Моделирование систем массового обслуживания в среде MATLAB + Simulink + Stateflow

Цель работы: практическое изучение технологий визуального программирования моделей с использованием подсистемы Simulink и пакета расширения Stateflow; освоение навыков проведения экспериментальных исследований с моделями систем массового обслуживания в интересах оценки их эффективности и влияния основных факторов.

Работа выполняется в среде MATLAB и оформляется в виде m-файла управляющей программы, содержащей обращение к mdl-файлу S-модели с вложенной SF-моделью (SF-диаграммой, SF-машиной) одной из возможных конфигураций СМО. Задания на выполнение работы предполагают проведение исследований различных типов и структур СМО с использованием разнородных показателей для оценки эффективности в зависимости от основных факторов влияния.

Роль УП, как и ранее, состоит в задании варьируемых и неварьируемых факторов, проведении стратегического и тактического планирования модельного эксперимента, реализации процедур статистической обработки результатов моделирования. Роль S-модели заключается в обеспечении динамики процесса функционирования SF-модели и регистрации его результатов на основе использования стандартных элементов подсистемы Simulink. Наконец, роль SF-модели (Stateflow-модели) состоит, собственно, в воспроизведении визуальной модели СМО с использованием формализма гибридных автоматов (карт состояний Харела).

Первоначально при выполнении работы проводится ознакомление с особенностями и возможностями встроенного в подсистему Simulink пакета Stateflow на примере простейших событийно управляемых систем. Далее осуществляется полномасштабная разработка SF-модели в соответствующей оболочке S-модели, в совокупности реализующих моделирование СМО заданного типа. В заключительной части работы осуществляется формирование УП в виде m-файла, реализующего, если это предусматривается заданием, план модельного эксперимента и

многократный прогон S-модели с регистрацией результатов для последующей оценки эффективности. Структура управляющей программы и используемые в ней синтаксические конструкции имеют аналогичный ранее рассмотренным в п. 6.2, 6.3 вид. При формировании этой программы осуществляется сопряжение с S-моделью и окончательная настройка ее блоков и модулей с учетом выбранных условий моделирования. Однако на начальном этапе удобно разработать упрощенный вариант УП, обеспечивающий предварительное тестирование S-модели с запуском из MATLAB.

Перед началом выполнения работы в соответствующем разделе создается рабочая папка. После запуска MATLAB данная папка устанавливается в окне «Current Directory» путем выбора из списка рабочих папок файловой системы.

Как уже отмечалось, первоначально целесообразно разработать m-файл упрощенного варианта УП, реализующий известные формы обращения к S-модели и задания ее основных параметров. Его примерный вид представлен на рис. 6.19 в стандартном окне редактора MATLAB.

Далее создается S-модель с вложенной SF-моделью. Для этого в соответствии с описанной в п. 6.3 последовательностью действий осуществляется запуск Simulink и создается пустое окно нового mdl-файла Untitled (впоследствии сохраняемого в различных модификациях под именем testo**.mdl). Осуществляется, как и ранее, установка общих параметров модели Configuration Parameters, однако теперь в поле Solver группы Solver options целесообразно установить метод «discrete», так как будет проводиться моделирование системы с набором дискретных основных состояний.

Следующим шагом является открытие раздела Stateflow в браузере библиотек Simulink. В результате в основном окне просмотра библиотеки появится единственный блок Chart (диаграмма), который с помощью мыши может быть перемещен в рабочее окно блок-диаграммы mdl-файла Simulink. Здесь также можно открыть собственную библиотеку Stateflow, щелкнув по наименованию раздела правой кнопкой мыши. В результате появится надпись Open the Stateflow library, щелкнув по которой левой кнопкой, получим дополнительно к блоку Chart подраздел Examples, содержащий примеры разработанных с помощью данного инструментария моделей систем.

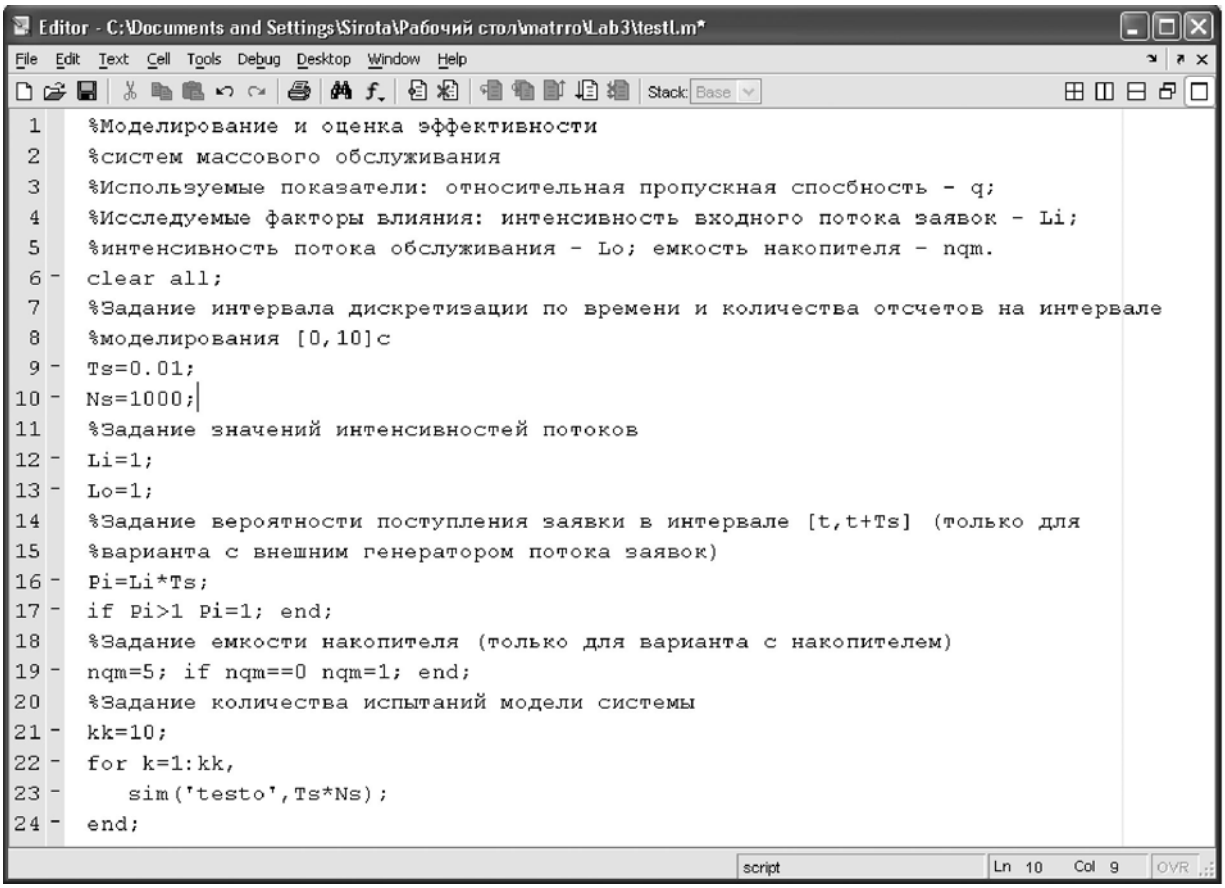


Рис. 6.19

Первоначально блок Chart, размещенный в рабочем окне mdl-файла, является пустым, то есть служит заготовкой для создания модели. Если активизировать мышью пустой блок Chart, то появится окно редактора SF-диаграммы (см. рис. 6.20), в котором можно начать сборку модели. Редактор имеет обычный для окон системы MATLAB вид.

Основные средства построения модели сосредоточены в пунктах меню File, Add, Tools, а также расположенной слева и сбоку панели инструментов, содержащей 8 кнопок (для версии MATLAB 7.0).

Все объекты SF-модели делятся на графические и неграфические. Для создания основных графических объектов используются кнопки боковой панели. Наиболее употребляемыми являются верхние четыре кнопки. Нажав каждую из кнопок, можно мышью перетащить в любое место окна соответствующий графический элемент модели, как это показано на рис. 6.20.

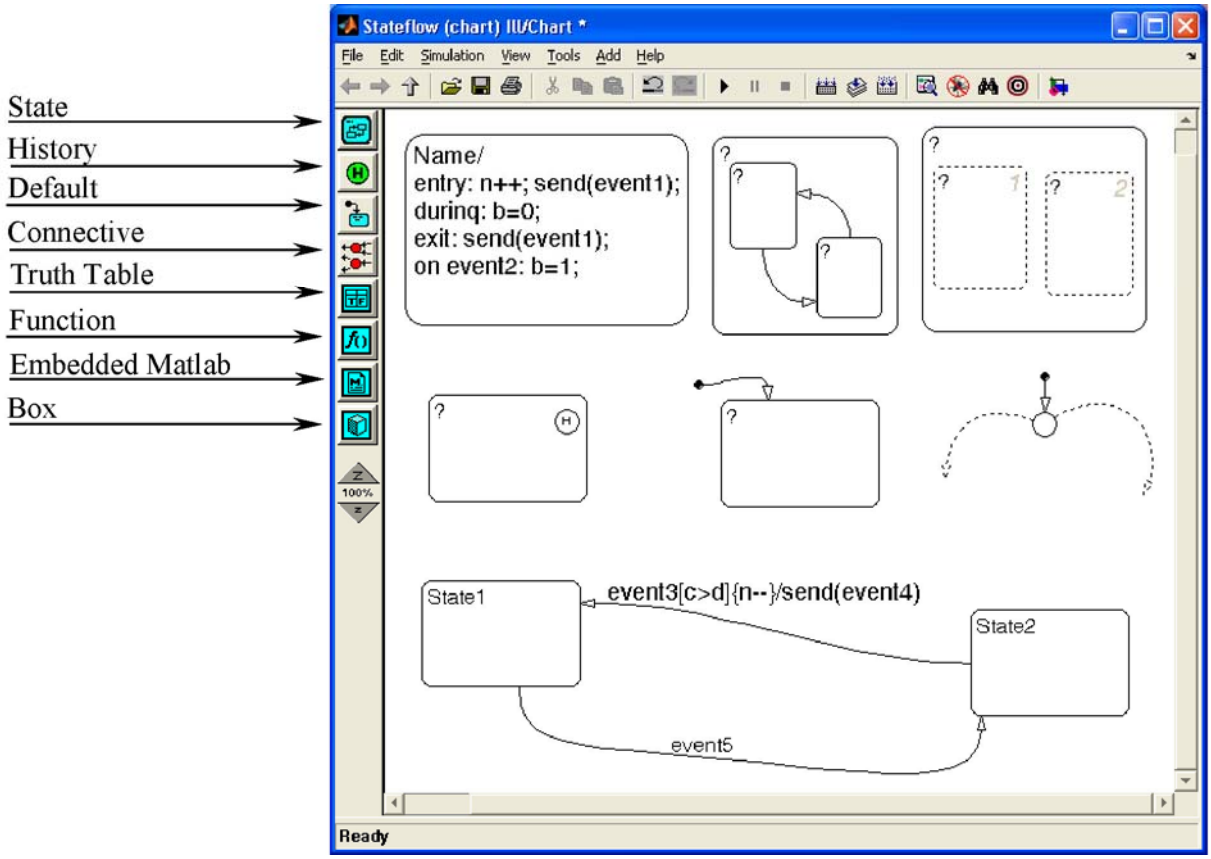


Рис. 6.20

Кнопки имеют следующее назначение:

