этом средняя частота передачи символов полезной информации в последовательности цифрового шума равна $2^{n-m}/(2^n - 1) \cdot \Omega \approx 0{,}5^m\Omega$.

Существенным достоинством предложенной системы передачи информации является то, что она не конкурирует с известными криптографическими методами, а может дополнять их, так как передаваемая информация может быть зашифрована.

4.3.2. Установление синхронизма ГПСК на концах канала связи

Важными моментами в работе рассматриваемых систем являются установление синхронизма ГПСК, управляемых матрицей $\boldsymbol{A}$, и начало передачи и приёма полезной информации. Поэтому были проведены исследования этих моментов для случая передачи по каналам *n*-разрядных кодов и независимых помехах в канале связи, вызывающих неверный приём двоичных символов (простейшая модель образования искажающих двоичных символов – рис. 4.1).

Критерием входа ГПСК запрашиваемого объекта в режим синхронизма с ГПСК запрашивающего его объекта на шаге t+1 может быть прием запрашиваемым объектом *n*-разрядных кодов $\boldsymbol{X}(t)$ и $\boldsymbol{X}(t+1)$ таких, что

$$\boldsymbol{X}(t+1) = \boldsymbol{A} \cdot \boldsymbol{X}(t). \qquad (4.1)$$

В этом случае ложный синхронизм наступит при таком неверном приёме кодов $\boldsymbol{X}(t)$ и $\boldsymbol{X}(t+1)$, при котором указанное условие выполнится. Для увеличения вероятности входа ГПСК запрашиваемого объекта в правильный синхронизм с ГПСК запрашивающего объекта на шаге $t+1$ можно ужесточить критерий (4.1), потребовав, например, дополнительно к этому критерию ещё и необходимость выполнения условия $\boldsymbol{X}(t) = \boldsymbol{A} \cdot \boldsymbol{X}(t-1)$, то есть принимать на запрашиваемом объекте решение о достижении указанного синхронизма по критерию

$$\boldsymbol{X}(t) = \boldsymbol{A} \cdot \boldsymbol{X}(t-1),\ \boldsymbol{X}(t+1) = \boldsymbol{A} \cdot \boldsymbol{X}(t). \qquad (4.2)$$

В диссертационных работах Т.В. Жгун и А.А. Жгун [11 – 14, 15], научным руководителем которых являлся автор, исследованы вопросы установления синхронизма ГПСК согласно критериям (4.1) и (4.2) при бинарной модели возникновения искажающих информацию независимых векторных помех [2]. При этом считалось, что вход ГПСК запрашиваемого объекта в синхронизм с ГПСК запрашивающего объекта осуществляется согласно критерию (4.2).

Было показано, что вероятность $W(t)$ выполнения критерия наступления синхронизма не позднее шага *t* определяется разностным уравнением

$$W(t) = W(t-1) + [1 - W(t-1)] \cdot p_t^n,$$

где p_t^n – вероятность приёма кода, соответствующего критерию наступления синхронизма на шаге *t,* а p – вероятность правильного приёма одного разряда этого кода. Решение данного уравнения имеет вид:

$$W(t) = 1 - (1 - p^{2n}) \cdot [(1 - p^{2n}/(1 + p^n)]^{t-2}, \quad \text{для } t \geq 2.$$

Если одновременно в синхронизм вводятся *k* объектов (система из k объектов), то $W_k(t) = [W(t)]^k$. При $t \to \infty$ значение $W(t)$ стремится к единице, так как оно не учитывает возможность установления ложного синхронизма.

Аналогично для вероятности *Q*(*t*) выполнения критерия наступления ложного синхронизма не позднее шага *t* получаем:

$$Q(t) = U(t-1) + [1 - U(t-1)] \cdot q_t^n,$$

где Q_t – вероятность приёма кода, соответствующего критерию наступления ложного синхронизма на шаге *t*, а q – вероятность правильного приёма одного разряда этого кода. Решая данное уравнение, получаем:

$$Q(t) = 1 - \left[1 - \frac{(2^n - 1) \cdot (1 - q^n)^2}{2^{2n}}\right] \cdot \left[1 - \frac{(2^n - 2) \cdot (1 - q^2)}{2^{2n} - (2^{2n} - 2) \cdot (1 - q^n)^2}\right]^{t-2}.$$

При $t \to \infty$ значение $Q(t)$ стремится к единице, так как оно не учитывает возможность установления правильного синхронизма.

Используя отношение вероятностей $W(t)$ и $Q(t)$ на первых шагах процесса установления синхронизма, можно примерно судить об отношении вероятностей установления правильного или ложного синхронизма. С высокой достоверностью ответить на этот вопрос и найти оценки характеристик процесса установления синхронизма ГПСК, в том числе при различных моделях помех в канале связи, позволяет только имитационное моделирование.

С помощью имитационного моделирования, общая методика которого поясняется рис. 4.1, автором и Д.В. Кирьяновым исследована работоспособность Рассмотренной технологии установления синхронизма ГПСК при указанной выше бинарной модели возникновения искажающих информацию независимых векторных помех, инвертирующих принимаемые из канала двоичные символы 0 и 1 с задаваемой вероятностью q. Структура ГПСК на обоих концах канала связи определялась вектором их обратной связи $\boldsymbol{V}$ (рис. 2.2*b* и рис. 2.2.*c*), выбираемым по задаваемому в программе моделирования адресу из разработанной локальной базы примитивных многочленов [33].

ГПСК на приёмном конце канала (на запрашиваемом объекте) начинал работать в режиме синхронизма с ГПСК на передающем конце канала (на запрашивающем объекте) при выполнении условия (4.1).

Рассмотрим структуру модели, разработанной в среде Delphi и технологию работы на этой модели. Число шагов настройки ГПСК запрашиваемого объекта, отсчитываемых от начала посылки в канал настраивающей его последовательности, будем называть временем установления его синхронизма ГПСК этого объекта.

4.3.3. Моделирование процессов установления синхронизма ГПСК

В общем случае детальное исследование вопросов установления правильного и ложного синхронизма ГПСК проектируемых систем скрытой передачи информации может быть выполнено только с помощью имитационного моделирования. Для оценки числа шагов t передачи настроечных кодов, необходимых для надёжного ввода ГПСК запрашиваемых объектов в режим синхронизма с ГПСК запрашивающего объекта при симметричной бинарной модели помех в канале связи [2, 28] была разработана программа имитации работы модели системы связи на этапе установления синхронизма ГПСК [36, 37]. Программа оценивает число шагов, при которых устанавливался правильный или ложный синхронизм ГПСК в системе связи с несколькими объектами, и другие параметры процесса установления синхронизма, а также строит гистограммы числа шагов, на которых ГПСК запрашиваемых объектов входил в правильный или в ложный синхронизм. Модель предоставляет исследователю возможность выбора числа объектов системы, разрядности ГПСК этих объектов, типа примитивного многочлена, управляющего работой ГПСК. При каждом запуске программа выполняет 100000 сеансов настройки ГПСК запрашиваемого объекта, по результатам которых она строит гистограммы указанных результатов.

Кроме того, исследователю предоставляется возможность выполнять и одиночные сеансы. Программу можно просто дополнить для имитации помех в канале связи согласно различным моделям их образования.

На рис. 4.4 приведено окно исследователя системы связи с выведенными в него результатами ввода в синхронизм ГПСК некоторого объекта этой системы с ГПСК запрашивающего его объекта при очень сильных помехах в канале связи (реально ошибок в приёме одного бита кода с вероятностью более 0,01 не бывает). Высоты нижних прямоугольников выводимой гистограммы пропорциональны значениям статистических оценок вероятностей установления правильного синхронизма, а верхние прямоугольники – значениям статистических оценок вероятностей установления ложного синхронизма. Шкала числа шагов гистограммы автоматически подстраивается под полученное максимальное значение числа шагов установления синхронизма.

На верхней панели окна исследователя располагаются две кнопки radiobutton, с помощью которых выбирается одиночный (1 сеанс) или автоматический (100000 сеансов настройки связи) режим работы модели, разрядность и номер примитивного многочлена, вызываемого из локальной базе данных, а также вероятность неправильного приёма двоичного символа настраивающей М-последователь-ности. Каждый сеанс настройки связи продолжается до установления на ГПСК запрашиваемого объекта правильного или ложного синхронизма с ГПСК запрашивающего объекта.

На правую панель окна исследователя выводятся кнопка button запуска указанных сеансов и результаты, получаемые в последней серии 100000 сеансов настройки связи: максимальные значения числа шагов, потребовавшихся для установления правильного или ложного синхронизма, и другие, указанные в ней параметры процесса установления синхронизма ГПСК. При запуске новой серии 100000 сеансов настройки связи все старые данные, включая выведенную в окно гистограмму, автоматически стираются.

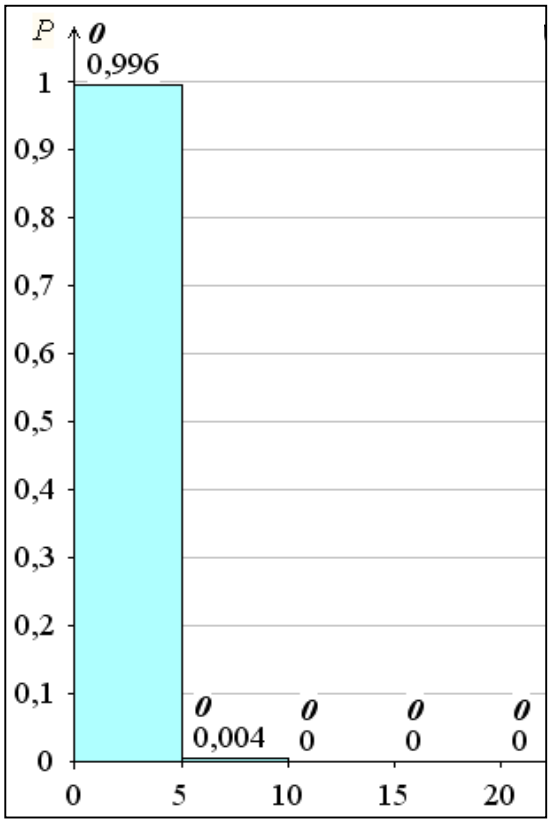
Проведённые с помощью модели исследования показали, что при использовании неразложимых многочленов одинаковой разрядности получаемые результаты практически не изменяются (в разработанную локальную базу данных [34] записаны все неразложимые многочлены с числом разрядов 8 (33 многочлена), 10 (113 многочленов) и 12 (434 многочлена).

Рис.4.4. Окно исследователя с результатами синхронизации ГПСК запрашиваемого объекта

При работе на модели можно проследить последовательность выполнений критериев, по которым ГПСК запрашиваемого объекта переключается в режим синхронной работы с ГПСК запрашивающего объекта. Это делается в режиме одиночных сеансов связи, при котором на левую информационную панель окна исследователя выводится число шагов синхронизирующей последовательности до момента установления правильного или ложного синхронизма. Причём в зависимости от того, в какой синхронизм установился ГПСК запрашиваемого объекта, указанное число шагов окрашивается в различный цвет.

Приведённый вариант модели, предназначенный для исследования процесса ввода в синхронизм одного ГСПК, можно использовать и случая ввода в синхронизм *k* ГПСК ($(k > 1)$. В этом случае программа должна повторить процесс синхронизации *k* раз, сравнить получены результаты и получить из них наихудший случай, так как вход ГСПК *k* запрашиваемых объектов системы в синхронизм с ГСПК запрашивающего объекта завершается тогда, когда в указанный синхронизм войдёт ГСПК каждого запрашиваемого объекта.

Ниже приведены результаты моделирования рассматриваемой установки синхронизма ГПСК со 100000 сеансами настройки при рабочих ($q \leq 0{,}05$) и при относительно больших ($0{,}05 < q \leq 0{,}1$) уровнях помех.



$T_{\text{син пр макс}} = 11$

$T_{\text{син лож макс}} = 0$

Оценки значений матем-х ожиданий $T_{\text{син пр}}$ и $T_{\text{син лож}}$

$\overline{M}(T_{\text{син пр}}) = 1{,}5$

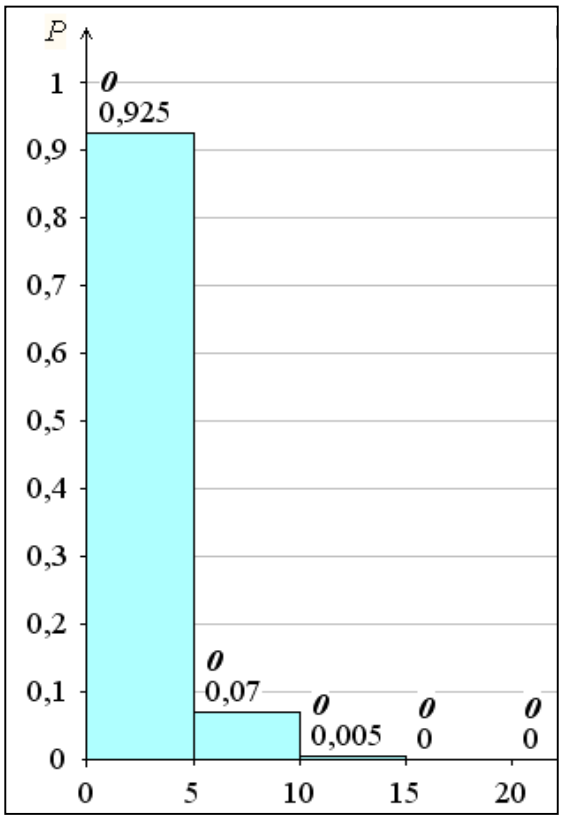
$\overline{M}(T_{\text{син лож}}) = 0$

Число сеансов (N), приведших к установлению правильного или или ложного синхронизма:

$N_{\text{син пр}} = 100000$

$N_{\text{син лож}} = 0$

а) Результаты при *q* = 0,025



$T_{\text{син пр макс}}=12$

$T_{\text{син лож макс}}=\boldsymbol{0}$

Оценки значений матем-х ожиданий $T_{\text{син пр}}$ и $T_{\text{син лож}}$

$\overline{M}(T_{\text{син пр}})=1,5$

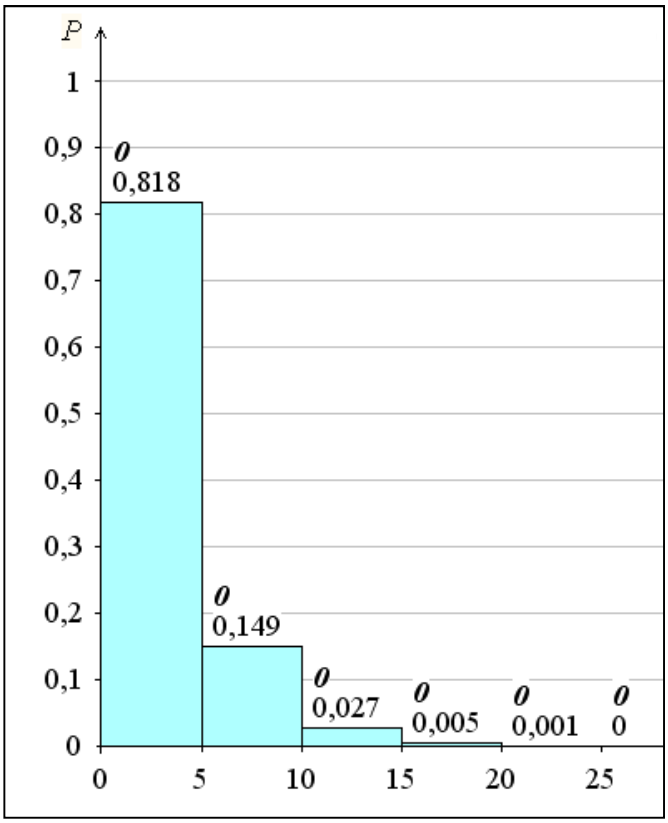
$\overline{M}(T_{\text{син лож}})=\boldsymbol{0}$

Число сеансов (N), приведших к установлению правильного или или ложного синхронизма:

$N_{\text{син пр}}=100000$

$N_{\text{син лож}}=\boldsymbol{0}$

b) Результаты при q = 0,05



$T_{\text{син пр макс}}=42$

$T_{\text{син лож макс}}=\boldsymbol{11}$

Оценки значений матем-х ожиданий $T_{\text{син пр}}$ и $T_{\text{син лож}}$

$\overline{M}(T_{\text{син пр}})=3,5$

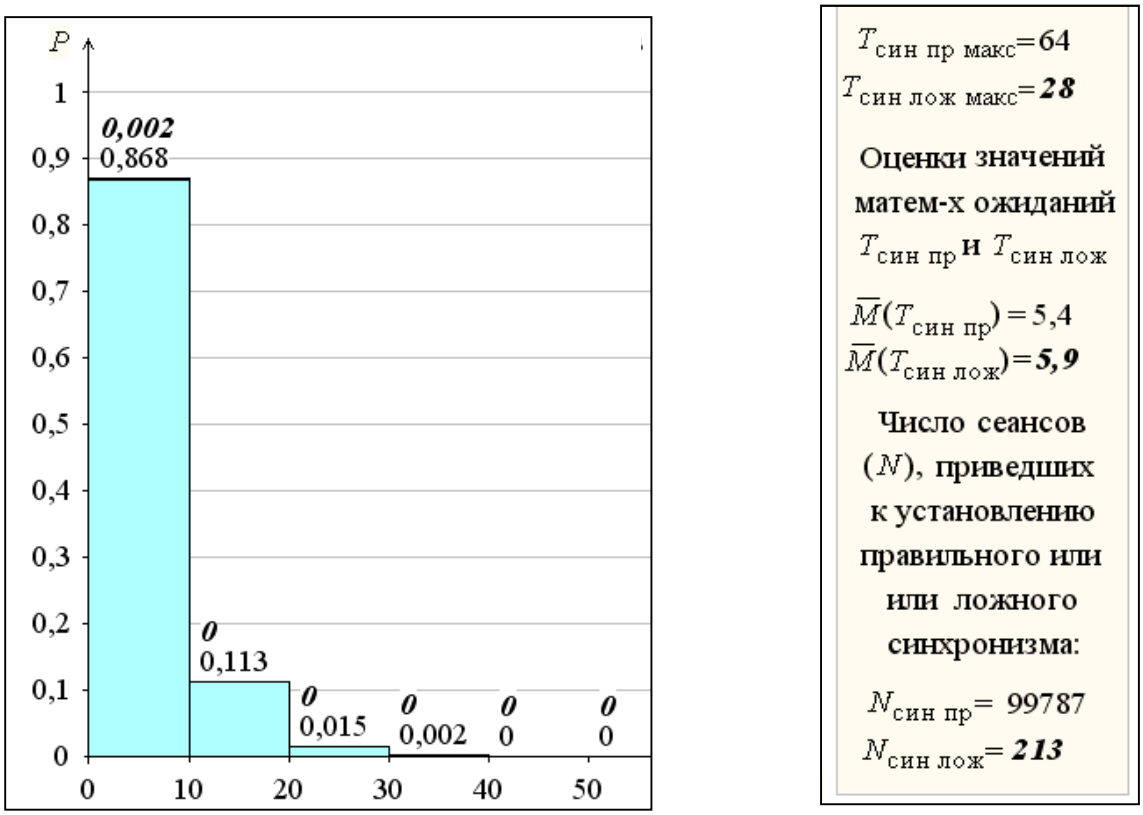
$\overline{M}(T_{\text{син лож}})=\boldsymbol{3,5}$

Число сеансов (N), приведших к установлению правильного или или ложного синхронизма:

$N_{\text{син пр}}=99975$

$N_{\text{син лож}}=\boldsymbol{25}$

c) Результаты при q = 0,75



d) Результаты при q = 0,1

Рис. 4.5. Результаты процессов установления синхронизма ГПСК при разных уровнях помех в канале связи

После вхождения ГПСК всех запрашиваемых объектов в режим синхронизма может быть начата передача им полезной информации в смеси с цифровым шумом. Предварительно запрашивающий объект должен убедиться в том, что переключения ГПСК всех запрашиваемых объектов в режим синхронизма произошло. Это запрашиваемые объекты могут сообщить, например, по обратным каналам связи или по каналам передачи им цифрового шума, пока они для этой цели не используются.

