МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

Е.В. Попова, Д.А. Замотайлова

КУРС ЛЕКЦИЙ
по дисциплине

Исследование и адаптация математических моделей и вычислительных методов

Краснодар, 2014

Лекция 1
Вычислительный эксперимент и математическое моделирование

Технический цикл вычислительного эксперимента можно условно разбить на несколько этапов:

- выбор физического приближения и математическая формулировка задачи (построение математической модели изучаемого явления или объекта);
- разработка вычислительного алгоритма решения задачи;
- реализация алгоритма в виде программы для ЭВМ;
- проведение расчетов на ЭВМ;
- обработка, анализ и интерпретация результатов расчетов, сопоставление с физическим экспериментом и, в случае необходимости, уточнение или пересмотр математической модели, то есть возвращение к первому этапу и повторение цикла вычислительного эксперимента.

Следует еще раз подчеркнуть, что деление вычислительного эксперимента на указанные пять этапов имеет в значительной мере условный характер. На самом деле все эти основные этапы тесно связаны между собой и служат одной цели - получению с необходимой точностью за возможно меньшее машинное время адекватного количественного описания изучаемого физического явления или процесса.

Сама структура вычислительного эксперимента показывает, что это сложный научно-производственный процесс, в котором участвует большой коллектив специалистов различного профиля.

Успех дела зависит от согласованного взаимодействия всех участников вычислительного эксперимента и, в частности, от умения находить компромиссные решения вопросов в областях, где перекрещиваются интересы различных специалистов.

Изучение математической модели сначала проводится обычным средствами. Прежде всего, исследуется вопрос о постановке задачи. Задача должна быть поставлена математически грамотно. Нужно убедиться в том, что существует единственное решение, выяснить характер его зависимости от входных данных (корректна или некорректна задача), для того чтобы определить возможность и метод работы с этой модель.

Для предварительного исследования модели вначале используются все традиционные методы: качественный размерностный анализ, поиск частных решений для специальных случаев, рассмотрение предельных случаев. Таким образом, добывается первичная (быть может, грубая) информация о качественном характере явления.

Полученные на этом этапе точные решения необходимы, кроме того, как тесты для проверки качества вычислительных алгоритмов, которые будут строиться для решения полной задачи. Математическая модель включает помимо основных уравнений некоторые дополнительные соотношения,

описывающие конкретные свойства и являющиеся фактически коэффициентами уравнений.

Для анализа сформулированной математической модели с помощью ЭВМ необходимы экономичные вычислительные алгоритмы, позволяющие получать решение задачи за допустимое (по возможности минимальное) время. Такое требование приобретает особую важность в связи с многовариантным характером вычислительного эксперимента.

Если существует аналитическое решение задачи, то зависимость от параметров выступает в явном виде. В вычислительном эксперименте необходимо проводить большие серии однотипных расчетов для изучения влияния различных параметров задачи. Поэтому необходимое условие вычислительного эксперимента - экономичность лежащего в его основе алгоритма. Конструирование вычислительного алгоритма подразумевает два этапа: построение разностной схемы для математической модели, то есть аппроксимацию исходной системы дифференциальных уравнений системой разностных сеточных (алгебраических) уравнений, и построение метода для быстрого решения полученных разностных сеточных уравнений.

Построение разностной схемы можно рассматривать как замену непрерывной среды некоторым ее дискретным аналогом. При этом возникают новые параметры - шаги разностной сетки (по времени и пространству), вводимой для замены области непрерывного изменения аргументов, в которой ведется поиск решения исходной задачи, множеством точек, (узлов) сетки. Представляется естественным желание использовать грубые сетки с большими шагами (с небольшим числом узлов), так как машинное время, необходимое для решения разностных уравнений, возрастает с увеличением числа узлов (с уменьшением шага сетки). Однако при неограниченном уменьшении шагов сетки разностная схема близка к исходной дифференциальной задаче лишь асимптотически.

При конечных же шагах сетки разностные уравнения, представляющие собой законы, в соответствии с которыми происходит эволюция «дискретной среды», могут заметно отличаться от дифференциальных уравнений, описывающих поведение непрерывной среды. Возможно возникновение различных, нежелательных эффектов разностного происхождения.

В настоящее время сложилась новая методология научных исследований - математическое моделирование и вычислительный эксперимент. Сущность этой методологии состоит в замене исходного объекта его математической моделью и исследовании современными вычислительными средствами математических моделей. Методология математического моделирования бурно развивается, охватывая все новые сферы - от разработки больших технических систем и управления ими до анализа сложнейших экономических и социальных процессов.

Широкое применение математических методов позволяет поднять общий уровень теоретических исследований, дает возможность проводить их в более тесной связи с экспериментальными исследованиями.

Математическое моделирование может рассматриваться как новый метод познания, который сочетает в себе многие достоинства как теории, так и эксперимента. Работа не с самим объектом (явлением, процессом), а с его моделью дает возможность безболезненно, относительно быстро и без существенных затрат исследовать его свойства и поведение в любых мыслимых ситуациях (преимущества теории). В то же время вычислительные (компьютерные, имитационные) эксперименты с моделями объектов позволяют подробно и глубоко изучать объекты в достаточной полноте, недоступной чисто теоретическим подходам (преимущества эксперимента).

Технические, экологические, экономические и иные системы, изучаемые современной наукой, больше не поддаются исследованию обычными теоретическими методами. Прямой натурный эксперимент над ними долог, дорог, часто либо опасен, либо попросту невозможен. Вычислительный эксперимент позволяет провести исследование быстрее и дешевле. Без применения этой методологии в развитых странах не реализуется ни один крупномасштабный технологический, экологический или экономический проект.

Рождение и становление методологии математического моделирования пришлось на конец 40 - начало 50-х г. XX века и было обусловлено двумя причинами. Первым, но не основным, побудительным мотивом послужило появление компьютеров, которые избавили исследователей от огромной по объему рутинной вычислительной работы.

Второй, более важной, причиной явился беспрецедентный социальный заказ - выполнение национальных программ СССР и США по созданию ракетно-ядерного щита. Эти сложнейшие научно-технические проблемы не могли быть реализованы традиционными методами без широкого использования вычислительных средств. Ядерные взрывы и полеты ракет и спутников были промоделированы сначала на компьютерах и лишь затем претворены на практике.

Основу математического моделирования составляет триада: модель - алгоритм - программа. Математические модели реальных исследуемых процессов сложны и включают системы нелинейных функционально-дифференциальных уравнений. Ядро математической модели составляют уравнения с частными производными.

На первом этапе вычислительного эксперимента выбирается (или строится) модель исследуемого объекта, отражающая в математической форме важнейшие его свойства - законы, которым он подчиняется, связи, присущие составляющим его частям, и т. д. Математическая модель (ее основные фрагменты) исследуется традиционными аналитическими средствами прикладной математики для получения предварительных знаний об объекте.

Второй этап связан с выбором (или разработкой) вычислительного алгоритма для реализации модели на компьютере. Необходимо получить искомые величины с заданной точностью на имеющейся вычислительной технике. Вычислительные алгоритмы должны не искажать основные

свойства модели и, следовательно, исходного объекта, они должны быть адаптирующимися к особенностям решаемых задач и используемых вычислительных средств. Изучение математических моделей проводится методами вычислительной математики, основу которых составляют численные методы решения задач математической физики - краевых задач для уравнений с частными производными.

На третьем этапе создается программное обеспечение для реализации модели и алгоритма на компьютере. Программный продукт должен учитывать важнейшую специфику математического моделирования, связанную с использованием ряда (иерархии) математических моделей, многовариантностью расчетов. Это подразумевает широкое использование комплексов и пакетов прикладных программ, разрабатываемых, в частности, на основе объектно-ориентированного программирования.

Успех математического моделирования определяется одинаково глубокой проработкой всех основных звеньев вычислительного эксперимента. Опираясь на триаду модель - алгоритм - программа, исследователь получает в руки универсальный, гибкий и недорогой инструмент, который вначале отлаживается, тестируется и калибруется на решении содержательного набора пробных задач. После этого проводится широкомасштабное исследование математической модели для получения необходимых качественных и количественных свойств и характеристик исследуемого объекта.

Вычислительный эксперимент по своей природе носит междисциплинарный характер. В совместных исследованиях участвуют специалисты в прикладной области, прикладной и вычислительной математике, по прикладному и системному программному обеспечению.

Вычислительный эксперимент проводится с опорой на широкое использование самых разных методов и подходов - от качественного анализа нелинейных математических моделей до современных языков программирования.

Решение проблем жизнеобеспечения на современном этапе основывается на широком использовании математического моделирования и вычислительного эксперимента. Вычислительные средства (компьютеры и численные методы) традиционно хорошо представлены в естественно - научных исследованиях, прежде всего в физике и механике. Идет активный процесс математизации химии и биологии, наук о земле, гуманитарных наук и т.д.

Современные информационные технологии используются в медицине. Сбор и анализ диагностических данных позволяет провести своевременную диагностику заболеваний. Например, компьютерный томограф является примером того, как использование математических методов обработки больших массивов данных позволило получить качественно новый медицинский инструментарий.

При математизации научных знаний выделяется этап абстрагирования от конкретной природы явления, идеализации и выделения его математической формы (строится математическая модель).

Вторым этапом математизации является исследование математических моделей как математических (абстрактных) объектов.

Третий этап применения математики в прикладных исследованиях характеризуется интерпретацией - приданием конкретного прикладного содержания математическим абстракциям. Специалист по прикладному математическому моделированию, работая бок о бок со специалистами в прикладной области, всегда за математическими абстракциями видит конкретное прикладное содержание.

Эвристическая роль математического моделирования проявляется в том, что вместо натурного эксперимента проводится математический эксперимент. Вместо исследования проявления того или иного воздействия на исследуемый объект используется параметрическое изучение математической модели, устанавливается зависимость решения от того или иного параметра. Такой эксперимент, дополняя натурный, позволяет значительно глубже исследовать явление или процесс.

Вычислительные средства, под которыми мы понимаем компьютеры и вычислительные методы, позволили решить с приемлемой точностью и за разумное время задачи, которые ранее были недоступны для исследования, дали возможность реализовать крупнейшие научно-технические проекты.

Сама математическая модель может быть достаточно сложной, нелинейной. Это зачастую делает невозможным ее качественное исследование традиционными методами прикладной математики. Именно поэтому в громадном большинстве случаев проводится качественное исследование на более простых, но обязательно содержательных по отношению к исходной математической модели задачах. В этом случае мы должны говорить о модельных (упрощенных) задачах для основной математической модели (моделей для модели).

Большое внимание при качественном исследовании математических моделей (или модельных задач для них) уделяется вопросам корректности.

Прежде всего рассматривается проблема существования решения. Соответствующие строгие результаты (теоремы существования) дают уверенность в корректности математической модели. Кроме того, конструктивные доказательства теорем существования могут быть положены в основу приближенных методов решения поставленной задачи.

При прикладном математическом моделировании важным является вопрос об устойчивости решения относительно малых возмущений входных данных. Неустойчивость (неограниченный рост решения при малых возмущениях) наиболее характерна для обратных задач и должна учитываться при построении приближенного решения.

Для нелинейных математических моделей может быть характерна множественность, неединственность решения. При качественном

исследовании математических моделей изучаются точки ветвления, бифуркации решения, вопросы выделения нужного искомого решения и многое другое.

Методы качественного исследования для различных типов математических моделей разработаны с неодинаковой полнотой. Среди моделей, где качественные методы принесли наиболее впечатляющие результаты, отметим обыкновенные дифференциальные уравнения. В теории уравнений с частными производными качественные методы также используются, хотя и не в такой большой степени.

Точное или приближенное решение находится с использованием аналитических и численных методов. В этой связи среди классических примеров аналитических методов отметим методы разделения переменных, интегральных преобразований для линейных задач математической физики.

Для нелинейных математических моделей особое значение имеют методы линеаризации, различные варианты методов возмущений. Теория возмущений базируется на использовании асимптотических разложений по выделенному малому параметру.

Сложные нелинейные многопараметрические модели могут быть исследованы на компьютере численными методами. В отличие от аналитического решения, которое может давать явную параметрическую зависимость решения от тех или иных условий задачи, при численном решении требуется многократное решение задачи при изменении того или

иного параметра. Но ведь численное решение может быть получено и для тех задач, для которых аналитического решения нет.

Применять компьютеры можно и на этапе качественного исследования математической модели, этапе отыскания аналитических решений модельных задач. Например, компьютер можно использовать для нахождения автомодельных решений. При выделении автомодельной переменной исходная задача для уравнения в частных производных сводится, например, к обыкновенному дифференциальному уравнению, происходит понижение размерности. Общее решение последнего находится на основе использования систем аналитических вычислений на компьютере (методов вычислительной алгебры), широко представленных в современных математических пакетах.

В применении компьютеров при математическом моделировании можно выделить, по крайней мере, два этапа, два уровня. Первый из них характеризуется исследованием достаточно простых математических моделей. На этом этапе (уровне) применения компьютеров вычислительные средства используются наряду и наравне с другими методами (чисто математическими) прикладной математики. Для этого уровня применения компьютеров в прикладном математическом моделировании характерен лозунг Р. Хеминга: "Цель расчетов - понимание, а не числа".

Второй этап (уровень) применения компьютеров характеризуется исследованием сложных нелинейных математических моделей. В этих

условиях вычислительные средства становятся основными, абсолютно преобладающими.

Традиционные средства прикладного математического моделирования выполняют вспомогательную, обслуживающую роль (качественное исследование задачи в сильно упрощенных постановках - модельные задачи, тестирование вычислительных алгоритмов и т.д.).

Именно возможность исследования сложных математических моделей на основе численных методов и компьютеров позволяет с новых позиций рассмотреть методологию научных исследований. Мощные компьютеры, высокоэффективные вычислительные алгоритмы, современное программное обеспечение позволяют в настоящее время организовать научные исследования в рамках единой технологии вычислительного эксперимента, который включает в себя теоретические и экспериментальные исследования.

В широком (методологическом) смысле под вычислительным экспериментом мы понимаем новую технологию научных исследований.

Для исследуемого объекта сначала строится математическая модель.

Она базируется на известных фундаментальных моделях. Вычислительный эксперимент, по своей сути, предусматривает исследование группы близких моделей. Вначале строится простая, но достаточно содержательная и полная с точки зрения описания исследуемых процессов и близости к экспериментальным данным модель.

В процессе проведения вычислительного эксперимента, на его последующих циклах модель уточняется, учитываются новые факторы и т.д. Поэтому мы всегда можем говорить (более того, должны говорить) о наборе, упорядоченном наборе (об иерархии) математических моделей, каждая из которых с той или иной точностью описывает действительность.

И в рамках наиболее простой модели необходимо добиваться согласия с экспериментом. Это и является, в конце концов, целью вычислительного эксперимента.

Суть вычислительного эксперимента, его содержательное зерно состоит в исследовании на компьютере математических моделей численными методами.

Основное содержание предварительного исследования математической модели состоит в выделении более простых (модельных) задач и их всестороннем исследовании, так как полная математическая модель слишком сложна. Модельные математические задачи в цикле вычислительного эксперимента строятся для двух различных целей: во-первых, для качественного исследования полной задачи (а опосредованно и исследуемого объекта), во-вторых - для проверки, тестирования вычислительных алгоритмов приближенного решения полной задачи.

Программное обеспечение вычислительного эксперимента базируется на использовании комплексов и пакетов прикладных программ.

Комплекс программ предназначен для решения близких по своей математической природе задач из одной предметной области. Он включает в

себя библиотеку программных модулей (в большой или меньшей степени независимых), из которых комплектуются рабочие программы. В комплексах прикладных программ сборка программ из модулей осуществляется вручную. Затем в цикле вычислительного эксперимента проводится серия расчетов на компьютерах при изменении тех или иных параметров задачи.

Полученные данные анализируются и интерпретируются с участием специалистов в прикладной области. Обработка результатов проводится с учетом имеющихся теоретических представлений и экспериментальных данных. Она осуществляется во многом в традициях классического натурного эксперимента. Сами опытные данные представляются в виде таблиц, графиков, фотографий с дисплея, кинофильмов и т.д.

Надо только всегда иметь в виду, что объем обрабатываемой информации, детализация полученных результатов в вычислительном эксперименте несравненно больше. В вычислительном эксперименте проблемы хранения и обработки информации имеют все возрастающее значение.

На этапе анализа результатов становиться ясным, удачно ли выбрана математическая модель, ее вычислительная реализация. Если есть необходимость, модели и численные методы уточняются, и весь цикл вычислительного эксперимента повторяется, то есть совершается новый виток спирали в познании истины.

Чрезвычайно важно отметить универсальность вычислительного эксперимента, которая позволяет легко переносить эту технологию на исследование других объектов. Это обстоятельство порождено тем, что многие явления и процессы имеют одни и те же математические модели.

Второй особенностью вычислительного эксперимента как технологии научных исследований является его междисциплинарный характер. Мы постоянно подчеркиваем это обстоятельство, говоря о том, что прикладной математик объединил теоретика и экспериментатора для более быстрого достижения общей цели.

Можно отметить следующие отличительные особенности и преимущества вычислительного эксперимента перед натурным экспериментом.

Во-первых, вычислительный эксперимент проводится даже тогда, когда натурный эксперимент невозможен. Такая ситуация имеет место с крупномасштабными экологическими экспериментами. Отметим в этой связи моделирование глобальных климатических изменений при использовании атомного оружия. Другой пример - исследование процессов при термоядерных параметрах (кроме взрыва атомной бомбы пока нет других возможностей достичь их).

Во-вторых, при использовании вычислительного эксперимента резко снижается стоимость разработок и экономится время. Это обеспечивается многовариантностью выполняемых расчетов, простотой модификации математических моделей для имитации тех или иных реальных условий.

В традициях экспериментального исследования мы воздействуем на математическую модель и обрабатываем результаты (вот почему мы говорим об эксперименте, хотя и вычислительном). И лишь изредка мы контролируем точность своего "прибора", сравнивая его с эталоном. В традициях теоретического исследования в вычислительном эксперименте мы имеем дело с математической моделью, а не с самим объектом.

Математическое моделирование традиционно развивается в недрах фундаментальных наук: механике и физике, для которых отмечается наивысший уровень теоретических исследований (другими словами, уровень математизации).

Значительно менее совершенен математический арсенал инженера и технолога. В современных условиях необходимо обеспечить повсеместное непосредственное внедрение математических методов в науку и технологию. Математическое моделирование технологических процессов сулит огромную выгоду, переход на новый качественный уровень самой технологии. Наиболее благодатное поле для приложения методов математического моделирования и вычислительного эксперимента - техника и промышленность, технология. Особое внимание заслуживают отрасли, определяющие научно-технический прогресс сегодня, и прежде всего микроэлектроника и нанотехнологии.

Отметим еще один аспект в применении вычислительного эксперимента. В настоящее время мировая общественность совершенно справедливо обеспокоена экологическими последствиями крупномасштабных проектов, обеспечением безопасности функционирования работающих установок и проектируемых объектов.

Вычислительный эксперимент на базе адекватных моделей позволяет испытать модель экологически опасного объекта в мыслимых и немыслимых условиях, дать практические рекомендации обеспечения условий безопасной работы, дать, если хотите, гарантии такой работы.

.

Лекция 2
Вычислительный эксперимент – новая технология научных исследований.

В широком (методологическом) смысле под вычислительным экспериментом мы понимаем новую технологию научных исследований.

Для исследуемого объекта сначала строится математическая модель. Она базируется на известных фундаментальных моделях. Вычислительный эксперимент, по своей сути, предусматривает исследование группы близких моделей. В начале строится простая, но достаточно содержательная и полная с точки зрения описания исследуемых процессов и близости к экспериментальным данным модель.

В процессе проведения вычислительного эксперимента, на его последующих циклах модель уточняется, учитываются новые факторы и т.д. Поэтому мы всегда можем говорить (более того, должны говорить) о наборе, упорядоченном наборе (об иерархии) математических моделей, каждая из которых с той или иной точностью описывает действительность.

И в рамках наиболее простой модели необходимо добиваться согласия с экспериментом. Это и является, в конце концов, целью вычислительного эксперимента.

Суть вычислительного эксперимента, его содержательное зерно состоит в исследовании на компьютере математических моделей численными методами.

Основное содержание предварительного исследования математической модели состоит в выделении более простых (модельных) задач и их всестороннем исследовании, так как полная математическая модель слишком сложна. Модельные математические задачи в цикле вычислительного эксперимента строятся для двух различных целей: во-первых, для качественного исследования полной задачи (а опосредованно и исследуемого объекта), во-вторых - для проверки, тестирования вычислительных алгоритмов приближенного решения полной задачи.