кнопка State обеспечивает установку всех необходимых состояний модели в виде блоков, имеющих вид прямоугольников со скругленными углами;

кнопка History Junction обеспечивает установку в любом блоке состояния признака состояния с памятью в виде кружка с вложенной буквой H;

кнопка Default Transition реализует установку перехода по умолчанию в виде стрелки с черным кружком в начале, что позволяет указать начальное состояние в случае, если имеется неоднозначность между взаимоисключающими состояниями одного уровня иерархии SF–диаграммы;

кнопка Connective Junction устанавливает признак альтернативных переходов без захода в состояния в виде пустого кружка, что, в ряде случаев, существенно упрощает построение модели;

кнопка Truth Table обеспечивает программирование и включение в модель специальных функций – таблиц истинности, реализующих описание сложного логического поведения элементов модели и устанавливаемых изначально в виде блоков – прямоугольников с надписью truthtable и окошком для указания имени таблицы;

кнопка Function обеспечивает программирование и включение в модель графических функций, определенных графом потока вычислений для описания реализуемого алгоритма и устанавливаемых в виде блоков – прямоугольников с заготовкой для имени функции «function»;

кнопка Embedded Matlab function обеспечивает программирование и включение в модель функций на языке MATLAB, устанавливаемых в виде прямоугольников с надписью eM и окошком для указания имени функции;

кнопка Box (ящик) обеспечивает при необходимости группирование элементов модели в блоки (подсистемы), первоначально (после перетаскивания мышью) имеющих вид пустых прямоугольников.

Помимо указанных графических объектов, в SF-диаграмме используются переходы (Transition); они не имеют своей кнопки и отображаются в виде стрелок, идущих от одного объекта к другому. Переходы формируются с помощью мыши путем ее перемещения из любой точки границы одного графического объекта к любой точке границы другого. Для графических обьектов – состояний допускается введение внутренних переходов. Эти переходы начинаются или заканчиваются на внутренних границах состояния-источника.

К числу основных неграфических объектов SF-диаграммы относятся **события, данные и действия**.

События (Event) управляют работой SF-диаграммы и должны быть определены пользователем. События имеют различные области видимости (Scope):

локальные (видимые только в пределах SF-диаграммы или ее состояний);

входные (передаваемые в SF-диаграмму из Simulink);

выходные (передаваемые в Simulink из SF-диаграммы);

экспортируемые (передаваемые во внешнюю УП);

импортируемые (передаваемые из внешней УП).

События создаются путем выбора пункта меню Add – Event и далее последовательности Event – Local, Event – Input from Simulink и т.д. В результате появляется диалоговое окно Event, где определяется конкретное имя события и его характеристики, в перечне которых важнейшей является установка вида события в поле Trigger. В большинстве случаев события могут рассматриваться как скачкообразные изменения (переключения) состояний, сопровождающиеся резкими изменениями уровня соответствующих сигналов; при этом могут быть события следующего вида: Rising – с повышением уровня сигнала; Falling – с понижением уровня; Either – с любым направлением изменения уровня. События могут также создаваться и модифицироваться в диалоговом окне обозревателя Stateflow Explorer, выбираемом в рамках пункта основного меню Tools – Explorer.

Другими основными неграфическими объектами SF-модели являются **данные** (Date). Они представляют числовые значения переменных, используемых в модели. Данные имеют свойства (классические типы), а также область видимости (Scope). Как и события, данные могут быть локальными, входными, выходными, импортируемыми и экспортируемыми. Кроме того, данные могут определяться как константы и временные (промежуточные) переменные. Для создания и модификации данных следует воспользоваться пунктом меню Add – Date, а также обозревателем Stateflow Explorer, выбираемом в рамках пункта меню Tools – Explorer.

Еще одним классом неграфических объектов SF-моделей являются **действия** или процедуры (Action). Они определяют любые операции, связанные с преобразованием данных, работой SF-модели и управлением процессом ее функционирования. Для описания процедур служит специальный язык Action Language, построенный на основе синтаксиса языка C и содержащий арифметические и логические операторы, функции, определяемые пользователем, а также некоторые специальные функции. Следует отметить следующие процедуры, часто используемые при построении SF-моделей:

chg (data_name) – процедура генерации локального события в случае изменения значений переменной data_name;

in (state_name) – логическая функция, имеющая значение true, когда состояние state_name активно;

send (event_name, state_name) – процедура пересылки спецификации события event_name состоянию state_name (прямая передача событий);

ml (′function_name (a, b,…)′) – процедура, вызывающая функцию, сформированную в MATLAB и выполняющую вычисления для значений перечисленных аргументов;

ml.datam_name – процедура, предоставляющая доступ к переменной datam_name рабочей области MATLAB.

Процедуры используются для описания SF-диаграммы в рамках двух моделей конечных автоматов:

модели Мура, связывающей процедуры с состояниями;

модели Мили, связывающей процедуры с переходами.

Для того чтобы реализовать выбранный алгоритм работы SF-модели, необходимо выполнить описание введенных состояний и переходов между ними, а также используемых данных.

При создании состояния с помощью кнопки State в левом верхнем углу его графического образа появляется знак вопроса. На его место вводится описание состояния, которое в простейшем случае должно определять имя состояния. В общем случае описание состояния имеет следующую структуру (см. пример рис. 6.20). Первым вводится имя состояния. Далее через косую черту может вводится: entry: и группа действий, выполняемых при входе в состояние; during: и группа действий, выполняемых пока данное состояние активно; exit: и группа действий, выполняемых при выходе из состояния; on event_name: и группа действий, выполняемых в момент возникновения события event_name при пребывании в данном состоянии.

Для описания перехода следует поместить указатель мыши на соединительную линию и нажать левую кнопку. Около линии появится знак вопроса. На его место следует ввести описание или метку перехода, которая определяет условия срабатывания перехода и выполняемые при этом процедуры. В общем случае структура метки перехода состоит из нескольких частей и имеет следующий вид (см. пример рис. 6.20):

event_name [condition] {condition_action}/transition_action ,

где event_name – имя события, которое инициирует переход; condition – условие перехода в виде булевского выражения, инициирующего переход

в случае его истинности; condition_action – действие условия, выполняемое мгновенно после того, как стало истинным условие перехода, но до того, как выполнен весь переход (не определилось состояние-адресат); transition_action – действие перехода, совершаемое при переходе и уже найдено состояние-адресат.

При программировании метки перехода может отсутствовать любая ее часть. Переход происходит при наступлении события, но с учетом истинности условия, если оно определено. Определение условия не обязательно. Также может отсутствовать имя события. Тогда переход происходит при выполнении условия и наступлении любого события.

SF-диаграмма может реализовывать иерархическую структуру вложенных состояний (рис. 6.20). Это означает, что каждое состояние имеет состояние-родителя. Для SF-диаграммы, состоящей из одного состояния, родителем является сама SF-диаграмма. Для того, чтобы получить вложенные состояния необходимо с помощью мыши перетащить один блок-состояние в другой, предварительно увеличив размеры последнего. Эти размеры регулируются также с помощью мыши, для чего необходимо поместить ее указатель в один из скругленных углов прямоугольника.

При выполнении действий адрес любого вложенного состояния с именем state_name1 указывается как state_name.state_name1, где state_name – имя состояния-родителя.

Stateflow поддерживает создание как взаимоисключающих (OR), так и параллельных (AND) состояний. Для определения параллельных состояний, предварительно помещенных в окне SF-диаграммы, необходимо, щелкнув левой кнопкой мыши в поле внешнего состояния-родителя, выбрать в раскрывающемся меню пункт Decomposition – Parallel (AND). В результате границы параллельных состояний отображаются пунктирной линией, а в правых верхних углах блоков автоматически устанавливаются их номера.

При подготовке SF-модели к работе требуется установить параметры модели с помощью диалогового окна, вызываемого при выборе команды Chart Properties, входящей в пункт меню File графического редактора. Окно содержит элементы, определяющие: имя SF-диаграммы; имя S-модели и/или SF-диаграммы более высокого уровня, в которых размещается данная модель; раскрывающийся список Update method,

определяющий используемый метод управления динамикой работой SF-диаграммой из Simulink; поле параметра дискретизации по времени Sample time; вспомогательные флажки, устанавливающие возможности использования данных Simulink, варианты инициализации и задания точек остановки модели, возможности модификации модели и т.п.

Весьма существенным является выбор установки Update method. Она имеет три основных варианта:

Inherited – метод, реализующий управление SF-диаграммы внешними событиями, что приводит к активизации модели каждый раз, когда на триггерный порт диаграммы поступает управляющий сигнал;

Discrete – метод, при котором периодичность активизации SF-диаграммы задается параметром Sample time, при этом Simulink автоматически генерирует управляющие события с соответствующей периодичностью;

Continuous – непрерывный метод активизации SF-диаграммы на каждом шаге моделирования, установленном для Simulink.

Помимо рассмотренных компонентов разработки и управления SF-моделей, следует также выделить встроенный отладчик, вызываемый в пункте меню Tools – Debug, синтаксический анализатор ошибок (Tools – Parse), а также средства поиска объектов заданного типа (Tools – Find). Принципы их использования подробно рассматриваются в специальной литературе.

Рассмотрим теперь технологию создания SF-моделей на примере простейших систем массового обслуживания.

Первой рассмотрим систему с одним каналом без накопителя (M/M/1/0). Такая система подробно описана в п. 4.1. Она имеет два состояния: канал обслуживания свободен (free); канал обслуживания занят (busy). Факторами, определяющими эффективность обработки заявок, являются интенсивность входного потока заявок Li и интенсивность потока обслуживания Lo. Значения этих параметров определяются в m-файле УП (см. рис. 6.19). Показателем эффективности является относительная пропускная способность системы q, определяемая как отношение среднего количества обслуженных заявок к среднему количеству поступивших заявок в единицу времени.