После окончания передачи полезной информации запрашивающий объект может сообщить об этом всем принимающим объектам обусловленным кодом по каналам передачи цифрового шума с полезной информацией.

4.3.5 Модуль задания параметров, запуска модели и выдачи результатов

В программе работы модели, разработанной автором совместно с Д.В. Кирьяновым [36, 37], реализуются модульный, структурный и объектно-ориентированный принципы построения программ. Программа использует

разработанную специально для неё локальную базу примитивных многочленов [35] PolinomsBase, для подключения которой к основной программе имя базы указывается в списке вызываемых программой утилит.

Наиболее информативным модулем программы является модуль реакции на событие “Нажатие кнопки Ввод и запуск” (кнопка Button 3). Нажатие этой кнопки вызывает ввод в программу указанных исследователем исходных данных (значений *q*, *n* и номера неприводимого многочлена, то есть вектора V – рис. 2.2), а также выполнение программы с выводом полученных результатов. Ниже приводится программа этого модуля (procedure TForm1.Button3Click) с пояснениями, отделяемыми от программы знаком //.

```
procedure TForm1.Button3Click(Sender: TObject);
 var X1,X2,Y1,Y2,Z1,Z2: array[0..8] of integer;  // Регистры ГПСС передатчика (X1,
                    // X2) и приемника (основные Y1 и Y2, вспомогательные Z1, Z2),
    D,L,P0,P1,P2,U,T0,T1,T2,R1,R2,Prmax,Lomax,G: integer;
    S: string;  W: real;
    H1,H2: array[1..10] of real;      // Значения разрядов гистограмм
              // K - номер вектора ОС ГПСС (глобальная переменная)
              // L - число циклов,
              // N - разрядность полиномов (глобальная переменная)
              // P - текущий шаг данного запуска
              // T - счетчик шагов (число шагов входа в синхронизм)
              // D - максимальное значение переменной по оси абсцисс
    Tsin: array[1..100000] of integer; // Число шагов до входа ГПСК в прав. синхр.
    Tlog: array[1..100000] of integer; // Число шагов до входа ГПСК в ложн. синхр.
    label Met1,Met2, Met3;
 begin
    Image5.Canvas.Font.Name:='Times New Roman';
    Image5.Canvas.Font.Style:=[ ];
    Image5.Canvas.Font.Size:=14;
          // m=2 и n=8  учитывается в программе
    S:=MaskEdit8.Text;              // Чтение выбранного номера вектора V
    K:=Round(StrToFloat(S));        // Получение этого номера в виде числа
    if K>33) or (K<1) then goto Met3;    // Ограничение выбора номеров для n=8
    S:=MaskEdit7.Text;              // Ввод вероятности ошибки в приеме
                                    // N-разрядного кода
```

```
    Q:=StrToFloat(S);                  // Перевод текста в число
    if Q>0.5 then begin Q:=0.5; MaskEdit7.Text:='0,5'; end;    // Ограничение Q
    V08(K);     // Выбор из базы PolinomsBase 8-координатного вектора V
                //  с номером K в виде N-разрядного целого числа VN
    G:=VN;      // VN – номер вектора, используемый в приложении PolinomsBase
                // (в данной процедуре VN – глобальная переменная)
    if G>9999999 then begin V[1]:=1;  G:=G-10000000; end else V[1]:=0;
    if G>999999 then begin V[2]:=1;  G:=G-1000000; end else V[2]:=0;
    if G>99999 then begin V[3]:=1;  G:=G-100000; end else V[3]:=0;
    if G>9999 then begin V[4]:=1;  G:=G-10000; end else V[4]:=0;
    if G>999 then begin V[5]:=1;  G:=G-1000; end else V[5]:=0;
    if G>99 then begin V[6]:=1;  G:=G-100; end else V[6]:=0;
    if G>9 then begin V[7]:=1;  G:=G-10; end else V[7]:=0;
    if G>0 then V[8]:=1 else V[8]:=0;          // Определили значения всех разрядов
      // кода V, управляющего ГПСК на вызывающем объекте (на передатчике)
      // Далее устанавливаем начальное состояние генераторов передатчика
      //    (для удобства реализации побайтовой передачи кодов в канал связи
      //    используем в передатчике 2 8-разрядных регистра: X1 и X2
    X1[1]:=0;  X1[2]:=0;  X1[3]:=0;  X1[4]:=0;  X1[5]:=0;  X1[6]:=0; X1[7]:=0;  X1[8]:=1;
     // Вводим генератор X2 (с него идёт передача кодов в канал связи) в
     // опережение на 8 шагов:
    for U:=1 to 7 do X2[U]:=V[U]; X2[8]:=1;
      // Установка начальных состояний регистров ГПСК передатчика окончена
      // Устанавливаем число сеансов (L):
    if RadioButton5.Checked then L:=1 else
          if RadioButton6.Checked then L:=100000 else
              ShowMessage ('Задайте число запусков');
      // Начало работы с L=1
    if L=1 then begin
     O:=false;
      // Реализуем 1 шаг работы ГПСК передатчика
     T0:=0;
Met1:  for I:=0 to 7 do X1[I]:=X1[I+1]; X1[8]:=0;
    if X1[0]=1 then begin
       X1[8]:=1;
       for I:=1 to 7 do X1[I]:=X1[I]+V[I]-2*X1[I]*V[I];
    end;
```

```
for I:=0 to 7 do X2[I]:=X2[I+1]; X2[8]:=0;
if X2[0]=1 then begin
   X2[8]:=1;
   for I:=1 to 7 do X2[I]:=X2[I]+V[I]-2*X2[I]*V[I];
end;              // Шаг ГПСК выполнен
// Далее реализуется передача двух байтов на приемник
for I:=1 to 8 do begin
   Y1[I]:=X1[I];
   W:=random;
   if W<Q then Y1[I]:=1-Y1[I];       // При выполненении указанного условия
   Y2[I]:=X2[I];                     // помеха инвертирует разряд кода Y1
   W:=random;                        // Аналогичные действия выполняются и
    if W<Q then Y2[I]:=1-Y2[I];      //  для кода Y2
 end;     // Приемник получил коды, которые могут быть искажены
 T0:=T0+1;            //  Увеличение на 1 показания счётчика выполненных
                      //  сеансов связи
// Начинается проверка выполнения критерия входа ГПСК в синхронизм
 for I:=1 to 8 do Z1[I]:=Y1[I];            // Скопирован код Y1 в Z1
 for U:=1 to 8 do Z2[U]:=Z1[U];
 for J:=1 to 8 do begin        // Выполняется 8 шагов работы ГПСК
   for I:=0 to 7 do Z2[I]:=Z2[I+1];  Z2[8]:=0;
   if Z2[0]=1 then begin
     Z2[8]:=1;
     for I:=1 to 7 do Z2[I]:=Z2[I]+V[I]-2*Z2[I]*V[I];
   end;
 end;     // Получен код Z2, опережающий исходный код  Z1 на 8 шагов.
          // Если этот код совпадает с кодом Y2, то это означает, что
          // ГПСС приемника вошёл в синхронизм. Проверка этого условия:
 J:=1;
 for I:=1 to 8 do if Z1[I]<>Y1[I] then J:=0;
 for I:=1 to 8 do if Z2[I]<>Y2[I] then J:=0;
 if J=0 then goto Met1;              // Если J=0, то синхронизма нет,
                                     // и выполняются следующие 8 шагов:
 if J=1 then begin    // Проверка: является ли это вхождение верным
    U:=1;
    for I:=1 to 8 do if Z1[I]<>X1[I] then U:=0;
```

```
        for I:=1 to 8 do if Z2[I]<>X2[I] then U:=0;
        if U=1 then Image5.Canvas.Font.Color:=clBlue
           else Image5.Canvas.Font.Color:=clRed;
        Image5.Canvas.TextOut(70,320+24*H,IntToStr(T0));
        H:=H+1;
        if H=18 then H:=0;
        Image5.Canvas.TextOut(70,320+24*H,'           ');
     end;
   end;  // Закончился 1 цикл работы модели при L=1 (одиночные сеансы)
   // Начинается цикл с 100000 сеансами связи (L=100000)
   T1:=0; T2:=0; P1:=0; P2:=0; Prmax:=0; Lomax:=0;
   for I:=1 to 100000 do Tsin[I]:=0;
   if (L=10000) or (L=100000) then for G:=1 to L do begin
      O:=true;
      P0:=0;   // Номер шага в текущем цикле
Met2: P0:=P0+1;
      for I:=0 to 7 do X1[I]:=X1[I+1];     // Начинается выполнение шага
      X1[8]:=0;                            //  генератора X1
      if X1[0]=1 then begin
         X1[8]:=1;
         for I:=1 to 7 do X1[I]:=X1[I]+V[I]-2*X1[I]*V[I];
      end;                                 // Шаг генератора X1 выполнен
     for I:=0 to 7 do X2[I]:=X2[I+1];     // Начинается выполнение шага
     X2[8]:=0;                            // генератора X2
     if X2[0]=1 then begin
        X2[8]:=1;
        for I:=1 to 7 do X2[I]:=X2[I]+V[I]-2*X2[I]*V[I];
     end;                  // Шаг генератора X2 выполнен
          // Далее реализуется передача двух байтов на приемник
     for I:=1 to 8 do begin
        Y1[I]:=X1[I];
        W:=random; if W<Q then Y1[I]:=1-Y1[I];   // При выполнении этого
        Y2[I]:=X2[I];      // условия помеха инвертирует i-й разряд кода
         W:=random; if W<Q then Y2[I]:=1-Y2[I];
     end;                  // Приемник получил коды (байты), отдельные разряды
                           // которых могут быть приняты неверно
```

```
   for U:=1 to 8 do Z2[U]:=Z1[U];
   for M:=1 to 8 do begin        // Выполняется 8 шагов ГПСК Z2
     for I:=0 to 7 do Z2[I]:=Z2[I+1]; Z2[8]:=0;
      if Z2[0]=1 then begin
         Z2[8]:=1;
         for I:=1 to 7 do Z2[I]:=Z2[I]+V[I]-2*Z2[I]*V[I];
      end;
   end;     // Получен код Z2, опережающий исходный Z1 на 8 шагов
   M:=1;   // Начинается проверка правильности приёма кода в приёмник
   for I:=1 to 8 do if Z1[I]<>Y1[I] then M:=0;
   for I:=1 to 8 do if Z2[I]<>Y2[I] then M:=0;
   if M=0 then begin P:=P+1; goto Met2; end;
      // Если M=0, то синхронизма нет, выполняется возврат  к началу цикла.
   if M=1 then begin          // Начало проверки: является ли это вхождение
     U:=1;                    // в синхронизм правильным?
     for I:=1 to 8 do if Z1[I]<>X1[I] then U:=0;
     for I:=1 to 8 do if Z2[I]<>X2[I] then U:=0;
                  // Итак, если U=1, то вход в синхронизм правильный, если же
                  // U=0, то вход в синхронизм – неверный (ложный)
   end;
   if U=1 then begin T1:=T1+1; Tsin[G]:=P0; Tlog[G]:=0;
                if P0>P1 then P1:=P0; end
        else begin T2:=T2+1; Tsin[G]:=0; Tlog[G]:=P0;
                if P0>P2 then P2:=P0; end;
end;   //  Конец очередного цикла по переменной G, кончая последним
       //  100000-м
       // Начинается анализ статистики, стираются старые и записываются
       // новые данные
Image5.Canvas.Brush.Color:=clCream;
Image5.Canvas.Font.Color:=clBlue;
Image5.Canvas.TextOut(1034,307,'         ');
if O=true then Image5.Canvas.TextOut(1034,307,IntToStr(P1))
Image5.Canvas.Font.Style:=[fsItalic]+[fsBold];
Image5.Canvas.TextOut(1034,335,'         ');
if O=true then Image5.Canvas.TextOut(1034,335,IntToStr(P2));
P0:=P1; if P2>P1 then P0:=P2;     // Нашли максимальное значение цикла
```

```
        // Далее находятся оценки математических ожиданий M(T) числа
        // циклов входа в правильный и в ложный синхронизм
 X1[0]:=0; Y1[0]:=0;
 for I:=1 to L do begin X1[0]:=X1[0]+Tsin[I]; Y1[0]:=Y1[0]+Tlog[I]; end;
 if T1>=1 then W:=round(X1[0]/T1*10)*0.1 else W:=0;
 Image5.Canvas.Brush.Color:=clCream;
 Image5.Canvas.TextOut(1029,468,'          ');
 Image5.Canvas.Font.Style:=[];
 if O=true then Image5.Canvas.TextOut(1030,468,FloatToStr(W));
      // Вывод значения M(Tсинхр. правильный)
 if T2>=1 then W:=round(Y1[0]/T2*10)*0.1 else W:=0;;
 Image5.Canvas.Font.Color:=clRed;
 Image5.Canvas.TextOut(1029,494,'          ');
 Image5.Canvas.Font.Style:=[fsItalic]+[fsBold];
 if O=true then Image5.Canvas.TextOut(1031,494,FloatToStr(W));
      // Вывод значения M(Tсинхр. ложный)
     // Начинаются стирание прежней гистограммы и разметка оси X
 Image5.Canvas.Brush.Color:=clWhite;
 Image5.Canvas.Pen.Color:=clWhite;
 Image5.Canvas.RectAngle(236,284,290,390);
 Image5.Canvas.RectAngle(231,323,831,730);   // Стёрли старую гистограмму
 Image5.Canvas.Pen.Color:=clSilver;
 for J:=1 to 10 do begin                    // Рисуется новая координатная сетка
    Image5.Canvas.MoveTo(231,730-40*J);
    Image5.Canvas.LineTo(831,730-40*J);
 end;
Image5.Canvas.MoveTo(831,330);      // Рисуется линия правой границы
Image5.Canvas.LineTo(831,710);      // координатной сетки
Image5.Canvas.TextOut(240,740,'                                                    '+
   '                          ');  // Стирание значений старой разметки оси абсцисс
if P0<=10 then D:=1 else if P0<=50 then D:=5 else if P0<=100 then D:=10
    else if P0<=500 then D:=50 else if P0<=1000 then D:=100 else
    if P0<=5000 then D:=500 else if P0<=10000 then D:=1000 else if P0<=50000
    then D:=5000 else if P0<=100000 then D:=10000 else if P0<=500000
    then D:=50000 else if P0<=1000000 then D:=100000 else D:=200000;
          // Нашли шаг D новой разметки оси абсцисс
```

```
U:=0;
Image5.Canvas.Font.Color:=clBlack;
Image5.Canvas.Font.Style:=[];
for J:=1 to 10 do begin // Корректировка расположения значений оси абсцисс
    if (D*J>9) and (D*J<100) then U:=4;
    if (D*J>99) and (D*J<1000) then U:=8;
    if (D*J>999) and (D*J<10000) then U:=12;
    if (D*J>9999) and (D*J<100000) then U:=16;
    if (D*J>99999) and (D*J<500000) then U:=21;
    Image5.Canvas.TextOut(226-U+60*J,740,IntToStr(D*J));
end;                    // Записана новая разметка оси абсцисс
R1:=0; for I:=1 to 10 do if (P1>(I-1)*D) and (P1<=I*D) then R1:=I;
R2:=0; for I:=1 to 10 do if (P2>(I-1)*D) and (P2<=I*D) then R2:=I;
  //  Продолжается анализ значений полученной статистики вероятностей
  //  ложного синхронизма
for J:=1 to 10 do begin
  Tsin[J]:=0; Tlog[J]:=0;
  for I:=1 to L do if (Tsin[I]>D*(J-1)) and (Tsin[I]<=D*J) then Tsin[J]:=Tsin[J]+1;
  for I:=1 to L do if (Tlog[I]>D*(J-1)) and (Tlog[I]<=D*J) then Tlog[J]:=Tlog[J]+1;
end;                // Получили значения разрядов гистограмм правильного и
                    // ложного синхронизма
                    // Рисуются гистограммы
for I:=1 to 10 do H1[I]:=400/L*Tsin[I];     // Значения
for I:=1 to 10 do H2[I]:=400/L*Tlog[I];     // гистограмм
for I:=1 to 10 do W:=W+H2[I];
Image5.Canvas.Brush.Color:=$00FFFFAF;
Image5.Canvas.Pen.Color:=clBlack;
for J:=1 to 10 do begin if J=1 then I:=1 else I:=0;
Image5.Canvas.Rectangle(231-I+(J-1)*60,
730-round(H1[J]),232+J*60,731); end;
Image5.Canvas.Brush.Color:=clSilver;
for J:=1 to 10 do begin if J=1  then I:=1 else I:=0;
Image5.Canvas.Rectangle(231-I+(J-1)*60,
730-round(H1[J])-round(H2[J]),232+J*60,731-round(H1[J])); end;
Image5.Canvas.Brush.Color:=clWhite;
for I:=R1+1 to 10 do Tsin[I]:=0;
```

```
  if O=true then for J:=1 to 10 do Image5.Canvas.TextOut(238+(J-1)*60,708-
      round(H1[J])-round(H2[J]),FloatToStr(0.001*round(1000*Tsin[J]/L)));
  for I:=R2+1 to 10 do Tlog[I]:=0;
  W:=0;   // Начальное значение суммы вероятностей ложного синхронизма
  Image5.Canvas.Font.Style:=[fsItalic]+[fsBold];
  if O=true then for J:=1 to 10 do begin
     P:=0.001*round(1000*Tlog[J]/L);
     Image5.Canvas.TextOut(238+(J-1)*60,690-round(H1[J])-round(H2[J]),
      FloatToStr(P));
      W:=W+P;
 end;
 Image5.Canvas.Brush.Color:=clCream;
 Image5.Canvas.Font.Color:=clBlack;
 Image5.Canvas.TextOut(858,293,'        ');
 Image5.Canvas.TextOut(861,293,FloatToStr(W));
 Image5.Canvas.TextOut(1014,696,'            ');
 Image5.Canvas.Font.Style:=[];
 if O=true then Image5.Canvas.TextOut(1016,696,IntToStr(T1));
 Image5.Canvas.TextOut(1014,723,'            ');
 Image5.Canvas.Font.Style:=[fsItalic]+[fsBold];
 if O=true then Image5.Canvas.TextOut(1018,723,IntToStr(T2));
 Image5.Canvas.Pen.Color:=clBlack;
 Image5.Canvas.MoveTo(230,730); Image5.Canvas.LineTo(870,730);
Met3: end;  // Конец процедуры
end.
```

4.3.4 Модернизация модели

В последние годы предложенная модель модернизируется [56, 63]. Необходимость модернизации была вызвана тем, что потенциальные “взломщики” сети, не зная указанного пароля, но зная используемую для связи последовательность *n*-разрядных кодов ГПСК передатчика, имеют возможность анализировать последовательность случайных символов или кодов с внедрёнными в неё символами или кодами полезной информации. Как уже указывалось в п.4.3.1, количество вариантов последовательностей, которые нужно для этой цели проанализировать с помощью современных ЭВМ, очень велико. К тому же передаваемая информация может быть ещё зашифрована. Следовательно,

вероятность выделения взломщиком полезной информации из смеси её с цифровым шумом ничтожно мала. Однако её можно уменьшить в огромное число раз, увеличив соответственно число последовательностей, которые необходимо проанализировать взломщику. Это достигается одновременным переходом ГПСК, вошедших в режим синхронизма, на другой алгоритм их работы. Поэтому изменятся и моменты передачи фрагментов полезной информации в смеси с цифровым шумом. В этом случае для выделения полезной информации из указанной смеси взломщику потребуется во много раз большее время, а течение которого система передачи информации может не один раз сменить алгоритм работы ГПСК.