Программное обеспечение вычислительного эксперимента базируется на использовании комплексов и пакетов прикладных программ. Комплекс программ предназначен для решения близких по своей математической природе задач из одной предметной области. Он включает в себя библиотеку программных модулей (в большой или меньшей степени независимых), из которых комплектуются рабочие программы. В комплексах прикладных программ сборка программ из модулей осуществляется вручную.

Затем в цикле вычислительного эксперимента проводится серия расчетов на компьютерах при изменении тех или иных параметров задачи.

Полученные данные анализируются и интерпретируются с участием специалистов в прикладной области. Обработка результатов проводится с учетом имеющихся теоретических представлений и экспериментальных данных. Она осуществляется во многом в традициях классического натурного эксперимента. Сами опытные данные представляются в виде

таблиц, графиков, фотографий с дисплея, кинофильмов и т.д. Надо только всегда иметь в виду, что объем обрабатываемой информации, детализация полученных результатов в вычислительном эксперименте несравненно больше. В вычислительном эксперименте проблемы хранения и обработки информации имеют все возрастающее значение.

На этапе анализа результатов становиться ясным, удачно ли выбрана математическая модель, ее вычислительная реализация. Если есть необходимость, модели и численные методы уточняются, и весь цикл вычислительного эксперимента повторяется, то есть совершается новый виток спирали в познании истины.

Чрезвычайно важно отметить универсальность вычислительного эксперимента, которая позволяет легко переносить эту технологию на исследование других объектов. Это обстоятельство порождено тем, что многие явления и процессы имеют одни и те же математические модели.

Второй особенностью вычислительного эксперимента как технологии научных исследований является его междисциплинарный характер. Мы постоянно подчеркиваем это обстоятельство, говоря о том, что прикладной математик объединил теоретика и экспериментатора для более быстрого достижения общей цели.

Можно отметить следующие отличительные особенности и преимущества вычислительного эксперимента перед натурным экспериментом.

Во-первых, вычислительный эксперимент проводится даже тогда, когда натурный эксперимент невозможен. Такая ситуация имеет место с крупномасштабными экологическими экспериментами. Отметим в этой связи моделирование глобальных климатических изменений при использовании атомного оружия. Другой пример - исследование процессов при термоядерных параметрах (кроме взрыва атомной бомбы пока нет других возможностей достичь их).

Во-вторых, при использовании вычислительного эксперимента резко снижается стоимость разработок и экономится время. Это обеспечивается многовариантностью выполняемых расчетов, простотой модификации математических моделей для имитации тех или иных реальных условий. В традициях экспериментального исследования мы воздействуем на математическую модель и обрабатываем результаты (вот почему мы говорим об эксперименте, хотя и вычислительном). И лишь изредка мы контролируем точность своего "прибора", сравнивая его с эталоном. В традициях теоретического исследования в вычислительном эксперименте мы имеем дело с математической моделью, а не с самим объектом.

Математическое моделирование традиционно развивается в недрах фундаментальных наук: механике и физике, для которых отмечается наивысший уровень теоретических исследований (другими словами, уровень математизации).

Значительно менее совершенен математический арсенал инженера и технолога. В современных условиях необходимо обеспечить повсеместное непосредственное внедрение математических методов в науку и технологию. Математическое моделирование технологических процессов сулит огромную выгоду, переход на новый качественный уровень самой технологии. Наиболее благодатное поле для приложения методов математического моделирования и вычислительного эксперимента - техника и промышленность, технология. Особое внимание заслуживают отрасли, определяющие научно-технический прогресс сегодня, и прежде всего микроэлектроника и нанотехнологии.

Отметим еще один аспект в применении вычислительного эксперимента. В настоящее время мировая общественность совершенно справедливо обеспокоена экологическими последствиями крупномасштабных проектов, обеспечением безопасности функционирования работающих установок и проектируемых объектов. Вычислительный эксперимент на базе адекватных моделей позволяет испытать модель экологически опасного объекта в мыслимых и немыслимых условиях, дать практические рекомендации обеспечения условий безопасной работы, дать, если хотите, гарантии такой работы.

При исследовании нового процесса или явления обычный подход связан с построением математической модели и проведением расчетов при изменении параметров задачи. В этом случае мы имеем поисковый вычислительный эксперимент.

В результате проведения поискового вычислительного эксперимента дается описание наблюдаемым явлениям, прогнозируется поведение исследуемого объекта в тех или иных условиях, возможно и не достижимых в реальных условиях. Такой тип вычислительного эксперимента характерен при проведении теоретических исследований в фундаментальных науках.

С другой стороны, при математическом моделировании

технологических процессов в качестве основного может быть выбран оптимизационный вычислительный эксперимент. Для него характерно решение задачи оптимизации по уменьшению затрат, облегчению конструкции и т.д. Для сформулированной математической модели ставится соответствующая задача оптимального управления, задача оптимизации.

При обработке данных натурных экспериментов используется диагностический вычислительный эксперимент. По дополнительным косвенным измерениям делается вывод о внутренних связях явления или процесса. В условиях, когда структура математической модели исследуемого процесса известна, ставится задача идентификации модели, например, определяются коэффициенты уравнений. Диагностическому вычислительному эксперименту обычно ставится в соответствие обратная задача математической физики.

Часто приходится сталкиваться с положением, когда математической модели исследуемого процесса или явления нет и создать ее не представляется возможным. Такая ситуация характерна, в частности, при

обработке данных натурного эксперимента. Тогда обработка проводится в режиме «черного ящика» и мы имеем дело с аппроксимационными моделями. При отсутствии математических моделей на основе широкого использования компьютеров проводится имитационное моделирование.

Лекция 3
Моделирование случайных событий и величин.

Моделирование случайных событий

Моделирования простого события

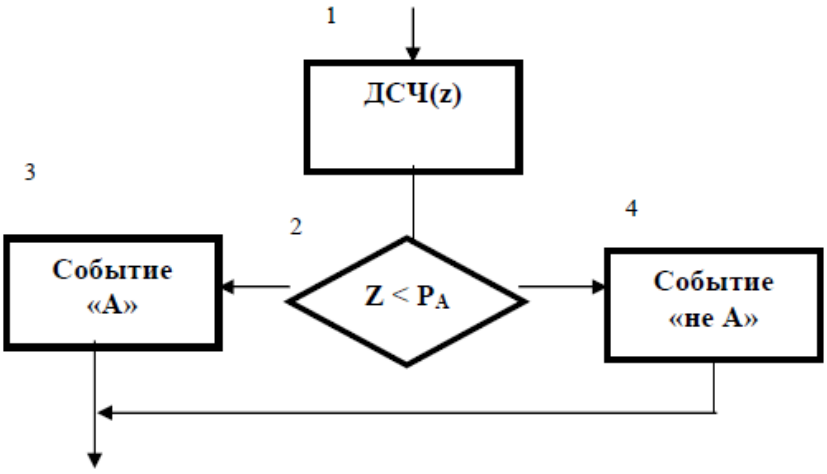
Пусть имеется событие A, вероятность наступления которого равна P_A. Требуется выработать правило, при многократном использовании которого частота появления события стремилась бы к его вероятности. Выберем с помощью датчика случайных чисел, равномерно распределенных в интервале (0,1), некоторое число z и определим вероятность того, что

$$P(z < P_A) = \int_0^{P_A} f(x)dx = P_A.$$

$z < P_A$. Для случайной величины z с равномерным распределением справедлива следующая зависимость:

Таким образом, вероятность попадания случайной величины в интервал $(0,P_A)$ равна величине P_A. Поэтому если при розыгрыше число z попало в этот интервал, то следует считать, что событие A произошло. Противоположное событие (**не** A) произойдет с вероятностью $(1 - P_A)$ в том случае, если $z >= P_A$.

Процедура моделирования простого события в имитационной модели описывается алгоритмом, схема которого показана на рис. 3.3.



Моделирование простого события

Оператор 1 обращается к датчику случайных чисел, генерирующему случайную величину z. Оператор 2 проверяет условие $z < P_A$. Если оно выполняется, считается, что произошло событие A. В противном случае считается, что произошло противоположное событие (не A).

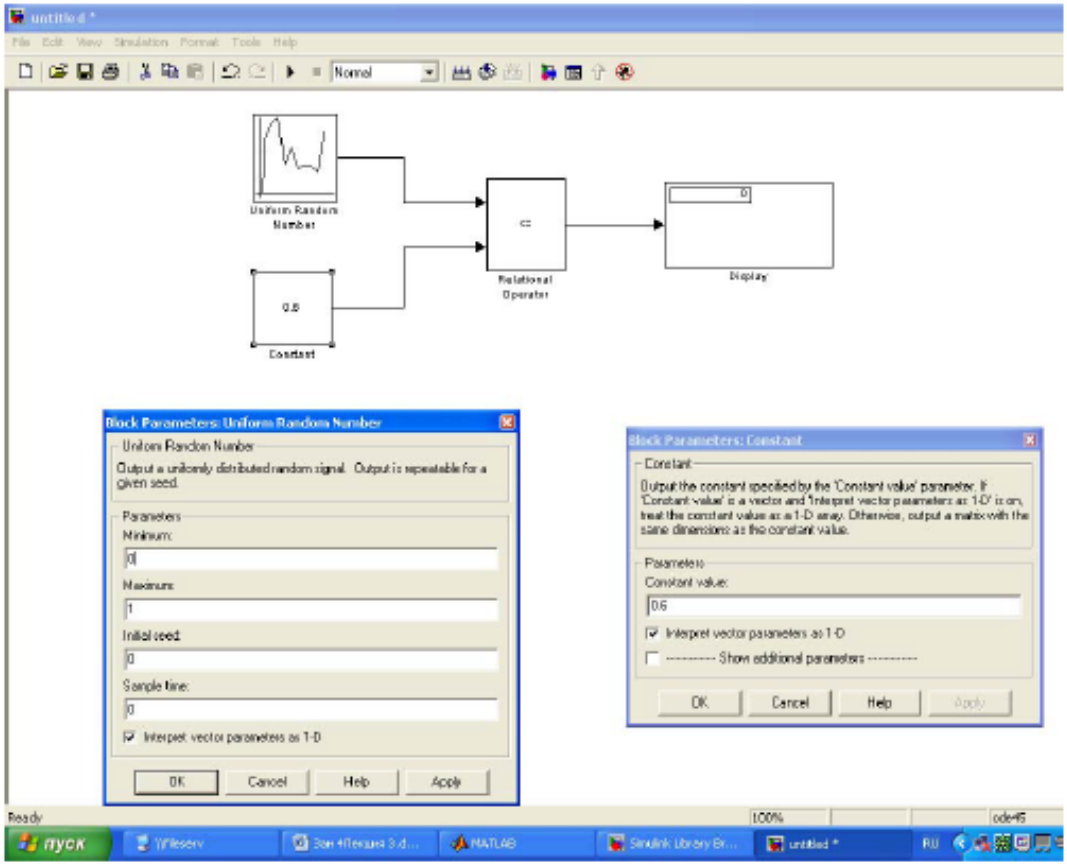
На рис. 3.4 показан пример реализации модели случайного события с помощью Simulink.

Блок Uniform Random Number генерирует случайные числа (СЧ), равномерно распределенные на интервале [0;1]. Для этого выполнены следующие исходные установки его параметров:

- Minimum (нижняя граница диапазона): 0;
- Maximum (верхняя граница диапазона) : 1.

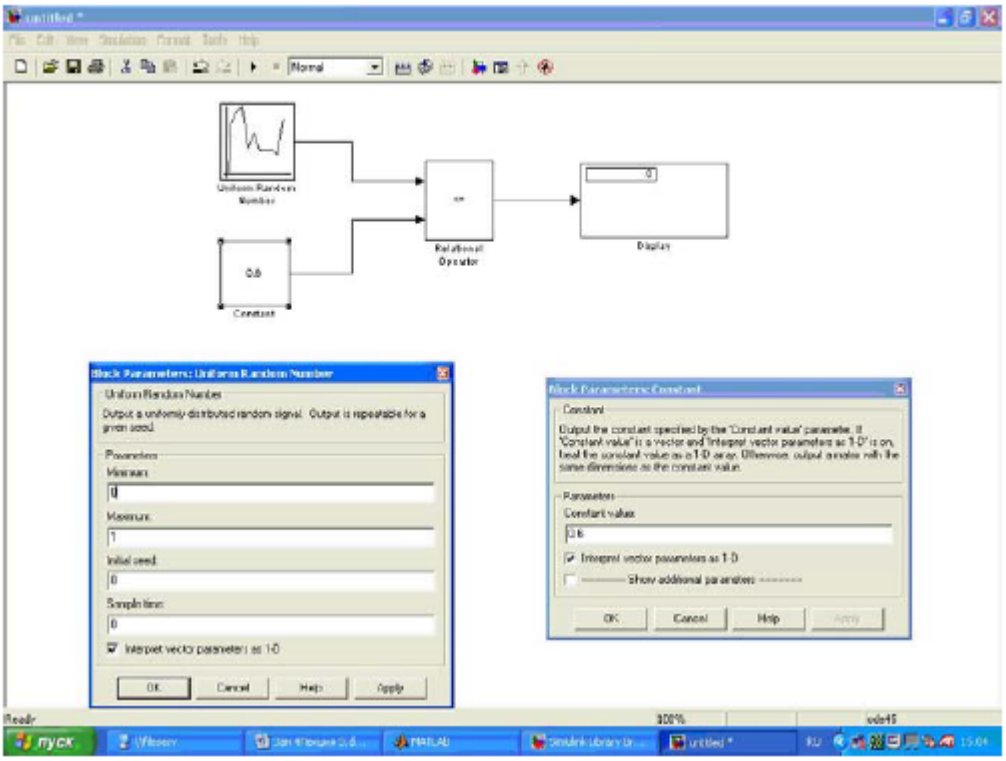
Третий параметр – Sample time – в данном случае позволяет задавать количество СЧ, которые будут сформированы

блоком в течение интервала моделирования: при Sample time, равном 0, новое случайное число генерируется через каждые 0,02 единицы модельного времени; при Sample time, равном интервалу моделирования, генерируется только одно СЧ. На рис. 3.4 приведен пример блок-диаграммы, позволяющей моделировать появление случайного события A при $P_A = 0{,}4$.



Моделирование случайного события с заданной вероятностью наступления

На рис. 3.5 показано моделирование нескольких случайных событий на интервале моделирования 6 единиц модельного времени.



Моделирование ряда случайных событий

Особенностью блока Uniform Random Number является то, что он в каждом сеансе моделирования генерирует одну и ту же последовательность СЧ. Для изменения генерируемой последовательности необходимо вручную изменить значение его параметра Initial seed. При проведении большого числа повторных экспериментов с целью накопления статистических данных это не очень удобно. Поэтому для моделирования случайных событий можно воспользоваться генераторами СЧ, входящими в состав компоненты Matlab, которая называется Toolboxes-Statistics (средства статистического анализа). Данная компонента доступна для использования в том случае, если она включена в рабочую конфигурацию пакета Matlab. Toolboxes-Statistics, как и другие инструментальные приложения, представляет собой набор специализированных функ-

ций (см. рис.), реализованных в виде М-файлов. Ее особенностью является то, что для нее отсутствует набор блоков, который включался бы в библиотеку Simulink. Поэтому в процессе моделирования статистические функции следует использовать один из двух способов:

- выполнять в командном окне Matlab;
- включать в вычисляемое выражение в тех блоках S-модели, в которых разрешено использование М-функций.

Категория функций Random Namber Generation (генераторы случайных чисел) обеспечивает формирование значений случайной величины, распределенной по определенному закону с задаваемыми параметрами. Генератор непрерывной СВ, равномерно распределенной в заданном интервале, называется *unifrnd*. Обращение к данной функции имеет вид *unifrnd* (A, B, M, N), где A, B – границы диапазона распределения, а параметры M, N задают размер генерируемой матрицы случайных чисел. Если параметры M, N опущены, то генерируется единственное значение случайной величины.

При моделировании случайного события функция *unifrnd* может быть указана в качестве параметра настройки следующих блоков:

- Matlab Fcn (раздел Function&Tables);
- Fcn (из того же раздела);
- Constant (раздел Sources).

Напоминание из теории вероятностей

Случайная величина (СВ) – величина, которая в результате опыта может принимать некоторое неизвестное заранее значение.

Дискретная случайная величина (ДСВ) – принимает конечное (счетное) множество возможных значений.

Непрерывная случайная величина (НСВ) – может принимать любые значения из некоторого интервала.

Случайная величина задается **функцией распределения** $F(x) = P(X < x)$. Если $F(x)$ непрерывна и дифференцируема, то непрерывная случайная величина задается **плотностью вероятностей** $f(x)$, которая является производной от $F(x)$.

$$F(x) = \int_{-\infty}^{x} f(x)dx$$

f(x)=F'(x).

Свойства функции распределения

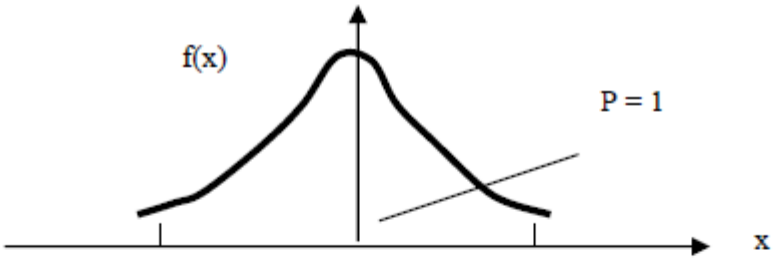
1. При $-\infty < x < +\infty$, $0 =< F(x) =< 1$;
2. Неубывающая, т.е при $x_2 < x_1$ $F(x_2) > F(x_1)$;
3. Имеет место $F(+\infty) = 1$ и $F(-\infty) = 0$;
4. Вероятность попадания СВ (x) в интервал (a,b).

Свойства плотности распределения вероятностей

1. Неотрицательна $f(x) >= 0$

$$P(a \le x \le b) = F(b) - F(a)$$

2. $\int_{-\infty}^{\infty} f(x)dx = F(\infty) - F(-\infty) = 1$



3. Вероятность попадания в интервал (a,b)

$$P(a \le x \le b) = \int_{-\infty}^{\infty} f(x)dx$$

4. Математическое ожидание непрерывной случайной величины

$$M(X) = \int_{-\infty}^{\infty} x \cdot f(x)dx$$

5. Дисперсия непрерывной случайной величины

$$D(X) = M\left[x - M(X)\right]^2 = \int_{-\infty}^{\infty} \left[x - M(X)\right]^2 \cdot f(x)dx$$

6. Среднее квадратическое отклонение

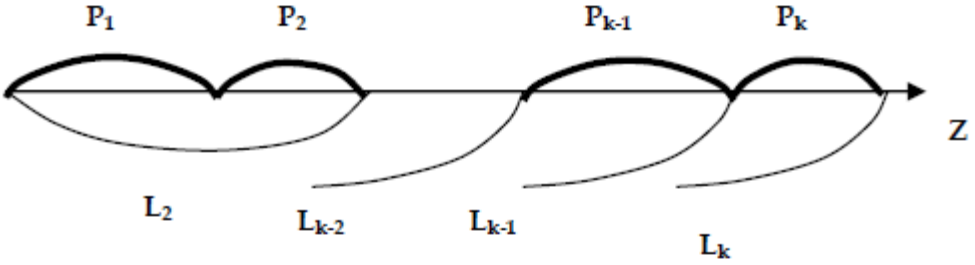
$$\sigma(X) = +\sqrt{D(X)}.$$

1.2. Моделирование полной группы несовместных событий

Пусть имеется полная группа несовместных событий (ПГНС) A_1, A_2..., A_k с вероятностями P_1 , P_2.., P_k . При этом выполняется условие

$$\sum_{i=1}^{k} P_i = 1.$$

Разделим интервал (0,1) на k отрезков, длины которых составляют P_1 , P_2.., P_k (рис. 3.6).



Моделирование полной группы несовместных событий

Если случайное число z, генерированное датчиком случайных чисел с равномерным распределением в интервале (0,1), попало, например, на участок P_{k-1}, то это должно означать, что произошло событие A_{k-1}.

Действительно, если обозначить

$$L_j = \sum_{i=1}^{j} P_i ,$$

то окажется справедливым выражение

$$P(L_{k-2} < z < L_{k-1}) = \int_{L_{k-2}}^{L1} 1 \cdot dx = P_{k-1}$$

Следовательно, произойдет событие, которое имеет вероятность P_{k-1}.