При построении S-моделей, в которых используются блоки Stateflow, содержащие в себе переходы, инициируемые истечением

некоторых временных интервалов (что характерно для СМО), требуется обеспечить синхронизацию внутреннего системного времени Stateflow и Simulink. Для этого необходимо в качестве входного параметра подавать в SF-диаграмму системное время (systime) из Simulink и именно это время использовать для составления условий переходов.

Другой возможный способ состоит в том, чтобы использовать специальный символ « t », изначально определенный в Stateflow для проверки выполнения условий истечения временных интервалов. Однако, в этом случае, если требуется осуществить многократное обращение к модели в цикле статистического моделирования, необходимо отключить флажок Execute (enter) Chart Initialization в диалоговом окне Chart Properties. И тот, и другой подход может быть использован в равной степени. Реализация с внешним источником предоставляет дополнительные возможности применения неравномерной шкалы системного времени (с замедлением и ускорением процесса моделирования).

Далее возможно рассмотреть два варианта построения общей S-модели.

В рамках первого варианта входной поток заявок формируется в Simulink и подается на вход SF-модели в виде последовательности событий-заявок. На рис. 6.21 и 6.22 представлены блок-диаграмма S-модели и диаграмма вложенной в нее SF-модели, реализованные в рамках этого варианта.

Здесь в S-модели размещается блок Clock, формирующий значение времени systime (Input from Simulink date), подаваемого на вход SF-модели (блок Chart). Поток заявок формируется на выходе блока Fcn, реализующего сравнение входного сигнала с величиной $Pi = Li * Ts$ (логическое условие $u < Pi$).

Амплитуда входного сигнала u имеет равномерное распределение в диапазоне [0,1] и формируется на выходе генератора шума Uniform Random Number. Значение Ts определяет интервал дискретизации Sample time во всех блоках S-модели. В результате с вероятностью Pi на выходе блока Fcn в каждом элементарном интервале $[t, t + Ts]$ формируется сигнал с единичной амплитудой. Поток заявок при этом имеет вид последовательности коротких импульсов, имеющих длительность Ts. Каждое событие, связанное с поступлением входной заявки, определяется в SF-модели под именем zayvka (Input from Simulink event).

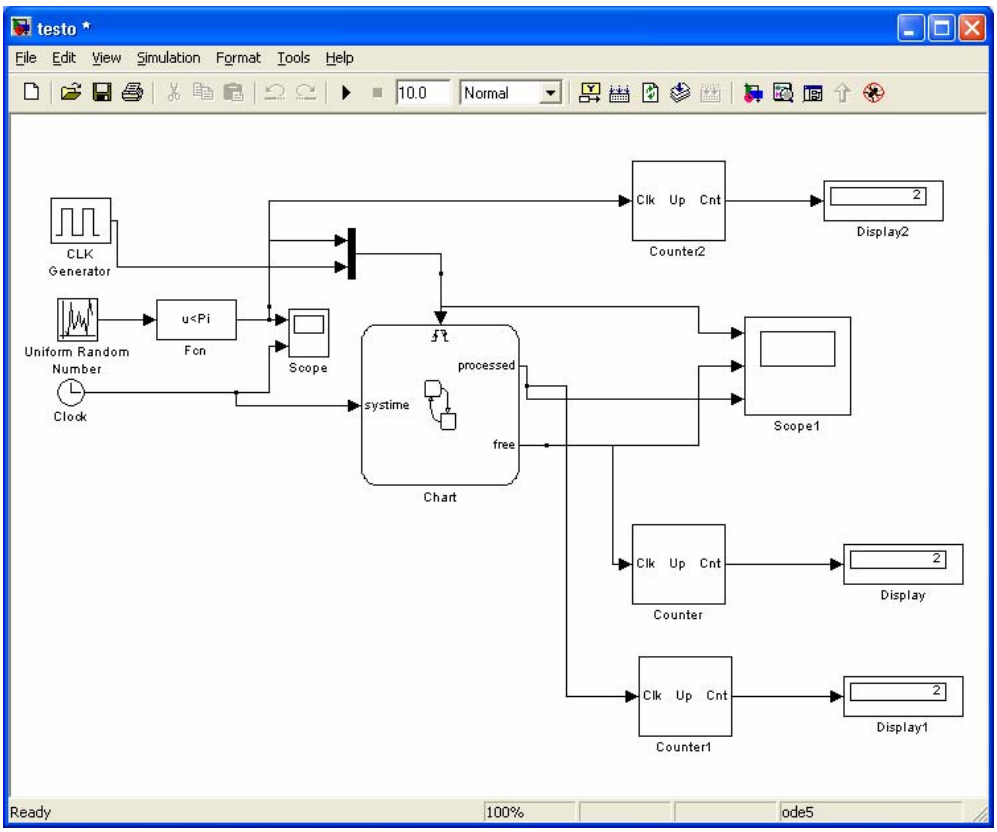


Рис. 6.21

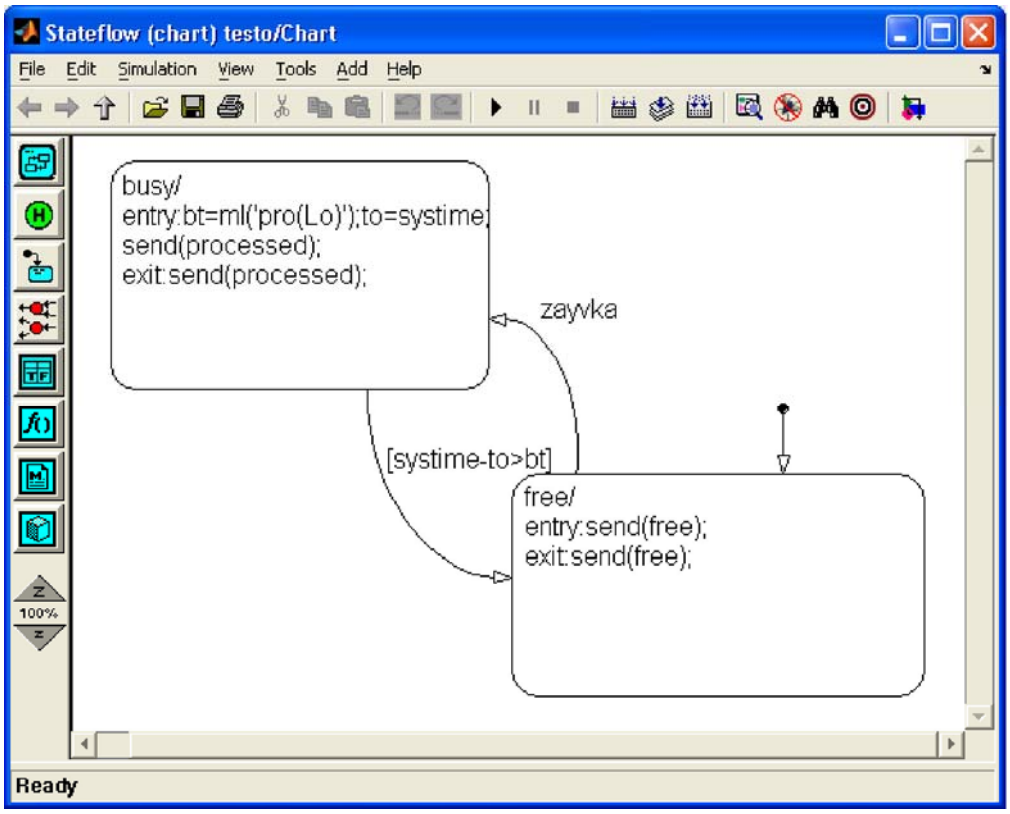


Рис. 6.22

Событие zayvka инициирует переход из состояния free схемы рис. 6.22 в состояние busy. При определении этого входного для SF-модели события в пункте меню Add на изображении блока Chart появится значок триггерного входа. Это означает, что активизация SF-модели будет происходить только при реализации метода обновления (Update method) Inherited и только в моменты поступления заявок. Данная ситуация не позволяет обеспечить правильную работу SF-модели, так как в ней, как видно из рис. 6.22, при нахождении в состоянии busy, требуется постоянно проводить сравнение текущего значения systime со временем окончания процесса обслуживания, определяемым как $to + bt$. Здесь to – фиксируемое при входе в состояние время systime, а bt – значение случайной величины интервала времени обслуживания с экспоненциальным законом распределения, вызываемое с помощью m-функции pro (Lo), которая определяется в MATLAB следующим образом:

```
function u = pro (Lambda);
% экспоненциальное распределение с параметром Lambda
u = – (1/Lambda)*log (rand);
```

Таким образом, чтобы обеспечить инициирование событий и переходов, связанных с истечением временных интервалов, в схеме рис. 6.22 требуется обеспечить активизацию SF-диаграммы на каждом шаге изменения времени S-модели. Для этого в блок-диаграмме рис. 6.21 вводится специальный генератор событий CLK Generator, формирующий периодическую последовательность импульсов с длительностью Ts и периодом $2 * Ts$. Соответственно, в SF-диаграмме определяется Input from Simulink event под именем CLK с установкой Either. В результате поток этих событий-активаторов SF-модели вместе с потоком заявок подается на триггерный порт SF- модели и активизирует ее с периодичностью Ts. Помимо рассмотренных объектов в SF-диаграмме вводятся два события processed и free, имеющие статус Output to Simulink event. Эти события фиксируют начало и окончание пребывания в соответствующих состояниях и используются в S-модели для визуализации и регистрации процесса работы СМО.

В соответствии с изложенным на рис. 6.23 представлено окно обозревателя Stateflow, где размещается древовидная структура,

описывающая иерархию моделей и внутренних состояний блока Chart, таблица определений событий и данных, а в правой части – диалоговое окно Chart properties.