Можно предложить несколько вариантов одновременного перехода ГПСК, работающих в режиме синхронизма, на другой алгоритм формирования их выходных последовательностей без передачи об этом какой-либо информации в каналы связи. Например:

- по некоторой кодовой комбинации в генерируемой их ГПСК последовательности;
- по последовательности кодовых комбинаций в генерируемой их ГПСК последовательности;
- по числу кодов, выданных ГПСК с момента вхождения их в синхронизм.

Указанное переключение можно производить несколько раз.

Окончание передачи и приёма полезной информации также может выполняться одновременно по посылке в канал определённой последовательности кодов.

ПРИЛОЖЕНИЯ

П1. Основы работы в среде Delphi

П1.1. Общие сведения о средах программирования

Понятие “среда программирования” появилось в конце прошлого века. Рассматриваемые среды являются важным помощником пользователей в разработке программ и других задач. Например: Microsoft Office Promt 2007 – профессиональная среда для перевода, Microsoft Office PowerPoint – среда для подготовки слайдов и т.д.

В конце 20-го – в начале 21-го века стали появляться и среды, предназначенные именно для разработки программ – Code Gear. Наиболее популярными средами являются C++ и Delphi (разработчиком этих сред являются основанная в 1982 году компания Borland, в которую входят фирмы Motorola – основной разработчик системы C++, Legadero и другие). Основная цель этих систем – упростить работу программистов при разработке программных продуктов. На английском языке слово “Delphi” произносится как Делфай или Делпфай, в России произносят “Делфи”.

Система C++ более удобна для системного программирования, а Delphi – для прикладного. Каждая из них имеет свои достоинства. Известно несколько модификаций как Делфи, так и C++ (Delphi 5, Delphi 7, Delphi 2007, Delphi NET, Delphi 2010). В конце 2004 года компания Borland объявила, что продолжит развитие классического C++ Builder {Builder – создатель программ, составитель программ}, объединив его со средой Delphi и прекратив разработку самостоятельной (отдельной) системы C++. Постепенно компания перешла на среды, использующие возможности интернета (NET).

Delphi – это программная среда визуального программирования и создания как приложений клиент/сервер, так и общих приложений для ОС Windows, то есть для ОС, разработанных компанией Microsoft. Эта среда сочетает в себе высокопроизводительный компилятор с языком Object Pascal (объектно-ориентированный паскаль), визуальные механизмы программирования, инструмент создания приложений клиентов для работы с различными локальными и удаленными базами данных, а также для подготовки научных документов.

Delphi – это система быстрой разработки приложений для Windows, которую нередко называют система *RAD* (*Rapid Application Development*): {rapid – быстрый, скорый, development – усовершенствование, развитие}. *Основой Delphi является графическая среда разработки приложений, кратко назы-*

ваемая интегрированной средой разработки (Integrated Development Environment). Последние модификации системы Delphi (Delphi 2007, Delphi 2010 и Delphi NET) требуют установки на персональном компьютере компонентов Microsoft.NET Framework 2.0, Microsoft.NET Framework SDK 2.0 и Microsoft.

Delphi 2010 существует в трёх вариантах: Professional, Enterprise и Architect. Каждый комплект включает набор средств и компонентов, представляющих собой готовые программные решения и обеспечивающих разработку высокоэффективных приложений различного назначения, в том числе для работы с базами данных. Комплекты Enterprise и Architect позволяют работать с удалёнными базами данных, то есть подключаться к ним через сервер. *Комплект Professional работает только с локальными базами данных.*

В последние годы разработчики Delphi активно продвигают так называемые NET-технологии (новые технологии на основе интернета). Эти технологии имеют хорошие перспективы развития, так как они используют мощную, быстро развивающуюся базу интернета.

П1.2. Структура среды Delphi

Интегрированная среда разработки Delphi является многооконной системой. Она включает в себя все необходимое для быстрой разработки Windows-приложений, может гибко настраиваться и имеет некоторый стандартный вид. Она является прекрасным средством для исследования широкого класса задач с помощью математического моделирования на ЭВМ.

Разрабатываемые в среде Delphi программы, как правило, состоят из нескольких программных модулей, имеющих соответствующие расширения. При этом создаваемые пользователями программы фактически являются приложениями к проекту – к главной, короткой программе комплекса программных модулей, создаваемой средой Delphi при первом запуске среды.

Рабочие окна и вкладки. Разработка программ ведётся в среде, состоящей из функциональных окон (рис. П1.1). В отдельных окнах имеются вкладки, занимающие часть окон. Вкладки представляют собой панели элементов (инструментов), используемых при разработке программ. Каждое окно и вкладка имеют названия. На рис. П1.1 приведены все окна, отмеченные цифрами:

1 – главное окно, в заголовке которого отображаются название проекта;
2 – окно дизайнера формы, в котором находится форма (3) – окно конструктора форм (заготовка окна приложения);
4 – окно дерева проектов;
5 – окно структуры проекта (Project Manager);

6 – окно инспектора объектов;

7 – окно палитры инструментов (Tool Palette).

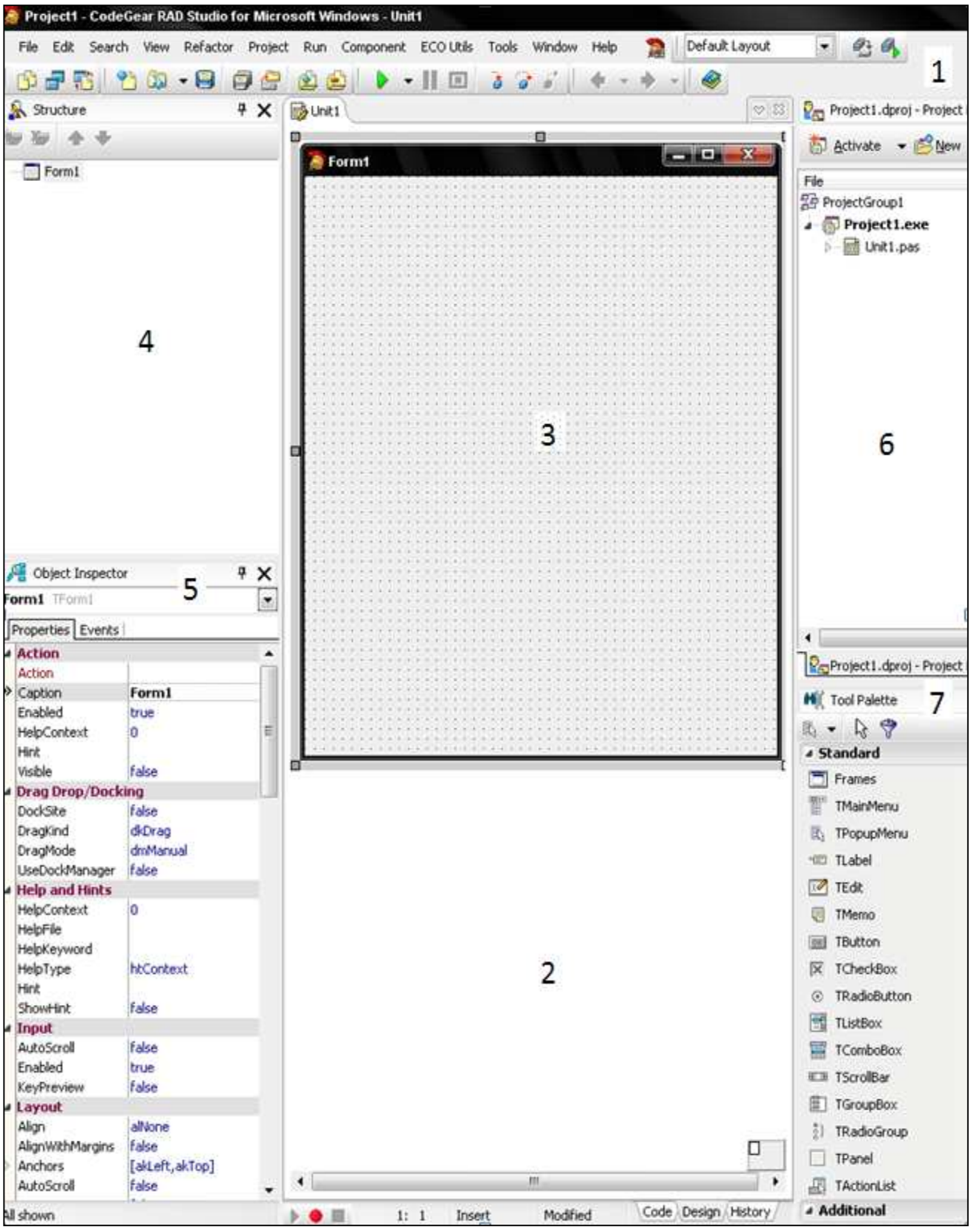


Рис. П1.1. Окна Delphi 2007 в начале работы над проектом

Элементы вкладок окон используются при разработке приложений к проекту. Каждый из этих элементов реализован программно в среде. Поэтому выбирая нужные элементы, программист значительно упрощает свою работу.

Главное окно предоставляет разработчикам приложений к проектам широкое меню набора команд для доступа к функциям сред Delphi. В различных средах эти наборы могут несколько отличаться. При этом в ряде случаев одну и ту же команду можно выполнить с помощью различных инструментов (кнопок).

Перечислим основные функции наиболее часто используемых вкладок главного окна Delphi 2007 (перечень этих функций открывается при щелчке по названию вкладки):

- File (файл) – вызов нового файла (программы), в том числе дополнительно к вызванному проекту; изменение имени проекта и открытого приложения, их запись и стирание;
- Edit (редактирование) – копирование и удаление файлов, вырезание их фрагментов и вставка в них нужных фрагментов;
- Viev (вид) – используется для быстрого нахождения форм и файлов проекта;
- Tools (инструменты) – рабочая среда; с помощью инструментов этого окна можно переключать различные настройки рабочей среды Delphi;
- Window (окно) – подключение окон Object Inspector, Tool Palette и др.

Под окном дизайнера формы находится окно редактора кода – окно разрабатываемой программы приложения. Эти окна можно переключать кнопками Code и Design (проект), расположенными ниже окна дизайнера формы. Рядом с этими кнопками находится кнопка History, нажав которую можно ознакомиться с историей создания данного проекта.

В среде Delphi 2007 центральную часть общего окна занимает окно дизайнера формы – подложки, на которую при проектировании устанавливаются все элементы. В нижней части общего окна находятся кнопки выбора режима центрального окна: Code (коды, программа), Design (дизайн), в котором видны форма и окно Tool Palette – палитра инструмента, то есть перечня вкладок с устанавливаемыми на форму элементами) и History (история создания проекта).

Для установки на форму необходимого элемента нужно найти его на вкладках, щёлкнуть по нему левой клавишей мыши, а затем этой же клавишей щёлкнуть в нужное место формы. При этот на форму будет установлен выбранный элемент, а на вкладке Properties (свойства) окна инспектора объектов

(Object Inspector) откроются параметры свойства) установленного объекта, которые можно откорректировать. После изменения значений параметров щелчком левой клавиши мыши по объекту на форме можно установить новые значения параметров объекта. На вкладке Events этого окна указаны события, на которые может реагировать объект, указанный в шапке окна Object Inspector.

В окне Projects Manager отображается структура проекта (приложения, над которым в данный момент идёт работа).

Из вкладок окна Tool Palette (палитра средств, перечень компонентов) можно выбирать любые компоненты для установки их на форму и запуска в работу.

Если какое-либо из перечисленных окон не отображается, то в меню View (просмотр) главного окна системы Delphi надо выбрать соответствующую команду для открытия этого окна.

П1.3. Подготовка проекта

Для создания ***нового*** проекта с приложением VCL (Visual Component Library – библиотека визуальных компонентов) необходимо открыть папку RAD Studio, запустить программу Delphi 2007 с помощью её ярлыка и в открывшемся окне CodeGear RAD Studio по пути File / New / VCL Forms Application – Delphi for Win32 или по пути New Project / VCL Forms Application – Delphi for Win32 установить начальный модуль проекта Project1. Этот модуль начинается с вызова стандартного перечня утилитов (ресурсов) и объекта типа Forma, который приписывается к классу форм.

Важно отметить, что объектное построение языка Паскаль предусматривает понятия *объекта* и *класса*. Класс – это тип данных, тип объектов, а объект – это конкретный экземпляр класса. Если, например, переменная *S* принадлежит классу вещественных чисел (real), то её запись (пояснение) должно быть вначале программы (для глобальных переменных) или вначале подпрограммы (для локальных переменных). Примеры задания переменных и их массивов: R: real; S: string; J: integer; K: array[1..10] of integer; a: array[1..6,1..4] of real;

В начале программы задаются классы объектов. Можно задавать и свои классы, например:

Type T1 = **class**TMyClass;
T2 = class(T1); - T2 принадлежит классу T1 (или: переменная с именем
T2 является объектом класса T1).

Пример уточнения в программе класса формы 1: Tform1 = class(Tform) – объект форма 1-я принадлежит классу форм (типа форм). Буква Т – это первая буква слова Type {Тáйп}.

Для объектно-ориентированного программирования характерна иерархия классов, то есть использование различных классов.

При создании новой программы ЭВМ формирует её основу (Project1):

```
unit Unit1;                                    // Приложение, модуль 1-й
interface                                      // Начало интерфейса
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Di-
alogs;                                         //  Вызов программных модулей
Type                                           //  Блок объявлений используемых типов
Tform1 = class(Tform)                          //  Форма 1-я принадлежит классу Tform
  . . . . . .
Interface . . .
```

В число вызываемых утилит можно включать и свои приложения (Unit). Ниже приведён пример такой программы с локальными базами данных UnitMedStat и UnitLetters:

```
uses Windows, SysUtils, Classes, Graphics, Controls, Forms, UnitMedStat, UnitLet-
ters, ClipBrd,  ExtCtrls;    // Две последние утилиты используются соответственно
type                         // для работы с буфером данных и для вычерчивания
  Tform1 = class(Tform)                        // прямых линий и прямоугольников
  Image1: Timage;
  procedure FormActivate(Sender: Tobject);  // Процедуры, записанные машиной
  private
   { Private declarations }
  public                                  //   Общие пояснения, выводимые машиной
   { Public declarations }     // (они могут выводиться в любом месте программы)
end;                                           // Завершение этапа указания type
var
 Form1: Tform1;
 I, J, GrNum: integer;                         //  Глобальные переменные
 IP: real;
implementation                                 // Выполнение программы
{$R *.dfm}              // Указание-подсказка компилятору о том, что описание
```

```
                    // формы, сформированной Delphi в процессе её создания,
                    // находится в приложении Unit1.dfm  (dfm – delphi-forma).
...                 // Далее приводятся коды подпрограмм, то есть их содержание
end.
```

Пример начала программы приложения “Статьи”:

```
unit Un_Artikles;
interface
uses
  Windows, SysUtils, Classes, Graphics, Controls, Forms, UnitMedStat, UnitLetters,
  ClipBrd, ExtCtrls;
Type                    // Названия процедур в раздел type вводятся только ЭВМ!
   Tform1 = class(Tform)
   Image1: Timage;
   procedure FormActivate(Sender: Tobject);
   private
     { Private declarations }
   public
     { Public declarations }
end;                                          // Конец раздела Type
var
  Form1: Tform1;
  I, J, GrNum: integer;                       // Глобальные переменные
  IP: real;
Implementation                                // Начало выполнения программы
{$R *.dfm}                  // Пояснявшееся выше указание компилятору,
 procedure Pr1(c: Tcanvas);          // за которым следуют программные модули
   var S: string; X: array[1..5,1..3] of real;
   begin
   . . . . .                  // Тело процедуры
   end;                       //  Конец первой процедуры
```

Создавая программу, прежде всего нужно выбрать соответствующие свойства формы: Name (имя, идентификатор формы), Caption (надпись-заголовок на форме), Height (высота формы), Width – ширина, Top (расстояние

от верхней границы формы до верхней границы экрана), Left (расстояние от левого края экрана до левой границы формы) и др. Все размеры указываются в пикселях.

Перечисленные свойства используются и для ряда устанавливаемых на форму компонентов, то есть они имеют тот же смысл. Но если эти компоненты ставятся на другие компоненты, например, на компонент Panel, то указанные расстояния измеряются не от краёв формы, а от сторон компонента, на который они устанавливаются. Большинство компонентов (Edit, MaskEdit, Memo, Button, Radiobutton, Panel и др.), устанавливаются непосредственно на форму.

Приведённый порядок указания координат компонентов сохраняется и для координат рисунков, надписей и т.д. Их координаты и параметры также можно устанавливать с помощью инспектора объектов.

При разработке больших программ полезно приводить пояснения к соответствующим участкам программы. Дело в том, что со временем назначение этих участков забывается и приходится тратить время на уточнение их назначения. Это особенно необходимо, с программой должны знакомиться другие пользователи.

Указанные пояснения приводятся в тексте программы на отдельных строках или справа от соответствующих операторов после знака //. При настройке программы отдельные её части можно отключать, помещая их в фигурные скобки.

П1.4. Структура проекта

Открыть проект, разработка которого уже начата, можно как из окна CodeGear RAD Studio, открывающегося при запуске программы Delphi 2007, так и из папки с этим проектом, расположенной в папке RAD Studio. На рис. П1.2 приводится содержание папки Pr_Articles, содержащей файлы проекта “Статьи”, в том числе двух с личными базами данных. Для открытия проекта удобнее всего щёлкнуть по значку файла с расширением droj, что не только открывает файл Un_Articles, но и позволяет запустить проект. Данный проект предназначен для подготовки рисунков и графиков для научных статей).

Подготовленный проект состоит из нескольких модулей (файлов). Обычно все файлы одного проекта располагают в одном каталоге, то есть в одной папке. В противном случае сложнее будет находить файлы, входящие в каждый из проектов.

Собственно проект не является объединением файлов в соответствии со стрелками. Он вызывает эти файлы и при запуске отправляет их на сборку и компиляцию. Отображение кода файла проекта в окне Редактора кода можно задать командой Project / View Source (Проект / Просмотр источника). Но проще эта команда выполняется с помощью кнопки Code, расположенной в нижней части формы.