Процедура моделирования полной группы несовместных событий описывается алгоритмом, схема которого показана на рис. 3.7.

Оператор 1 обращается к датчику случайных чисел с равномерным распределением в интервале (0,1). Условный оператор 2 проверяет условие попадания случайной величины z в интервал (O, L_1). Если это условие выполняется, то считается, что произошло событие A_1. Если условие в операторе 2 не выполняется, то алгоритм осуществляет проверку условий попадания случайной величины в другие интервалы. Одно из событий A_1, A_2..., A_k обязательно произойдет.

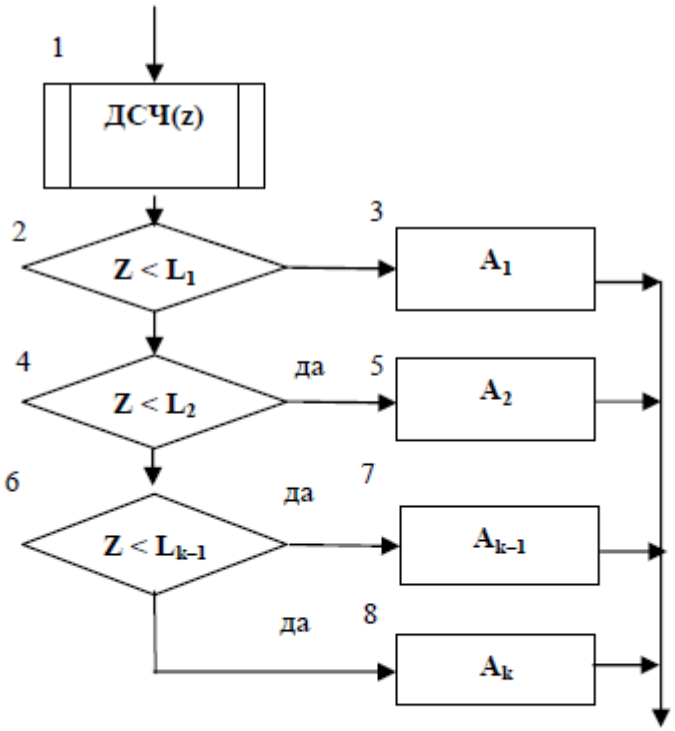


Схема алгоритма моделирования ПГНС

При разработке имитационных моделей с использованием универсальных языков программирования используются разработанные методы моделирования различных законов распределения СВ. Если закон СВ распределения известен, то она может быть достаточно адекватно представлена в имитационной модели. Ниже в качестве дополнительного материала представлены методы моделирования СВ, которые могут быть использованы про создании имитационных моделей на основе универсальных языков программирования.

Метод обратной функции

Пусть имеется некоторая непрерывная случайная величина x, заданная функцией распределения $F(x)$. Можно доказать, что значения этой функции равномерно распределены в интервале (0,1). Поэтому между случайной величиной z, равномерно распределенной в том же интервале, и функцией распределения случайной величины x существует взаимно однозначное соответствие, т.е.

$$z = F(x). \quad (1)$$

Отсюда следует, что

$$x = F^{-1}(z), \quad (2)$$

где F^{-1} – обратная функция.

Следовательно, если уравнение (1) имеет аналитическое решение, то для моделирования случайной величины x можно использовать датчик случайных чисел, генерирующий величину z, и затем осуществить расчет по формуле (2).

Моделирование случайных величин с показательным распределением

Пусть имеется случайная величина x с показательным распределением. Функция распределения имеет вид

$$F(x)=1-e^{-\lambda x},$$

где λ – параметр распределения.

Применив метод обратной функции, получим

$$z = F(x)=1-e^{-\lambda x},$$

откуда

$$x = -\frac{1}{\lambda} ln(1-z). \qquad (3)$$

Учитывая, что случайная величина (1 – z) имеет также равномерное распределение в интервале (0,1), соотношение (3) можно заменить соотношением

$$x = -\frac{1}{\lambda} ln(z).$$

Моделирование случайных величин с равномерным распределением

Датчик случайных чисел генерирует случайные величины с равномерным распределением в интервале (0,1). Если же нужно моделировать случайные величины с равномерным распределением в интервале (a,b), то можно воспользоваться методом обратной функции.

Для рассматриваемого случая выражение (1) примет вид

$$z = F(x) = \frac{x-a}{b-a},$$

откуда $x = a + z(b - a)$.

На практике применяется и другой способ задания равномерного распределения. Вместо границ интервала задаются среднее значение случайной величины $x_{ср}$ и величина интервала Δx. Тогда определение возможного значения случайной величины с равномерным распределением может быть произведено по формуле

$$X = x_{ср} + \Delta x(z - 0,5).$$

Моделирование случайных величин с нормальным распределением

Нормальное распределение НСВ – это распределение, которое имеет плотность распределения вероятностей

$$f(y) = \frac{1}{\sigma_y \sqrt{2\pi}} exp\left[- \frac{(y - m_y)^2}{2\sigma_y^2} \right],$$

где m_y – математическое ожидание;

σ_y – среднее квадратическое отклонение.

Функция распределения

$$F(y) = \frac{1}{\sigma_y \sqrt{2\pi}} \int_{-\infty}^{y} e^{\left[-\frac{(y-m_y)^2}{2\sigma_y^2} \right]} dy.$$

Метод обратной функции для нормального распределения неприменим, так как после подстановки соответствующей функции распределения выражение (2) не имеет аналитического решения. Поэтому в данном случае применяется другой метод.

Согласно центральной предельной теореме теории вероятностей при сложении достаточно большого числа одинаково распределенных независимых случайных чисел получается случайная величина, имеющая нормальное распределение.

Напоминание

Центральная предельная теорема теории вероятностей (теорема Ляпунова) гласит: если СВ x_1, x_2,... x_n независимы, одинаково распределены и имеют конечные математическое ожидание и дисперсию, то распределение суммы этих СВ при увеличении n приближается к нормальному (теорема применима при $n > 10$).

Как показали исследования, уже при сложении более десяти случайных величин с равномерным распределением в интервале (0,1) получается случайная величина, которая с точностью, достаточной для большинства практических задач, может считаться распределенной нормально.

Процедура розыгрыша нормально распределенной случайной величины заключается в следующем.

$$v = \sum_{i=1}^{12} z_i.$$

1. Сложим 12 случайных величин с равномерным распределением в интервале (0,1), т.е. составим сумму

Использовав известные теоремы о сумме математических ожиданий и дисперсий независимых случайных величин, можно установить, что в данном случае случайная величина v имеет следующие характеристики:
математическое ожидание

$$M(V) = \sum_{i=1}^{12} M(z) = 12 \cdot \frac{1}{2} = 6;$$

дисперсия:

$$D(V) = \sum_{i=1}^{12} D(z) = 12 \cdot \frac{1}{12} = 1;$$

среднее квадратическое отклонение

$$\sigma(V) = +\sqrt{D(V)} = 1$$

2. Нормируем и центрируем случайную величину v, т.е. перейдем к величине

$$\eta = [v - M(V)] / \sigma(V) = v - 6.$$

3. От нормированной и центрированной величины η перейдем к случайной величине y с заданными параметрами M(Y) и σ(Y) по формуле
где M(Y) – известное математическое ожидание случайной величины y;

$$y = M(Y) + \sigma(Y) \cdot \eta.$$

σ(Y) – известное квадратическое отклонение случайной величины y.

Моделирование случайных величин с усеченным нормальным распределением

Усеченное нормальное распределение случайной величины x задается четырьмя параметрами: математическим ожиданием M(X), средним квадратическим отклонением σ(X), а также минимальным и максимальным значениями x_1 и x_2 (точками усечения).

Функции распределения случайной величины x определяются равенством

$$F(x)=\begin{cases}0 & при\ x<x_1\\ \left[\Phi_0(t)-\Phi_0(t_1)\right]\cdot A & при\ x_1<x\le x_2\\ 1 & при\ x>x_2\end{cases},$$

где

Функция Лапласа Φ_0 – это функция распределения нормированной и центрированной случайной величины t. Эта функция табулирована.

$$t=\frac{x-M(X)}{\sigma(X)1},\qquad t_1=\frac{x_1-M(X)}{\sigma(X)},\qquad t_2=\frac{x_2-M(X)}{\sigma(X)}.$$

$$A=\frac{1}{\Phi_0(t_2)-\Phi_0(t_1)};$$

$$\Phi_0(t)=\frac{1}{\sqrt{2\pi}}\int_0^t e^{-\frac{t^2}{2}}dt$$

$$t=\frac{y-M_y}{\sigma_y}.$$

Существуют также формулы для расчета математического ожидания, дисперсии и среднего квадратического отклонения случайной величины x. Однако с достаточной для практики точностью при моделировании случайной величины с усеченным нормальным распределением можно обой-

тись без расчетов по формулам. Для определения возможных значений случайной величины с этим распределением можно использовать алгоритм, схема которого приведена на рис. 3.8.

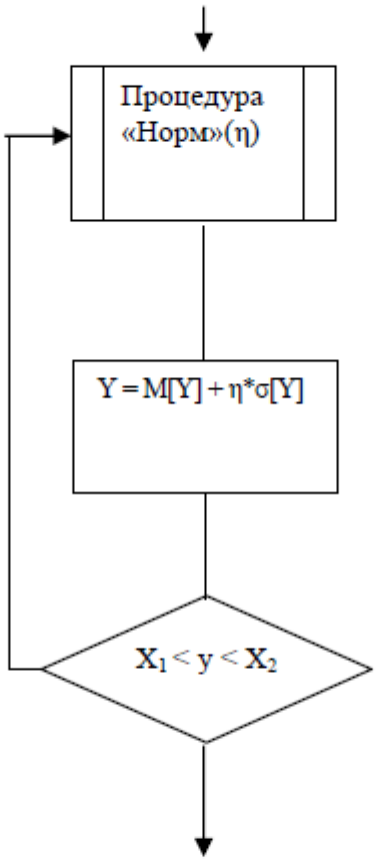


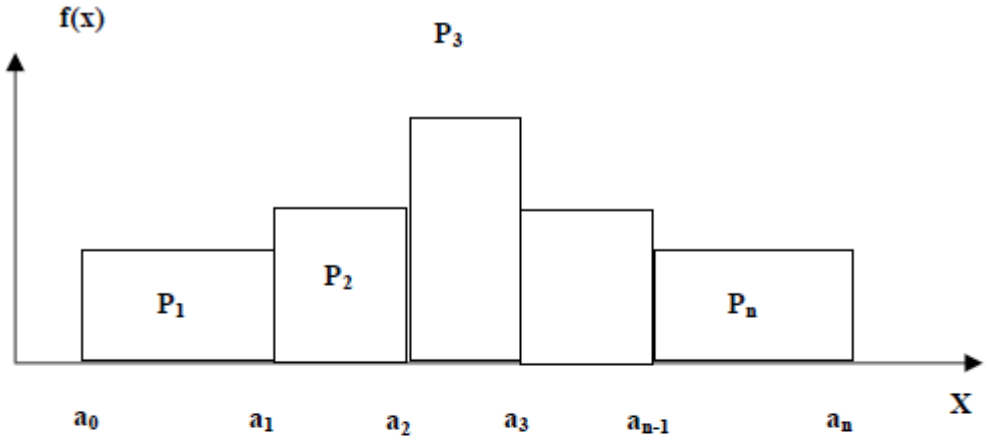
Схема алгоритма моделирования случайной величины с усеченным нормальным распределением

Оператор 1 обращается к процедуре моделирования возможных значений нормированной и центрированной случайной величины η с нормальным распределением. Оператор 2 вычисляет значение случайной величины с заданными параметрами *M(Y)* и *σ(Y)*.

Условный оператор 3 проверяет условие попадания случайной величины y в неусеченную область. При выполнении этого условия значение случайной величины y с усеченным нормальным распределением считается найденным. В противном случае управление в алгоритме передается вновь на вход оператора 1 и генерируется другая случайная величина.

Моделирование случайных величин с произвольным распределением

Пусть случайная величина x задана в интервале (a_0, a_n) кусочно-постоянной функцией $f(x)$. Это значит, что интервал разбит на n частичных интервалов и плотность распределения $f(x)$ на каждом из них постоянна



Плотность распределения произвольной функции

Целесообразно выбрать величины a_k так, чтобы вероятности попадания в любой частичный интервал P_k были одинаковы, т.е.

$$\int_{a_{k-1}}^{a_k} f(x)dx = \frac{1}{n}$$

Моделирование случайных величин

Использование случайных величин является наиболее универсальным и поэтому наиболее распространенным способом учета в модели случайных факторов, присущих реальным экономическим системам или процессам. Примерами случайных величин могут служить: интервал времени до появления очередного клиента, длительность проведения технического обслуживания автомобиля, объем данных, считываемых из оперативной памяти ЭВМ и т.д. Случайные величины могут быть дискретные или непрерывные. Рассмотрим моделирование тех и других величин.

Моделирование дискретной случайной величины

Дискретная случайная величина может быть задана табличной зависимостью:

X	x_1	x_2	...	x_n
P	p_1	p_2	...	p_n

Здесь p_j – вероятность того, что дискретная случайная величина X примет значение x_j. При этом $p_1 + p_2 + ... + p_n = 1$. Разделим интервал (0,1) на n отрезков, длины которых пропорциональны заданным вероятностям. Если случайное число z, вырабатываемое датчиком случайных чисел, равномерно распределенных в интервале (0,1), попадет в интервал p_k, то случайная величина X примет значение x_k. Таким образом, при моделировании дискретных случайных величин фактически используется та же процедура, что и при моделировании ПГНС.

Лекция 4
Концепция и возможности объектно-ориентированной моделирующей системы Matlab.

Общие сведения о MATLAB/SIMULINK. Библиотека блоков SIMULINK

Экономическая теория рассматривает в основном динамические проблемы, гипотезы и закономерности.

В математических и технических вузах РФ широко преподается математический программный инструмент Matlab. Для него как надстройки (Toolboxes) разработаны многие спецприложения для анализа различных систем. Он также предоставляет экономистам финансовый пакет FinancialToolbox, связь с Excel – Excellink, связь с Word – Notebook.

Особый интерес для экономистов представляет инструмент Simulink, разработанный специально для моделирования динамических систем. Он имеет библиотеку стандартных графических блоков с встроенными математическими функциями. Иногда его называют инструментом графического (визуального) программирования. Для проведения моделирования достаточно с помощью мышки перетащить из библиотеки блоки в окно модели, соединить их информационными линиями. Создав таким образом модель, запустив ее, можно наблюдать результаты моделирования в окнах графопостроителей и цифровых дисплеев.

При работе с SIMULINK в основном используются файлы трех типов.

M-файлы (с расширением **.m**) – файлы, содержащие тексты программ на языке MATLAB. В виде M-файлов реализованы все библиотечные функции MATLAB. По умолчанию M-файлы открываются с помощью собственного редактора/отладчика MATLAB.

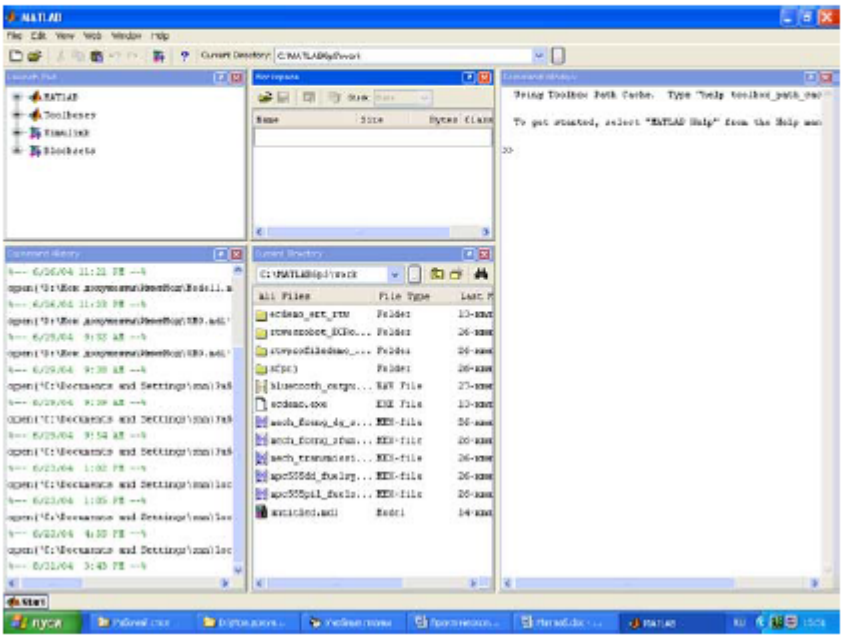
Mdl-файлы (с расширением **.mdl**) – файлы моделей SIMULINK. Они могут быть открыты либо с помощью SIMULINK (в виде графического окна с блок-схемой), либо в виде текста с помощью редактора/отладчика MATLAB.

MAT-файлы (с расширением **.mat**) – файлы данных, хранящиеся в рабочем пространстве (Workspace) MATLAB. Они либо вводятся вручную в командном окне, либо записываются в эту оперативную память из любого пакета MATLAB или из любого приложения, взаимодействующего с MATLAB, например, Excel или Word. Любые приложения могут читать эти данные.

Запуск MATLAB, интерфейс

Для запуска MATLAB надо дважды щелкнуть левой кнопкой мыши на его ярлыке.

Затем появляется рабочий стол MATLAB с вложенными окнами (рис. 4.1).



Окна MATLAB

Чтобы не путать рабочие столы MATLAB и Windows, будем называть стол MATLAB главным окном.

Слева в главном окне расположено окно Launch Pad – первое блюдо для освоения MATLAB. Мы видим здесь продукты, которые заказали во время инсталляции. Раскроем папки продуктов и увидим папки Help (текстовая помощь), Demos (демонстрационные примеры), Product Page (Web–страница продуктов фирмы Math/Works Inc. на сайте фирмы в Internet).

В этом же окне можно переключиться на лист Workspace. Это рабочее пространство MATLAB. В этой оперативной памяти сохраняются все данные рабочей сессии MATLAB, Simulink и других инструментов. Их всегда можно просмотреть или обработать любым инструментом.

В центре расположено окно Current Direktory – текущий, рабочий справочник (папка). Терминология сохранилась от старой операционной системы MS DOS. Чтобы MATLAB мог увидеть вашу программу (м-файл) или функцию, надо установить текущей папку, в которой находится эта функция.

Внизу слева находится окно Command History (история команд) – это протокол вашей работы.

Справа расположено Command Windows – окно для ввода и исполнения команд.

Окно MATLAB представляет собой стандартное окно Windows-приложения и содержит все основные компоненты такого окна:

- √ строку заголовка с кнопками управления окном;
- √ строку меню (основное меню приложения);
- √ панель инструментов;
- √ рабочее поле;
- √ строку состояния;
- √ вертикальную и горизонтальную полосы прокрутки.

Строка меню MATLAB содержит следующие команды:

File (файл) – команды для работы с файлами и настройки системы;

Edit (правка) – команды редактирования информации, отображенной в рабочем поле окна;

View (вид) – команды управления форматом окна;

Web – связь по интернету с фирмой по многим вопросам приобретения, регистрации, консультаций, применения MATLAB;

Windows (окно) – список открытых окон приложения;

Help (справка) – команды вызова средств поддержки пользователя.

Команды меню File.

New – создать;

Open – открыть;

Close Launch Pad – закрыть окно начального знакомства MATLAB;

Import Data – прием данных из других приложений;

Save Workspace As – сохранить рабочую область как...;

Set Path – выбор рабочей папки;

Preferences (предпочтения) – настройка форматов чисел, экрана и других параметров для умолчания, что в офисных продуктах обычно делается в меню Сервис.

Print – печать;

Print Selection – печать выделенного.

Далее список файлов, открывавшихся в прошлом и текущем сеансе работы с MATLAB. Вначале этот список пуст.

Exit MATLAB – выход из MATLAB.

Команды меню Edit.

Они аналогичны командам офисных пакетов.

Команды Clear стирают содержимое окон: командного, истории и рабочего пространства.

Команды меню View (вид).

Здесь расположены команды показа или скрытия окон, а также команда Undock, которая позволяет вывести любое текущее окно из главного окна на рабочий стол Windows.

Help – справочная система.

Help browser – это Web-browser, интегрированный в экран MATLAB и отображающий документы в формате интернета HTML. Все продукты Math\Works Inc. можно получить по интернету.