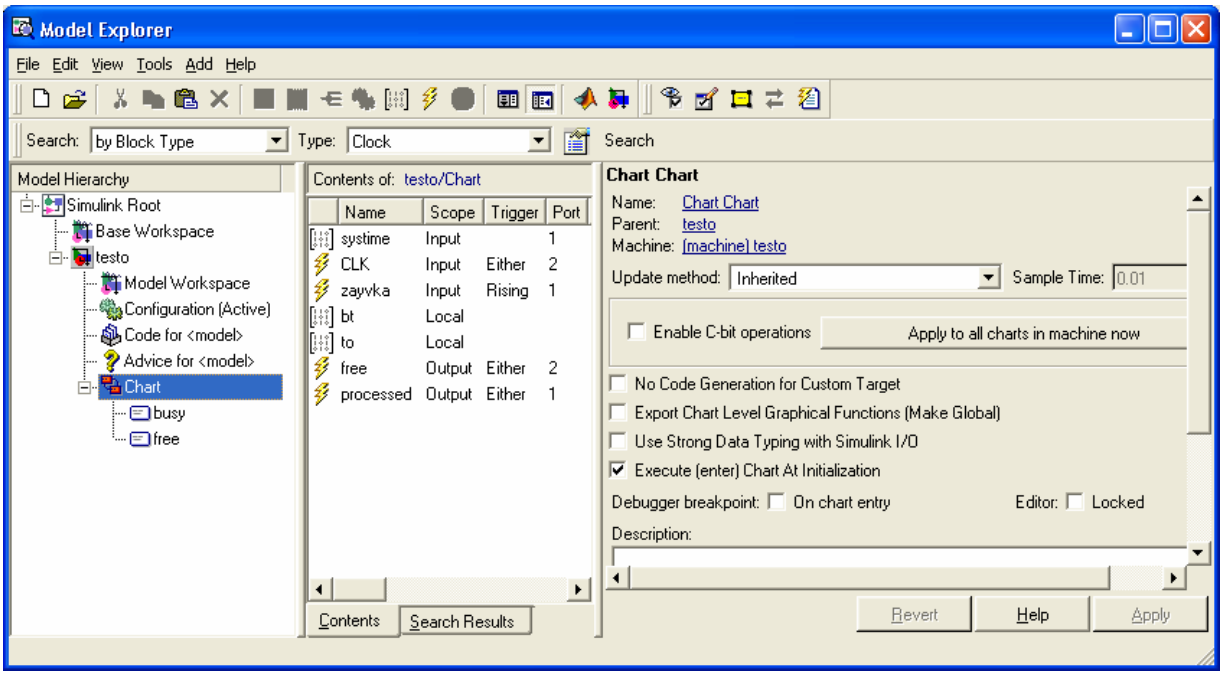


Рис. 6.23

Процесс функционирования SF-модели характеризуется визуальным отображением активных состояний и переходов, что позволяет наглядно определить динамику работы СМО. На выходе SF-модели в S-модели фиксируются событийные процессы processed и free. Они вместе с входным потоком заявок и потоком импульсов CLK визуализируются в блоке Scope1, как это показано на рис. 6.24 ($\mathrm{Li} = 0{,}5\ \mathrm{c}^{-1}, \mathrm{Lo} = 1\ \mathrm{c}^{-1}$).

Здесь видно, что при указанном способе определения событий processed и free, соответствующие процессы описывают пребывание системы во взаимоисключающих состояниях. При этом количество переходов в состояние busy, описываемых повышением уровня processed, определяет количество обслуженных заявок. Поэтому в S-модели для регистрации общего количества заявок и количества обслуженных заявок (см. рис. 6.21) используются соответствующим образом настроенные двоичные счетчики импульсов (блоки Counter подраздела Swiches and

Counter библиотеки Signal Processing). Результаты счета отображаются в блоках Display (раздел Sinks).

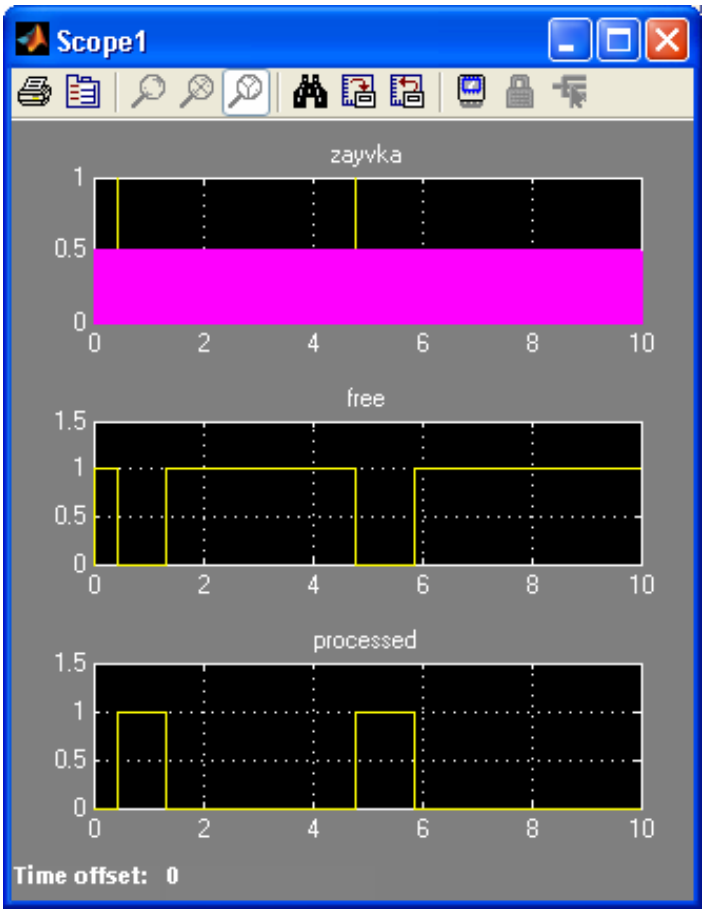


Рис. 6.24

Для приближенной оценки пропускной способности достаточно при длительной реализации входного потока оценить отношение

$$\tilde{q} = n_{pr} / n_{\Sigma}$$

количества зарегистрированных счетчиком Counter1 обслуженных заявок n_{pr} к общему числу заявок n_{Σ}, зарегистрированных счетчиком Counter2.

Запуск всей модели осуществляется либо из управляющего m-файла, либо из Simulink (если все значения переменных определены в блоках S-модели).

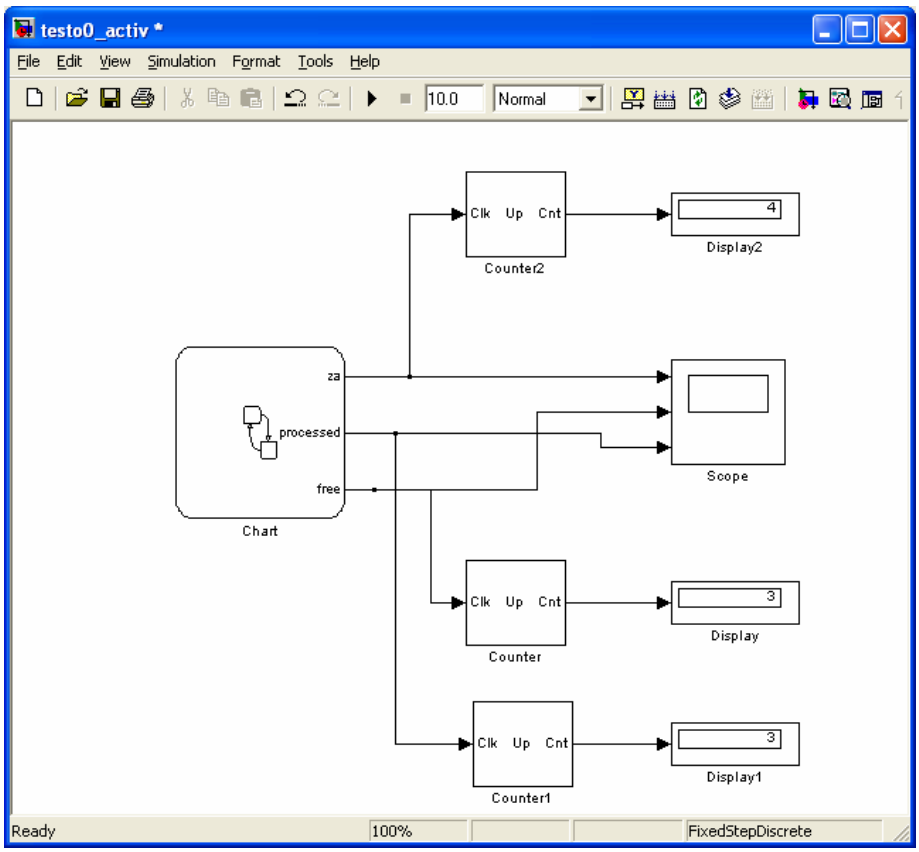


Рис. 6.25

Рассмотрим теперь второй вариант реализации модели, в рамках которого поток заявок формируется внутри SF-модели. Одновременно представляется реализация модели с использованием внутреннего источника времени на основе специального символа « t ». Соответствующие блок-диаграммы S и SF-моделей представлены на рис. 6.25 и рис. 6.26.

SF-модель имеет теперь два параллельных состояния: состояние Activ, где осуществляется генерация событий zayvka (статус Local) пуассоновского потока с интенсивностью Li и одновременно формируется выходной сигнал SF-модели za (Output to Simulink date); состояние QP, где размещается собственно модель СМО с аналогичной предыдущему примеру конструкцией. На рис. 6.27 представлено соответствующее окно Stateflow Explorer, описывающее иерархию модели, события и данные. Следует только отметить, что в рамках данного варианта должны быть реализованы Update methods: Discrete или Coutinues.