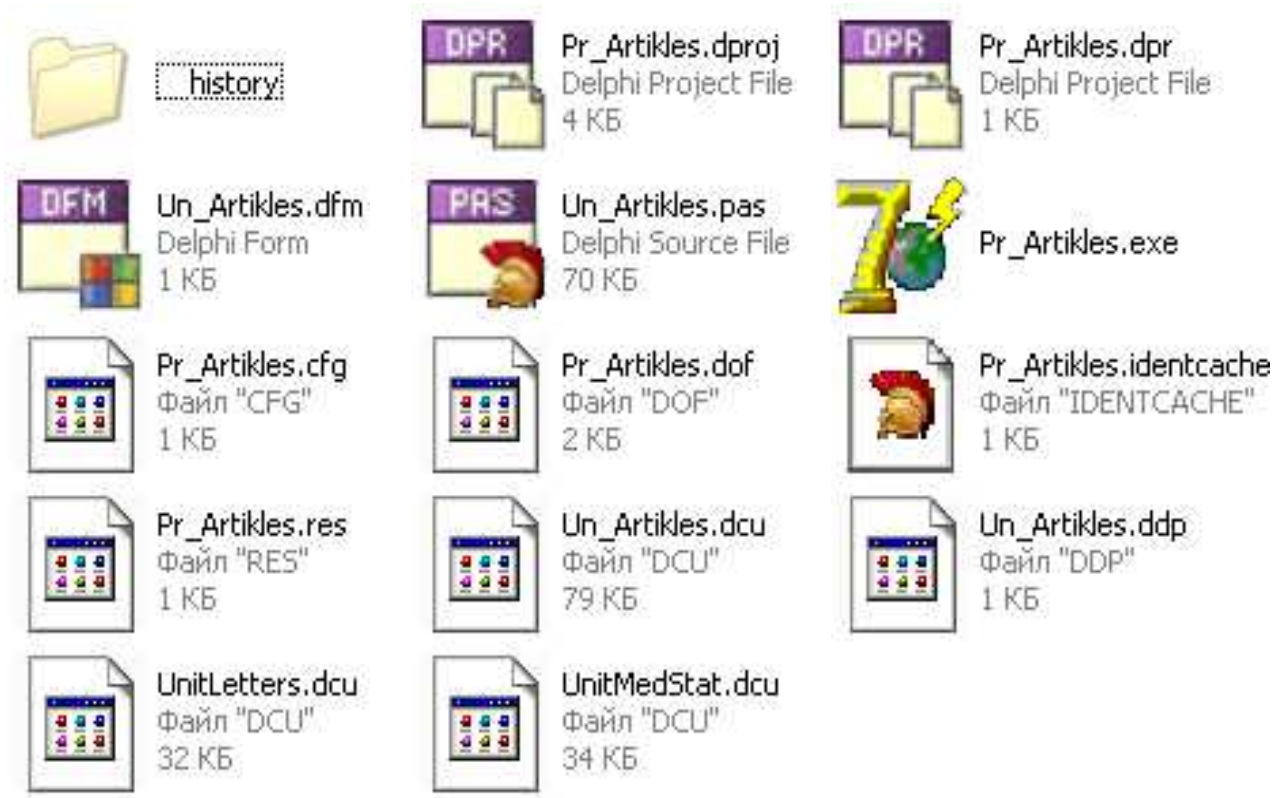


Рис. П1.2. Пример комплекта файлов проекта

Приведённые на рис. П1.2 модули имеют следующие расширения:

- проект – dpr (delphi-project);
- описания форм – dfm (delphi-form);
- модули форм и модули – pas (pascal);
- откомпилированные модули и их параметры – dcu (delphi-compil-unit);
- параметры проекта – dof (delphi-object-file);
- параметры среды – cfg (configuration);
- описание ресурсов – res (resources);
- проект-приложение – exe (окончательный exe-файл в машинных кодах, готовый к исполнению в операционной системе MS Windows).

В большинстве случаев для размещения устанавливаемых на форму элементов бывает достаточно одной формы. Однако иногда специфика разрабатываемого проекта (специфика решаемой задачи) требует двух и даже большего количества форм. На следующем рисунке приведена структура проекта с двумя формами. Аналогичную структуру имеют проекты и с несколькими формами. Стрелками на рисунке показана последовательность обработки информации.

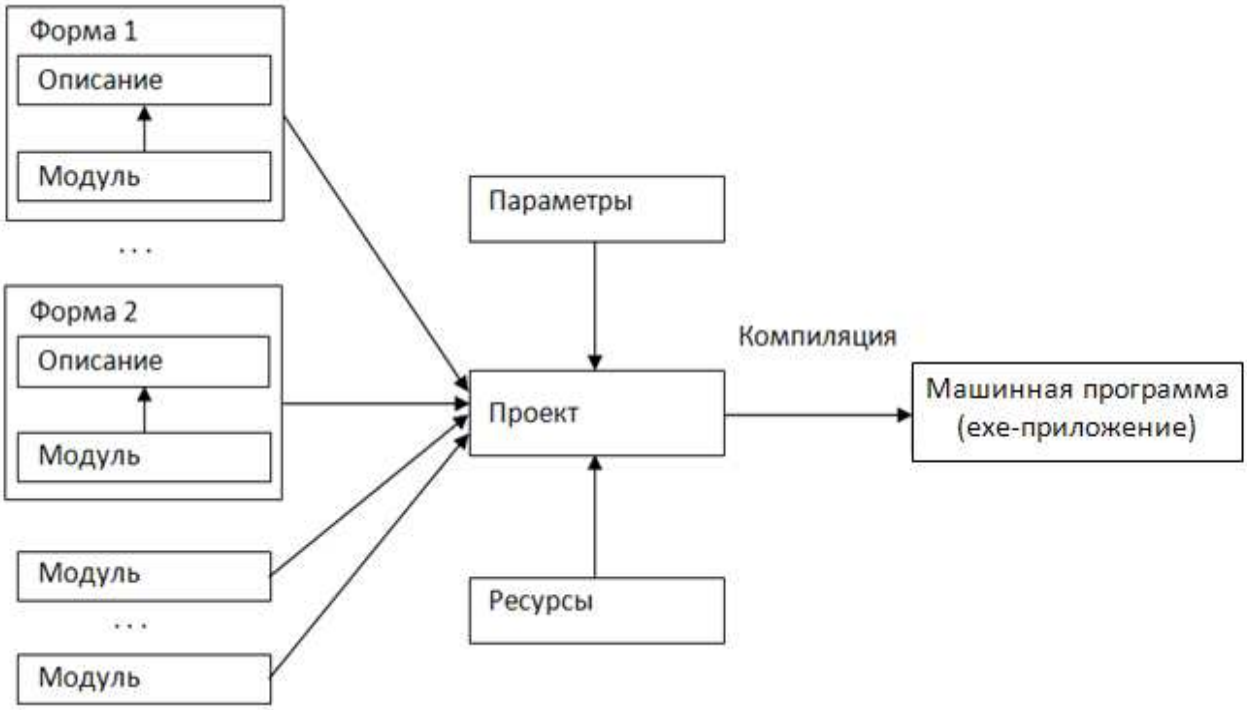


Рис. П1.3. Пример комплекта файлов проекта с двумя формами

Наибольшим из модулей проекта, как правило, является окончательное приложение, имеющее расширение exe. Оно получается путем быстрой компиляции (объединения) откомпилированных модулей с расширениями dcu. Среди них могут быть, например, локальные базы данных на основе приложений Unit. Кроме модулей, указанных в приведенном выше перечне, могут формироваться ещё копии отдельных модулей. В случае зависания и порчи программы проекта по этим копиям можно восстановить проект на момент последней его записи.

Таким образом, проект представляет собой совокупность файлов, которые используются компилятором. *Основу проекта* образуют файл главного модуля, который обычно обозначается Project.dpr (description – описание) и один или несколько других модулей (по числу используемых форм, pas).

Помимо этого в процессе компиляции используется файл ресурсов (res), файлы описания форм (отдельный файл для каждой формы с расширением dfm), а также файл конфигурации проекта (cfg).

Рекомендация. При разработке двух и более проектов во избежание путаницы с принадлежностью файлов к указанным проектам файлы каждого проекта следует размещать в отдельных папках. Эти папки должны создаваться и размещаться внутри папки Projects, располагаемой в папке RAD Studio. В противном случае файлы проекта записываться в выбранную папку не будут.

При открытии проекта появляются окно редактора формы (форма), наложенное на окно редактора кода (окно приложения Unit), а также вкладки управления (сверху), палитры инструментов (справа) и Object Inspector (инспектор объектов, слева), над которым есть ещё (маленькое) окно структуры приложения. При этом в некоторых версиях Delphi файл главного модуля для открытия не предлагается. Чтобы его увидеть надо в меню Project выбрать команду **View Source.**

Пример главного модуля программы лабораторной работы М3: (запускается с помощью кнопки **run** – зелёный треугольник):

```
Program LaborM3;            //  Название запускаемого проекта
Uses
  Forms
  UnitM3 in ’UnitM3.pas’ {Form1};   //  Указание запускаемого приложения
{SR *.RES}                  //  Директива компилятору подключить файл ресурсов
begin
  Application.Initialize;     //  Запуск приложения.
```

Приложение (Unit) открывается в главном окне редактора. Под этим окном имеются 3 указателя: Code (программа), Design (план, планирование) и History (история). Ими можно переключать файл истории создания данного приложения, план расположения объектов на форме и собственно модуль программы с расширением *pas*.

П1.5. Понятие холста (класса Tcanvas)

Canvas в переводе с английского языка Canvas означает холст. Данный класс используется почти во всех программах, так как непосредственно на

форму выводить тексты, графику и рисунки нельзя. Эта информация выводится на специальные объекты с использованием холста, то есть в общем случае *Canvas – это поверхность, на которую можно вывести текст и графику*. Пример: c.TextOut(120,74,'Обозначения'); – вывод слова с указанием местоположения в задаваемой области холста (левой верхней точки этой области).

В списке подпрограмм и в заголовках подпрограмм обычно указывается: c: Tcanvas. Например: procedure Fig 3_2(k: integer; c: Tcanvas);

В последнем примере ввод параметра *k* в заголовок процедуры используется в случае многократного запуска данной процедуры из другой процедуры с различными значениями некоторого параметра, в данном примере – k.

В связи с тем, что в среде Delphi буква "c" используется для сокращённого обозначения слова "Canvas", в качестве математической переменной она не применяется. Ввод буквы "c" в качестве идентификатора будет ошибкой, фиксируемой средой.

Отметим, что в среде Delphi некоторые классы рассматриваются как подклассы или как свойства некоторых классов. Например, класс Font (Шрифт) может служить свойством более общих классов, например, классов Graphics, Form, Edit, Label и др. В некоторых литературных источниках Canvas рассматривается как свойство, а не как класс.

Для задания операторов и функций в среде Delphi используется объектно-ориентированный язык Паскаль.

По умолчанию область холста совпадает с областью формы или элемента, на который производится вывод. Класс Tcanvas имеет набор стандартных свойств и методов, позволяющих выполнять различные графические операции. Выведенная на холст графическая информация отображается до тех пор, пока данная область холста не будет закрыта другим холстом или свёрнута. Примеры компонентов для холста: Image (изображение), PaintBox (набор цветов). Эти компоненты для установки на холст берутся из вкладок окна Tool Palette.

Совместно с классом Canvas используется несколько других классов, или точнее подклассов класса Canvas. Например: c.Font.Size:=12 (задание размера шрифта в подклассе Шрифт класса Canvas), c.Pen.Color:= (задание цвета линий в подклассе Карандаш – Pencil класса Canvas).

П1.6. Операторы и функции класса Canvas

Ниже поясняются некоторые свойства основных классов (подклассов), необходимые для выполнения исследований и лабораторных работ (кроме

указанных выше свойств), методы их задания, а также наиболее часто используемые функции. В распоряжение программистов предоставляется широкий выбор указанных свойств и операторов.

Отметим, что по умолчанию в системе Delphi принимаются следующие параметры: размер шрифта – 12-й, цвет шрифта и линий – чёрный, стиль шрифта – прямой, стиль линий – прямой.

Основные свойства и методы (операторы) класса Canvas	
Свойство	Подкласс Tfont (Шрифт)
Название шрифта	Примеры: c.Font.Name:=’Times New Roman’; (по умолчанию) c.Font.Name:=’Cambria’; c.FontName:=’Cambria Math’;
Color (здесь и в других классах)	*Цвета, имеющие названия в ЭВМ*: Black, Silver, Maroon (каштановый), Green, Red, Lime (салатный), Blue, Olive, Nary (тёмно-синий), Fuchsia (ярко-розовый), Purple (розовый), Aqua (бирюзовый), Teal (зелёно-голубой), White, Gray. Примеры задания: c.Font.Color:=clBlue; (cl – буквы слова color); c.Brush.Color:=clRed; c.Pen.Color:=clGreen; *Цвета, формируемые пользователем*. Общий вид записи: $00D5E3F5 (в 16-ричной системе задаётся смесь трёх основных цветов – D5 (синий), E3 (зелёный) и F5 (красный) – вместо этих чисел нужно подобрать свои числа по получаемому цвету). Первая цифра < 8, вторая = 0 (иначе будет чёрный цвет). Примеры записи цветов: c.Font.Color:=$00F8E7D8; c.Pen.Color:=$00A4D6C1; c.Brush.Color:=$00B2F5F3.
Size	Размер. Пример задания: c.Font.Size:=14;
Style	1) *Для шрифта*: [ ] – прямой, fsItalic – наклонный (курсив), fsBold – полужирный, fsUnderline – подчёркнутый. Пример задания двухстильного шрифта c.Font.Style:=[fsItalic] + [fsBold]; (fs означает Font.Style). 2) *Для линий:* psSolid – сплошная, psDash – пунктирная, psDot – точечная, psDashDot – пунктирно-точечная, psDashDotDot – линия с чередованием одного штриха и двух точек; (ps означает PenStyle, pencil – карандаш). Пример задания цвета красителя: c.Pen.Style:=psDash; пример задания цвета линии: c.Pen.Color:= clRed;

Свойство	Подкласс Tbrush (Кисть) – для заливки фигур
Brush.Style Brush.Color	psSolid – сплошная заливка, psHorisontal – горизонтальная штриховка, psVertical – вертикальная штриховка, psFDiagonal – диагональная штриховка с наклоном линий вперёд, psBDiagonal – диагональная штриховка с наклоном линий назад, psCross – горизонтально-вертикальная штриховка в клетку, psDiagCross – диагональная заливка в клетку, (ps означает стиль заливки). Примеры заданий стиля и цвета заливки: c.Brush.Style:=psHorisontal; c.Brush.Color:=clAqua;
Свойства	Подкласс Tpen (Карандаш) – для линий
Width	Толщина линии. Пример задания: c.Pen.Width:=1 (в пикселях);
Height	Свойства фигур. Примеры задания: Button1.Width:=15; Edit2.Height:=12; Image3.Height:=50 (в пикселях);
Оператор	Результат применения
c.TextOut(*k*, *l*, *s*)	Вывод текста *s* от точки с координатами (*k*, *l*).
c.MoveTo(*k*, *l*); c.LineTo(*m*, *n*);	Вычерчивание линии от точки (*k*, *l*) к точке (*m*, *n*). После оператора Line можно задать c.LineTo(*p*, *r*), не повторяя оператора c.MoveTo(*m*, *n*).
c.RectAngle(*x*1, *y*1, *x*2, *y*2);	Вычерчивание прямоугольника с задаваемыми координатами левого верхнего (*x*1, *y*1) и нижнего правого (*x*2, *y*2) его углов.
c.RoundRec(x1, y1, x2, y2, x3, y3);	Вычерчивание прямоугольника со скруглёнными углами. Координаты *x*1, *y*1, *x*2 и *y*2 задают верхний левый и нижний правый углы прямоугольника без округлений, а *x*3 и *y*3 определяют радиусы скруглений углов по координатам *x* и *y*.
C.Ellipse(x1, y1, x2, y2);	Вычерчивание эллипса или круга с указанными координатами верхнего левого и нижнего правого углов прямоугольника, в который он вписывается.
c.Arc(*x*1, *y*1, x2, *y*2, *x*3, *y*3, *x*4, *y*4);	Вычерчивается дуга эллипса, имеющего параметры *x*1, *y*1, *x*2, *y*2. Из эллипса вырезается дуга от точки (*x*3, *y*3) к точке (*x*4, *y*4) против часовой стрелки.

Некоторые функции (операторы) среды Delphi	
1) **Математические функции**	
Функция	Получаемый результат
abs(*x*)	Абсолютное значение *x*.
sqrt(*x*)	Значение квадратного корня из величины *x* > 0.

Exp(*x*)	Значение экспоненты от *x*, то есть e^x.
Ln(*x*)	Значение натурального логарифма от $x > 0$.
$Y = \exp(a \cdot \ln(x))$	Значение степенной функции $y = x^a$.
$Y = \ln(x)/\ln(a)$	Значение логарифма $y = \log_a x$ с основанием *a*.
Random	Значение случайной величины, принадлежащей (0, 1).
Random(*n*)	Значение целого случайного числа, принадлежащего [0, n-1].
X:=0.01*round(100**x*)	Округление числа *x* до двух разрядов после запятой.
2) **Функции преобразования**	
Функция	Получаемый результат
StrToInt(*s*)	Целое число, изображением которого является строка *s*.
StrToFloat(s)	Вещественное число, изображением которого является *s*.
IntToStr(*k*)	Строка, являющаяся изображением целого числа *k*.
FloatToStr(*x*)	Строка, являющаяся изображением вещественного числа *x*.
Round(*x*)	Значение целой величины, ближайшей к вещественной *x*.
Trunk(*x*)	Значение целой части вещественной величины *x*.
3) **Операторы чтения и записи в элементы ввода-вывода**	
Оператор	Выполняемое действие
S:=Edit2.Text	Чтение числа S в программу из элемента Edit1.
MaskEdit.2.text:=S	Запись числа S из программы в элемент MaskEdit2.

Упражнения

1. Процедура вычисления значения $x = a/b$ = 1,12^4990/1,14^4980 составлена следующим образом:

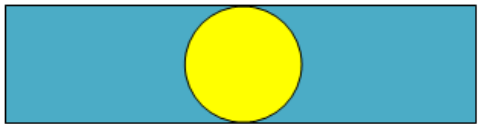
```
a := 1.12;
for I:= 1 to 4989 do a:=a*1.12;
b := 1.14;
for I:= 1 to 4979 do b := b*1.14;
x := a/b;
```

При запуске этой программы оказалось, что происходит выход величин *a* и *b* за пределы разрядной сетки, так как вещественные числа можно задавать в интервалах [5e-324 … 1.7e+308] для чисел типа real и [5e-4951 … 1.1e+4 932] для чисел типа extended (расширенный вещественный тип).

Предложить программу решения указанной задачи, позволяющую получать правильное значение величины *x.*

2. Написать для программы формулу для вычисления $y = |\sin x|^{e^{-x}}$.

3. Подготовить программу рисования фигуры, изображённой на рисунке справа.



4. Найти ошибку в программе (в процедуре Average arithmetic – среднее арифметическое), определяющей значение *C* – среднего арифметического двух записываемых в программу чисел *A* и *B* и выводящей результат в используемое окно.

```
. . .
Procedure Average arithmetic(c: TCanvas);
 var  a, b, c: real;
 begin
    a:=2.34;  b:=-0,65;  c:=1,73;
    c:=(a+b+c)/3;
    c.TextOut(20, 10, 'C = '+FloatToStrI);
 end;
```

П1.7. Управляющие процедуры

П1.7.1. Процедура активизации формы

В отдельных случаях, например, когда создаваемая программа не предусматривает вывод информации на холстовое покрытие формы, рассматриваемую процедуру можно не вводить. С помощью процедуры активизации формы можно:

- выполнять рабочие процедуры в нужной последовательности;
- многократно повторять любые рабочие процедуры с различными значениями параметров;
- управлять выводом графической информации в разные окна и в буфер данных (для перемещения её из буфера в файлы текстового процессора Word, в программы подготовки презентаций Power Point и т.д.);
- просто переключать элементы вывода информации и др.

Для того, чтобы с расположенными на форме инструментами можно было выполнять указанные действия, форму нужно активизировать. Эта активизация

достигается генерированием и запуском процедуры активизации формы ***procedure Tform1.FormActivate(Sender: Tobject*** – номер формы здесь может быть и иным. Для генерирования этой процедуры нужно в режиме Design [дизайн] щёлкнуть по форме для открытия её параметров в окне Object Inspector. Затем следует выбрать мышкой событие OnActivate в левой вкладке окна и дважды щёлкнуть по открывшемуся справа полю в правой вкладке окна. При этом в создаваемой программе будет создана заготовка для рассматриваемой процедуры.