Предоставляется также справочная система в формате PDF. Справочник по функциям MATLAB содержится в разделе Reference. В нем сведения о назначении и параметрах, а также примеры использования функций MATLAB, входящих в состав рабочей конфигурации пакета.

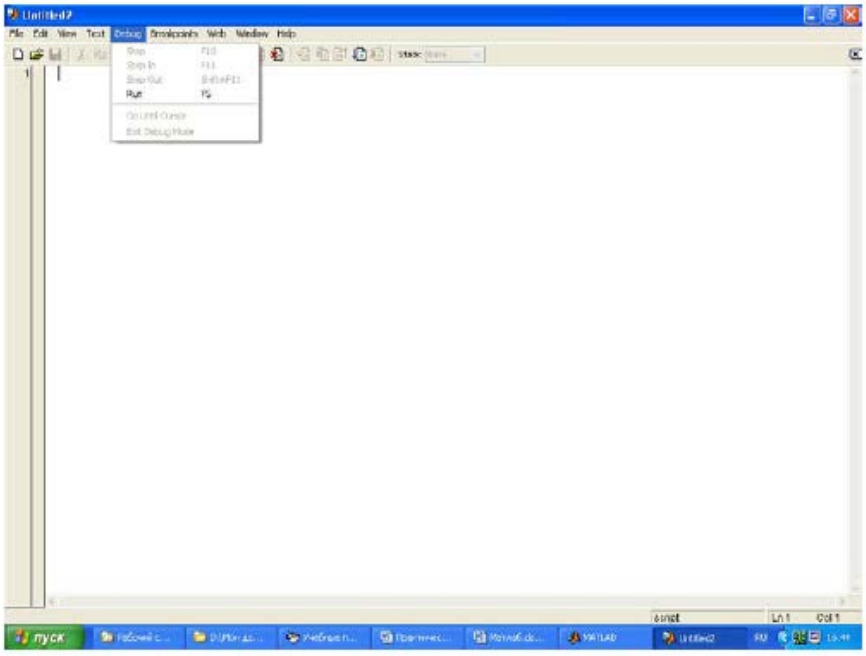
Изменение конфигурации приводит к изменению списка функций, по которым может быть получена справка.

Чтобы получить полную информацию по интересующему разделу, достаточно щелкнуть в соответствующей строке. В поле просмотра появится список функций, входящих в этот раздел, с указанием назначения каждой из них. Щелкнув на нужной функции, можно получить о ней более подробные сведения.

Ниже строки меню расположены кнопки команд меню. Они стандартны для Windows-приложений. Но предпоследняя (перед кнопкой «?») зеленая с красным левым нижним уголком – это кнопка вызова Simulink.

Editor/debugger – редакгор/отладчик программ

Для автоматизации управления экономическими экспериментами с моделями Simulink приходится писать программы на языке MATLAB. Программы (м-файлы) пишутся и отлаживаются в редакторе/отладчике. Он вызывается, когда из меню Файл MATLAB мы открываем новый или существующий м-файл



Окно для написания m-файла

Программа пишется как в обычном текстовом редакторе. В меню имеются лишь два раздела, относящиеся к отладке: Debug и Breakpoints.

Для выполнения лабораторных работ на отлаженной модели экономисту наиболее полезна команда Run из раздела Debug. Она запускает программу на исполнение и манипуляции с моделью Simulink. Этой же программой обрабатываются и отображаются результаты экспериментов.

Breakpoints – точки остановки. Ими помечают строки программы для остановки и анализа поэтапного исполнения программы при ее отладке.

Более подробно рассматривать работу с редактором/ отладчиком будем на последующих занятиях.

Простые вычисления в командном режиме

В MATLAB можно различать два режима работы: вычисления в командном режиме и исполнение программ, записанных на его языке.

В командном окне представлен символ «>>», означающий готовность MATLAB к исполнению команд оператора. Они выполняются как в любом калькуляторе, например, Бэйсика или Excel.

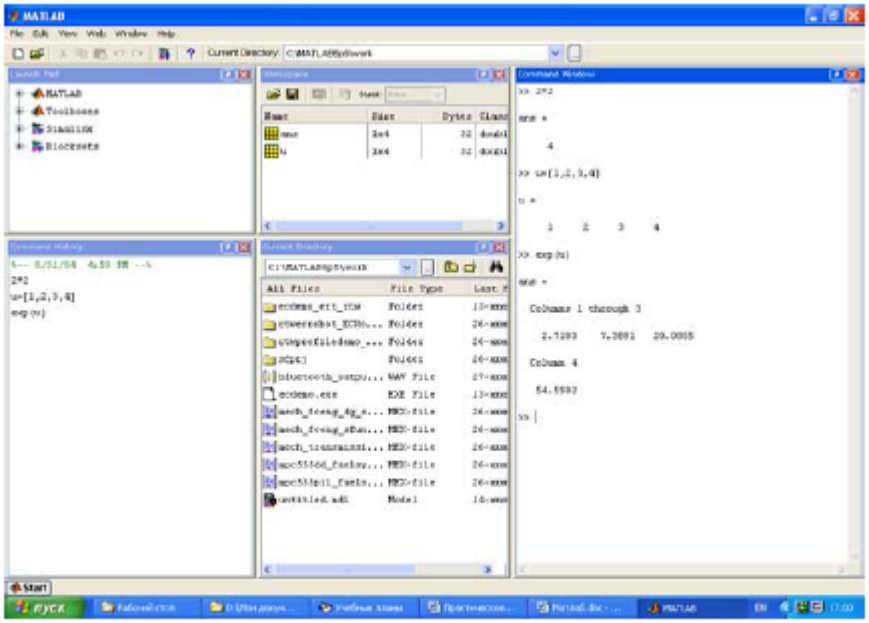
В командном окне (рис.) мы ввели выражение 2*2. Чтобы его вычислить (исполнить), нажмите клавишу Enter. Получим ответ ans = 4 (от англ. answer – ответ). Затем MATLAB показал знак готовности к исполнению новых команд «>>».

Вместо числа мы можем ввести матрицу или вектор, например:

u = [1,2,3,4].

Нажмем клавишу исполнения Enter и получим ответ

u = 1 2 3 4.



Простые вычисления в командном режиме

MATLAB может вычислять практически все математические функции. Например, на нашем рисунке в окне он использует наш вектор **u** и вычисляет вектор экспоненциальных функций exp(**u**) в векторной переменной ans.

Обратите внимание на информацию, выводимую в окнах Workspace, Command History.

Можно ввести последовательность команд. Если команда не заканчивается символом точки с запятой (;), то она выполняется сразу же после нажатия клавиши Enter.

Использование разделителя в виде точки с запятой позволяет вводить в рабочем поле последовательность команд, которая будет выполнена только в том случае, если после очередной команды не стоит этот символ. Если выполнение команды приводит к вычислению некоторого значения (скалярного или матрицы), то оно запоминается в рабочей области MATLAB в переменной с именем ans. Значение, занесенное в переменную ans, выводится на экран сразу после вычисления в форме ans = значение (число, вектор, матрица).

Введение в Simulink

Программа Simulink является приложением к пакету MATLAB. При моделировании с использованием Simulink реализуется принцип визуального программирования, в соответствии с которым пользователь на экране из библиотеки стандартных блоков создает модель устройства, процесса или системы и осуществляет расчеты. При этом, в отличие от классических способов моделирования, пользователю не нужно досконально изучать язык программирования и численные методы математики, а достаточно общих знаний, требующихся при работе на компьютере, и, естественно, знаний той предметной области, в которой он работает.

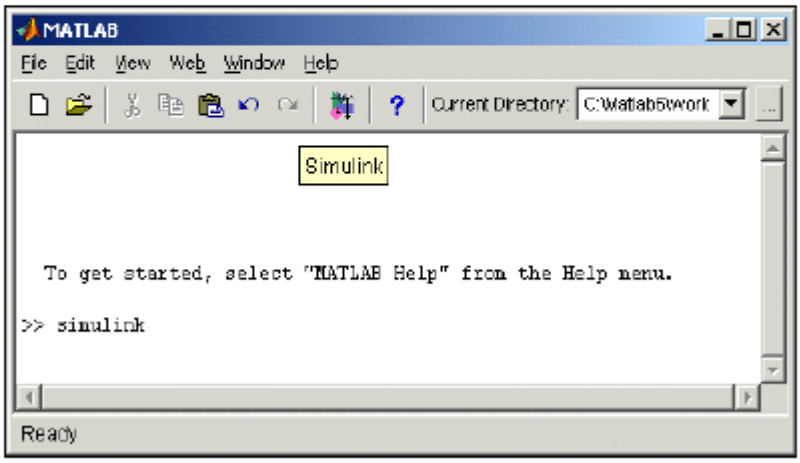
Simulink является достаточно самостоятельным инструментом MATLAB и при работе с ним совсем не требуется знать сам MATLAB и остальные его приложения. С другой стороны, доступ к функциям MATLAB и другим его инструментам остается открытым и их можно использовать в Simulink. При работе с Simulink пользователь имеет возможность модернизировать библиотечные блоки, создавать свои собственные, а также составлять новые библиотеки блоков.

При моделировании пользователь может выбирать метод решения дифференциальных уравнений, а также способ изменения модельного времени (с фиксированным или переменным шагом). В ходе моделирования имеется возможность следить за процессами, происходящими в системе. Для этого используются специальные устройства наблюдения, входящие в состав библиотеки Simulink. Результаты моделирования могут быть представлены в виде графиков или таблиц.

Преимущество Simulink заключается также в том, что она позволяет пополнять библиотеки блоков с помощью подпрограмм, написанных как на языке MATLAB, так и на языках C++, Fortran и Ada.

Работа с Simulink

Для запуска программы необходимо предварительно запустить пакет MATLAB. Основное окно пакета MATLAB показано на рис. Там же показана подсказка, появляющаяся в окне при наведении указателя мыши на ярлык Simulink в панели инструментов.

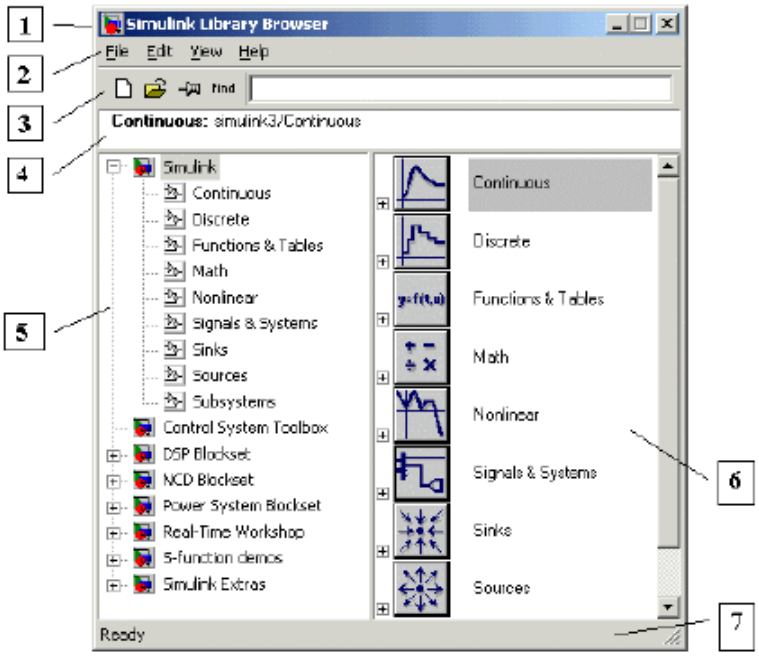


Основное окно программы MATLAB

После открытия основного окна программы MATLAB нужно запустить программу Simulink. Это можно сделать одним из трех способов:

- Нажать кнопку (Simulink) на панели инструментов командного окна MATLAB.
- В командной строке главного окна MATLAB напечатать Simulink и нажать клавишу Enter на клавиатуре.
- Выполнить команду Open… в меню File и открыть файл модели (mdl–файл).

Последний вариант удобно использовать для запуска уже готовой и отлаженной модели, когда требуется лишь провести расчеты и не нужно добавлять новые блоки в модель. Использование первого и второго способов приводит к открытию окна обозревателя разделов библиотеки Simulink



Окно обозревателя разделов библиотеки Simulink

Обозреватель разделов библиотеки Simulink

Окно обозревателя библиотеки блоков содержит следующие элементы:

1. Заголовок с названием окна – Simulink Library Browser.
2. Меню с командами File, Edit, View, Help.
3. Панель инструментов с ярлыками наиболее часто используемых команд.
4. Окно комментария для вывода поясняющего сообщения о выбранном блоке.
5. Список разделов библиотеки, реализованный в виде дерева.
6. Окно содержимого раздела библиотеки (список вложенных разделов библиотеки или блоков).
7. Строка состояния, содержащая подсказку по выполняемому действию.

Библиотека Simulink содержит следующие основные разделы:

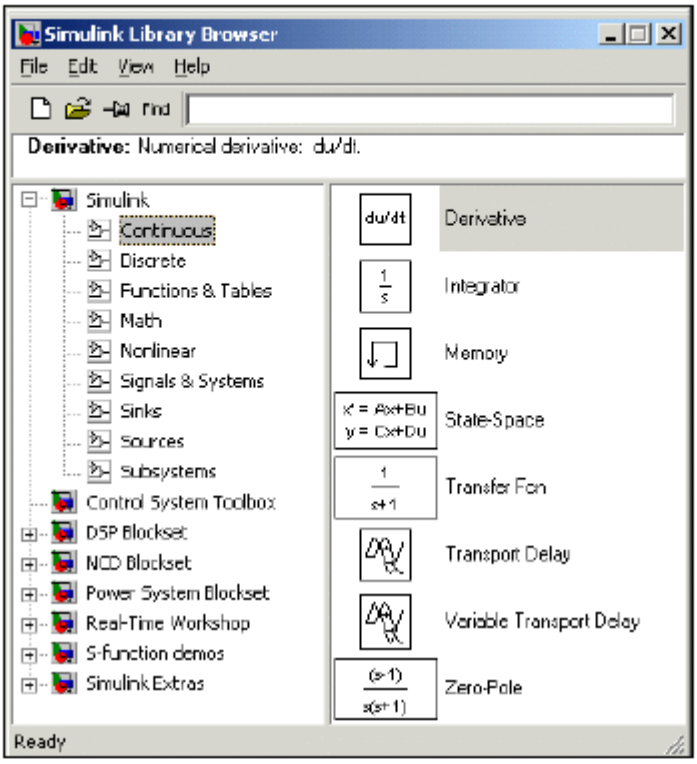
1. Continuous – линейные блоки.
2. Discrete – дискретные блоки.
3. Functions & Tables – функции и таблицы.
4. Math – блоки математических операций.
5. Nonlinear – нелинейные блоки.
6. Signals & Systems – сигналы и системы.
7. Sinks – регистрирующие устройства.
8. Sources – источники сигналов и воздействий.
9. Subsystems – блоки подсистем.

Список разделов библиотеки Simulink представлен в виде дерева, и правила работы с ним являются общими для списков такого вида:

• Пиктограмма свернутого узла дерева содержит символ «+», а пиктограмма развернутого содержит символ «–».

• Для того чтобы развернуть или свернуть узел дерева, достаточно щелкнуть на его пиктограмме левой клавишей мыши (ЛКМ).

При выборе соответствующего раздела библиотеки в правой части окна отображается его содержимое



Окно обозревателя с набором блоков раздела библиотеки

Для работы с окном используются команды, собранные в меню. Меню обозревателя библиотек содержит следующие пункты:

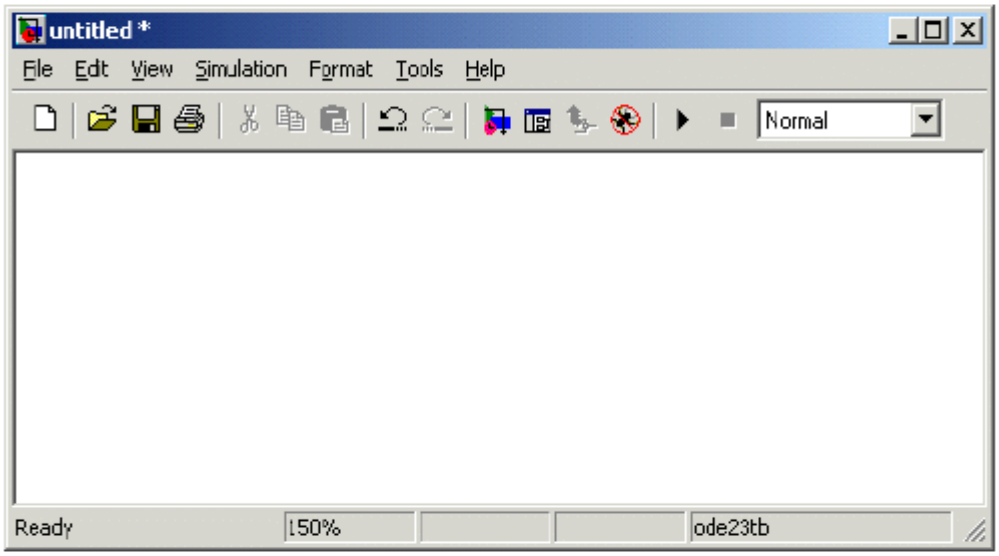
- File (Файл) – работа с файлами библиотек.
- Edit (Редактирование) – добавление блоков и их поиск (по названию).
- View (Вид) – управление показом элементов интерфейса.
- Help (Справка) – вывод окна справки по обозревателю библиотек.

Для работы с обозревателем можно также использовать кнопки на панели инструментов.

Создание модели

Для создания модели в среде SIMULINK необходимо последовательно выполнить ряд действий:

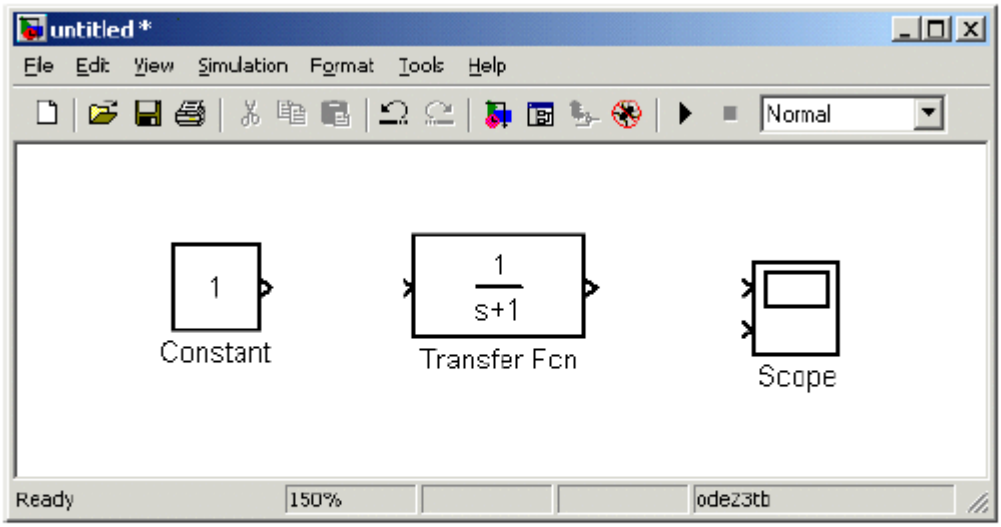
1. Создать новый файл модели с помощью команды File/New/Model или используя кнопку на панели инструментов (здесь и далее, с помощью символа «/», указаны пункты меню программы, которые необходимо последовательно выбрать для выполнения указанного действия). Вновь созданное окно модели показано на рис.



Пустое окно модели

2. Расположить блоки в окне модели.

Для этого необходимо открыть соответствующий раздел библиотеки (например, Sources – Источники). Далее, указав курсором на требуемый блок и нажав на левую клавишу мыши, – «перетащить» блок в созданное окно. *Клавишу мыши нужно держать нажатой*. На рис. показано окно модели, содержащее блоки.