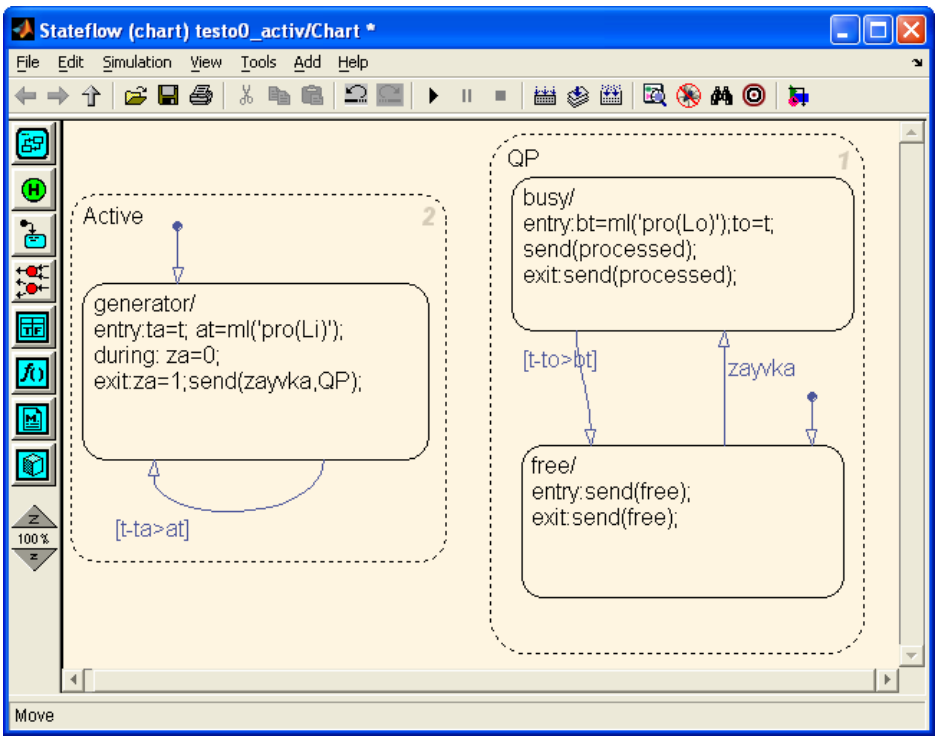


Рис. 6.26

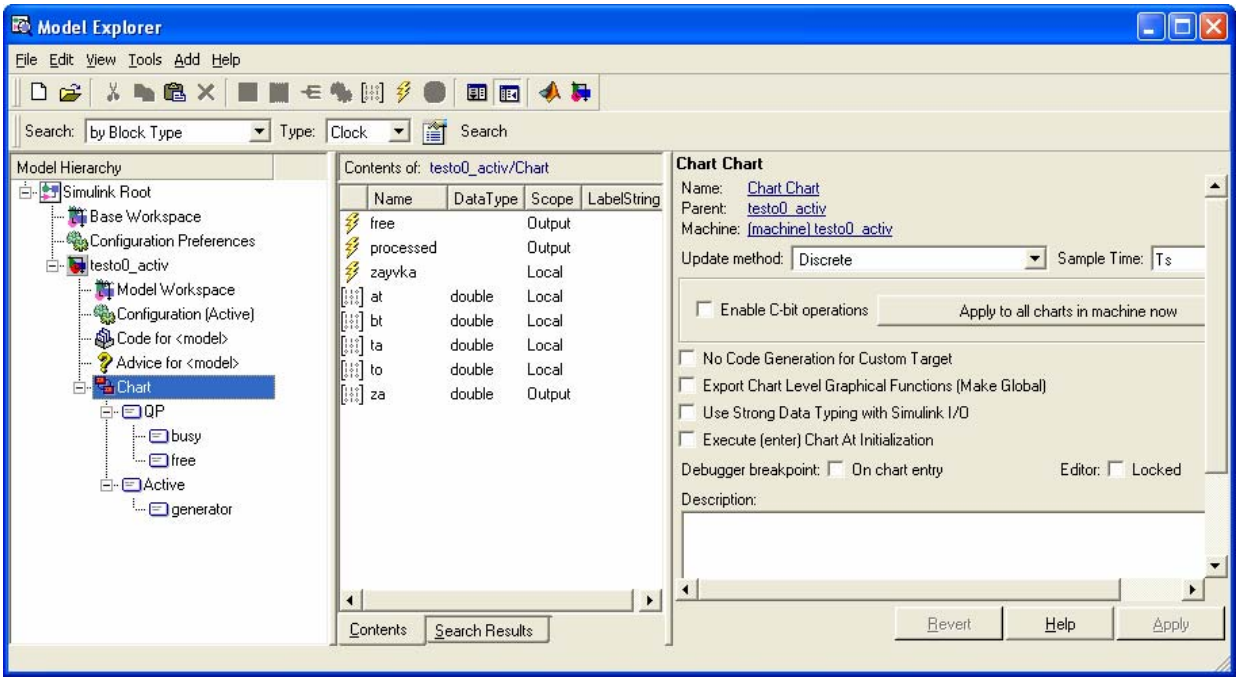


Рис. 6.27

Наконец, на рис. 6.28 представлены осциллограммы сигналов на выходе данного варианта SF-модели, регистрируемые блоком Scope. Как видно из этого рисунка, процесс работы SF-модели в этом варианте аналогичен ранее рассмотренному с внешним генератором. Однако общая S-модель в данном случае представляется более простой.

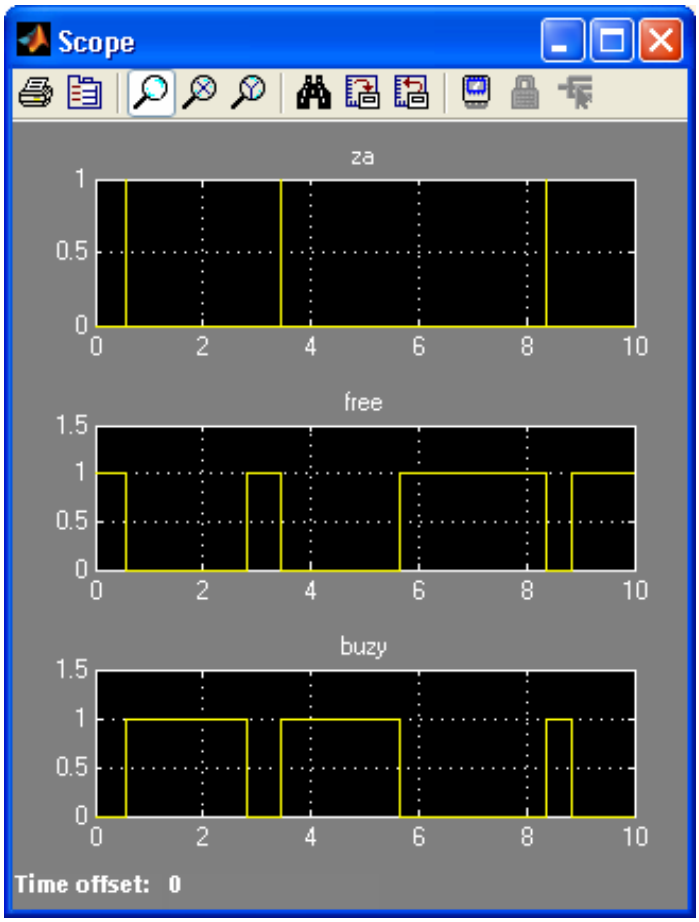


Рис. 6.28

В качестве следующего примера рассмотрим построение модели одноканальной СМО с накопителем. Для нее, помимо интенсивностей потоков Li, Lo, в управляющей MATLAB программе зададим переменную nqm, определяющую предельную длину очереди или емкость накопителя.

Блок-диаграммы S-модели, разработанные для подобной системы, представлены на рис. 6.29 и рис. 6.30. Данная модель является **базовой** для построения любых, более сложных моделей систем, состоящих из комбинаций элементарных приборов массового обслуживания.

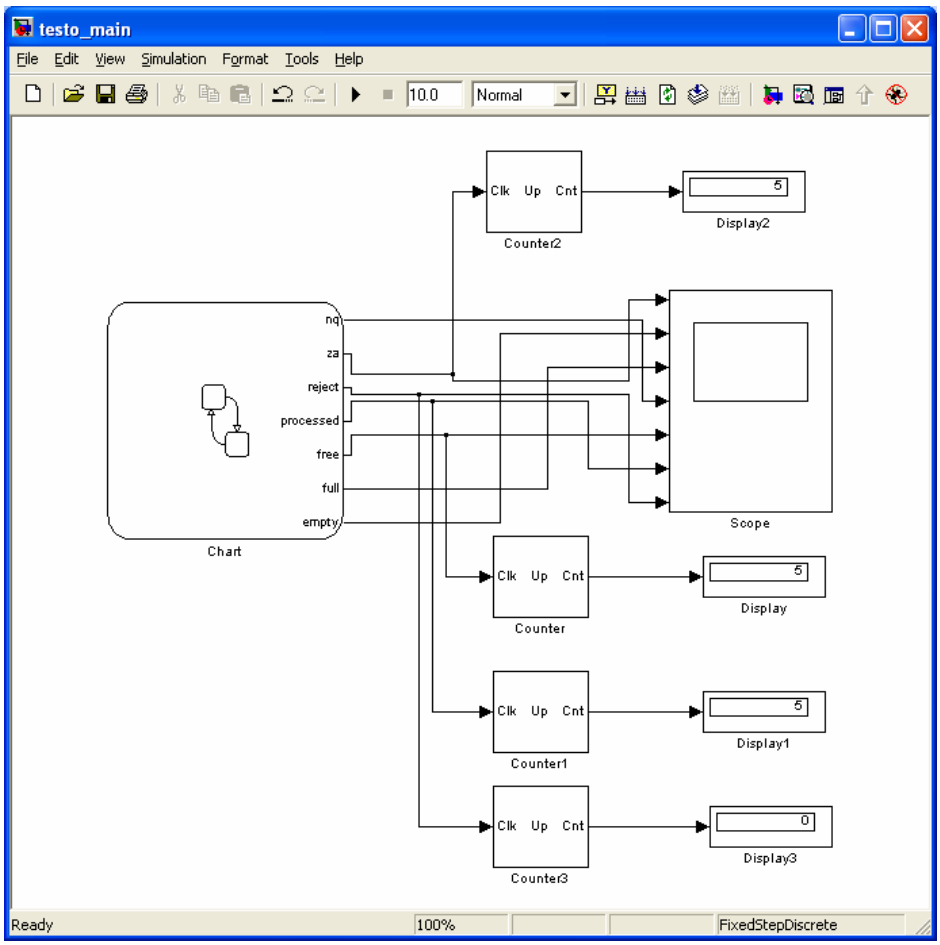


Рис. 6.29

Здесь появляется дополнительные выходные переменные в блоке Chart S-модели, что связано с усложнением ее SF-диаграммы. SF-модель имеет теперь три параллельных процесса: изменения состояния генератора заявок (Active); изменения состояния канала обслуживания (Process) и изменения состояния очереди накопителя (Queue). Для упрощения анализа работы системы функции генерации случайных интервалов времени между заявками и временем обслуживания в схеме рис. 6.30 заменены переменными at, bt (Local), принимающими детерминированные значения. Блок Active по сравнению с предыдущим примером изменился в том плане, что теперь указывается адресация при посылке события zayvka. Текущее количество заявок в очереди (блок Queue) описывается переменной nq, имеющей видимость в Simulink. Новый блок Queue имеет три вложенных состояния, выделяемые для накопителя: пусто (empty); хранение (stored); заполнено (full). Для описания входа и выхода в состояния используются события empty, full (Output to Simulink), а для

регистрации отказа в обслуживании новой заявки при пребывании в состоянии full используется переменная reject (Output to Simulink date). Событие prf (Local) фиксирует переход на обслуживание новой заявки с выбором ее из очереди, при этом в блоке Queue осуществляется уменьшение значения nq на единицу.