В последних версиях системы Delphi 2007 в указанной последовательности создания заготовки процедуры активизации формы вместо упомянутого двойного щелчка в открывшееся поле можно вписать слово FormActivate и нажать клавишу Enter.

Ниже приводятся создаваемая ЭВМ заготовка программы FormActivate и вариант её заполнения для программы с рабочими процедурами Pr1, Pr2 и Stat Russ. Последняя процедура вызывается из внутренней базы данных.

Заготовка:
```
procedure Tform1.FormActivate(Sender: TObject);
begin

end;
```

Кроме создания заготовки программы активизации формы ЭВМ вписывает в блок объявлений используемых типов (в начале программы Unit) название созданной ею программы, то есть procedure FormActivate(Sender: TObject).

Ниже приведён пример заполнения рассматриваемой процедуры без задания в ней локальных переменных. Внутренние процедуры вызываются для исполнения в порядке их записи. Одна из них предусматривает вывод информации в окно Image1.

```
Procedure Tform1.FormActivate(Sender: TObject);
  var  . . .
  begin
    StatRuss;              //  Вызов файла базы данных
    Pr1;                   //  Процедуры текущей программы
    . . .
    Pr2 (Image1.Canvas);
  end;
```

Отметим, что все глобальные переменные, введённые в вызываемых извне программах (в рассмотренном примере – из процедуры StatRuss), имеют силу и в создаваемой программе, то есть заново вводить их не нужно.

Координаты компонентов, устанавливаемых непосредственно на форму, определятся от верхнего левого угла формы. Но если эти компоненты ставятся на другие компоненты, например, на окно (Image), то координаты их местоположения измеряются от левого верхнего угла компонента, на который их установили, а не от левого верхнего угла формы. К таким компонентам могут относиться, например, Edit, **MaskEdit, Memo**, Button, Radiobutton, Panel и др.

П1.7.2. Процедура значений параметров элементов, установленных на форме

Рассматриваемая процедура называется FormCreate {форм кр*éе*т}– создатель формы, на которой устанавливаются элементы. У некоторых элементов не все параметры можно задать с помощью окна ObjectInspector. Например, в текстовом поле (в шаблоне ввода) MaskEdit маска вводимого кода, то есть разрядность и допустимый смысл каждого разряда кода, в свойствах окна ObjectInspector отсутствуют. Такие параметры обычно задаются в процедуре FormCreate. Если некоторые элементы берутся из баз данных, то значения их параметров также указываются в этой процедуре.

Процедура FormCreate имеет и ещё одно важное свойство. Оно заключается в стабилизации установленных значений параметров элементов. Дело в том, что перемещение любого элемента по форме автоматически отражается и в значениях свойств этого элемента в окне ObjectInspector. Поэтому при случайном сдвиге установленных на форме элементов нужна новая настройка их свойств с помощью окна ObjectInspector. Если же значения свойств элементов записаны в процедуре FormCreate, то при новом запуске проекта их значения, написанные в процедуре FormCreate, сохранятся.

Рассмотрим в качестве примера начало процедуры FormCreate из программы одной из лабораторных работ по математическому моделированию:

```
procedure Tform1.FormCreate(Sender: TObject);
 begin
   But11:=0; Bat11:=0;
   MaskEdit1.EditMask:=’9999’;  MaskEdit1.Text:=’....’;
   MaskEdit4.EditMask:=’9’;  MaskEdit4.Text:=’ ‘;
   MaskEdit5.EditMask:=’9’;  MaskEdit5.Text:=’ ‘;
```

```
  . . .
MaskEdit24.EditMask:='9'; MaskEdit24.Text:=' ';
MaskEdit25.EditMask:='cccccccc'; MaskEdit25.Text:=' ';
  . . .
Lambda(96,290,'Black',Image2.Canvas);
Lambda(169,333,' ',Image2.Canvas);
Tau12(350,479,'Red',Image2.Canvas);
Tau8(394,472,'Red',Image2.Canvas);
SigmaM(253,434,' ',Image3.Canvas);
SigmaB(356,478,' ',Image3.Canvas);
  . . .
Tau8(278,195,' ',Image6.Canvas);
Tau12(125,101,' ',Image7.Canvas);
Tau12(624,411,' ',Image7.Canvas);
  . . .
for I:=0 to 6 do Lambda(311+34*I,155,'Red',Image11.Canvas);
Lambda(207,318,'Red',Image11.Canvas);
  . . .
end;
```

В этой процедуре, в частности, предусмотрен вывод греческих букв из базы данных UnitLetters, имя которой включается в перечень утилитов, вызываемых разрабатываемой программой (оператором uses). Такая база является временной необходимостью: греческие буквы в алфавит системы Delphi пока не включены. Слова '9999', '9' и 'cccccccc' являются шаблонами ввода. Количество цифр '9' и букв 'c' в этих словах соответственно указывает, какое максимальное число десятичных цифр или произвольных символов можно ввести в элемент ввода MaskEdit. При этом часть разрядов этого элемента может оставаться пустой.

Если в разрабатываемом проекте состав число рабочих процедур небольшое, то функции процедур FormActivate и FormCreate обычно совмещают в одной процедуре – в FormActivate. В принципе можно вообще обойтись без рабочих процедур, разместив все их операторы в процедуре FormActivate. Однако такая программа будет сложной для понимания и для внесения в неё изменений. Введение же в программу приложения к проекту рабочих процедур позволяет удобно разрабатывать программу по частям.

П1.7.3. Процедуры обработки событий

Существует большое количество событий, на которые могут соответственно реагировать форма и установленные на ней элементы, то есть в ответ на каждое событие выполнять соответствующую программу-процедуру. Такими событиями могут быть, например, одинарный или двойной щелчок по форме, по окну Image, по окну ввода-вывода текста и по другим элементам, нажатие любой клавиши клавиатуры или отпускание этой клавиши или клавиши мыши и т.д. На практике в подавляющем большинстве случаев в качестве указанного события используется щелчок по кнопке Button.

При разработке программы проекта с процедурами обработки событий ЭВМ по команде программиста on Click (двойной щелчок на вкладке Events в окне объектного инспектора) создаёт стандартную заготовку для написания такой процедуры. Рассмотрим подготовку процедуры обработки события Button-Click.

Вначале необходимо в режиме Design щёлкнуть по выбранной кнопке Button. При этом в окне Object Inspector появятся значения свойств этой кнопки. Далее нужно перейти на вкладку Events (события), из списка возможных событий выбрать On click (на щелчок) и дважды щёлкнуть по открывшемуся на вкладке справа окну. При этом средой Delphi будет создана заготовка процедуры Tform1.Button1Click(Sender: TObject), в которой номер кнопки Button может быть другим. Заготовка этой процедуры имеет вид:

```
procedure Tform1.Button1Click(Sender: TObject);
begin

end;
```

Упражнение: составить программу смены цветов светофора, управляемого из будки постового ГАИ на перекрёстке двух улиц. При каждом нажатии кнопки Button программа должна обеспечивать смену цветов светофора на перекрёстке 1-й и 2-й улиц согласно последовательностям:

1) … жёлтый – зелёный – жёлтый – красный – жёлтый – зелёный – жёлтый …
2) … жёлтый – красный – жёлтый – зелёный – жёлтый – красный – жёлтый …

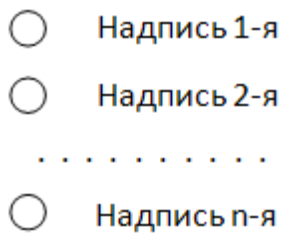
В данной программе светофор можно представить двумя кругами (эллипсами с одинаковыми диагоналями), один из которых условно соответствует двум противоположным сторонам светофора, а другой – двум другим. Предполагается, что при включении светофора постовым ГАИ по обоим направлениям

на светофоре зажигается жёлтый свет, что может быть, например, реализовано присваиванием глобальной целочисленной переменной *N* (или двум таким переменным N1 и N2) значения 0. При уходе постового из будки по обоим направлениям должен зажигаться жёлтый свет.

П1.8. Переключение задач и их параметров при выполнении программ

Рассматриваемая задача обычно выполняется с помощью переключателей RadioButton, получивших в своём названии приставку радио- в связи с их аналогией с радиотехническими (в том числе с телевизионными) многопозиционными переключателями. Каждый из указанных переключателей имеет 2 положения: включён (checked) и выключен. Несколько таких переключателей, расположенных на одном носителе, например, на форме, на окне Image, объекте Panel и на др., образуют группу переключателей, в которой одновременно включён один и только один переключатель.

Переключатели RadioButton, расположенные на разных носителях, образуют независимые указанные группы. В качестве такого объекта, предназначался объект RadioGroup (группа переключателей). Но он ещё недостаточно отработан. Однако указанные группы переключателей можно создавать и самим. Наиболее просто это можно сделать с помощью группового компонента (объекта) Group-Box {Group – группа, Box – коробка}. Переключатели RadioButton удобно располагать внутри этого компонента. Они устанавливаются столбиком. При этом справа от каждого из переключателей с помощью его свойства Caption (надпись) в окне Object Inspector можно написать переключаемое значение соответствующей величины, вид переключаемой функции и т.д. При щелчке по кнопке переключателя в ней появляется точка.



Обычно рассматриваемые переключатели стоятся на основе кнопок Button. Например:

```
If RadioButton1.Checked then B:=0.2;
If RadioButton2.Checked then begin  B:=0.5;   и т. Д.
```

П1.9. Вывод информации в буфер

Подготовленные графики, таблицы и т.д. могут программно выводиться из системы Delphi в буфер обмена ПК (ClipBoard) для дальнейшего использования.

Как известно, изображения, включая графики, могут быть представлены в растровом или в векторном формате. В настоящее время мониторы почти всех ПК работают с растровым форматом. В растровом формате каждому пикселю отводится 3 байта ($2^{24} = 16^6$), задаваемые в программах 6-ю шестнадцатеричными цифрами. Общее количество цветов – около 16 млн.

В программах можно воспользоваться и “стандартными” цветами. В системе Delphi 2007 их 52: clBlue, clGreen, clRed и др. (приставка cl означает Color). В настоящее время цвета кодируются уже восемью шестнадцатеричными цифрами. При этом перед кодом цвета записывается символ $, который указывает, что за ним записаны биты графической информации, то есть код выбранного в программе цвета:

$ 00 D9 8F C2 (00-синий-зелёный-красный).

Код 00 означает отсутствие в данной двухразрядной позиции соответственно синего, зелёного или красного цвета, а код FF – его максимальную интенсивность. Так, код 000000 – чёрный свет, а увеличение интенсивности какого-либо цвета связано с ослаблением чёрного цвета.

В системе Delphi 2007 начали использоваться и 2 первых разряда кода цвета: если в этих разрядах записать код отличный от 00, то получим черный цвет. Другие цвета при одинаковой их интенсивности дают серый цвет различной интенсивности. Если их общая интенсивность нулевая, то итоговый цвет получается черным. При максимальной интенсивности синего, зеленого и красного цветов получается белый цвет.

Примеры: $00FFFFFF – белый цвет; $00CCCCCC – светло-серый цвет; $00999999 – серый; $00666666 – темно-серый; $00333333 – черный неинтенсивный; $00000000 – черный интенсивный цвет.

Пример задания цвета в программах: c.Brush.Color:=$00A89E6F или c.Brush.Color:=clAqua (заливка выбранным цветом фигуры, в которой находится изображающая точка).

Кодирование цвета каждого пикселя изображения приводит к необходимости иметь для хранения изображений весьма большую память. Так при размере поля изображения $1024 \times 1024 = 2^{20} \cong 10^6$ пикселов необходимо не менее 24 мегабайт памяти. То есть, если в тексте созданного документа содержится 40 рисунков указанного размера, то для них потребуется 1 гигабайт памяти.

В связи с указанным прорабатываются вопросы сжатия графической информации на этапе записи её в буфер ЭВМ, занимающий часть ОЗУ. При этом используется стандарт сжатия GIF (Graphic Interchange Format – формат графического обмена), сжимающий 3 шестнадцатеричные цифры в одну.

Для обращения к буферу в перечень системных утилитов, вызываемых ПО при запуске проекта, необходимо включить драйвер буфера обмена ClipBrd (Clip – клип, кусок предварительной съёмки кинофильма, вариант фотоизображения и т.д., Board – доска для записи, для ЭВМ – устройство для промежуточного запоминания во внешнем участке ОЗУ – буферная память или буфер обмена).

Обращаться к буферу непосредственно из рабочих процедур нельзя. Такое обращение возможно из процедуры активизации формы Form1Activate или из процедур отработки щелчка на кнопке типа ButtonClick. Рассмотрим пример команды обращения к буферу из процедуры Form1Activate:

```
unit Un_Artikles;
uses
  Windows, SysUtils, Classes, Graphics, Controls, Forms,
  UnitMedStat, UnitLetters, ClipBrd, ExtCtrls;
. . . . . .
procedure Tform1.FormActivate(Sender: Tobject);
  var ProcName: String;
     W: word;  H: Thandle;  P: Hpalette;  K: integer;
  begin
    ProcName:='Dokl_Sekz_4';
. . . . . .
    if ProcName='Novgorod_2000_2007' then begin
      Image1.Height:=288; Image1.Width:=200;
      StatNovObl; // StatRuss;                    // Вызов статистических данных по
      Novgorod_2000_2007(Image1.Canvas);    // Новгородской обл. или по России
    end;
    Image1.Picture.Graphic.SaveToClipBoardFormat(W, H, P);
    Clipboard.SetAsHandle(W,H);
  end;
end.
```

В приведённой программе курсивом выделены операторы, выполняющие запись информации в буфер компьютера.

П1.10. Запись битовых образов в среду Delphi

Под битовыми образами понимают небольшие рисунки, хранящиеся в памяти компьютера в виде файлов с расширениями bpm (bitmap – битовый рисунок, битовая карта). В большинстве случаев необходимые рисунки или фотографии записываются в указанных файлах с расширениями, не позволяющими непосредственно использовать их в Delphi-проектах, например, в мультипликации. В связи с этим для демонстрации битовых рисунков в Delphi-программах и для создания ресурса таких образов файлы с выбранными изображениями нужно преобразовать в bitmap-файлы, которые могут сохраняться и демонстрироваться в среде Delphi. Рассмотрим, как это выполняется в приведённой ниже программе.

```
Unit Unit1;
 Interface;
 Uses SysUtils, Classes, Graphics, Controls, Forms, Dialogs, ExtCtrls;
      {утилита ExtCtrls нужна для управления передачей битовых образов в ЭВМ}
 type
    Tform = class(Tform)
     Image1 = Timage;
     procedure FormActivate(Sender: Tobject);
     private     { Private declarations }
     public      { Public declarations }
 end;
  var
     Form1:  Tform1;
     bm:  Tbitmap;              // Подготовка объекта типа “битовый образ”
 implementation                 // Реализация (подготовка записи “битового образа”)
  {$R *.dfm}
procedure Tform1.FormActivate(Sender: Tobject);
  begin
     Form1.Image1.Picture;      //  Подготовка окна Image1 под битовый образ
     bm:=Tbitmap.Create;        //  Подготовка объекта bm для приёма образа
     bm.LoadForm1File('NovSU.bm');  // Считывание в bm цифрового изображе
                                    //  ния НовГУ (битового образа)
     Form1.Image1.Canvas.Draw(10,20,bm);
     Form1.Image1.Canvas.Draw(10,20,bm);  // Запись изображения в окно Image1
  end;                      // с координатой левого верхнего угла изображения (10, 20)
```

Теперь битовый образ bm можно просто переписать и в модуль ресурсов, который подключается командой {$R Images res}, и в другие модули среды.

П1.11 Реализация движения

Для реализации движения некоторого объекта он систематически стирается и вновь рисуется несколько смещённым в нужном направлении. При этом величина и время смещения должны быть такими, чтобы дискретность смещения не была бы заметна человеческому глазу.

В системе Delphi минимальный интервал времени между моментами смещения может быть выбран равным одной миллисекунде, а минимальное перемещение по координатным осям – одному пикселю.

В принципе небольшую задержку стирания объекта в последнем его месте и рисования его в новом месте не сложно выполнить путём периодического выполнения некоторых вычислений. Однако при этом сложно настраивать программы на нужное время задержки. Поэтому в системе Delphi для реализации задержек удобнее воспользоваться системным таймером, то есть генератором временных импульсов. Элемент Timer расположен на вкладке System окна палитры инструментов. Он выставляется на форму и может оставаться невидимым, то есть находиться за пределами окна Image, в котором проводится моделирование. При моделировании обычно используют генерируемую ЭВМ процедуру типа Form1.Timer1Timer(Sender: Tobject) и кнопки Старт и Стоп на основе элементов Button. Указанная процедура реализует периодический процесс, не требуя дополнительной организации цикла. Каждый шаг этого процесса выполняется с появлением очередного временного импульса. Период следования временных импульсов в микросекундах устанавливается на вкладке окна Object Inspector при отражении в нём параметров элемента Timer1.

Кнопки Пуск и Стоп задают соответственно состояния генератора временных интервалов Timer1.Enabled:=True (таймер работает) и Timer1.Enabled:=False (генератор выключен). Во втором случае движение моделируемого объекта не выполняется (объект останавливается). Движение объекта может быть продолжено при нажатии кнопки Пуск.

Ниже приводится программа Unit1 прямолинейного движения квадратного объекта.

```
Unit Unit1;
interface
uses
```

```
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, ExtCtrls, StdCtrls;
type
 Tform1 = class(Tform)
  Image1: Timage;
  Timer1: Ttimer;
  Button1: Tbutton;
  Button2: Tbutton;
  procedure Timer1Timer(Sender: Tobject);
  procedure FormCreate(Sender: Tobject);
  procedure FormActivate(Sender: Tobject);
  procedure Button1Click(Sender: Tobject);
  procedure Button2Click(Sender: Tobject);

 private
  { Private declarations }
 public
  { Public declarations }
end;

var
 Form1: Tform1;
 T: integer;

implementation

{$R *.dfm}

procedure Tform1.Button1Click(Sender: Tobject);
  begin
    Timer1.Enabled:=True;
  end;

procedure Tform1.Button2Click(Sender: Tobject);
  begin
    Timer1.Enabled:=False;
  end;

procedure Tform1.FormActivate(Sender: Tobject);
  begin
    T:=0;
```

```
      Image1.Canvas.Pen.Color:=clBlue;
      Image1.Canvas.Rectangle(0,20,20,40);
      Timer1.Enabled:=False;
   end;

procedure Tform1.FormCreate(Sender: Tobject);
   begin
      Timer1.Interval:=20;          // миллисекунды
   end;

procedure Tform1.Timer1Timer(Sender: Tobject);
   var DateTime: TdateTime;
   begin
      Timer1.Enabled:=False;
      Image1.Canvas.Pen.Color:=clWhite;
      Image1.Canvas.Rectangle(T,20,T+20,40);
      T:=T+1;
      Timer1.Enabled:=True;          // Таймер включен
      Image1.Canvas.Pen.Color:=clBlue;
      Image1.Canvas.Rectangle(T,20,T+20,40);
  end;
end.
```

П1.12. Базы данных

Универсальные базы данных. Такие базы обычно располагаются за пределами ПК и подключении к ним производится через серверы. В этом случае их называют удалёнными базами. К универсальным базам в составе программного обеспечения ПК относится, например, база “Контакты”, используемая электронной почтой. Некоторые из универсальных баз являются однонаправленными, то есть запись в них не возможна.