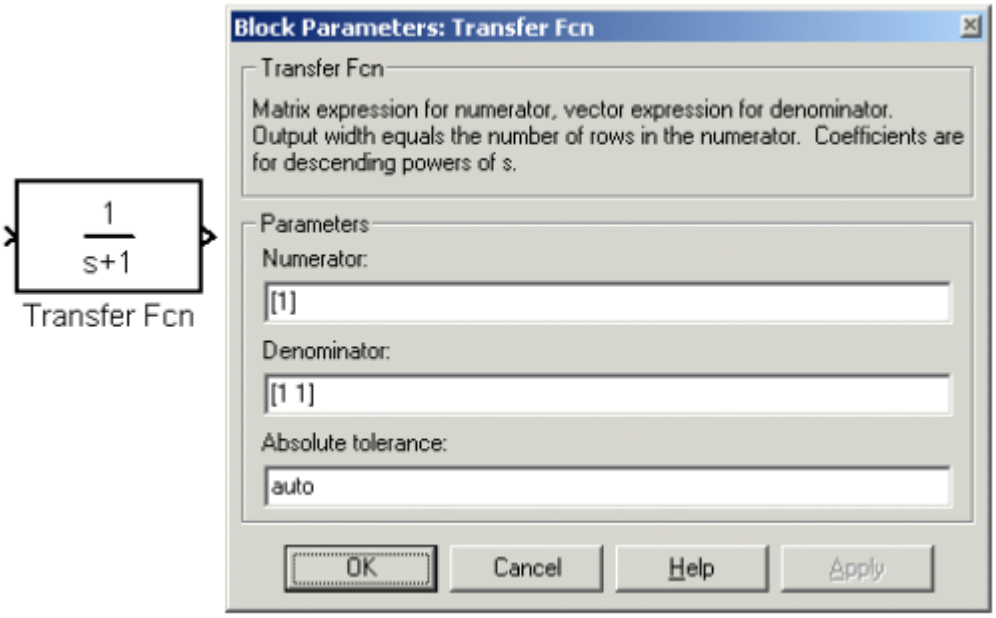
Окно модели, содержащее блоки

Для удаления блока необходимо выбрать блок (указать курсором на его изображение и нажать левую клавишу мыши), а затем нажать клавишу Delete на клавиатуре.

Для изменения размеров блока требуется выбрать блок, установить курсор в один из углов блока и, нажав левую клавишу мыши, изменить размер блока (курсор при этом превратится в двухстороннюю стрелку).

3. Далее, если это требуется, нужно изменить параметры блока, установленные программой «по умолчанию».

Для этого необходимо дважды щелкнуть левой клавишей мыши, указав курсором на изображение блока. Откроется окно редактирования параметров данного блока. При задании численных параметров следует иметь в виду, что в качестве десятичного разделителя должна использоваться точка, а не запятая. После внесения изменений нужно закрыть окно кнопкой OK. На рис. в качестве примера показаны блок, моделирующий передаточную функцию, и окно редактирования параметров данного блока.



Блок, моделирующий передаточную функцию, и окно редактирования параметров блока

4. После установки на схеме всех блоков из требуемых библиотек нужно выполнить соединение элементов схемы

Для соединения блоков необходимо указать курсором на «выход» блока, а затем нажать и, не отпуская левую клавишу мыши, провести линию к входу другого блока. После чего отпустить клавишу. В случае правильного соединения изображение стрелки на входе блока изменяет цвет. Для создания точки разветвления в соединительной линии нужно подвести курсор к предполагаемому узлу и, нажав *правую* клавишу мыши, протянуть линию. Для удаления линии требуется выбрать линию (так же, как это выполняется для блока), а затем нажать клавишу Delete на клавиатуре.

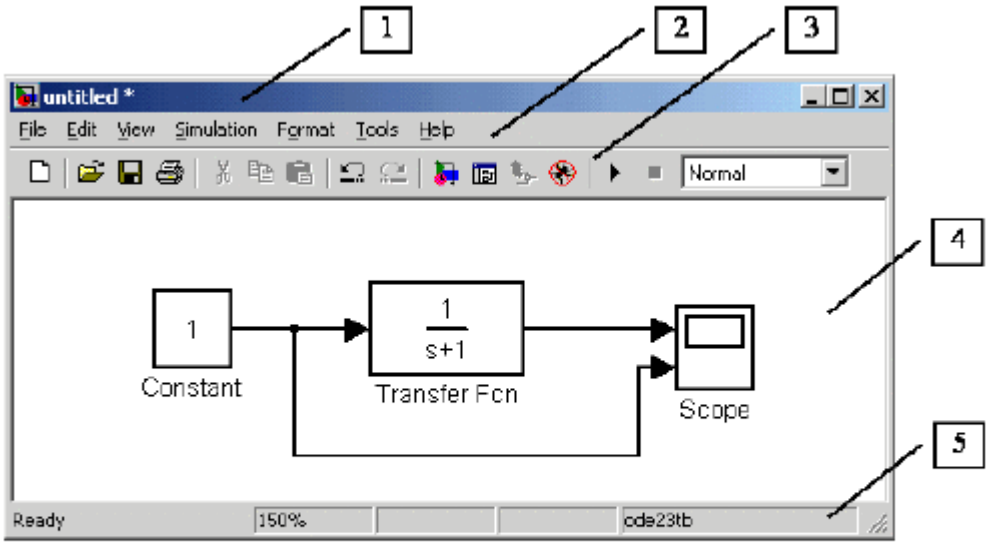


Схема модели с выполнением соединением блоков

5. После составления расчетной схемы необходимо сохранить ее в виде файла на диске, выбрав пункт меню File/Save As... в окне схемы и указав папку и имя файла. Следует иметь в виду, что имя файла не должно превышать 32 символов, должно начинаться с буквы и не может содержать символы кириллицы и спецсимволы. Это же требование относится и к пути файла (к тем папкам, в которых сохраняется файл). При последующем редактировании схемы можно пользоваться пунктом меню File/Save. При повторных запусках программы SIMULINK загрузка схемы осуществляется с помощью меню File/Open... в окне обозревателя библиотеки или из основного окна MATLAB.

Окно модели

Окно модели содержит следующие элементы

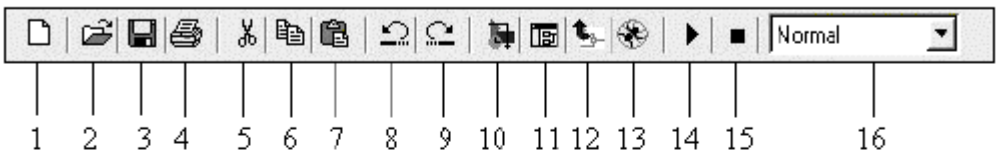
1. Заголовок с названием окна. Вновь созданному окну присваивается имя Untitled с соответствующим номером.
2. Меню с командами File, Edit, View и т.д.

3. Панель инструментов.
4. Окно для создания схемы модели.
5. Строка состояния, содержащая информацию о текущем состоянии модели.

Меню окна содержит команды для редактирования модели, ее настройки и управления процессом расчета, работы файлами и т.п.:

- File (Файл) – работа с файлами моделей.
- Edit (Редактирование) – изменение модели и поиск блоков.
- View (Вид) – управление показом элементов интерфейса. Simulation (Моделирование) – задание настроек для моделирования и управление процессом расчета.
- Format (Форматирование) – изменение внешнего вида блоков и модели в целом.
- Tools (Инструментальные средства) – применение специальных средств для работы с моделью (отладчик, линейный анализ и т.п.).
- Help (Справка) – вывод окон справочной системы.

Для работы с моделью можно также использовать кнопки на панели инструментов



Панель инструментов окна модели

Кнопки панели инструментов имеют следующее назначение:

1. New Model – открыть новое (пустое) окно модели.
2. Open Model – открыть существующий mdl-файл.
3. Save Model – сохранить mdl-файл на диске.
4. Print Model – вывод на печать блок-диаграммы модели.
5. Cut – вырезать выделенную часть модели в буфер промежуточного хранения.

6. Copy – скопировать выделенную часть модели в буфер промежуточного хранения.
7. Paste – вставить в окно модели содержимое буфера промежуточного хранения.
8. Undo – отменить предыдущую операцию редактирования.
9. Redo – восстановить результат отмененной операции редактирования.
10. Library Browser – открыть окно обозревателя библиотек.
11. Toggle Model Browser – открыть окно обозревателя модели.
12. Go to parent system – переход из подсистемы в систему высшего уровня иерархии («родительсую систему»). Команда доступна только, если открыта подсистема.
13. Debug – запуск отладчика модели.
14. Start/Pause/Continue Simulation – запуск модели на исполнение (команда Start).
15. Stop – закончить моделирование. Кнопка становится доступной после начала моделирования, а также после выполнения команды Pause.
16. Normal/Accelerator – обычный/ускоренный режим расчета. Инструмент доступен, если установлено приложение Simulink Performance Tool.

В нижней части окна модели находится строка состояния, в которой отображаются краткие комментарии к кнопкам панели инструментов, а также к пунктам меню, когда указатель мыши находится над соответствующим элементом интерфейса. Это же текстовое поле используется и для индикации состояния Simulink: Ready (Готов) или Running (Выполнение). В строке состояния отображаются также:

- масштаб отображения блок-диаграммы (в процентах, исходное значение равно 100%);
- индикатор степени завершенности сеанса моделирования (появляется после запуска модели);
- текущее значения модельного времени (выводится также только после запуска модели);
- используемый алгоритм расчета состояний модели (метод решения).

Лекция 5
Управление модельным временем.

Управление модельным временем

Виды представления времени в модели

С построенной имитационной моделью проводятся имитационные эксперименты, которые фактически представляют собой наблюдение за поведением экономической системы в течение некоторого промежутка времени. Эффективность экономической системы с помощью модели оценивается путем наблюдения за ней во времени. Характерной особенностью большинства практических задач является то, что скорость протекания рассматриваемых экономических процессов значительно ниже скорости реализации модельного эксперимента. Например, если моделируется работа супермаркета в течение месяца, вряд ли кому-то придет в голову воспроизводить этот процесс в модели в таком же масштабе времени. С другой стороны, даже те имитационные эксперименты, в которых временные параметры функционирования системы не учитываются, требуют для своей реализации определенных затрат времени работы компьютера.

В связи с этим при разработке практически любой имитационной модели и планировании проведения модельных экспериментов необходимо соотносить между собой три представления времени:

- реальное, в котором происходит функционирование имитируемой системы;
- модельное (или, как его еще называют, системное), в масштабе которого организуется работа модели;
- машинное, отражающее затраты времени ЭВМ на проведение имитации.

С помощью механизма модельного времени решаются следующие задачи:

- Отображается переход моделируемой системы из одного состояния в другое.
- Производится синхронизация работы компонент модели.

- Изменяется масштаб времени «жизни» (функционирования) исследуемой системы.
- Производится управление ходом модельного эксперимента.
- Моделируется квазипараллельная реализация событий в модели (приставка «квази» в данном случае отражает последовательный характер обработки событий (процессов) в ИМ, которые в реальной системе возникают (протекают) одновременно).

Необходимость решения последней задачи связана с тем, что в распоряжении исследователя, как правило, находится однопроцессорная вычислительная система, а модель может содержать значительно большее число одновременно работающих подсистем. Поэтому действительно параллельная (одновременная) реализация всех компонент модели невозможна. Реализация квазипараллельной работы компонент модели является достаточно сложной технической задачей. Пакет Matlab позволяет избавиться от этой проблемы. В данном курсе вопросы, связанные с этой проблемой, не рассматриваются.

Существуют два метода реализации механизма модельного времени:

- моделирование с постоянным шагом;
- моделирование по особым состояниям.

Изменение времени с постоянным шагом

При использовании данного метода отсчет системного времени ведется через фиксированные, выбранные исследователем интервалы времени. События в модели считаются наступившими в момент окончания этого интервала. Погрешность в измерении временных характеристик системы в этом случае зависит от величины шага моделирования ΔT.

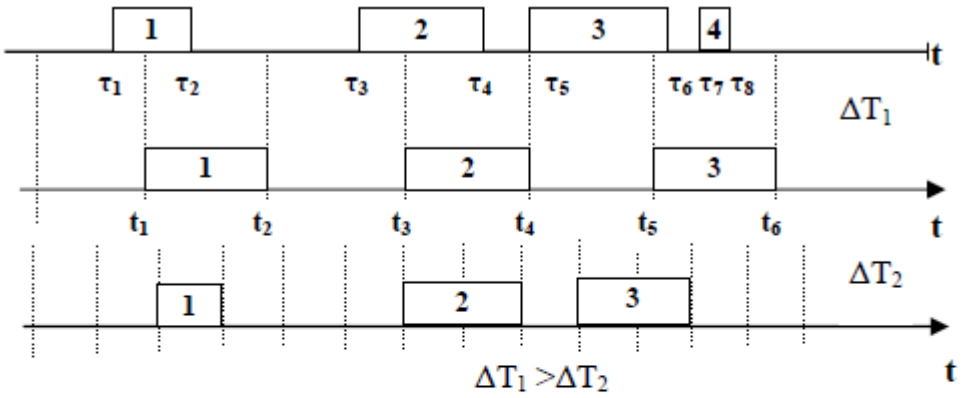
Метод моделирования с постоянным шагом используется на практике, если:

1. События появляются регулярно, их распределение во времени достаточно равномерно.
2. Число событий велико и моменты их появления близки.
3. Моменты появления событий заранее определить невозможно.

Данный метод управления модельным временем достаточно просто реализуется в том случае, когда условия появления событий всех типов в модели можно представить как функцию времени (например, если моделируется система массового обслуживания).

Рассмотрим в качестве примера систему массового обслуживания, процессы которой мы хотим моделировать.

Процесс функционирования такой системы можно рассматривать как последовательную смену ее состояний. Пусть, например, в одноканальной системе массового обслуживания происходит процесс обслуживания поступающих заявок (рис. 5.1).



Моделирование способом ΔT

Введем следующие обозначения:

τ_1 – момент начала обслуживания 1-й заявки;

τ_2 – момент конца обслуживания 1-й заявки;

τ_3 – момент начала обслуживания 2-й заявки;

τ_4 – момент конца обслуживания 2-й заявки;

τ_5 – момент начала обслуживания 3-й заявки;

τ_6 – момент конца обслуживания 3-й заявки;

τ_7 – момент начала обслуживания 4-й заявки;

τ_8 – момент конца обслуживания 4-й заявки.

Выберем шаг ΔT и будем анализировать состояние системы через промежутки времени t_1, t_2,..., отстоящие друг от друга на ΔT. Этот способ иногда называют способом ΔT.

В момент t_1 будет обнаружено, что в системе началось обслуживание 1-й заявки. В момент $t_2 = t_1 + \Delta T$ будет установлено, что обслуживание 1-й заявки завершено. В момент $t_3 = t_2 + \Delta T$ будет обнаружено, что в системе началось обслуживание 2-й заявки. В момент $t_4 = t_3 + \Delta T$ будет установлено, что обслуживание 2-й заявки завершено. В момент $t_5 = t_4 + \Delta T$ будет обнаружено, что в системе началось обслуживание 3-й заявки. В момент $t_6 = t_5 + \Delta T$ будет установлено, что обслуживание 3-й заявки завершено. Факт поступления 4-й заявки и факт окончания ее обслуживания не будут обнаружены.

Эту логику наблюдения за реальной системой мы переносим из реального времени поведения системы в модельное время.

Для предотвращения потерь информации и повышения точности работы модели нужно уменьшить шаг ΔT. При малом ΔT можно достаточно точно описать процесс функционирования системы.

Однако способ ΔT является весьма неэкономичным с точки зрения расхода машинного времени. Достоинство способа состоит в том, что он позволяет моделировать любые процессы: детерминированные, непрерывные, случайные, с зависимыми или независимыми событиями и т.п.

Выбор ΔT – задача очень важная и нелегкая. Необходимо:

1. ΔT принимать равной средней интенсивности возникновения событий различных типов.
2. ΔT выбирать равной среднему интервалу между наиболее частыми (или наиболее важными) событиями.

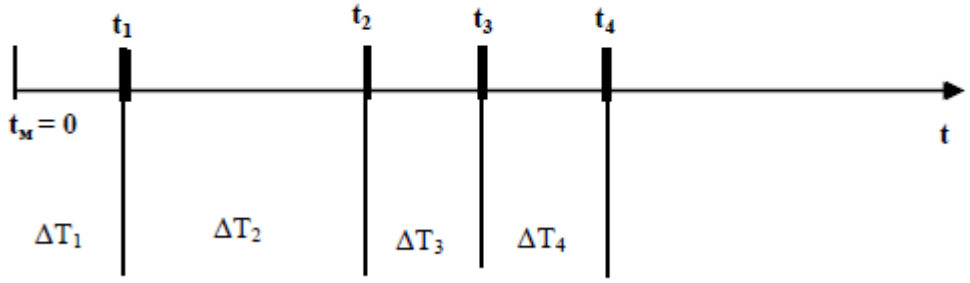
Продвижение времени по особым состояниям

При моделировании по особым состояниям системное время каждый раз изменяется на величину, строго соответствующую интервалу времени до момента наступления очередного события. В этом случае события обрабатываются в порядке их наступления, а одновременно наступившими считаются только те, которые являются одновременными в действительности.

Для реализации моделирования по особым состояниям требуется разработка специальной процедуры планирования событий (так называемого календаря событий).

Если известен закон распределения интервалов между событиями, то такое прогнозирование труда не составляет: достаточно к текущему значению модельного времени добавить величину интервала, полученную с помощью соответствующего датчика.

Пусть, например, при моделировании системы массового обслуживания очередные заявки поступают в случайные моменты времени, но по известному закону, допустим показательному (именно так часто бывает на практике) с параметрами: λ – интенсивность потока заявок и T – среднее время между соседними заявками. Иллюстрация к такой ситуации приведена на рис.
($t_1 - t_4$ – моменты формирования заявок для системы массового обслуживания; $\Delta T1 \ldots \Delta T4$ – случайные интервалы времени, имеющие показательный закон распределения СВ).



Изменение модельного времени по особым состояниям

При моделировании таким методом сложности возникают, если имеет место несколько взаимосвязанных событий.

Моделирование по особым состояниям целесообразно использовать, если:

- события распределяются во времени неравномерно или интервалы между ними велики;
- предъявляются повышенные требования к точности определения взаимного положения событий во времени;
- необходимо учитывать наличие одновременных событий.

Дополнительное достоинство метода заключается в том, что он позволяет экономить машинное время, особенно при моделировании систем периодического действия, в которых события длительное время могут не наступать.

Моделирование параллельных процессов

Практически любая более или менее сложная система имеет в своем составе компоненты, работающие одновременно, или, как принято говорить, параллельно. Примерами являются одновременное обслуживание клиентов, например, в парикмахерской, работа по обслуживанию и одновременно устранению неисправности в работе, например, одного из кассовых аппаратов, и т.д.

Итак, если в составе системы имеются компоненты (подсистемы), выполняющие свои функции одновременно, то можно утверждать, что в такой системе существуют параллельные процессы. Параллельно работающие подсистемы могут взаимодействовать самым различным образом либо вообще работать независимо друг от друга. Способ взаимодействия подсистем определяет вид параллельных процессов, протекающих в системе. В свою очередь, вид моделируемых процессов влияет на выбор метода их имитации.

Виды параллельных процессов

Асинхронный параллельный процесс – такой процесс, состояние которого не зависит от состояния другого параллельного процесса (ПП).

Пример асинхронных ПП, протекающих в рамках одной системы: подготовка и проведение рекламной кампании фирмой и работа сборочного конвейера; пример из области вычислительной техники: выполнение вычислений процессором и вывод информации на печать.

Синхронный ПП – такой процесс, состояние которого зависит от состояния взаимодействующих с ним ПП.

Пример синхронного ПП: работа торговой организации и доставка товара со склада (нет товара – нет торговли).

Независимый ПП – процесс, который не является подчиненным ни для одного из процессов. Пример независимого ПП: процессы обслуживания покупателей в кассах супермаркета.

Методы описания параллельных процессов

В пакете Matlab имеется собственный язык моделирования, позволяющий моделировать параллельные процессы.

Рассмотрим механизм реализации ПП на основе транзактов.

В этом случае под событием понимается любое перемещение транзакта по системе, а также изменение его состояния (обслуживается, заблокирован и т.д.).

Событие, связанное с данным транзактом, может храниться в одном из следующих списков:

- **Список текущих событий**. В этом списке находятся события, время наступления которых меньше или равно текущему модельному времени. События с «меньшим» временем связаны с перемещением тех транзактов, которые должны были начать двигаться, но были заблокированы.
- **Список будущих событий**. Этот список содержит события, время наступления которых больше текущего модельного времени, то есть события, которые должны произойти в будущем (условия наступления которых уже определены – например, известно, что транзакт будет обслуживаться некоторым устройством 10 единиц времени).

- **Список прерываний**. Данный список содержит события, связанные с возобновлением обработки прерванных транзактов. События из этого списка выбираются в том случае, если сняты условия прерывания.

Рассмотрим использование двух первых списков событий в динамике, при моделировании параллельных процессов.