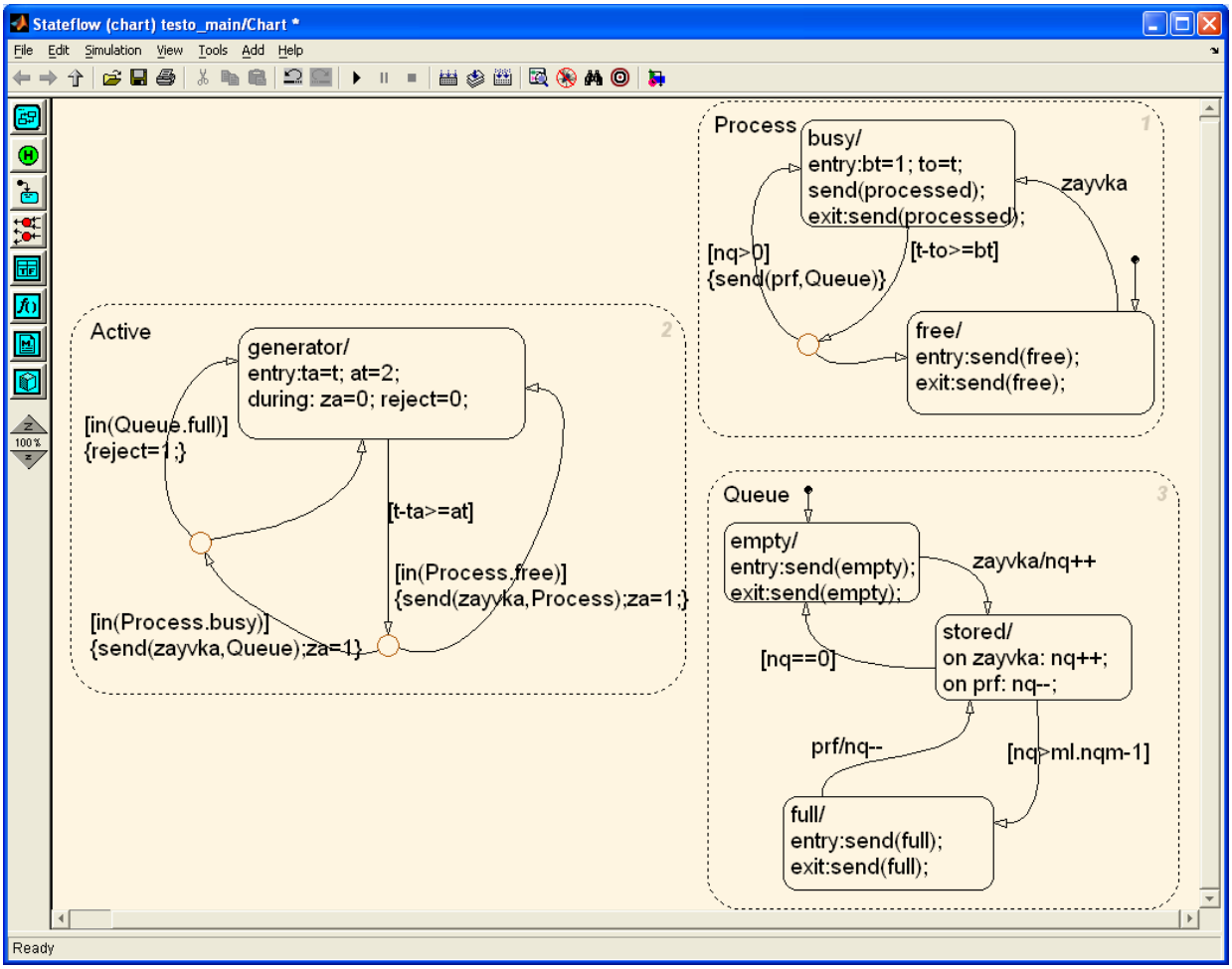


Рис. 6.30

Соответствующее разработанной модели окно Stateflow Explorer, содержащее иерархию моделей и состояний, таблицу определений событий и данных, а также окно Chart Properties приведено на рис. 6.31.

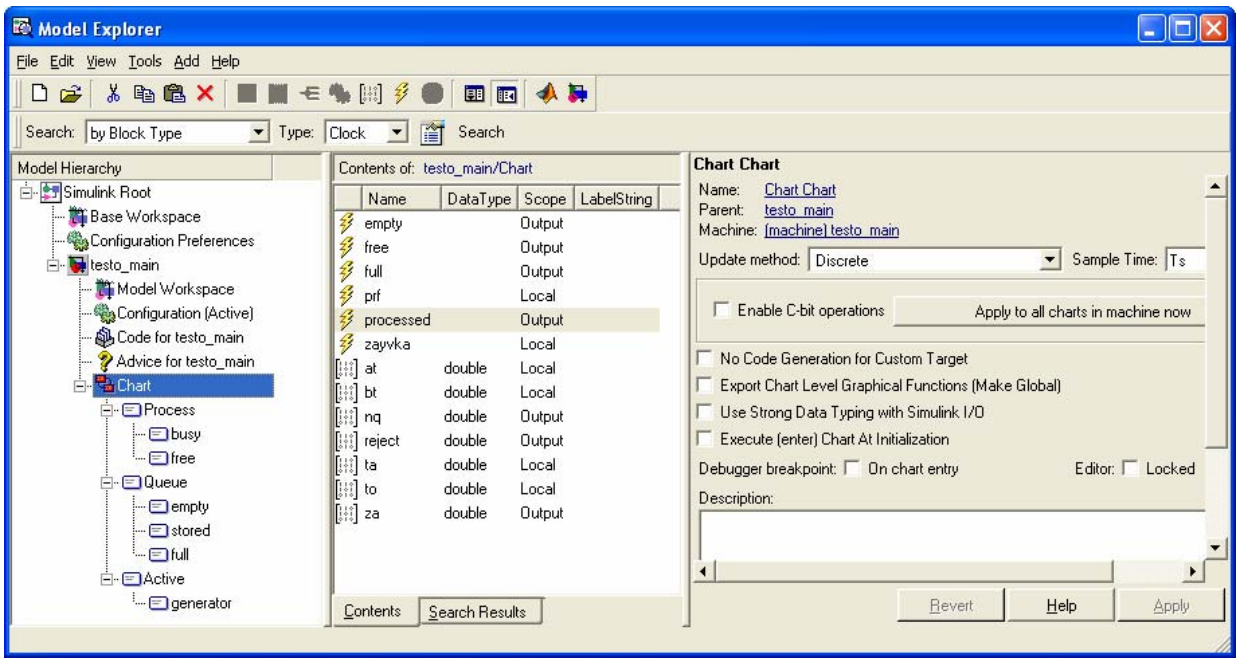
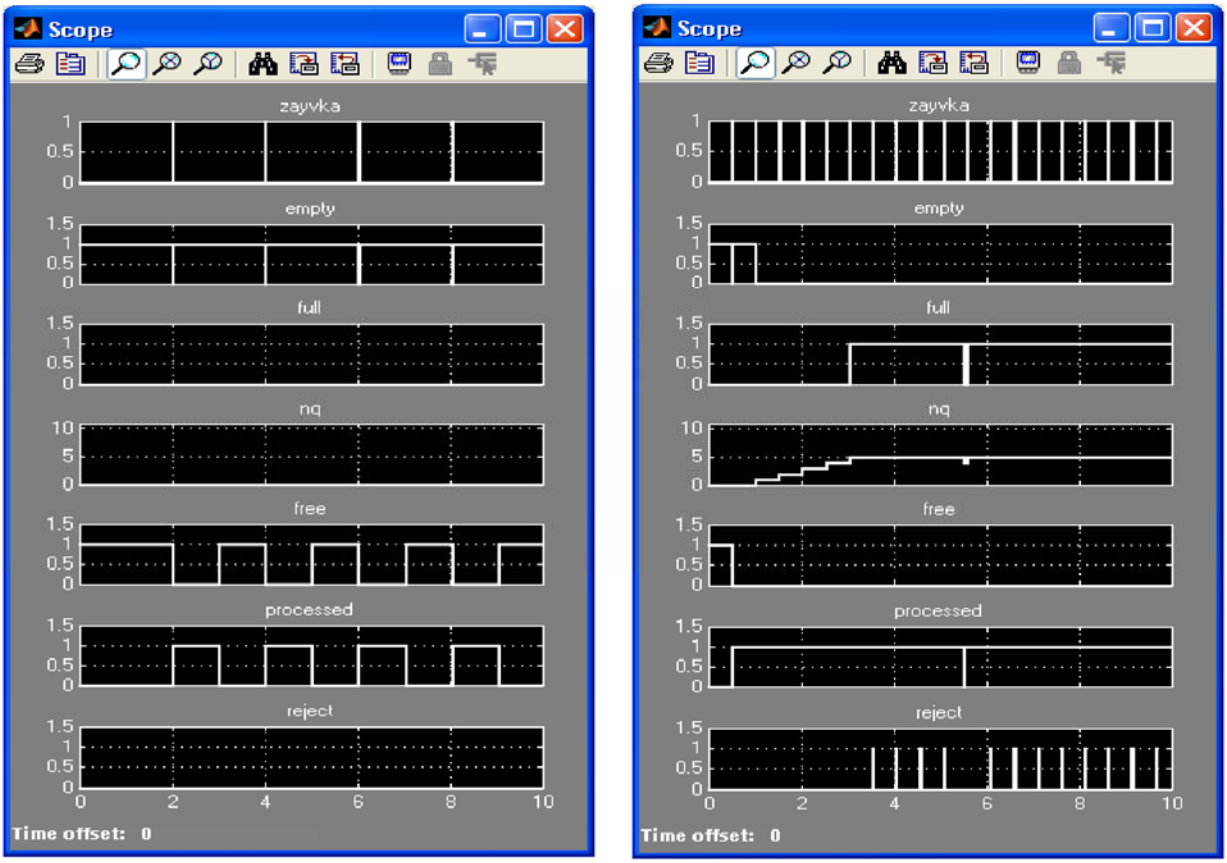


Рис. 6.31



а) б)

Рис. 6.32

На рис. 6.32 представлены сигналы, формируемые на выходе SF-модели и регистрируемые в блоке Scope. Рис. 6.32а соответствует значениям $at = 2$ с и $bt = 1$ с. Естественно, что при этом очереди нет и все заявки обслужены. На рис. 6.32б представлена другая ситуация, когда $at = 0{,}25$ с и $bt = 5$ с, а $nqm = 5$. Соответственно, накопитель быстро заполняется ($nq = 5$) и после определенного момента времени заявки получают отказ. После того, как через 5 с заканчивается обслуживание первой заявки, величина nq уменьшается и очередная заявка поступает в накопитель. Эта ситуация фиксируется в осциллограммах full, nq и reject.