В системе Delphi 2007 компоненты, обеспечивающие связь с однонаправленными БД, находятся на вкладках dbGo, dbExpress, InterBase и BDE. На вкладках DataControls и DataAcces располагаются компоненты, обеспечивающие отображение, а также запись и редактирование данных.

В большинстве случаев обращение к редактируемым удалённым БД относительно сложно, в них нельзя хранить конфиденциальную информацию.

Поэтому целесообразно научиться создавать свои базы данных на используемом ПК с удобной структурой и простым обращением к ним.

Создание приложений – баз данных. Для создания локальных БД, то есть БД в составе интерфейса собственного компьютера, удобно воспользоваться приложениями Unit – Delphi for Win32 к проекту, на основе которых можно создавать библиотеку визуальных компонентов. Они имеют очень простую структуру. Пример такого приложения:

```
unit MedStat;                                  // приложение Медицинская статистика
interface
var   Nms,Nmso,Nmsx,Nds,Ndso,Ndsx,Nvs,Nvso,Nvsx,Ncmd,Ncmv,Ninv,
      SSP, OPG: array[1980..2015] of real;                    // идентификаторы
      Nrg, Ncmo, Ncmm, Nos: array[1975..2015] of real;  // показателей здоровья
      FONum, OblNum: integer; IFS: array[2005..2006] of real;          // населения
procedure StatRuss;
procedure StatNovObl;
procedure StatFedOkrugi(FONum: integer);
procedure StatOblCS(OblNum: integer);   // статистические данные по областям
        . . .                                  // Северо-Западного Федерального округа
 implementation
procedure StatRuss;                    // Модуль статистических данных показателей
  begin                                // здоровья  населения  РФ
      Nrg[1980]:=14.9;   Nrg[1981]:=15.0;    Nrg[1982]:=16.6;
          . . .
  end;
          . . .                 //   Другие модули-процедуры
end.
```

Для вызова данных локальных баз (в данном случае база MedStat) их имена включаются в список вызываемых системных утилитов. Все глобальные переменные в приложении становятся глобальными и в основной программе. Повторение их в списке глобальных переменных основной программы является ошибкой.

При запуске рабочих процедур в процедуру FormActivate включаются и процедуры БД (см. п. П1.7.1).

П1.13. Запуск проекта. Изменение названий проекта и приложений

Законченные проекты почти всегда подвергаются испытаниям (проверке) в разных режимах работы. Запуск подготовленного проекта проще всего выполнить, нажимая кнопку с зелёным треугольником на вкладке **run** главного окна. Программа проверки правильности работы нового проекта зависит от назначения и содержания проекта. Лучше всего для её реализации знать правильные контрольные значения, которые должны быть получены при выполнении программы. Поэтому остановимся только на изменении названия проекта и приложения Unit.

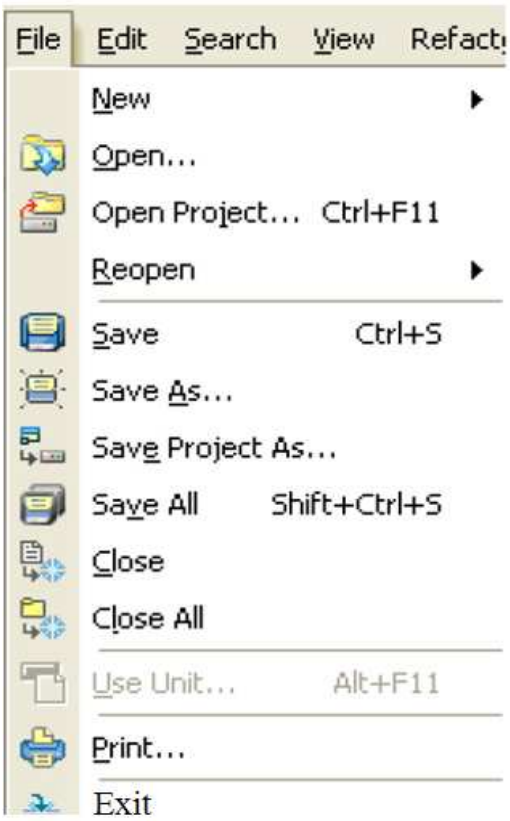


Рис. П1.4. Открытая вкладка File главного окна

При нажатии на кнопку File главного окна открывается его вкладка, приведённая на рис. П1.4. Для изменения название проекта или приложения к нему следует щёлкнуть соответственно по надписи Save Project As … или по надписи Save As … Тогда программа Delphi откроет окно для записи нового названия проекта или приложения.

Если проект и приложение желательно назвать одинаково, то для их отличия по аналогии с вариантом, приведённым на рис. П1.2, можно, например, добавить к их названиям приставки Pr_ и Un_ (но не Pr- и Un-, что является ошибкой, так как ввод символов “-“ в названия файлов не допускается).

П1.14. Страничная организация документов и презентаций

Страничная организация материалов предусматривает задание номеров страниц на общем поле изображений и возможность перехода с любой страницы на любую другую. В редакторе презентаций Microsoft Power Point это делается по номерам страниц на левом поле изображений, а в среде Delphi – на верхнем.

Для решения рассматриваемой задачи, прежде всего, необходимо установить на форме инструмент поле контроля страниц TPageControl – с вкладки Win32 палитры инструментов (Tool Palette). Размеры этого поля не должны быть меньше размеров страниц, создаваемых обычно в окнах Image. В противном случае некоторые части полей страниц будут неконтролируемыми и может нарушиться управление страницами.

Создание очередной страницы программы выполняется с помощью щѐлчка правой клавишей мыши по полю TPageControl или по открытому (выбранному) значку (PageControl) в окне дерева проекта (Project Manager) – рис. П1.5 или окно 4 на рис. П1.1. При этом в открывшемся окне следует выбрать New Page или Next Page. Параметры открывающегося поля страницы (TabSheet) и его имя устанавливаются с помощью Инспектора объектов. Для этого следует в окне Project Manager щѐлкнуть по квадрату данной страницы. При этом Object Inspector откроет вкладку её элементов, в которой в правое поле Caption (заголовок) можно записать желаемый текст или номер страницы. Для выбора типа и размера шрифта этой записи следует нажать прямоугольную кнопку в правой части строки Font вкладки Events.

Элементы, устанавливаемые *на поле открытой страницы*, в том числе окна Image, отражаются в окне Project Manager. На рис. П1.5 приведён пример такого отражения для трёхстраничного приложения.

Если нужно, чтобы в начале работы первой автоматически открывалась *N*-я страница, в процедуру активизации формы следует включить оператор PageControl1.ActivePage:= TabSheetN.

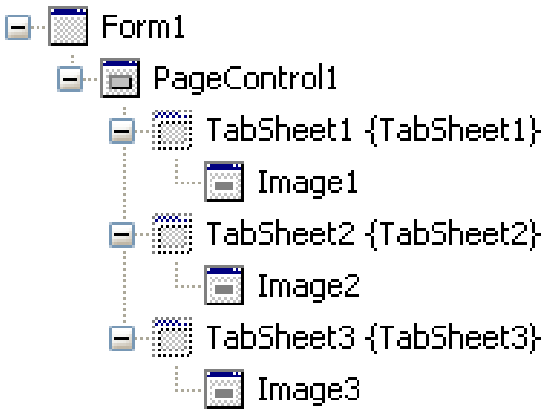


Рис.П1.5. Вид окна Project Manager

Вопросы для самоконтроля по разделу П1

1. Определить понятия Типа и Объекта в системе Delphi. В какой части программы они указываются: в начале, в середине или в конце?
2. Какой вариант развития системы Delphi считается наиболее перспективным и почему?
3. Какие функции выполняет процедура активизации формы? Можно ли всю программу выполнить только на основе этой процедуры?
4. Какие функции выполняет утилита ClipBrd?

5. Записать команду (оператор) округления вещественного числа с тремя знаками после запятой до числа с одним знаком после запятой.
6. Какой формат задания цветов используется в средах Delphi?
7. С какого символа начинается код нестандартного цвета?
8. Как в среде Delphi устанавливаются параметры объектов, выставляемых
9. Как в среде Delphi можно создать локальную базу данных?
10. Какие объекты в системе Delphi относятся к типу Tbitmap?
11. Какую структуру имеет типовая программа в среде Delphi?
12. Каким образом изображения из среды Delphi выводятся в буфер ОС MS?
13. Составление программ, выполняющих:
    - рисование простых фигур и их раскрашивание;
    - поиск вещественных корней нелинейных алгебраических уравнений;
    - вычерчивание графиков;
    - рисование таблиц и их заполнение.

П2. Таблица неприводимых многочленов
5-й, 6-й, 8-й, 10-й и 12-й степени

№ мн.	Коэффициенты многочленов 5-й степени
1	0 0 1 0 1
2	0 1 0 0 1
3	0 1 1 1 1
4	1 0 1 1 1
5	1 1 0 1 1
6	1 1 1 0 1

№ мн.	Коэффициенты многочленов 6-й степени
1	0 0 0 0 1 1
2	0 0 1 0 0 1
3	0 1 0 0 1 1
4	0 1 0 1 1 1
5	0 1 1 0 1 1
6	1 0 0 0 0 1
7	1 0 0 1 0 1
8	1 0 0 1 1 1
9	1 0 1 1 0 1
10	1 1 0 0 1 1
11	1 1 0 1 0 1

№ мн.	Коэффициенты многочленов 8-й степени
1	0 0 0 1 0 1 1 1
2	0 0 0 1 1 0 1 1
3	0 0 0 1 1 1 0 1
4	0 0 1 0 1 0 1 1
5	0 0 1 0 1 1 0 1
6	0 0 1 1 1 0 0 1
7	0 0 1 1 1 1 1 1
8	0 1 0 0 1 1 0 1

№ мн.	Коэффициенты многочленов 12-й степени
1	0 0 0 0 0 0 0 0 1 0 0 1 1
2	0 0 0 0 0 0 0 0 1 1 1 1
3	1 0 0 0 0 0 0 0 1 0 1 1 1
4	0 0 0 0 0 0 1 0 0 0 0 1
5	0 0 0 0 0 0 1 1 0 0 1 1
6	0 0 0 0 0 0 1 1 0 1 0 1
7	0 0 0 0 0 0 1 1 1 1 1 1
8	0 0 0 0 0 1 0 0 1 0 0 1
9	0 0 0 0 0 1 0 0 1 1 0 1
10	0 0 0 0 0 1 0 1 0 0 1 1
11	0 0 0 0 0 1 0 1 1 1 1 1
12	0 0 0 0 0 1 1 0 1 0 0 1
13	0 0 0 0 0 1 1 1 0 1 1 1
14	0 0 0 0 0 1 1 1 1 0 1 1
15	0 0 0 0 0 1 1 1 1 1 0 1
16	0 0 0 0 1 0 0 0 0 0 0 1
17	0 0 0 0 1 0 0 0 0 1 1 1
18	0 0 0 0 1 0 0 0 1 0 1 1
19	0 0 0 0 1 0 0 1 1 0 0 1
20	0 0 0 0 1 0 0 1 1 1 1 1
21	0 0 0 0 1 0 1 0 0 0 1 1
22	0 0 0 0 1 0 1 0 0 1 0 1
23	0 0 0 0 1 1 0 0 1 1 1 1
24	0 0 0 0 1 1 0 1 0 0 0 1
25	0 0 0 0 1 1 1 0 1 0 1 1
26	0 0 0 0 1 1 1 0 1 1 0 1
27	0 0 0 0 1 1 1 1 1 1 1 1
28	0 0 0 1 0 0 0 0 0 1 1 1
29	0 0 0 1 0 0 0 1 1 0 0 1

9	0 1 0 1 1 1 1 1	30	0 0 0 1 0 0 0 1 1 1 1 1
10	0 1 1 0 0 0 1 1	31	0 0 0 1 0 0 1 0 0 0 1 1
11	0 1 1 0 0 1 0 1	32	0 0 0 1 0 0 1 1 0 0 0 1
12	0 1 1 0 1 0 0 1	33	0 0 0 1 0 0 1 1 0 1 1 1
13	0 1 1 1 0 0 0 1	34	0 0 0 1 0 0 1 1 1 0 1 1
14	0 1 1 1 0 1 1 1	35	0 0 0 1 0 0 1 1 1 1 1 1
15	0 1 1 1 1 0 1 1	36	0 0 0 1 0 1 0 0 1 1 1 1
16	1 0 0 0 0 1 1 1	37	0 0 0 1 0 1 0 1 0 1 1 1
17	1 0 0 0 1 0 0 0	38	0 0 0 1 0 1 0 1 1 1 0 1
18	1 0 0 0 1 1 0 1	39	0 0 0 1 0 1 1 0 0 0 0 1
19	1 0 0 1 1 1 1 1	40	0 0 0 1 0 1 1 0 1 0 1 1
20	1 0 1 0 0 0 1 1	41	0 0 0 1 0 1 1 0 1 1 0 1
21	1 0 1 0 1 0 0 1	42	0 0 0 1 0 1 1 1 1 0 0 1
22	1 0 1 1 0 0 0 1	43	0 0 0 1 1 0 0 0 0 0 1 1
23	1 0 1 1 1 0 1 1	44	0 0 0 1 1 0 0 0 0 1 0 1
24	1 1 0 1 1 1 0 1	45	0 0 0 1 1 0 0 1 0 0 0 1
25	1 1 0 0 0 0 1 1	46	0 0 0 1 1 0 1 0 1 0 1 1
26	1 1 0 0 1 1 1 1	47	0 0 0 1 1 0 1 1 0 0 1 1
27	1 1 0 1 0 0 0 1	48	0 0 0 1 1 1 0 1 1 0 0 1
28	1 1 0 1 0 1 1 1	49	0 0 0 1 1 1 0 1 1 1 1 1
29	1 1 0 1 1 1 0 1	50	0 0 0 1 1 1 1 0 0 0 1 1
30	1 1 1 0 0 1 1 1	51	0 0 0 1 1 1 1 0 1 0 0 1
31	1 1 1 1 0 0 1 1	52	0 0 0 1 1 1 1 0 1 1 1 1
32	1 1 1 1 0 1 0 1	53	0 0 0 1 1 1 1 1 0 0 0 1
33	1 1 1 1 1 0 0 1	54	0 0 0 1 1 1 1 1 0 1 1 1
№	Коэффициенты многочленов 10-й степени	55	0 0 1 0 0 0 0 0 0 0 0 1
		56	0 0 1 0 0 0 0 0 1 1 0 1
1	0 0 0 0 0 0 1 0 0 1	57	0 0 1 0 0 0 0 1 0 0 1 1
2	0 0 0 0 0 0 1 1 1 1	58	0 0 1 0 0 0 0 1 1 0 0 1
3	0 0 0 0 0 1 1 0 1 1	59	0 0 1 0 0 0 1 0 0 1 0 1
4	0 0 0 0 0 1 1 1 0 1	60	0 0 1 0 0 0 1 1 0 1 1 1
5	0 0 0 0 1 0 0 1 1 1	61	0 0 1 0 0 0 1 1 1 0 1 1
6	0 0 0 0 1 0 1 1 0 1	62	0 0 1 0 0 0 1 1 1 1 0 1

7	0 0 0 0 1 1 0 1 0 1	63	0 0 1 0 0 1 0 0 0 0 0 1
8	0 0 0 1 0 0 0 1 1 1	64	0 0 1 0 0 1 0 0 0 0 1 1
9	0 0 0 1 0 1 0 0 1 1	65	0 0 1 0 0 1 0 0 0 1 0 1
10	0 0 0 1 1 0 0 0 1 1	66	0 0 1 0 0 1 0 0 1 0 0 1
11	0 0 0 1 1 0 0 1 0 1	67	0 0 1 0 0 1 0 1 1 0 1 1
12	0 0 0 1 1 0 1 1 1 1	68	0 0 1 0 0 1 1 0 0 1 1 1
13	0 0 0 1 1 1 0 0 0 1	69	0 0 1 0 0 1 1 0 1 1 0 1
14	0 0 1 0 0 0 0 0 0 1	70	0 0 1 0 0 1 1 1 0 0 1 1
15	0 0 1 0 0 0 1 0 1 1	71	0 0 1 0 0 1 1 1 1 1 1 1
16	0 0 1 0 0 1 1 0 0 1	72	0 0 1 0 1 0 0 1 1 0 1 1
17	0 0 1 0 1 0 1 0 0 1	73	0 0 1 0 1 0 1 0 1 0 1 1
18	0 0 1 0 1 0 1 1 1 1	74	0 0 1 0 1 0 1 1 0 1 0 1
19	0 0 1 0 1 1 0 1 1 1	75	0 0 1 0 1 0 1 1 1 0 0 1
20	0 0 1 1 0 0 0 0 1 1	76	0 0 1 0 1 1 0 0 0 0 0 1
21	0 0 1 1 0 0 0 1 0 1	77	0 0 1 0 1 1 0 0 1 0 1 1
22	0 0 1 1 0 0 1 0 0 1	78	0 0 1 0 1 1 0 1 0 0 1 1
23	0 0 1 1 0 1 0 1 1 1	79	0 0 1 0 1 1 1 1 0 0 0 1
24	0 0 1 1 1 0 0 1 1 1	80	0 0 1 1 0 0 0 0 0 0 1 1
25	0 0 1 1 1 0 1 1 0 1	81	0 0 1 1 0 0 0 0 1 0 0 1
26	0 0 1 1 1 1 0 0 1 1	82	0 0 1 1 0 0 0 0 1 1 1 1
27	0 0 1 1 1 1 1 1 1 1	83	0 0 1 1 0 0 0 1 0 0 0 1
28	0 1 0 0 0 0 1 0 1 1	84	0 0 1 1 0 0 0 1 1 0 1 1
29	0 1 0 0 0 0 1 1 0 1	85	0 0 1 1 0 0 0 1 1 1 0 1
30	0 1 0 0 0 1 1 0 0 1	86	0 0 1 1 0 0 0 1 1 1 1 1
31	0 1 0 0 0 1 1 1 1 1	87	0 0 1 1 0 0 1 0 0 0 0 1
32	0 1 0 0 1 0 0 0 1 1	88	0 0 1 1 0 0 1 1 0 0 1 1
33	0 1 0 0 1 1 0 0 0 1	89	0 0 1 1 0 0 1 1 0 1 0 1
34	0 1 0 0 1 1 1 1 0 1	90	0 0 1 1 0 0 1 1 1 0 0 1
35	0 1 0 1 0 0 0 0 1 1	91	0 0 1 1 0 0 1 1 1 1 1 1
36	0 1 0 1 0 1 0 1 1 1	92	0 0 1 1 0 1 0 0 1 1 0 1
37	0 1 0 1 1 0 0 0 0 1	93	0 0 1 1 0 1 1 0 0 1 0 1
38	0 1 0 1 1 0 0 1 1 1	94	0 0 1 1 0 1 1 1 0 0 0 1
39	0 1 0 1 1 0 1 0 1 1	95	0 0 1 1 1 0 0 1 0 1 0 1