В списке текущих событий транзакты расположены в порядке убывания приоритета соответствующих событий; при равных приоритетах – в порядке поступления в список.

Каждое событие (транзакт) в списке текущих событий может находиться либо в активном состоянии, либо в состоянии задержки. Если событие активно, то соответствующий транзакт может быть продвинут по системе; если продвижение невозможно (например, из-за занятости устройства), то событие (и транзакт) переводится в состояние задержки.

Как только завершается обработка (продвижение) очередного активного транзакта, просматривается список задержанных транзактов и ряд из них переводится в активное состояние. Процедура повторяется до тех пор, пока в списке текущих событий не будут обработаны все активные события. После этого просматривается список будущих событий. Модельному времени присваивается значение, равное времени наступления ближайшего из этих событий. Данное событие заносится в список текущих событий. Затем просматриваются остальные события списка. Те из них, время которых равно текущему модельному времени, также переписываются в список текущих событий. Просмотр заканчивается, когда в списке остаются события, времена которых больше текущего модельного времени.

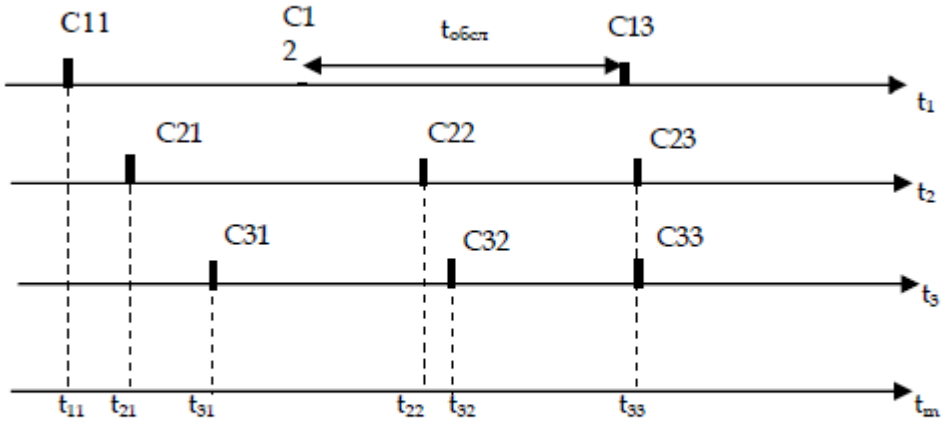
В качестве иллюстрации к изложенному рассмотрим небольшой пример. Пусть в систему поступают транзакты трех типов, каждый из которых обслуживается отдельным устройством. Известны законы поступления транзактов в систему и длительность их обслуживания. Таким образом, в системе существуют три параллельных независимых процесса (P1, P2, P3).

Временная диаграмма работы системы при обслуживании одного транзакта каждого типа показана на рис.

На рисунке события, относящиеся к процессу P1, обозначены как C1i; относящиеся к P2 и к P3 – соответственно как C2i и C3i.

Для каждого процесса строится своя цепь событий, однако списки событий являются общими для всей модели. Формирование списков начинается с заполнения списка будущих событий. Как было отмечено выше, в этот список помещаются события, время наступления которых превышает текущее значение модельного времени. Очевидно, что на момент заполнения списка время наступления прогнозируемых событий должно быть известно. На первом шаге $t_m = 0$, и в список будущих событий заносятся события C11, C21, C31. Затем, событие с наименьшим временем наступления – C11 – переносится в список текущих событий; если одновременных с ним событий нет, то оно обрабатывается и исключается из списка текущих событий. После этого вновь корректируется список будущих событий и т.д., пока не истечет заданный интервал моделирования.

Динамика изменения списков текущих и будущих событий для рассмотренного примера отражена в таблице.



Временная диаграмма параллельных процессов

Изменение списков событий

t	Список будущих событий	Список текущих событий
0	C11 C21 C31	0
t_{11}	C21 C31 C12	C11
t_{21}	C31 C12 C22	C21
t_{31}	C12 C22 C32	C31
t_{12}	C22 C32 C13	C12
t_{22}	C32 C13 C23	C22
t_{32}	C13 C23 C33	C32
t_{13}	C23 C33	C32
t_{23}		C23 C33

Применение сетевых моделей для описания параллельных процессов

Как будет показано на последующих занятиях, этапу программной реализации модели (т.е. ее описанию на одном из языков программирования) должен предшествовать так называемый этап алгоритмизации. Другими словами, прежде чем превратить имитационную модель в работающую компьютерную программу, ее создатель должен воспользоваться каким-то менее формальным и более наглядным средством описания логики работы будущей программы. Разумеется, это требование не является обязательным для всех разработчиков и для всех создаваемых моделей: при наличии достаточного опыта программа не очень сложной модели может быть написана сразу. Однако практика показывает, что человеческие возможности не безграничны, и при моделировании более сложных систем даже опытные разработчики бывают вынуждены немного «притормозить» на этапе алгоритмизации. Для описания логики работы модели могут быть использованы различные средства: либо русский язык (устный или письменный), либо традиционные схемы алгоритмов, либо какие-то другие «подручные» средства. Первые два варианта являются, как правило, наиболее знакомыми и наиболее часто используемыми. Однако, если вы попробуете описать в виде схемы алгоритма модель даже такой простой системы, которая использовалась в предыдущем примере, то это окажется напрасной тратой времени и сил. Прежде всего потому, что такие схемы совершенно не приспособлены для описания параллельных процессов.

Одним из наиболее элегантных и весьма распространенных средств описания параллельных процессов являются так называемые сети Петри.

Управление модельным временем в MATLAB

Задача корректного управления модельным временем, то есть «временем жизни» моделируемой системы, возлагается на разработчика независимо от того, какие инструменты создания модели он использует. Принципиально отличие Simulink от универсальных средств программирования состоит в том, что логическая структура S-модели не зависит от способа управления модельным временем. Более того, исследователь может выбирать способ изменения модельного времени для каждого сеанса моделирования индивидуально.

Тем не менее и при использовании Simulink модельное время остается «спинным мозгом», согласующим работу всех компонент S-модели. Поэтому при подготовке каждого модельного эксперимента должны быть получены ответы на три вопроса:

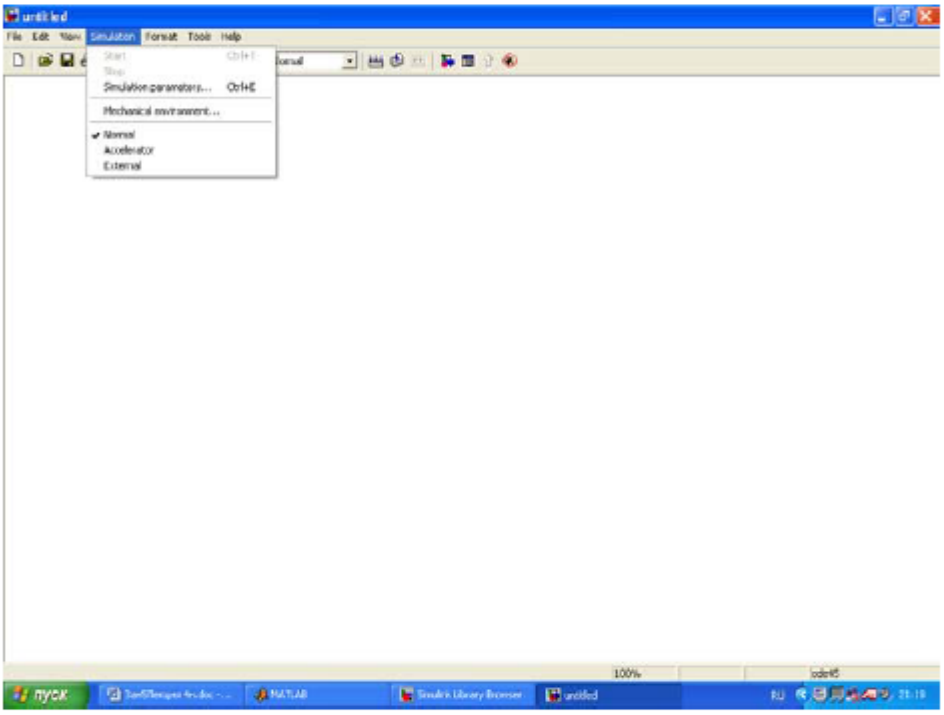
- Какой способ изменения (приращения) времени целесообразно использовать (с переменным или постоянным шагом).
- С какой дискретностью следует изменять модельное время.
- Какое событие будет служить условием окончания эксперимента.

Выбор шага моделирования

Как было рассмотрено ранее, в практике имитационного моделирования применяются два основных способа изменения модельного времени – с постоянным шагом и по особым состояниям. При выборе одного из этих методов важное значение имеет тип моделируемой системы: для непрерывных систем (с непрерывным временем смены состояний) по умолчанию используется переменный шаг приращения времени, а для дискретных систем следует устанавливать постоянный (фиксированный) шаг. Но такой подход не всегда оправдан, поскольку при моделировании непрерывных систем бывает

удобнее определять очередное состояние системы как функцию времени, изменяющегося с заданной дискретностью. И наоборот, при моделировании дискретных систем величина очередного приращения времени зачастую определяется прогнозируемым моментом изменения состояния системы; причем смена состояний происходит, как правило, нерегулярно. Поэтому полезно знать, каким образом при разработке моделей дискретных систем можно заставить модельное время изменяться по особым состояниям.

Ранее мы рассматривали то, что в окне Simulink имеется меню Simulation



Окно блок-диаграммы S-модели и вид меню Simulation

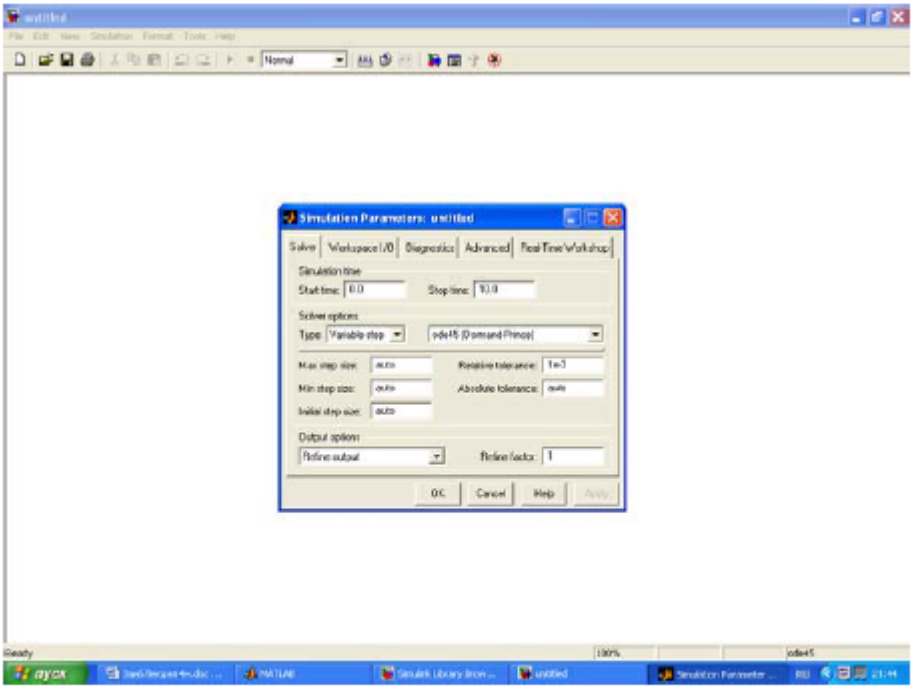
Данное меню играет основную роль при проведении исследования на модели. Посредством команд этого меню разработчик получает возможность не только динамически управлять сеансом моделирования, но и изменять многие важней-

шие параметры модели, такие как, например, способ изменения модельного времени и формат представления результатов моделирования.

Важной командой этого меню является Simulation Parameters. По данной команде открывается диалоговое окно настроек параметров моделирования. Для нас представляют интерес вкладки:

- *Solver* (Расчет) – установка параметров расчета модели;
- *Workspace I/O* (Ввод/вывод данных в рабочую область) – установка параметров обмена данными с рабочей областью MATLAB;
- *Diagnostics* (Диагностика) – выбор уровня диагностики.

Установка параметров управления модельным временем с помощью вкладки *Solver* рис.



Вкладка *Solver* окна установки параметров моделирования

Параметры моделирования разделены на три группы:

- *Simulation time* (интервал моделирования); величина интервала моделирования задается посредством указания начального (*Start time*) и конечного (*Stop time*) значений модельного времени;
- *Solver options* (параметры расчета) – выбор метода реализации (расчета) модели;
- *Output options* (Параметры вывода) – соответствующие элементы позволяют указать периодичность записи параметров модели в рабочую область (при моделировании с переменным шагом).

Под выбором метода расчета модели понимается следующее. Имея структуру исследуемой системы в виде блок-диаграммы, разработчик может в ходе моделирования выбирать метод отображения динамики системы. С помощью двух раскрывающихся списков *Type* (Тип) система может быть отнесена к одному из следующих классов:

- С дискретным состоянием и дискретным временем перехода из одного состояния в другое.
- С дискретным состоянием и непрерывным временем переходов.
- С непрерывным состоянием и дискретным временем переходов.
- С непрерывным состоянием и непрерывным временем переходов.

Первый список (слева) позволяет выбрать способ изменения модельного времени; он содержит два пункта:

- *Variable-step* (моделирование с переменным шагом);
- *Fixed-step* (моделирование с фиксированным шагом).

Как правило, *Variable-step* используется для моделирования непрерывных систем, а *Fixed-step* – дискретных.

Второй список (справа) позволяет выбрать метод расчета нового состояния системы. Первый вариант (discrete) обеспечивает расчет дискретных состояний системы (и для непрерывного, и для дискретного времени переходов из состояния в состояние).

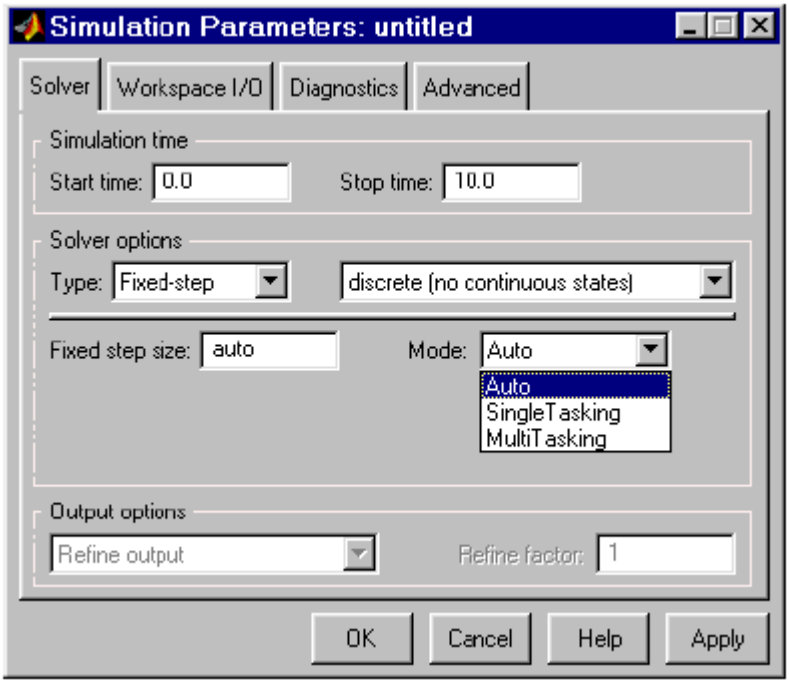
Остальные пункты списка обеспечивают выбор метода расчета нового состояния для непрерывных систем. Эти методы различны для переменного (*Variable-step)* и фиксированного (*Variable-step)* шага времени, но основаны на единой методике – решении обыкновенных дифференциальных уравнений (ODE).

Ниже двух раскрывающихся списков Type находится область, содержимое которой меняется в зависимости от выбранного способа изменения модельного времени. При выборе **Fixed-step** в данной области появляется текстовое поле Fixed-step size (величина фиксированного шага) позволяющее указывать величину шага моделирования (рис. 5.6). Величина шага моделирования по умолчанию устанавливается системой автоматически (auto). Требуемая величина шага может быть введена вместо значения auto либо в форме числа, либо в виде вычисляемого выражения (то же самое относится и ко всем параметрам, устанавливаемым системой автоматически).

При выборе Fixed-step необходимо также задать режим расчета (Mode). Для параметра Mode доступны три варианта:

- MultiTasking (Многозадачный) – необходимо использовать, если в модели присутствуют параллельно работающие подсистемы и результат работы модели зависит от временных параметров этих подсистем. Режим позволяет выявить несоответствие скорости и дискретности сигналов, пересылаемых блоками друг другу.
- SingleTasking (Однозадачный) – используется для тех моделей, в которых недостаточно строгая синхронизация работы отдельных составляющих не влияет на конечный результат моделирования.

Auto (Автоматический выбор режима) – позволяет Simulink автоматически устанавливать режим MultiTasking для тех моделей, в которых используются блоки с различными скоростями передачи сигналов, и режим SingleTasking для моделей, в которых содержатся блоки, оперирующие одинаковыми скоростями.



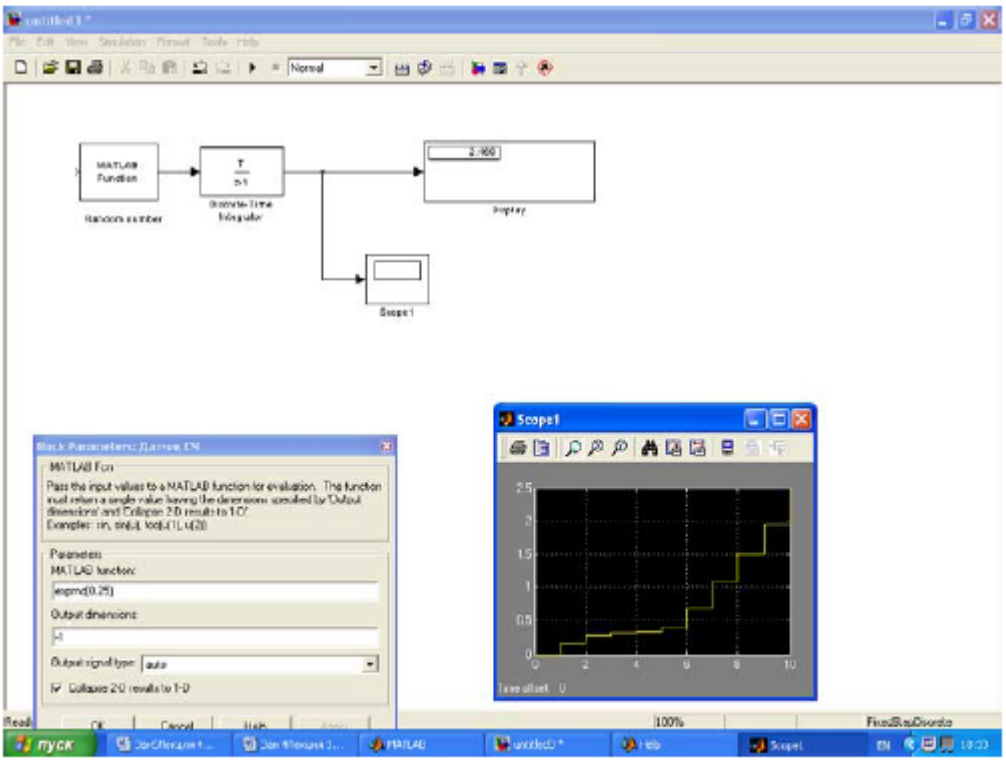
Вкладка Solver при выборе фиксированного шага расчета

При выборе Variable-step в области появляются поля для установки трех параметров:

- Max step size – максимальный шаг расчета. По умолчанию он устанавливается автоматически (auto) и его значение в этом случае равно (SfopTime – StartTime)/50. Довольно часто это значение оказывается слишком большим, и наблюдаемые графики представляют собой ломаные (а не плавные) линии. В этом случае величину максимального шага расчета необходимо задавать явным образом.
- Min step size – минимальный шаг расчета.
- Initial step size – начальное значение шага моделирования.

При моделировании непрерывных систем с использованием переменного шага необходимо указать точность вычислений: относительную (Relative tolerance) и абсолютную (Absolute tolerance). По умолчанию они равны соответственно 10^{-3} и auto.