Следующий этап предполагает проведение исследований, направленных на оценку эффективности рассматриваемой системы с использованием УП, реализующей полномасштабное стратегическое и тактическое планирование модельного эксперимента. В качестве показателей будем рассматривать относительную пропускную способность системы и среднее время нахождения заявки в системе (в очереди и на обслуживании). Типовой текст m-файла сценария имеет следующий вид

```
%Оценка эффективности системы массового обслуживания M/M/1/nqm
%Используемые показатели: относительная пропускная способность qm;
%среднее время обслуживания заявки taym.
%Исследуемые факторы влияния: интенсивности потока заявок Li и потока
%обслуживания Lo.
%Фиксируемый параметр - предельная длина очереди nqm.
clear all;
rand('state',10)
%Задание интервала дискретизации по времени и
%количества отсчетов на интервале моделирования [0,10] с.
Ts=0.01;
Ns=10000;
%Задание неварьируемых величин (предельная длина очереди)
nqm=5;
%Задание количества и диапазонов изменения факторов Li (a) и Lo (b)
nf=2;
minf=[0.4 0.4];
maxf=[1 1];
%формирование дробного двухуровневого плана эксперимента
%для учета взаимодействий
fracfact('a b ab' );
N=2^nf;
```

```
fracplan=ans;
fictfact=ones(N,1);
X=[fictfact ans]';
fraceks=zeros(N,nf);
for i=1:nf,
   for j=1:N,
fraceks(j,i)=minf(i)+(fracplan(j,i)+1)*(maxf(i)-minf(i))/2;
end;
end;
fraceks
%Тактическое планирование эксперимента
%Задание доверительного интервала и уровня значимости оценки qm
dm=0.01;
alpha=0.05;
%Определение t-критического
tkr_alpha=norminv(1-alpha/2);
%Цикл по совокупности экспериментов стратегического плана
for j=1:N,
   a=fraceks(j,1);
   b=fraceks(j,2);
   Li=a;
   Lo=b;
   rr=rand
   %Организация цикла статистических испытаний с переменным объемом
   %NE для достижения заданной точности оценки показателя qm;
   %показатель taym оценивается попутно
   NE=1;
   e=0;
   l=0;
   SQ=0;
   D=1;
      while NE < tkr_alpha^2*D/dm^2,
%Инициализация начальных значений массивов интервалов времени
      t1(1:Ns)=0;  tay(1:Ns)=0;
%Инициализация начальных значений массива номеров заявок в очереди
      ns(1:nqm)=0;
%Инициализация начального значения количества входящих заявок
      ob=0;
%Инициализация начального значения количества обслуженных заявок
      ok=0;


%Имитация функционирования системы
```

```
sim('testo_main2',Ts*Ns);

%Фиксация результатов каждой реализации
u=ok/ob;
v=sum(tay)/ok;
t1(1:ob);
tay(1:ob);

%Усреднение результатов и оценка выборочной
%дисперсии D измеряемого параметра
e=e+u;
l=l+v;
SQ=SQ+u^2;
if NE>1 D=SQ/(NE-1)-(e^2)/(NE*(NE-1)); end;
NE=NE+1;
   end;
   NE=NE-1
   %Оценка показателей (реакции) по NE реализациям
   qm=e/NE
   taym=l/NE
   Y(j)=qm;
   Y1(j)=taym;
end;

%Определение коэффициентов регрессии для qm и taym
C=X*X';
b_=inv(C)*X*Y'
b_1=inv(C)*X*Y1'
%Формирование зависимостей реакции системы на множестве
%значений факторов
A=minf(1):0.01:maxf(1);
B=minf(2):0.01:maxf(2);
[k N1]=size(A);
[k N2]=size(B);
for i=1:N1,
   for j=1:N2,
      an(i)=2*(A(i)-minf(1))/(maxf(1)-minf(1))-1;
      bn(j)=2*(B(j)-minf(2))/(maxf(2)-minf(2))-1;
      %Экспериментальная поверхность реакции для qm(Yc)и taym(Yc1)
   Yc(j,i)=b_(1)+an(i)*b_(2)+bn(j)*b_(3)+an(i)*bn(j)*b_(4);
   Yc1(j,i)=b_1(1)+an(i)*b_1(2)+bn(j)*b_1(3)+an(i)*bn(j)*b_1(4);
end;
```

```
end;
for i=1:N1,
   for j=1:N2,
      %Теоретическая поверхность реакции для qm(Yo)и taym(Yo1)
      ro=A(i)/B(j);
      ss=0;
      for p=1:nqm,
         ss=ss+p*ro^(p-1);
      end;

         if ro~=1
            Yo(j,i)=(1-ro^(nqm+1))/(1-ro^(nqm+2));
            Yo1(j,i)=((1-ro)/(1-ro^(nqm+2)))*(ro^2)*ss/B(j)+1/B(j);
         else %использование предельных соотношений
            %для устранения неопределенностей типа 0/0 при ro=1
            Yo(j,i)=(nqm+1)/(nqm+2);
            Yo1(j,i)=(1/(nqm+2))*(ro^2)*ss/B(j)+1/B(j);
         end;
end;
end;

%Отображение зависимостей в трехмерной графике для показателя qm
[x,y]=meshgrid(A,B);
figure;
subplot(1,2,1),plot3(x,y,Yc),
xlabel('fact a(Li)'),
ylabel('fact b(Lo)'),
zlabel('Yc'),
title('qm(ex)'),
grid on,
subplot(1,2,2),plot3(x,y,Yo),
xlabel('fact a(Li)'),
ylabel('fact b(Lo)'),
zlabel('Yo'),
title('qm(th)'),
grid on;

%Отображение зависимостей в трехмерной графике для показателя taym
figure;
subplot(1,2,1),plot3(x,y,Yc1),
xlabel('fact a(Li)'),
ylabel('fact b(Lo)'),
zlabel('Yc1'),
```

```
title('taym(ex)'),
grid on,
subplot(1,2,2),plot3(x,y,Yo1),
xlabel('fact a(Li)'),
ylabel('fact b(Lo)'),
zlabel('Yo1'),
title('taym(th)'),
grid on;
```

В данной программе реализуется многократный прогон S-модели в ходе статистических испытаний в соответствии с планом эксперимента, а также обработка и отображение результатов моделирования. Тактическое планирование выполняется для оценки пропускной способности, тогда как оценка среднего времени нахождения заявки в системе проводится попутно. Для регистрации результатов в каждой реализации процесса функционирования системы в программе вводятся массивы моментов времени поступления заявок на вход системы t1(:) и временных интервалов нахождения заявок в системе tay(:), а также переменные, определяющие количества входящих заявок ob и количества обслуженных заявок ok. Кроме того, для фиксации временных интервалов потребовалось ввести служебный массив ns(:), который должен хранить порядковые номера входящих заявок, в данный момент времени находящихся в очереди, и который должен постоянно обновляться по мере продвижения заявок в очереди в процессе обслуживания.

Основной вопрос, который возникает при разработке данной модели и ей подобных, состоит в выборе наиболее рационального и удобного для пользователя способа связи S- и SF-моделей с УП, написанной на языке MATLAB, с учетом потребностей выполнения необходимых вспомогательных процедур обработки данных для регистрации результатов моделирования и получения оценок эффективности системы. В предлагаемых ниже примерах рассматривается три возможных способа.

Первый способ не предполагает применение каких-либо дополнительных компонентов Stateflow по отношению к уже рассмотренным в предыдущих примерах. Он основан на реализации необходимых процедур и операций с максимальным использованием возможностей языка MATLAB. Пример SF-модели, реализующей данный подход представлен на рис.6.33

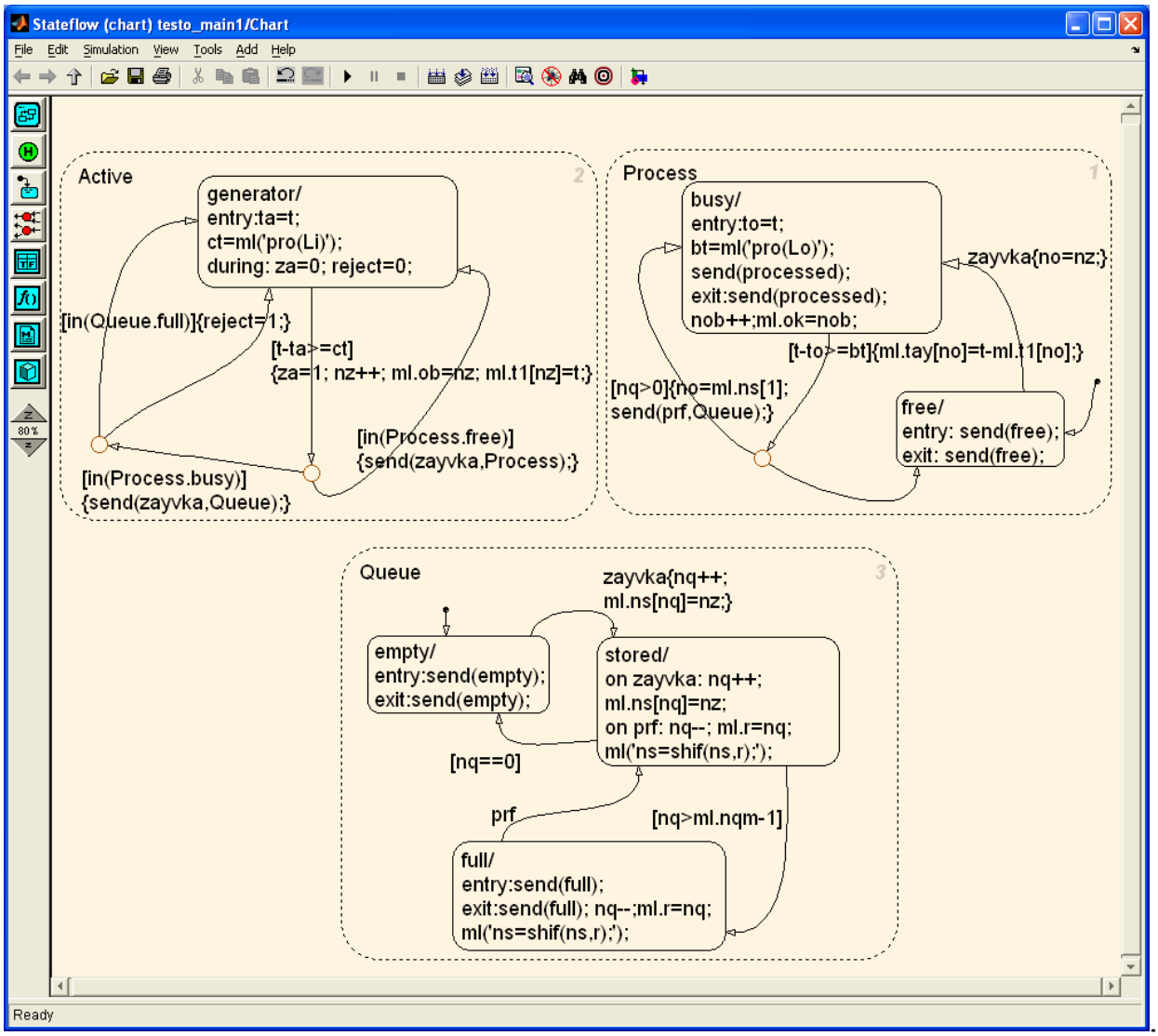


Рис.6.33

Эта модель фактически повторяет модель, представленную на рис.6.30, в которой дополнительно введены операции присвоения значений переменных и массивов, определенных в m-файле УП. Введены счетчики входящих заявок nz и обслуженных заявок nob, на основе которых определяются переменные ml.ob и ml.ok. Текущее обновление массива ml.ns (сдвиг очереди), на основе которого определяется очередной номер обслуживаемой заявки no=ml.ns[1] и рассчитывается время обработки заявки в системе ml.tay[no]=t-ml.t1[no] в блоке Processed, производится путем вызова m-функции, имеющей вид

```
function y=shif(ns,nq);
%сдвиг  элементов массива после уменьшения его длины nq
%на единицу
if nq~=0,
   for i=1:nq,
     ns(i)=ns(i+1);
   end;
end;
y=ns;
```