40	0 1 0 1 1 1 0 0 1 1	96	0 0 1 1 1 0 0 1 1 0 0 1
41	0 1 1 0 0 0 0 1 0 1	97	0 0 1 1 1 0 0 1 1 1 0 1
42	0 1 1 0 0 0 1 1 1 1	98	0 0 1 1 1 0 0 1 1 1 1 1
43	0 1 1 0 0 1 0 1 1 1	99	0 0 1 1 1 0 1 0 0 0 1 1
44	0 1 1 0 0 1 1 0 1 1	100	0 0 1 1 1 0 1 0 1 0 0 1
45	0 1 1 0 1 0 0 0 0 1	101	0 0 1 1 1 0 1 0 1 1 1 1
46	0 1 1 0 1 0 1 0 1 1	102	0 0 1 1 1 0 1 1 1 0 1 1
47	0 1 1 0 1 0 1 1 0 1	103	0 0 1 1 1 1 0 0 0 0 1 1
48	0 1 1 0 1 1 1 0 0 1	104	0 0 1 1 1 1 0 1 0 0 0 1
49	0 1 1 1 0 0 0 0 0 1	105	0 0 1 1 1 1 0 1 0 1 1 1
50	0 1 1 1 0 0 0 1 1 1	106	0 0 1 1 1 1 0 1 1 1 0 1
51	0 1 1 1 0 1 1 1 1 1	107	0 0 1 1 1 1 1 1 0 0 1 1
52	0 1 1 1 1 0 0 1 0 1	108	0 0 1 1 1 1 1 1 0 1 0 1
53	0 1 1 1 1 1 0 1 1 1	109	0 0 1 1 1 1 1 1 1 0 0 1
54	0 1 1 1 1 1 1 0 1 1	110	0 1 0 0 0 0 0 0 0 1 1 1
55	1 0 0 0 0 0 1 0 1 1	111	0 1 0 0 0 0 0 1 0 0 1 1
56	1 0 0 0 0 1 0 0 1 1	112	0 1 0 0 0 0 1 0 1 1 1 1
57	1 0 0 0 0 1 0 1 0 0	113	0 1 0 0 0 0 1 1 0 0 0 1
58	1 0 0 0 0 1 1 0 0 1	114	0 1 0 0 0 0 1 1 0 1 0 1
59	1 0 0 0 1 0 0 0 1 1	115	0 1 0 0 0 0 1 1 0 1 1 1
60	1 0 0 0 1 0 0 1 0 1	116	0 1 0 0 0 1 0 0 0 0 1 1
61	1 0 0 0 1 1 0 0 0 1	117	0 1 0 0 0 1 0 0 1 0 0 1
62	1 0 0 0 1 1 0 1 1 1	118	0 1 0 0 0 1 0 0 1 1 1 1
63	1 0 0 1 0 0 0 0 1 1	119	0 1 0 0 0 1 0 1 0 1 1 1
64	1 0 0 1 0 0 1 1 1 1	120	0 1 0 0 0 1 0 1 1 0 1 1
65	1 0 0 1 0 1 0 0 0 1	121	0 1 0 0 0 1 0 1 1 1 0 1
66	1 0 0 1 0 1 1 0 1 1	122	0 1 0 0 0 1 1 0 0 1 1 1
67	1 0 0 1 1 1 0 1 0 1	123	0 1 0 0 0 1 1 0 1 0 1 1
68	1 0 0 1 1 1 1 0 0 1	124	0 1 0 0 0 1 1 1 0 1 0 1
69	1 0 0 1 1 1 1 1 1 1	125	0 1 0 0 0 1 1 1 1 1 1 1
70	1 0 1 0 0 0 0 0 1 1	126	0 1 0 0 1 0 0 0 1 0 0 1
71	1 0 1 0 0 0 0 1 0 1	127	0 1 0 0 1 0 1 0 0 0 0 1
72	1 0 1 0 0 0 1 0 0 1	128	0 1 0 0 1 0 1 0 0 1 1 1

73	1	0	1	0	1	0	0	1	1	1	129	0	1	0	0	1	0	1	0	1	1	0	1
74	1	0	1	0	1	0	1	1	0	1	130	0	1	0	0	1	0	1	1	0	1	0	1
75	1	0	1	0	1	1	0	1	0	1	131	0	1	0	0	1	1	0	0	1	1	0	1
76	1	0	1	0	1	1	1	1	1	1	132	0	1	0	0	1	1	0	1	0	0	1	1
77	1	0	1	1	0	0	0	0	0	1	133	0	1	0	0	1	1	0	1	1	0	0	1
78	1	0	1	1	0	0	1	1	0	1	134	0	1	0	0	1	1	0	1	1	0	1	1
79	1	0	1	1	0	1	0	0	1	1	135	0	1	0	0	1	1	0	1	1	1	1	1
80	1	0	1	1	0	1	1	1	1	1	136	0	1	0	0	1	1	1	0	0	1	0	1
81	1	0	1	1	1	1	0	1	1	1	137	0	1	0	1	0	0	0	0	1	1	0	1
82	1	0	1	1	1	1	1	0	1	1	138	0	1	0	1	0	0	0	0	1	1	1	1
83	1	0	1	1	1	1	1	1	0	1	139	0	1	0	1	0	0	0	1	1	1	0	1
84	1	1	0	0	0	0	1	1	1	1	140	0	1	0	1	0	0	1	1	1	0	0	1
85	1	1	0	0	0	1	0	0	0	1	141	0	1	0	1	0	1	0	0	0	1	1	1
86	1	1	0	0	0	1	0	1	1	1	142	0	1	0	1	0	1	0	0	1	0	1	1
87	1	1	0	0	0	1	1	1	0	1	143	0	1	0	1	0	1	0	0	1	1	0	1
88	1	1	0	0	1	0	0	0	0	1	144	0	1	0	1	1	0	0	0	0	0	0	1
89	1	1	0	0	1	0	1	0	1	1	145	0	1	0	1	1	0	0	0	0	1	0	1
90	1	1	0	0	1	1	0	1	0	1	146	0	1	0	1	1	0	0	1	0	0	1	1
91	1	1	0	0	1	1	1	0	0	1	147	0	1	0	1	1	0	0	1	1	0	0	1
92	1	1	0	1	0	0	0	1	1	1	148	0	1	0	1	1	0	1	0	0	1	0	1
93	1	1	0	1	0	0	1	1	0	1	149	0	1	0	1	1	0	1	0	1	0	0	1
94	1	1	0	1	0	1	0	1	0	1	150	0	1	0	1	1	0	1	1	1	0	1	1
95	1	1	0	1	0	1	1	0	0	1	151	0	1	0	1	1	1	0	0	0	1	0	1
96	1	1	0	1	1	0	0	0	1	1	152	0	1	0	1	1	1	0	1	0	1	1	1
97	1	1	0	1	1	0	1	0	0	1	153	0	1	0	1	1	1	0	1	1	1	0	1
98	1	1	0	1	1	0	1	1	1	1	154	0	1	0	1	1	1	1	0	1	0	1	1
99	1	1	0	1	1	1	1	0	1	1	155	0	1	0	1	1	1	1	1	0	1	0	1
100	1	1	0	1	1	1	1	1	0	1	156	0	1	0	1	1	1	1	1	1	0	0	1
101	1	1	1	0	0	0	0	0	0	1	157	0	1	1	0	0	0	0	0	0	0	1	1
102	1	1	1	0	0	0	0	1	1	1	158	0	1	1	0	0	0	0	0	1	0	0	1
103	1	1	1	0	0	0	1	1	0	1	159	0	1	1	0	0	0	0	0	1	0	1	1
104	1	1	1	0	0	1	0	0	1	1	160	0	1	1	0	0	0	0	1	0	1	0	1
105	1	1	1	0	1	0	1	0	0	1	161	0	1	1	0	0	0	0	1	1	0	0	1

106	1 1 1 0 1 1 0 0 0 1	162	0 1 1 0 0 0 1 0 1 1 0 1
107	1 1 1 0 1 1 0 1 1 1	163	0 1 1 0 0 0 1 1 1 1 1 1
108	1 1 1 1 0 0 0 1 0 1	164	0 1 1 0 0 1 0 0 0 0 0 1
109	1 1 1 1 0 1 1 0 1 1	165	0 1 1 0 0 1 0 0 0 1 1 1
110	1 1 1 1 0 1 1 1 0 1	166	0 1 1 0 0 1 0 0 1 0 1 1
111	1 1 1 1 1 0 1 0 1 1	167	0 1 1 0 0 1 0 1 0 0 1 1
112	1 1 1 1 1 1 0 0 1 1	168	0 1 1 0 0 1 0 1 0 1 0 1
113	1 1 1 1 1 1 1 0 0 1	169	0 1 1 0 0 1 0 1 1 0 0 1
- - -		170	0 1 1 0 0 1 1 0 0 0 1 1
№	Коэффициенты многочленов 12-й степени (продолжение)	171	0 1 1 0 0 1 1 0 0 1 0 1
		172	0 1 1 0 0 1 1 0 1 1 1 1
173	0 1 1 0 0 1 1 1 1 0 1 1	174	0 1 1 0 1 0 0 0 1 1 0 1
175	0 1 1 0 1 0 0 1 0 0 1 1	176	0 1 1 0 1 0 1 0 0 1 0 1
177	0 1 1 0 1 0 1 1 1 1 0 1	178	0 1 1 0 1 1 0 0 0 0 1 1
179	0 1 1 0 1 1 0 0 1 0 0 1	180	0 1 1 0 1 1 0 1 0 0 0 1
181	0 1 1 0 1 1 0 1 1 1 0 1	182	0 1 1 0 1 1 1 0 0 0 0 1
183	0 1 1 0 1 1 1 0 0 1 1 1	184	0 1 1 0 1 1 1 1 0 0 1 1
185	0 1 1 0 1 1 1 1 1 1 1 1	186	0 1 1 1 0 0 0 0 1 0 1 1
187	0 1 1 1 0 0 0 1 0 1 0 1	188	0 1 1 1 0 0 0 1 1 0 0 1
189	0 1 1 1 0 0 1 0 1 1 1 1	190	0 1 1 1 0 0 1 1 1 0 0 1
191	0 1 1 1 0 0 1 1 1 1 0 1	192	0 1 1 1 0 1 0 0 0 0 1 1
193	0 1 1 1 0 1 0 0 0 1 0 1	194	0 1 1 1 0 1 0 0 1 1 1 1
195	0 1 1 1 0 1 0 1 0 0 0 1	196	0 1 1 1 0 1 0 1 0 1 1 1
197	0 1 1 1 0 1 0 1 1 1 0 1	198	0 1 1 1 0 1 1 0 1 1 0 1
199	0 1 1 1 0 1 1 1 0 0 1 1	200	0 1 1 1 0 1 1 1 0 1 0 1
201	0 1 1 1 0 1 1 1 1 0 0 1	202	0 1 1 1 1 0 0 0 1 0 0 1
203	0 1 1 1 1 0 0 0 1 1 1 1	204	0 1 1 1 1 0 0 1 0 1 1 1
205	0 1 1 1 1 0 0 1 1 1 0 1	206	0 1 1 1 1 0 1 0 0 1 1 1
207	0 1 1 1 1 0 1 0 1 1 0 1	208	0 1 1 1 1 0 1 1 0 0 1 1
209	0 1 1 1 1 0 1 1 1 1 1 1	210	0 1 1 1 1 1 0 0 0 0 0 1
211	0 1 1 1 1 1 0 1 1 1 1 1	212	0 1 1 1 1 1 1 0 0 0 1 1
213	0 1 1 1 1 1 1 1 1 0 1 1	214	1 0 0 0 0 0 0 0 0 1 1 1
215	1 0 0 0 0 0 0 0 1 1 0 1	216	1 0 0 0 0 0 0 1 1 0 0 1

217	1	0	0	0	0	0	0	1	1	1	1	1	218	1	0	0	0	0	0	1	0	0	0	1	1
219	1	0	0	0	0	0	1	0	0	1	1	1	220	1	0	0	0	0	0	1	1	0	0	0	1
221	1	0	0	0	0	0	1	1	0	1	1	1	222	1	0	0	0	0	1	0	0	0	1	0	1
223	1	0	0	0	0	1	0	0	1	0	0	1	224	1	0	0	0	0	1	0	1	0	1	1	1
225	1	0	0	0	0	1	0	1	1	1	0	1	226	1	0	0	0	0	1	1	0	1	1	0	1
227	1	0	0	0	0	1	1	1	1	0	0	1	228	1	0	0	0	0	1	1	1	1	1	1	1
229	1	0	0	0	1	0	0	0	0	0	1	1	230	1	0	0	0	1	0	0	1	0	0	0	1
231	1	0	0	0	1	0	0	1	0	1	1	1	232	1	0	0	0	1	0	1	0	0	0	0	1
233	1	0	0	0	1	0	1	1	1	0	0	1	234	1	0	0	0	1	0	1	1	1	1	1	1
235	1	0	0	0	1	1	0	0	1	0	1	1	236	1	0	0	0	1	1	0	0	1	1	0	1
237	1	0	0	0	1	1	1	0	0	0	1	1	238	1	0	0	0	1	1	1	0	1	1	1	1
239	1	0	0	0	1	1	1	1	0	0	0	1	240	1	0	0	0	1	1	1	1	1	0	1	1
241	1	0	0	0	1	1	1	1	1	1	0	1	242	1	0	0	1	0	0	0	0	0	1	0	1
243	1	0	0	1	0	0	0	0	1	0	0	1	244	1	0	0	1	0	0	0	1	1	0	1	1
245	1	0	0	1	0	0	1	0	0	1	1	1	246	1	0	0	1	0	0	1	0	1	0	1	1
247	1	0	0	1	0	0	1	0	1	1	0	1	248	1	0	0	1	0	0	1	1	0	1	0	1
249	1	0	0	1	0	1	0	0	0	0	0	1	250	1	0	0	1	0	1	0	0	1	0	1	1
251	1	0	0	1	0	1	0	0	1	1	0	1	252	1	0	0	1	0	1	0	1	1	1	1	1
253	1	0	0	1	0	1	1	0	0	1	0	1	254	1	0	0	1	0	1	1	0	1	0	0	1
255	1	0	0	1	0	1	1	0	1	1	1	1	256	1	0	0	1	0	1	1	1	1	0	1	1
257	1	0	0	1	1	0	0	0	0	0	0	1	258	1	0	0	1	1	0	0	0	1	0	1	1
259	1	0	0	1	1	0	0	1	1	0	0	1	260	1	0	0	1	1	0	1	1	0	0	0	1
261	1	0	0	1	1	0	1	1	0	1	1	1	262	1	0	0	1	1	0	1	1	1	1	0	1
263	1	0	0	1	1	1	0	0	1	0	0	1	264	1	0	0	1	1	1	0	0	1	1	1	1
265	1	0	0	1	1	1	0	1	0	1	1	1	266	1	0	0	1	1	1	0	1	1	1	0	1
267	1	0	0	1	1	1	1	0	0	1	1	1	268	1	0	0	1	1	1	1	0	1	1	0	1
269	1	0	0	1	1	1	1	0	1	1	1	1	270	1	0	0	1	1	1	1	1	1	0	0	1
271	1	0	1	0	0	0	0	0	1	1	0	1	272	1	0	1	0	0	0	0	1	1	0	1	1
273	1	0	1	0	0	0	0	1	1	1	0	1	274	1	0	1	0	0	0	1	0	0	0	0	1
275	1	0	1	0	0	0	1	0	1	0	1	1	276	1	0	1	0	0	0	1	1	0	0	1	1
277	1	0	1	0	0	1	0	0	1	0	1	1	278	1	0	1	0	0	1	0	0	1	1	0	1
279	1	0	1	0	0	1	0	1	0	0	1	1	280	1	0	1	0	0	1	0	1	0	1	0	1
281	1	0	1	0	0	1	0	1	1	1	1	1	282	1	0	1	0	0	1	1	0	0	0	1	1

283	1	0	1	0	0	1	1	0	1	0	0	1	284	1	0	1	0	0	1	1	0	1	1	1	1
285	1	0	1	0	0	1	1	1	1	0	1	1	286	1	0	1	0	1	0	0	0	1	0	1	1
287	1	0	1	0	1	0	0	0	1	1	1	1	288	1	0	1	0	1	0	0	1	0	0	1	1
289	1	0	1	0	1	0	0	1	1	1	1	1	290	1	0	1	0	1	0	1	0	1	0	0	1
291	1	0	1	0	1	0	1	1	0	0	0	1	292	1	0	1	0	1	1	0	0	0	1	0	1
293	1	0	1	0	1	1	0	1	0	0	0	1	294	1	0	1	0	1	1	1	0	0	0	0	1
295	1	0	1	0	1	1	1	0	0	1	1	1	296	1	0	1	0	1	1	1	0	1	0	1	1
297	1	0	1	0	1	1	1	1	0	1	0	1	298	1	0	1	1	0	0	0	0	1	0	1	1
299	1	0	1	1	0	0	0	0	1	1	0	1	300	1	0	1	1	0	0	0	1	0	0	1	1
301	1	0	1	1	0	0	0	1	1	0	0	1	302	1	0	1	1	0	0	0	1	1	1	1	1
303	1	0	1	1	0	0	1	0	1	0	0	1	304	1	0	1	1	0	0	1	0	1	1	1	1
305	1	0	1	1	0	1	0	0	0	1	0	1	306	1	0	1	1	0	1	0	0	1	0	0	1
307	1	0	1	1	0	1	0	1	0	1	1	1	308	1	0	1	1	0	1	0	1	1	0	1	1
309	1	0	1	1	0	1	1	0	0	1	0	1	310	1	0	1	1	1	0	0	0	1	0	0	1
311	1	0	1	1	1	0	0	0	1	1	1	1	312	1	0	1	1	1	0	0	1	0	0	0	1
313	1	0	1	1	1	0	1	0	0	1	1	1	314	1	0	1	1	1	0	1	1	0	1	0	1
315	1	0	1	1	1	0	1	1	1	0	0	1	316	1	0	1	1	1	0	1	1	1	1	1	1
317	1	0	1	1	1	1	0	0	0	0	0	1	318	1	0	1	1	1	1	0	0	1	0	1	1
319	1	0	1	1	1	1	0	0	1	1	0	1	320	1	0	1	1	1	1	0	1	0	0	1	1
321	1	0	1	1	1	1	1	0	0	0	1	1	322	1	0	1	1	1	1	1	1	0	1	1	1
323	1	0	1	1	1	1	1	1	1	1	0	1	324	1	1	0	0	0	0	0	0	0	0	1	1
325	1	0	0	0	0	0	0	0	0	1	0	1	326	1	1	0	0	0	0	0	0	1	1	1	1
327	1	1	0	0	0	0	0	1	0	0	0	1	328	1	1	0	0	0	0	0	1	0	1	1	1
329	1	1	0	0	0	0	1	0	0	0	0	1	330	1	1	0	0	0	0	1	0	0	1	1	1
331	1	1	0	0	0	1	0	0	0	1	1	1	332	1	1	0	0	0	1	0	0	1	1	0	1
333	1	1	0	0	0	1	0	1	0	1	0	1	334	1	1	0	0	0	1	0	1	1	1	1	1
335	1	1	0	0	1	0	0	0	0	0	1	1	336	1	1	0	0	1	0	0	0	0	1	1	1
337	1	1	0	0	1	0	0	1	0	0	1	1	338	1	1	0	0	1	0	0	1	0	1	1	1
339	1	1	0	0	1	0	0	1	1	1	1	1	340	1	1	0	0	1	0	1	0	0	1	0	1
341	1	1	0	0	1	0	1	1	0	1	1	1	342	1	1	0	0	1	0	1	1	1	0	1	1
343	1	1	0	0	1	0	1	1	1	1	0	1	344	1	1	0	0	2	2	0	0	0	1	0	1
345	1	1	0	0	1	1	0	0	1	0	0	1	346	1	1	0	0	1	1	0	0	1	1	1	1
347	1	1	0	0	1	1	0	1	0	1	1	1	348	1	1	0	0	1	1	1	0	1	0	1	1