Приведем пример моделирования потока заявок на обслуживание. Предположим, необходимо моделировать поток посетителей супермаркета, подходящих к кассе для оплаты покупки. Наблюдения показали (или из других источников поступила информация), что в вечернее время с 17.00 до 18.00 в среднем через каждые 15 мин. приходит очередной посетитель. Вопрос: как будет выглядеть очередность посетителей на протяжении данного промежутка времени? Модель потока посетителей может выглядеть следующим образом

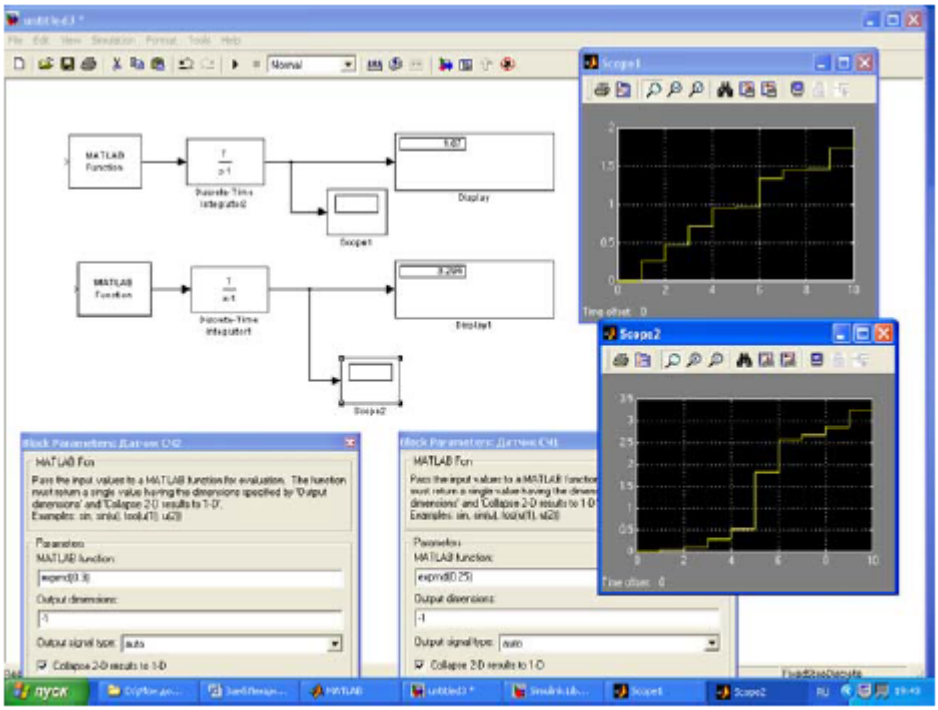


Модель потока посетителей

Блок MATLAB function имитирует случайные числа в соответствии с экспоненциальным законом распределения. С этой целью используется функция *exprnd* с параметром 0.25, что соответствует среднему времени 15 мин. (15/60). Блок Discrete Time Integrator суммирует случайные числа, т.е. формирует время работы кассы нарастающим итогом. Блок Display показывает суммарное время работы кассы (в нашем случае оно будет случайное). Блок Scope1 отражает: по вертикали – общее время работы, которое складывается из случайных временных интервалов; по горизонтали – количество посетителей. С этой осью совпадает шаг моделирования.

В случае моделирования потока покупателей и стоимости их покупки управление модельным временем осуществляется по особым состояниям.

Используя средства Simulink доработаем модель так, чтобы управлять величиной шага моделирования при изменении модельного времени по особым состояниям



Изменение модельного времени по особым состояниям

Нижняя часть модели обеспечивает формирование отрезков времени, длина которых распределена по заданному закону (экспоненциальному), и продвижение модельного времени. Верхняя часть модели имитирует случайную величину стоимости их покупки. Средняя стоимость покупки в модели принята за 0,3 тыс. руб. Из результатов одного прогона модели видно, что за 3 часа 20 минут в кассе супермаркета будет 1 тыс. 700 рублей. Понятно, что и моделируемая ситуация и сама модель очень просты. Но в данном случае для нас важен подход к решению подобных задач.

Синхронизация параллельных процессов

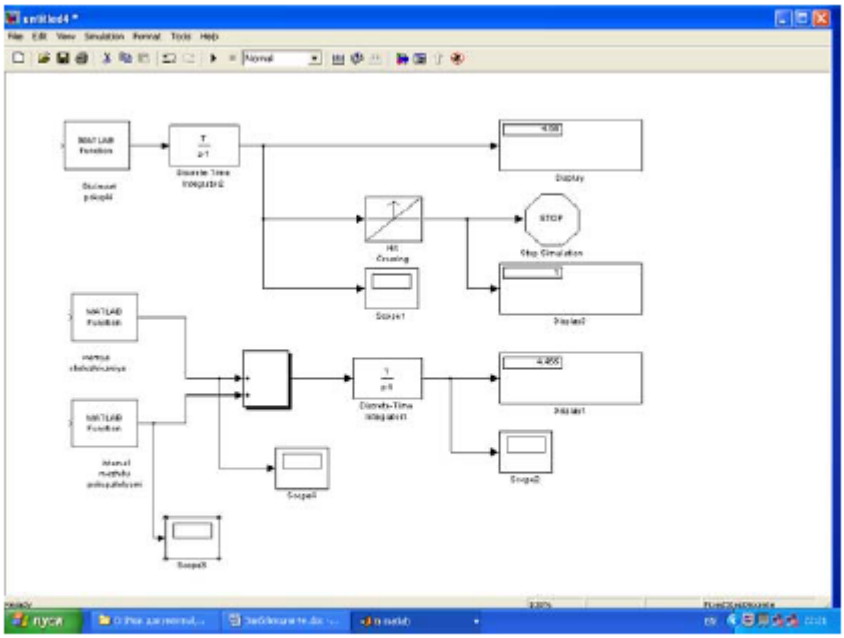
До этого мы рассматривали только асинхронные параллельные процессы, т.е. такие, которые не влияют друг на друга. «Привязку» таких процессов к единой оси модельного времени Simulink выполняет сам, освобождая от соответствующих проблем разработчика. Другое дело, когда имеют место синхронные параллельные процессы, состояние каждого из которых зависит от текущего состояния другого. При согласовании таких процессов должна учитываться специфика решаемой задачи, и без помощи разработчика Simulink здесь уже обойтись не может.

Для корректного управления модельным временем в таких моделях необходимо:

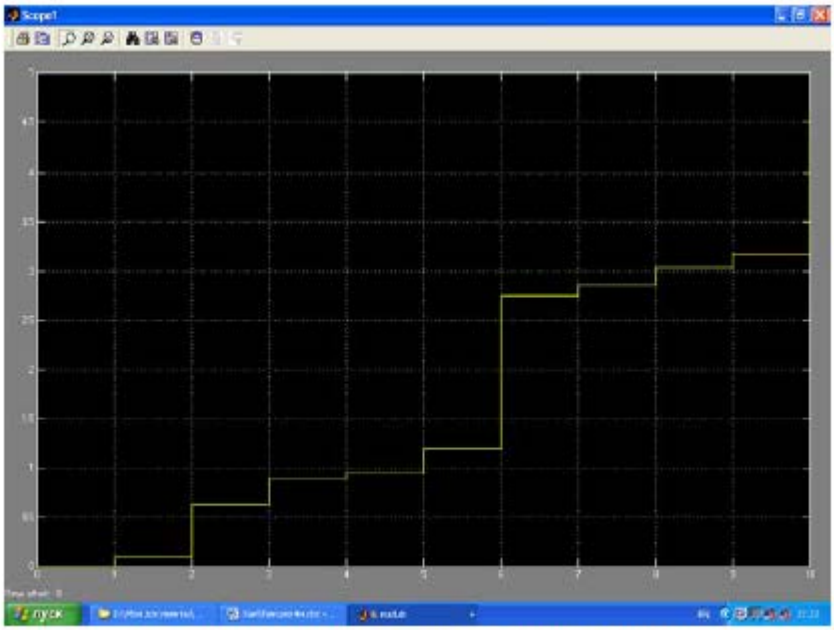
1. Установить, какой из взаимодействующих процессов является подчиненным по отношению к другому.
2. Определить, могут ли в подчиненном процессе происходить события, не связанные с изменением состояния управляющего процесса.
3. Обеспечить приращение модельного времени на интервал времени до ближайшего события в управляющем процессе.
4. Контролировать условия окончания сеанса моделирования.

Пример. В качестве иллюстрации к сказанному вновь воспользуемся моделью работы супермаркета. Очевидно, что оплата покупателя в кассу может начаться только при условии под-

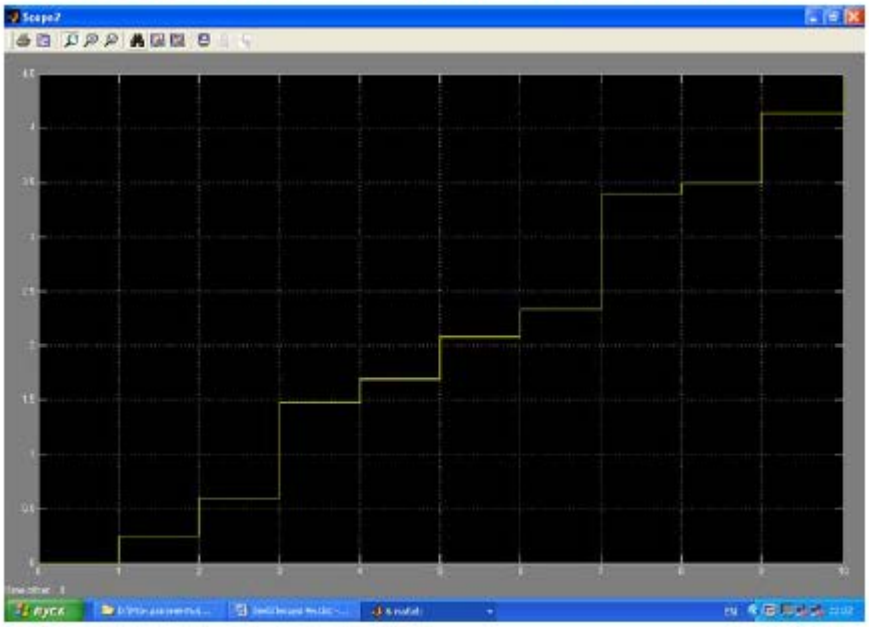
хода его к кассе. Потому процесс оплаты является подчиненным по отношению к процессу подхода покупателей к кассе. Если интервал между покупателями значительно больше длительности расчета в кассе, то продвижение модельного времени будет определяться только событиями управляющего процесса (потоком покупателей). Блок-диаграмма такой системы не будет отличаться от рассмотренной ранее (см. рис.). Если же интервал между покупателями в кассе соизмерим с временем обслуживания в кассе, то необходимо отразить в модели дополнительное условие: «обслуживание» очередного покупателя не может начаться до тех пор, пока не завершится «обслуживание» предыдущего. При такой постановке задачи оба процесса становятся равноправными с точки зрения влияния на значение модельного времени. Очередной шаг модельного времени в этом случае вычисляется как сумма двух временных интервалов: промежутка до нового покупателя и длительности обслуживания его в кассе. Модель будет иметь вид, показанный на рис. Результаты работы модели представлены на рис.



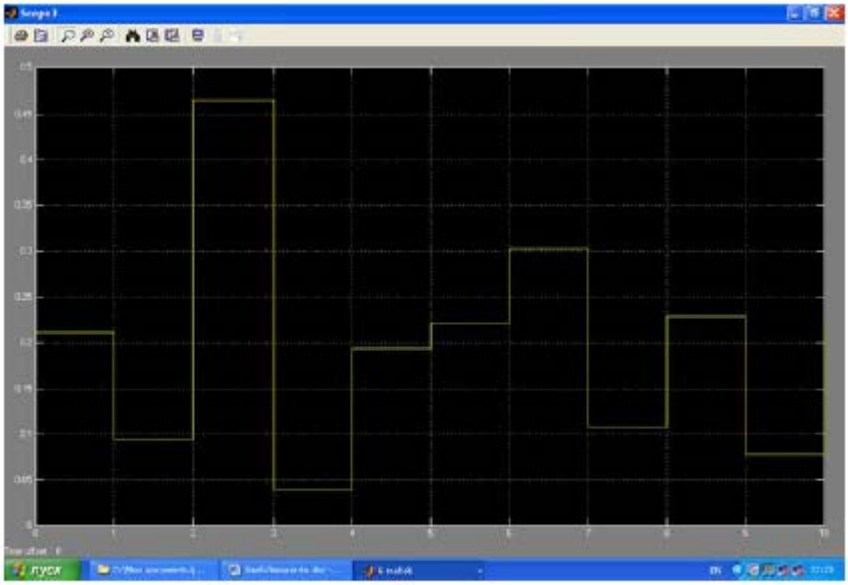
Управление модельным временем при наличии двух синхронных процессов



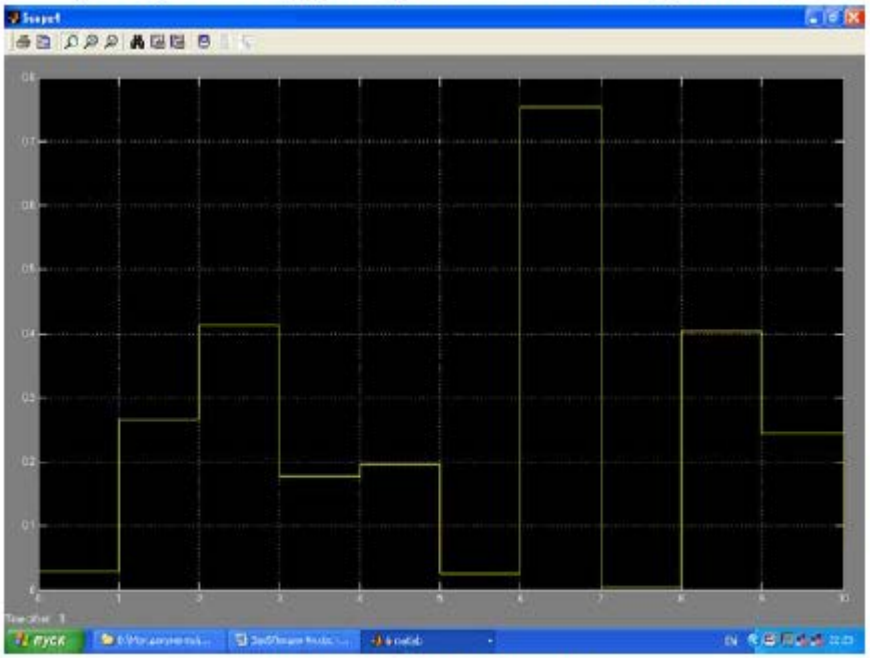
Результат работы модели (поступление денег в кассу)



Результаты работы модели (суммарный интервал между покупателями и время обслуживания покупателя – по вертикали)

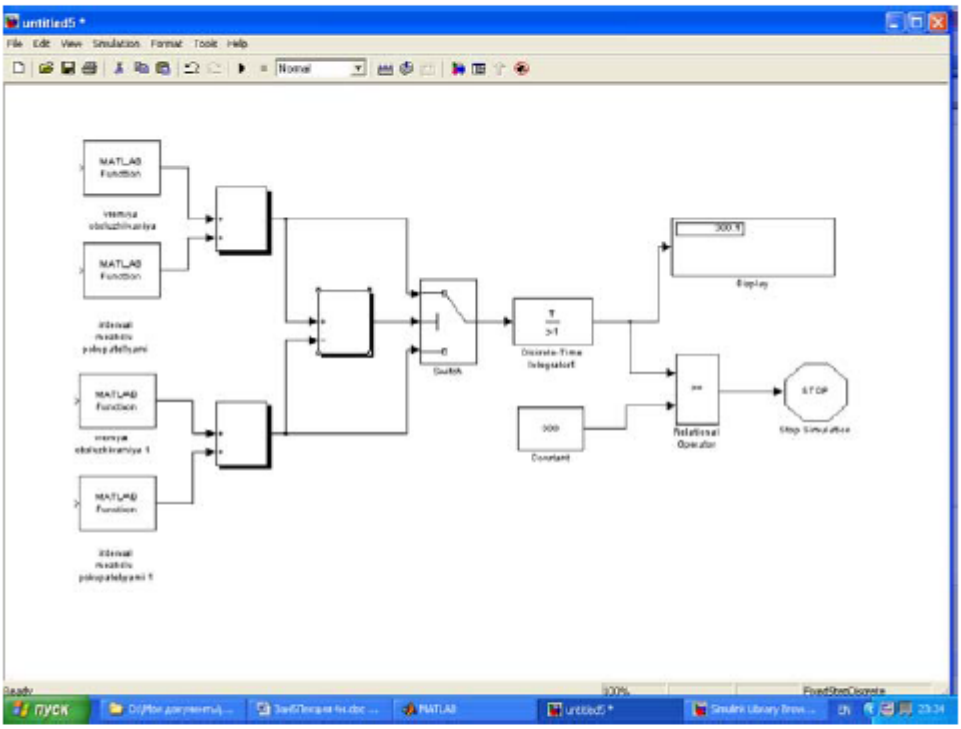


Результат работы модели
(интервал между покупателями – по вертикали)



Результат работы модели
(время обслуживания покупателя – по вертикали)

При моделировании аналогичного процесса, но с двумя кассами, необходимо учитывать, что они работают одновременно и совершенно независимо друг от друга, т.е. асинхронно. Предположим, что сеанс моделирования должен быть остановлен по истечении 300 единиц модельного времени. Модель будет иметь вид, представленный на рис.



Управление окончанием моделирования при наличии двух параллельных процессов

В данной модели используется способ продвижения модельного времени, суть которого состоит в том, что в каждом такте выбирается бóльшая из величин T1 и T2. Где T1 и T2 – время суммарного обслуживания очередного покупателя в 1–й и во 2-й кассе соответственно.

При этом исходим из того, что суммарное время обслуживания каждого покупателя состоит из двух составляющих:

- Ожидание (в течение времени $t_{ож}$);
- Обслуживание в кассе ($t_{обсл}$).

$$T1 = t_{ож1} + t_{обсл1}$$
$$T2 = t_{ож2} + t_{обсл2}.$$

Лекция 6
Планирование модельных экспериментов.

Планирование модельных экспериментов

Цели планирования экспериментов

Для правильной организации модельного эксперимента исследователь должен располагать следующей информацией:

1) к какому классу относится моделируемая система (статическая или динамическая, детерминированная или стохастическая и т.д.);

2) какой режим работы системы его интересует: стационарный (установившийся) или нестационарный;

3) в течение какого промежутка времени следует наблюдать за поведением (функционированием) системы;

4) какой объем испытаний (т.е. повторных экспериментов) сможет обеспечить требуемую точность оценок (в статистическом смысле) исследуемых характеристик системы.

Разумеется, можно пойти по такому пути: не особенно задумываясь над перечисленными вопросами, взять от модели все «по максимуму» – исследовать работу системы во всех режимах, для всех возможных сочетаний внешних и внутренних параметров и повторять каждый эксперимент по сотне раз. Однако польза от такого моделирования невелика, поскольку полученные данные будет очень сложно обработать и проанализировать, а еще труднее принять с их помощью какое-либо конкретное решение. Да и затраты времени на моделирование, даже с учетом быстродействия современных компьютеров, окажутся чрезмерными.

Таким образом, планирование модельных экспериментов преследует две основные цели:

- сокращение общего объема испытаний при соблюдении требований к достоверности и точности их результатов;
- повышение информативности каждого из экспериментов в отдельности.

Поиск плана эксперимента производится в так называемом факторном пространстве.

Факторное пространство – это множество внешних и внутренних параметров модели, значения которых исследователь может контролировать в ходе подготовки и проведения модельного эксперимента.

Во многих случаях факторы могут носить не только количественный, но и качественный характер. Поэтому значения факторов обычно называют уровнями. Если при проведении эксперимента исследователь может изменять уровни факторов, эксперимент называется активным, в противном случае – пассивным.

Введем еще несколько терминов, используемых в теории планирования эксперимента. Каждый из факторов имеет верхний и нижний уровни, расположенные симметрично относительно некоторого нулевого уровня. Точка в факторном пространстве, соответствующая нулевым уровням всех факторов, называется *центром плана*.