349	1	1	0	0	1	1	1	0	1	1	0	1	350	1	1	0	0	1	1	1	1	0	0	1	1
351	1	1	0	1	0	0	0	0	0	0	0	1	352	1	1	0	1	0	0	0	0	0	1	1	1
353	1	1	0	1	0	0	1	0	0	0	1	1	354	1	1	0	1	0	0	1	0	0	1	1	1
355	1	1	0	1	0	0	1	0	1	1	1	1	356	1	1	0	1	0	0	1	1	1	1	0	1
357	1	1	0	1	0	1	0	0	0	0	1	1	358	1	1	0	1	0	1	0	0	0	1	0	1
359	1	1	0	1	0	1	0	1	0	0	0	1	360	1	1	0	1	0	1	0	1	1	0	1	1
361	1	1	0	1	0	1	0	1	1	1	0	1	362	1	1	0	1	0	1	1	0	0	1	1	1
363	1	1	0	1	0	1	1	1	0	0	1	1	364	1	1	0	1	0	1	1	1	0	1	0	1
365	1	1	0	1	0	1	1	1	1	0	0	1	366	1	1	0	1	1	0	0	0	0	0	1	1
367	1	1	0	1	1	0	0	0	0	1	0	1	368	1	1	0	1	1	0	0	0	1	0	0	1
369	1	1	0	1	1	0	0	1	0	0	0	1	370	1	1	0	1	1	0	1	0	0	1	1	1
371	1	1	0	1	1	0	1	1	0	0	1	1	372	1	1	0	1	1	1	0	0	0	0	0	1
373	1	1	0	1	1	1	0	1	1	1	1	1	374	1	1	0	1	1	1	1	0	1	1	1	1
375	1	1	0	1	1	1	1	1	0	0	0	1	376	1	1	0	1	1	1	1	1	1	0	1	1
377	1	1	1	0	0	0	0	0	0	0	0	1	378	1	1	1	0	0	0	0	0	0	1	1	1
379	1	1	1	0	0	0	0	1	0	1	0	1	380	1	1	1	0	0	0	0	1	1	0	0	1
381	1	1	1	0	0	0	1	0	1	0	1	1	382	1	1	1	0	0	0	1	0	1	1	1	1
383	1	1	1	0	0	0	1	1	1	0	1	1	384	1	1	1	0	0	0	1	1	1	1	0	1
385	1	1	1	0	0	1	0	0	0	1	0	1	386	1	1	1	0	0	1	0	0	1	1	1	1
387	1	1	1	0	0	1	0	1	0	0	0	1	388	1	1	1	0	0	1	0	1	1	1	0	1
389	1	1	1	0	0	1	1	0	0	0	0	1	390	1	1	1	0	0	1	1	0	0	1	1	1
391	1	1	1	0	0	1	1	1	0	0	1	1	392	1	1	1	0	1	0	0	0	0	1	0	1
393	1	1	1	0	1	0	0	0	1	1	1	1	394	1	1	1	0	1	0	0	1	0	1	1	1
395	1	1	1	0	1	0	0	1	1	0	1	1	396	1	1	1	0	1	0	0	1	1	1	0	1
397	1	1	1	0	1	0	1	1	1	0	0	1	398	1	1	1	0	1	0	1	1	1	1	1	1
399	1	1	1	0	1	1	0	0	1	0	1	1	400	1	1	1	0	1	1	0	0	1	1	0	1
401	1	1	1	0	1	1	0	1	0	0	1	1	402	1	1	1	0	1	1	1	0	0	0	1	1
403	1	1	1	0	1	1	1	1	0	0	0	1	404	1	1	1	0	1	1	1	1	0	0	1	1
405	1	1	1	0	1	1	1	1	0	1	1	1	406	1	1	1	1	0	0	0	0	0	0	1	
407	1	1	1	1	0	0	0	1	0	0	0	1	408	1	1	1	1	0	0	0	1	1	0	0	
409	1	1	1	1	0	0	0	1	1	0	1	1	410	1	1	1	1	0	0	1	0	0	0	0	
411	1	1	1	1	0	0	1	0	0	1	1	1	412	1	1	1	1	0	0	1	0	1	0	1	
413	1	1	1	1	0	0	1	1	1	0	0	1	414	1	1	1	1	0	1	0	0	0	0	0	

415	1	1	1	1	0	1	0	0	0	1	1	1	416	1	1	1	1	0	1	0	0	1	0	1
417	1	1	1	1	0	1	0	1	0	0	1	1	418	1	1	1	1	0	1	1	0	0	1	0
419	1	1	1	1	0	1	1	1	0	0	0	1	420	1	1	1	1	0	1	1	1	0	1	1
421	1	1	1	1	0	1	1	1	1	1	0	1	422	1	1	1	1	1	0	0	0	0	0	0
423	1	1	1	1	1	0	0	0	1	1	0	1	424	1	1	1	1	1	0	0	1	0	0	0
425	1	1	1	1	1	0	0	1	1	0	0	1	426	1	1	1	1	1	0	1	0	0	0	1
427	1	1	1	1	1	0	1	0	1	1	1	1	428	1	1	1	1	1	0	1	1	1	0	1
429	1	1	1	1	1	0	1	1	1	1	0	1	430	1	1	1	1	1	1	0	0	0	0	1
431	1	1	1	1	1	1	0	0	0	1	0	1	432	1	1	1	1	1	1	0	0	1	0	0
433	1	1	1	1	1	1	1	0	0	0	0	1	434	1	1	1	1	1	1	1	0	1	1	0

Литература

1. Бахвалов Н.С. Численные методы. 1. М.: “Наука”, 1975. 632 с.
2. Блох И.П., Попов О.В., Турин В.Я. Модели источника ошибок в каналах передачи цифровой информации. М: Связь, 1971. 312 с.
3. Богомолов А.П. Основы математического моделирования. М.: МГУ. 2003. URL: http://www.MirKnig.com. – 2,98 Мб.
4. Бокс Дж., Дженкинс Г.Д. Анализ временных рядов. М.: Мир. 1974. 408 с.
5. Гилл А. Линейные последовательностные машины: Пер. с англ. [под ред. Я.З. Цыпкина]. М.: Наука. 1974. 286 с.
6. Данилов Д.Л., Жиглявский А.А. Главные компоненты временных рядов: Метод «Гусеница». СПб: Пресском. 1997. 308 с.
7. Добрис Г.В. Метод синтеза генератора псевдослучайных чисел для стохастических вычислительных машин на основе двух регистров сдвига // Автоматика и вычислительная техника. 1973. № 2. С. 1 – 7.
8. Dörfel G., Кирьянов Б.Ф.. Моделирование редких событий / Математическое моделирование и цифровая обработка изображений. Новгород: НПИ,
9. Ермаков С. М., Михайлов Г. А. Курс статистического моделирования. М.: Наука. 1976. 320 с.
10. Ермаков С.М., Михайлов Г.А. Статистическое моделирование. М.: Наука. 1982. 296 с.
11. Жгун А.А. Исследование длины настраивающей последовательности в модели скрытой передачи информации // Сб. материалов 9-ой Международной конференции «Информационные и коммуникационные технологии в образовании». Борисоглебск. 2008. С. 206 – 209.
12. Жгун А.А. Исследование ложной синхронизации приёма и передачи информации в модели скрытой передачи информации // Вестник Новгородского гос. ун-та им. Ярослава Мудрого. 2010. Вып. 60. С. 33 – 35.
13. Жгун А.А., Жгун Т.В. Определение длины настраивающей последовательности в модели скрытой передачи информации // Вестник Новгородского гос. ун-та им. Ярослава Мудрого. 2007. Вып. 44. С. 26 – 29.
14. Жгун А.А., Жгун Т.В., Осадчий А.Н. Исследование синхронизации приёма и передачи информации в стенографической системе // Научно-технические ведомости СПбГПУ, Сер. “Информатика. Телекоммуникации. Управление”. СПб: СПбГПУ. 2010. № 2. С. 7 – 11.

15. Жгун А.А., Кирьянов Б.Ф. Оценка вероятности синхронизации в модели скрытой передачи информации // Вестник Казанского гос. технич. ун-та им. А.Н. Туполева. 2009. Вып. 4. С. 78 – 81.
16. Жгун Т.В. Компьютерная модель скрытой передачи информации в локальных сетях. Деп. в ВИНИТИ 18.08.03. № 885 – В2003. 52 с.
17. Жгун Т.В., Кирьянов Б.Ф. Распознавание двоичной информации, перемешанной со случайными символами // Труды 6-й Междунар. конф. “Распознавание образов и анализ изображений: новые информационные технологии”. Вел. Новгород. 2002. Т.1. С. 231 – 232.
18. Жгун Т.В., Кирьянов Б.Ф. Модель скрытой передачи информации // Вестник Новгородского гос. ун-та им. Ярослава Мудрого. Сер. “Математика и информатика”. 2002. Вып. 22. С. 50 – 53.
19. Зарубин В.П. Математическое моделирование в технике. Вып. XX. Учебник для вузов. Физико-математические науки. Математическая кибернетика. Вычислительная математика. М: МВТУ им. Н.Э. Баумана.
20. Карпов Ю. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5 (+CD). СПб: БХВ-Петербург. 2006. 400 с.
21. Кирьянов Б.Ф. Основы теории стохастических вычислительных машин и устройств. Монография. Казань: КАИ им. А.Н. Туполева. 1975. 167 с. [Деп. в ЦНИИТЭИ приборостроения, ДР № 524, 21.05.1976].
22. Кирьянов Б.Ф. Микро-ЭВМ как средства имитации и обработки случайных процессов в радиоэлектронных системах. Монография. Новгород: Новгородский политехнич. ин-т. 1986. 213 с. [Деп. в ВИНИТИ, ДР № 7646-В86, 10.11.1986].
23. Кирьянов Б.Ф. Многоканальный генератор псевдослучайных символов // Известия АН СССР. Техническая кибернетика. 1970. Вып. 4. С. 107 –
24. Кирьянов Б.Ф. Эквивалентность систем, реализующих стохастический принцип вычислений // Известия АН СССР. Техническая кибернетика. 1972. Вып. 5. С. 121 – 128.
25. Кирьянов Б.Ф. Вопросы теории и проектирования многоканальных генераторов псевдослучайных последовательностей на одном сдвигающем регистре // Рига: Вероятностные автоматы и их применение. 1971.
26. Кирьянов Б.Ф. Микропроцессорные средства в задачах имитации и обработки случайных сигналов: Новгородский политехнич. ин-т. Учебное пособие. Часть 1. 1988. 52 с.
27. Кирьянов Б.Ф. Микропроцессорные средства в задачах имитации и обработки случайных сигналов: Новгородский политехнич. ин-т. Учебное пособие. Часть 2. 1989. 48 с.

28. Кирьянов Б.Ф. Математическое моделирование на ЭВМ: Новгородский гос. ун-т им. Ярослава Мудрого. Учебное пособие [Электронный ресурс]. 2011. 128 с.
29. Кирьянов Б.Ф. Математическое моделирование: Учебное пособие к лаб. работам: Новгородский гос. ун-т им. Ярослава Мудрого. 2006. 36 с.
30. Кирьянов Б.Ф. Параметрическая модель управления групповым порогом // Математическое моделирование и его приложения. Сб. трудов Новгородского политехнич. ин-та. Под ред. проф. Кирьянова Б.Ф. 1993.
31. Кирьянов Б.Ф. Обобщение теоремы о сдвигающем векторе и практические аспекты ее применения // Прикладная математика. Сб. трудов Новгородского политехнич. ин-та. 1994. Вып. 1. С. 57 – 60.
32. Кирьянов Б.Ф., Болотханов Э.Б. Усечение распределений в стохастических моделях. Реализация нормального распределения // Материалы докладов международ. научно-технич. конф. “Математическое моделирование в на-уке, промышленности и образовании”. Тирасполь: РИО
33. Кирьянов Б.Ф., Глова В.И., Леонтьев А.Г., Ризаев И.С. Формирование случайных последовательностей при физическом моделировании дискретных каналов связи // М: “Наука”, Кодирование и передача сообщений в системах связи (Академия наук СССР, Институт проблем передачи информации), 1976. – С. 141 – 151.
34. Кирьянов Б.Ф., Ильеня А.М. Моделирование случайных векторов с произвольным распределением координат // Вестник Новгородского гос. ун-та. Вып. 17. 2001. С. 59 – 60.
35. Кирьянов Б.Ф., Кирьянов Д.В. Простой способ генерирования локальных баз данных и технология обращения к ним // Вестник Новгородского гос. ун-та им. Ярослава Мудрого. Сер. Технич. науки. 2010. Вып. 60. С. 41 – 42.
36. Кирьянов Б.Ф., Кирьянов Д.В. Модель системы связи с высоконадёжной защитой информации в каналах её передачи // Сб. трудов региональной научно-технич. конф. “Информационные технологии и их приложения”. Казань: КГТУ им А.Н. Туполева. 2011. С. 207 – 211.
37. Кирьянов Б.Ф., Кирьянов Д.В. Моделирование системы связи с высоконадёжной защитой информации в каналах её передачи // Вестник Новгородского гос. ун-та им. Ярослава Мудрого. 2011. Вып. 65. С. 73 – 75.
38. Кирьянов Б.Ф., Кознов А.В. Процессы авторегрессии со случайными коэффициентами и их применение при моделировании

радиотехнических систем // Прикладная математика. Межвузовский сборник, вып. 1. Новгород: НовГУ. 1994. С. 3 – 8.

39. Кирьянов Б.Ф., Леонтьев А.Г. Физическое моделирование каналов связи // Повышение верности передачи цифровой информации по каналам связи. М.: “Наука”. 1974. С. 141 – 151.
40. Кирьянов Б.Ф., Мансуров Р.М. Генератор псевдослучайных чисел с многоразрядным сдвигом / Казань, КАИ, 1978. 7 с. Деп. в ЦНИИТЭИ приборостроения 05.05.78, № 923.
41. Кирьянов Б.Ф., Песошин В.А., Гришкин С.Г. К проблеме формирования некоррелированных последовательностей чисел на основе М-последовательностей // Академия наук Латвийской ССР. Автоматика и вычислительная техника. 1984. № 4. С. 70 – 75.
42. Кирьянов Б.Ф., Токмачев М.С. Математические модели в здравоохранении. Великий Новгород: НовГУ. 2009. 279 с.
43. Кирьянов Б.Ф., Dörfel G. Моделирование редких событий // Математическое моделирование и цифровая обработка изображений. Новгород: НПИ. 1991. С. 13 – 16.
44. Кнут Д.Е. Искусство программирования на ЭВМ. Т. 2. Получисленные алгоритмы. М.: Мир. 1977. 725 с.
45. Кропачева Н.Ю., Тихомиров А.С. Моделирование случайных величин. Вел. Новгород: НовГУ, 2004. – 47 с.
46. Культин Никита. Основы программирования в Delphi 2010. СПб.: БХВ-Петербург. 2010. 448 с.
47. Лифшиц А.Л., Мальц Э.А. Статистическое моделирование систем массового обслуживания. М.: «Советское радио», 1978. 248 с.
48. Лоу А.М., Кельтон В.Д. Имитационное моделирование. СПб: Питер, 2004. 847 с.// Пер. с англ. Под редакцией д.т.н. проф. В.Н. Томашевского.
49. Лукашин Ю.П. Адаптивные методы краткосрочного прогнозирования временных рядов. М.: Финансы и статистика. 2003. 416 с.
50. Павловский Ю.Н. Имитационные модели и системы. М.: ФАЗИС, ВЦ РАН, 2000. 144 с.
51. Песошин В.А., Кузнецов В.М. Генераторы псевдослучайных двоичных последовательностей // Вычислительные и управляющие системы летательных аппаратов. Казань: КАИ им. А.Н. Туполева. 1983. С. 51 – 58.
52. Песошин В.А., Кузнецов В.М. Генераторы псевдослучайных чисел на регистрах сдвига: Монография. Казань: Казан – изд. КГТУ. 2007. 294 с.

53. Песошин В.А., Мансуров Р.М., Кузнецов В.М. Комбинированный генератор случайных чисел // Вероятностные методы и кибернетика. Казань: КГУ. 1983. Вып. 19. С. 88 -99.
54. Петухов О.А., Морозов А.В., Петухова Е.О. Моделирование: системное, имитационное, аналитическое: Учебное пособие. СПб: СЗТУ. 2008. 288
55. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки. М.: Мир. 1976.
56. Под общей редакцией Ю.Д. Максимова. Вероятностные разделы математики. СПб: “Иван Федоров”. 2001. 589 с.
57. Под редакцией В.А. Райхлина. Эволюционное моделирование. Казань: ФЭН. 2004. Вып. 2. 304 с.
58. Полляк Ю.Г. Вероятностное моделирование на ЭВМ. М.: “Советское радио”. 1971. 400 с.
59. Райхлин В.А. Конструктивное моделирование систем. Казань: Фэн (Наука), 2005. 304 с.
60. Сави Г.И. Системное моделирование сложных процессов. М.: ФАЗИС, ВЦ РАН, 2000. 288 с.
61. Самарский А.А., Михайлов А.М. Математическое моделирование: Идеи. Методы. Примеры. (2). М. Физматлит. 2002. 350 с.
62. Советов Б.Я., Яковлев С.А. Моделирование систем. М.: Высшая школа. 1985. 271 с.
63. Строгалев В.П., Толкачёва И.О. Имитационное моделирование: Учебное пособие. М.: МГТУ им. Н.Э. Баумана, 2008. 280 с.
64. Токмачёв М.С. Временные ряды и прогнозирование. Великий Новгород: НовГУ. 2005. 193 с.
65. Тюрин Ю.Н., Макаров А.А. Статистический анализ данных на компьютере. М.: ИНФРА. 1998. 528 с.
66. Хомоненко А.Д., Гофман В.Э. Delphi. 2-е издание. СПб.: БХВ-Петербург. 2008. 576 с.
67. Элспас Б. Теория автономных линейных последовательных сетей. М.: ИЛ. Кибернетический сборник. 1963. Вып. 7. С. 90 – 128.
68. Anderson, T.W.: The Statistical Analysis of Time Series. John Wiley, New York (1994).
69. ASM Transactions on Modeling and Computer Simulations [January 1998, Vol. 8, № 1; October 2003, Vol. 13, № 4].
70. Carin, M.C. and B.L. Nelson: Autoregressive to Anything: Time-Series Input Processes for Simulation, *Operations Res. Letters, 19:* 51 – 58 (1996).

71. Chatfield C. The Analysis of Time Series: an Introduction. 4th ed. Chapmam and Hall. 1989. 242 p.
72. Dörfel G. Pseudostochastische Impulsgeneratoren – Strukturen und Anwendungen / Experimentelle Technik der Physik, 1987, Heft 1. S. 1 – 12.
73. Dyadkin I.G., Hamilton K.G. A study of 128-bit multipliers for congruential pseudorandom numbers generators / Computer Physics Communications. 2000. Vol. 125. P. 239 – 258.
74. Grange C.W.J., Newbold P. Forecasting Economic Time Series // Academic Press, Inc. 1986/ 338 p.
75. Kiryanov B.F., Koznov A.V. The modeling of random sequences with given normed autocovariation function and univariate distribution / International workshop on mathematical methods and tools in computer simulation. Preprint MM-94-01. St.-Petersburg, 1994, p. 10 – 11.
76. Kleijnen. J.P.C.: Regression Metamodels for Simulation with Common Random Numbers: Comparison of Validation Tests and Confidence Intervals, *Manugement Sci., 38:* p. 1164 – 1185 (1992).
77. Mikhailov G.A., Marchenko M.A. Parallel realization of statistical simulation and random number generators / Anal. and Math. Modell. 2002. Vol. 17. № 1. P. 113 – 124.
78. Niederreiter H. Random Number Generation and Quasi-Monte Carlo Methods. Philadelphia: SIAM, 1992.