Интервалом варьирования фактора называется некоторое число, прибавление которого к нулевому уровню дает верхний уровень, а вычитание – нижний. Как правило, план эксперимента строится относительно одного (основного) выходного скалярного параметра Y, который называется *наблюдаемой переменной*. Если моделирование используется как инструмент принятия решения, то в роли наблюдаемой переменной выступает показатель эффективности. При этом предполагается, что значение наблюдаемой переменной, полученное в ходе эксперимента, складывается из двух составляющих

$$Y = f(x) + e(x),$$

где $f(x)$ – функция отклика (неслучайная функция факторов);

$e(x)$ – ошибка эксперимента (случайная величина);

x – точка в факторном пространстве (определенное сочетание уровней факторов).

Очевидно, что y является случайной переменной, так как зависит от случайной величины $e(x)$.

Дисперсия Dy наблюдаемой переменной, которая характеризует точность измерений, равна дисперсии ошибки опыта: $Dy = De$.

Dy называют дисперсией воспроизводимости эксперимента. Она характеризует качество эксперимента. Эксперимент называется идеальным при $Dy = 0$.

Существует два основных варианта постановки задачи планирования имитационного эксперимента:

1. Из всех допустимых выбрать такой план, который позволил бы – получить наиболее достоверное значение функции отклика $f(x)$ при фиксированном числе опытов.

2. Выбрать такой допустимый план, при котором статистическая оценка функции отклика может быть получена с заданной точностью при минимальном объеме испытаний.

Решение задачи планирования в первой постановке называется стратегическим планированием эксперимента, во второй – тактическим планированием.

Стратегическое планирование имитационного эксперимента

Итак, цель методов стратегического планирования имитационных экспериментов – получение максимального объема информации об исследуемой системе в каждом эксперименте (наблюдении). Другими словами, стратегическое планирование позволяет ответить на вопрос, при каком сочетании уровней внешних и внутренних факторов может быть получена наиболее полная и достоверная информация о поведении системы.

При стратегическом планировании эксперимента должны быть решены две основные задачи:

1. Идентификация факторов.

2. Выбор уровней факторов.

Под *идентификацией факторов* понимается их ранжирование по степени влияния на значение наблюдаемой переменной (показателя эффективности).

По итогам идентификации целесообразно разделить все факторы на две группы – первичные и вторичные. *Первичные* –

это те факторы, в исследовании влияния которых экспериментатор заинтересован непосредственно. *Вторичные* – факторы, которые не являются предметом исследования, но влиянием которых нельзя пренебречь.

Выбор уровней факторов производится с учетом двух противоречивых требований:

- уровни фактора должны перекрывать (заполнять) весь возможный диапазон его изменения;
- общее количество уровней по всем факторам не должно приводить к чрезмерному объему моделирования. Поиск компромиссного решения, удовлетворяющего этим требованиям, и является задачей стратегического планирования эксперимента.

Способы построения стратегического плана

Эксперимент, в котором реализуются всевозможные сочетания уровней факторов, называется *полным факторным экспериментом* (ПФЭ).

Общее число различных комбинаций уровней в ПФЭ для k-факторов можно вычислить следующим образом:

$$N = l_1 \cdot l_2 \cdot l_3 \cdot \ldots \cdot l_k,$$

где l_k – число уровней k-го фактора.

Если число уровней для всех факторов одинаково, то $N = L^k$ (L – число уровней).

Недостаток ПФЭ – большие временные затраты на подготовку и проведение.

Например, если в модели отражены 3 фактора, влияющие на значение выбранного показателя эффективности, каждый из которых имеет 4 возможых уровня (значения), то план проведения ПФЭ будет включать 64 эксперимента ($N = 4^3$). Если при этом каждый из них длится хотя бы одну минуту (с учетом времени на изменение значений факторов), то на однократную реализацию ПФЭ потребуется более часа.

Поэтому использование ПФЭ целесообразно только в том случае, если в ходе имитационного эксперимента исследуется взаимное влияние всех факторов, фигурирующих в модели.

Если такие взаимодействия считают отсутствующими или их эффектом пренебрегают, проводят *частичный факторный эксперимент* (ЧФЭ).

Известны и применяются на практике различные варианты построения планов ЧФЭ. Мы рассмотрим только некоторые из них.

1. *Рандомизированный план* – предполагает выбор сочетания уровней для каждого прогона случайным образом. При использовании этого метода отправной точкой в формировании плана является число экспериментов, которые считает возможным (или необходимым) провести исследователь.

2. *Латинский план* (или «латинский квадрат») – используется в том случае, когда проводится эксперимент с одним первичным фактором и несколькими вторичными. Суть такого планирования состоит в следующем. Если первичный фактор А имеет *l* уровней, то для каждого вторичного фактора также выбирается *l* уровней. Выбор комбинации уровней факторов выполняется на основе специальной процедуры, которую мы рассмотрим на примере.

Пусть в эксперименте используется первичный фактор А и два вторичных фактора – В и С, число уровней факторов *l* равно 4. Соответствующий план можно представить в виде квадратной матрицы размером *l* x *l* (4 x 4) относительно уровней фактора А. При этом матрица строится таким образом, чтобы в каждой строке и в каждом столбце данный уровень фактора А встречался только один раз.

В результате имеем план, требующий 4 x 4 = 16 прогонов, в отличие от ПФЭ, для которого нужно $4^3 = 64$ прогона.

3. *Эксперимент с изменением факторов по одному.* Суть его состоит в том, что один из факторов «пробегает» все l уровней, а остальные n-1 факторов поддерживаются постоянными. Такой план обеспечивает исследование эффектов каждого фактора в отдельности. Он требует всего $N = l_1 + l_2 + l_3 + \dots l_k$ прогонов.

Для рассмотренного выше примера (3 фактора, имеющие по 4 уровня) N = 4 + 4 + 4 = 12.

Еще раз подчеркнем, что такой план применим (как и любой ЧФЭ) только при отсутствии взаимодействия между факторами.

4. Дробный факторный эксперимент. Каждый фактор имеет два уровня – нижний и верхний, поэтому общее число вариантов эксперимента $N = 2^k$, где k – число факторов. Матрицы планов для $k = 2$ и $k = 3$ приведены ниже.

Планы, построенные по такому принципу, обладают определенными свойствами (симметричность, нормированность, ортогональность и ротатабельность), обеспечивающими повышение качества проводимых экспериментов.

Пример латинского плана

Значение фактора B	**Значение фактора C**			
	C1	C2	C3	C4
B1	A1	A2	A3	A4
B2	A2	A3	A4	A1
B3	A3	A4	A1	A2
B4	A4	A1	A2	A3

Матрица плана дробного факторного эксперимента для k = 2

Номер эксперимента	**Значение факторов**	
	x1	*x2*
1	0	0
2	0	1
3	1	0
4	1	1

Матрица плана дробного факторного эксперимента для k = 3

Номер эксперимента	Значение факторов		
	x1	*x2*	*x3*
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Тактическое планирование эксперимента

Совокупность методов установления необходимого объема испытаний относят к тактическому планированию экспериментов.

Поскольку точность оценок наблюдаемой переменной характеризуется ее дисперсией, то основу тактического планирования эксперимента составляют так называемые методы понижения дисперсии. В связи с этим для восприятия последующего материала потребуются некоторые знания математической статистики.

Формирование простой случайной выборки

Поскольку имитационное моделирование представляет собой статистический эксперимент, то при его проведении необходимо не только получить достоверный результат, но и обеспечить его «измерение» с заданной точностью.

Различие понятий «достоверный результат» и «точный результат» можно пояснить с помощью приведенного ниже рисунка. На рисунке использованы следующие обозначения:

y, y_0, – истинное и ошибочное значения наблюдаемой переменной y;

b, b_0 – доверительные интервалы измерения величин y и y_0.

В общем случае объем испытаний (величина выборки), необходимый для получения оценок наблюдаемой переменной с заданной точностью, зависит от следующих факторов:

- вида распределения наблюдаемой переменной y (напомним, при статистическом эксперименте она является случайной величиной);
- коррелированности между собой элементов выборки;
- наличия и длительности переходного режима функционирования моделируемой системы.

Если исследователь не обладает перечисленной информацией, то у него имеется единственный способ повышения точности оценок истинного значения наблюдаемой переменной – многократное повторение прогонов модели для каждого сочетания уровней факторов, выбранного на этапе стратегического планирования эксперимента. Такой подход получил название «формирование простой случайной выборки» (ПСВ). Другими словами, при использовании ПСВ каждый «пункт» стратегического плана просто выполняется повторно определенное число раз. При таком подходе общее число прогонов модели, необходимое для достижения цели моделирования, равно произведению $N_C \times N_T$ (N_C – число сочетаний уровней факторов по стратегическому плану; N_T – число прогонов модели для каждого сочетания, вычисленное при тактическом планировании).

Например, если для полного факторного эксперимента $N_C = 64$, а для обеспечения требуемой точности оценок N_T должно быть равно 20, то общее число прогонов модели – 1280. Требуемое время для проведения всех испытаний (по минуте на каждое) – более 20 часов. То есть «модельер» должен трудиться почти сутки без сна и отдыха. Поэтому даже при использовании ПСВ до начала испытаний необходимо определить тот минимальный объем выборки, который обеспечит требуемую точность результатов.

Если случайные значения наблюдаемой переменной не коррелированы и их распределение не изменяется от прогона к прогону, то выборочное среднее можно считать нормально распределенным. В этом случае число прогонов N_T, необходимое для того чтобы истинное среднее наблюдаемой переменной лежало в интервале $y \pm b$ с вероятностью $(1 - \alpha)$, определяется следующим образом:

$$N_T = \frac{Z^2 \cdot D_y}{b^2},$$

где Z – значение нормированного нормального распределения, которое определяется по справочной таблице при заданном уровне значимости $\alpha / 2$;

Dy – дисперсия;

b – доверительный интервал.

Если требуемое значение дисперсии Dy до начала эксперимента неизвестно, целесообразно выполнить пробную серию из L прогонов и вычислить на ее основе выборочную дисперсию, значение которой подставляют в вышеприведенную формулу и получают предварительную оценку числа прогонов модели N_T, затем выполняют оставшиеся $N_T - L$ прогонов, периодически уточняя оценку и число прогонов N_T.

Возможности Matlab/Simulink по планированию и реализации модельных экспериментов

Разработка планов экспериментов

Планирование экспериментов осуществляется с помощью М-функции Design of Experiments приложения Statistics toolbox.

В состав раздела Design of Experiments входят функции, обеспечивающие разработку всех трех основных видов стратегического плана эксперимента:

- Полного факторного эксперимента (ПФЭ);
- Частичного факторного эксперимента (ЧФЭ);
- Дробного факторного эксперимента (ДФЭ).

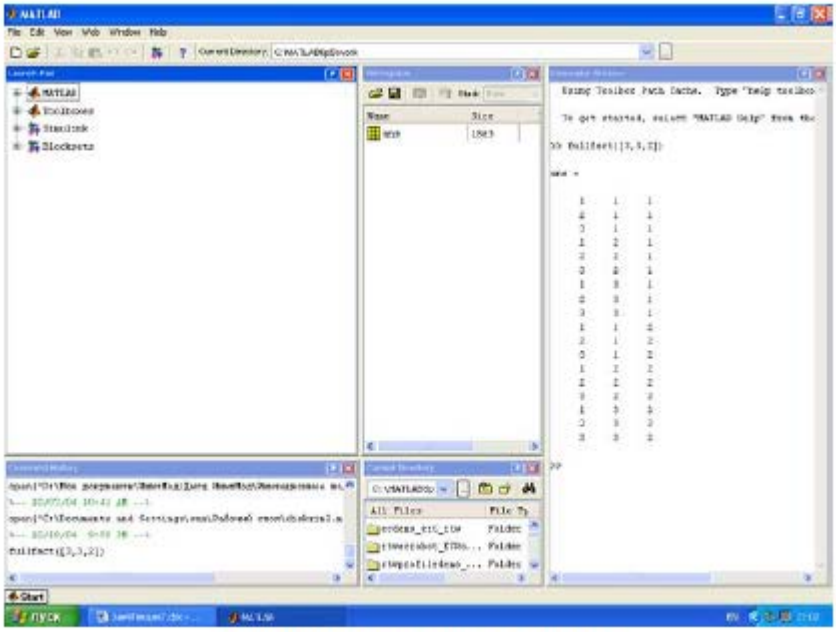
Для разработки **ПФЭ** служит функция *fullfact*. В качестве ее параметров необходимо указать число уровней каждого фактора, участвующего в эксперименте.

Например, если факторы А и В имеют 3 уровня, а фактор С 2 уровня, то обращение к функции *fullfact* выглядит так:

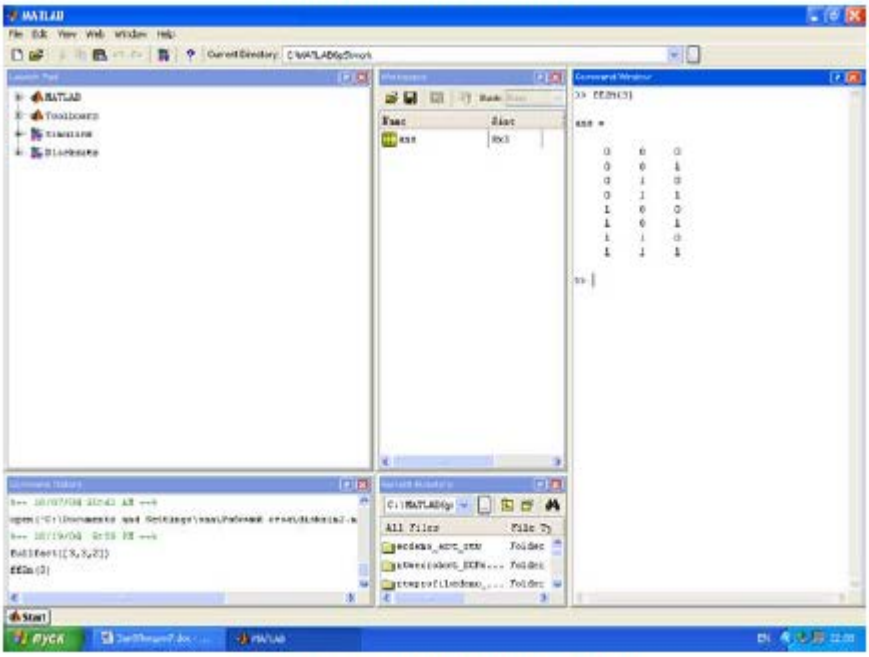
fullfact([3,3,2]).

Введя указанную конструкцию в командном окне *Matlab*, можно получить список всех возможных комбинаций уровней факторов. Список выводится в командном окне, а также сохраняется в рабочей области под именем *ans* (рис.). Он может быть использован в качестве подсказки либо в текущем сеансе работы с *Matlab*, либо записан в отдельный МАТ-файл для последующего применения.

Для формирования плана **ДФЭ** используется функция *ff2n*. Параметром функции является число факторов. Например, команда *ff2n(3)* обеспечивает вывод в командное окно следующего списка



Формирование плана ПФЭ



Формирование плана ДФЭ

Формирование **ЧФЭ** (рандомизированного плана) осуществляется с помощью функции *unidrnd*, представляющей собой генератор дискретной СВ, равномерно распределенной на интервале [1; N]. В общем случае она используется с тремя параметрами

unidrnd (N, k, m),

где *N* – верхняя граница интервала распределения;

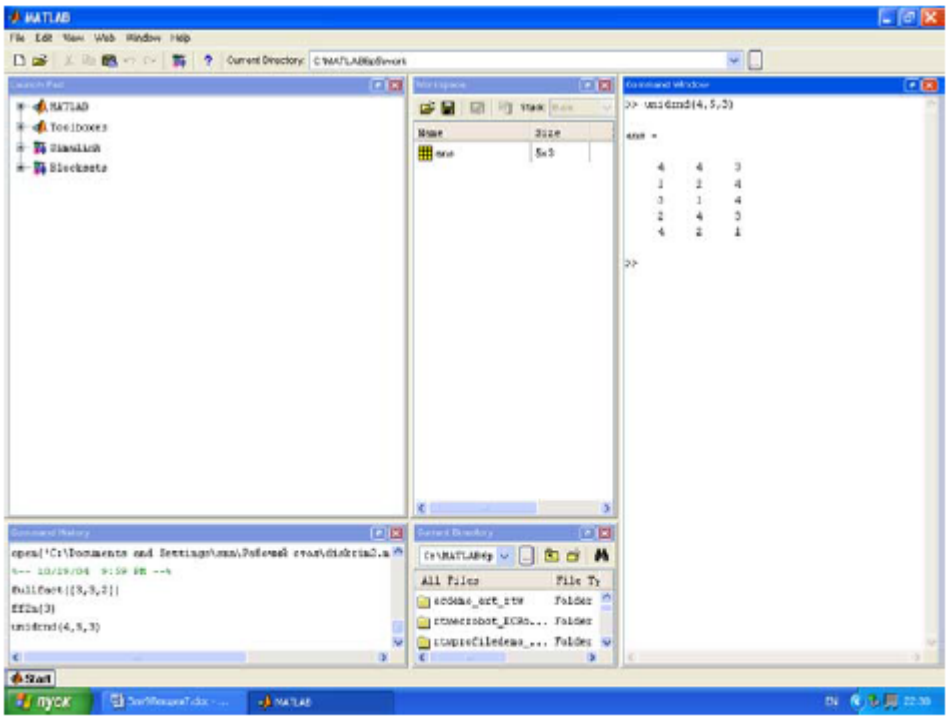
k, m – задают число строк и столбцов генерируемой случайной матрицы.

При генерации плана эксперимента эти величины интерпретируются следующим образом:

N – число уровней факторов, участвующих в эксперименте;

k – выбранное пользователем число экспериментов (различных сочетаний уровней факторов);

m – число факторов



Формирование рандомизированного плана эксперимента

Проведение имитационных экспериментов с использованием файлов сценариев

Процесс прогона модели при различных значениях параметров (факторов), обработку полученных результатов моделирования можно автоматизировать путем использования файлов сценариев. Matlab располагает механизмом, позволяющим создавать и сохранять устойчивые сценарии в виде специальных М-файлов, которые так и называются – файлы сценариев (Script files), или просто М-сценарии.

М-сценарий представляет собой последовательность команд (или операторов) Matlab, разделенных точкой с запятой (если они записаны в одной строке).

При написании М-сценариев следует учитывать следующее:

- М-сценарий не имеет входных параметров (аргументов);
- М-сценарий может содержать любые М-функции и операторы Matlab;
- Входящие в сценарий М-функции и операторы могут оперировать с данными, находящимися в рабочей области Matlab.

Основным инструментом разработки как М-сценариев, так и М-функций является Редактор/Отладчик Matlab – Editor/Debugger, хотя для этих целей может быть использован любой текстовый редактор.

Для улучшения визуального восприятия текста М-файла его различные компоненты имеют в окне Редактора/Отладчика разный цвет:

- комментарий – зеленый;
- ключевые слова Matlab – синий;
- остальные конструкции – черный.

Порядок использования команд рассмотрим на примере создания сценария, обеспечивающего запуск модели и построение графиков.

```
1. % Optimal profit tax rate simulation
2. % File: C:\Csr_MtLb\TxRt\TaxRate_DscM.m and TaxRate_
   Dsc.mdl
3. open_system(«TaxRate_Dsc») % Load TaxRate_Dsc.mdl
4. TaxRate=[0:0.05:0.7]% План-вектор эксперимента по став-
   ке налога
5. for Rntb=0.2:0.2:1 % Цикл и план-вектор по рентабельности
6. sim(«TaxRate_Dsc»)% Run model
7. plot(TaxRate, ScopeData(end,2:end)) %Чертить график по-
   ступления в бюджет
8. hold on% Разрешить дополнение графика кривыми
9. grid% Чертить сетку
10. end
11. hold off% Запретить дополнение графика
```

В m-файле программы за знаком процента всегда идут поясняющие комментарии. Они не являются командами и компьютером не исполняются.

В первой строке программы дается ее назначение, или смысловое название.

Во второй строке – полное имя m-файла, содержащего нашу программу для управления экспериментом над Simulink моделью, и имя файла Simulink модели с расширением .mdl.

Третья строка командой open_system загружает с диска модель в оперативную память.

Четвертая строка присваивает переменной модели TaxRate вектор плана эксперимента по налоговой ставке.

В строках с 5 по 10 выполняется for цикл для проведения экспериментов при различных величинах рентабельности бизнеса.

В шестой строке командой sim запускается модель и начинается моделирование, имитация налогового взаимодействия государства и предприятия.

После окончания имитации команда plot чертит один график, используя данные рабочего (work space) пространства Matlab, записанные туда графопостроителем Scope. Оператор hold on разрешает дополнять рисунок графиками кривых, рассчитанных для других значений циклов рентабельности.