Обновление массива производится после перехода первой в очереди заявки на обслуживание и уменьшения величины переменной nq единицу в блоках Queue.stored и Queue.full .

Таким образом, в представленной модели вся вспомогательная обработка данных в интересах оценки эффективности системы проводится с использованием переменных и функции, определенных в MATAB-программе.

На рис. 6.34 представлены результаты моделирования в виде экспериментальных зависимостей введенных показателей эффективности: относительной пропускной способности qm и среднего времени обслуживания заявки taym от интенсивности потока заявок Li и интенсивности потока обслуживания Lo. Для сравнения рядом даны теоретические зависимости, рассчитанные для стационарного режима работы системы на основе соотношений, приведенных в п. 4.1.

Второй способ организации работы модели рассматриваемой системы предполагает использование такого внутреннего компонента Stateflow, как графические функции, определяемые графом потока вычислений для описания реализуемого алгоритма и устанавливаемые в виде блоков – прямоугольников с заготовкой для имени функции «function». На рис.6.35 представлена блок-диаграмма модели, реализующей данный способ для организации внутренних процедур обработки данных.

.

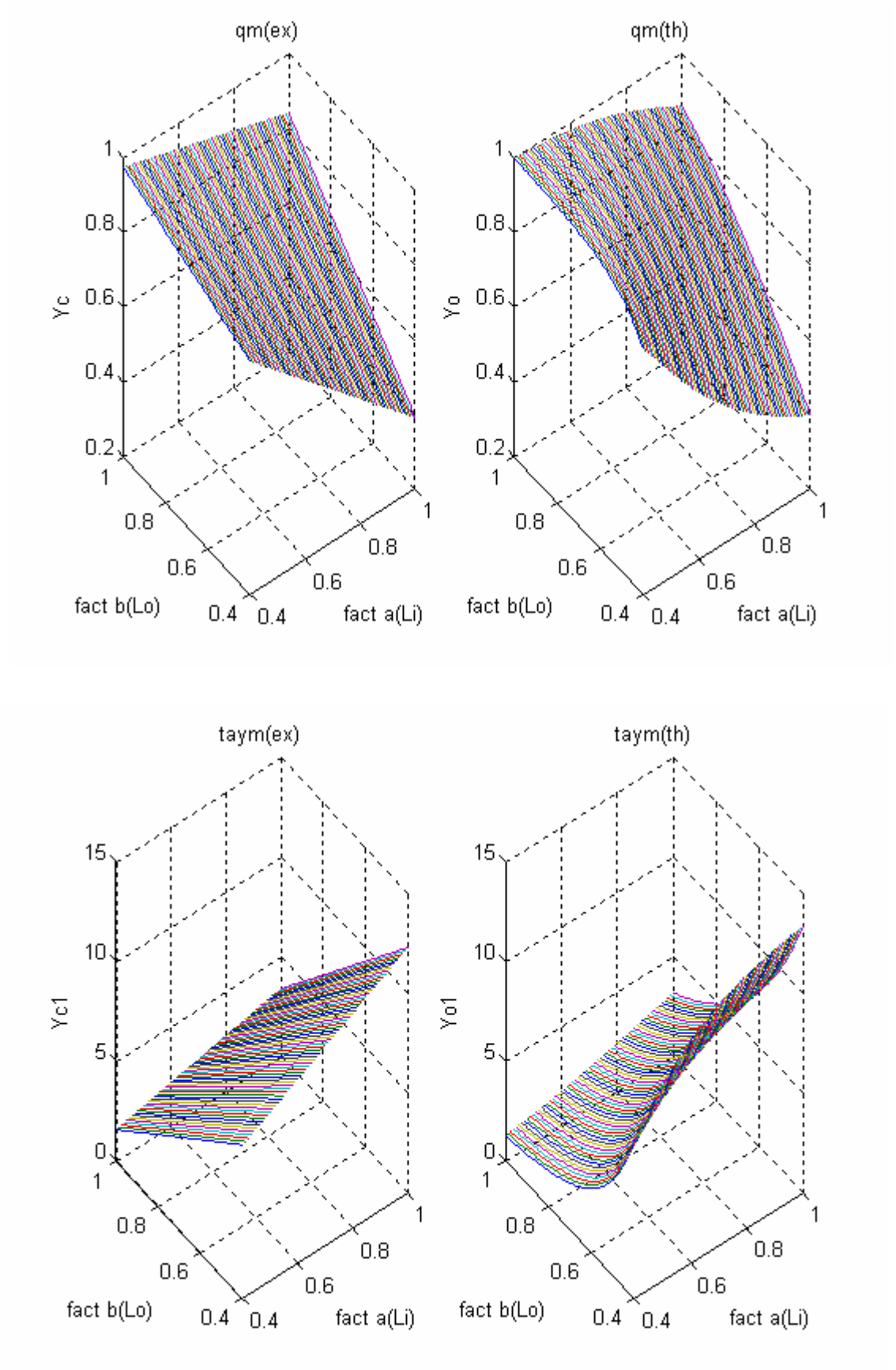


Рис.6.34

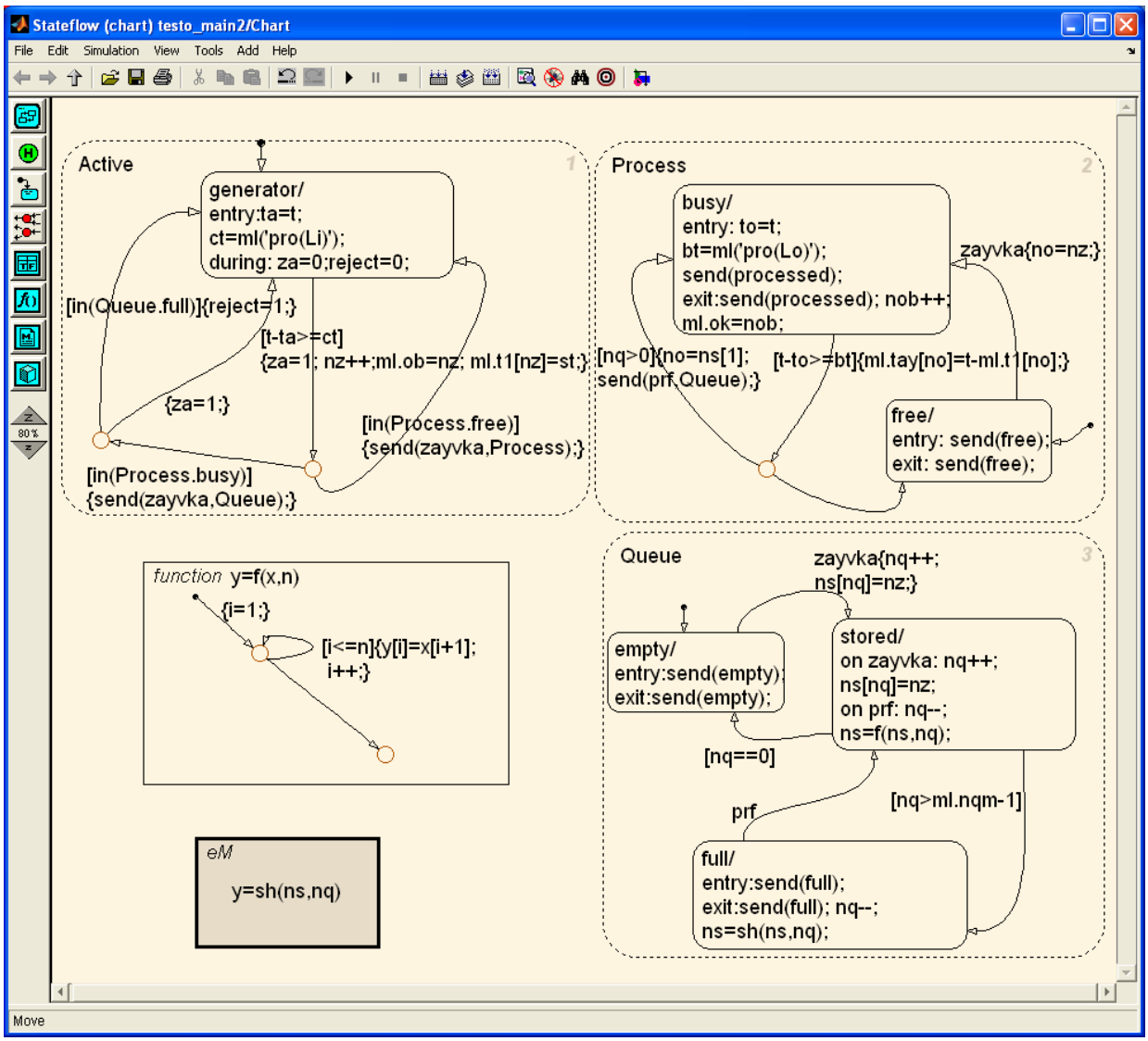


Рис.6.35

В рассматриваемом примере графическая функция используется в блоке Queue.stored для проведения обновления и перезаписи элементов массива ns, сохраняющего номера заявок в очереди для последующей фиксации времени пребывания заявок в системе. Для программирования графической функции соответствующий блок перемещается из панели инструментов в окно SF-диаграммы. Далее в появляющемся поле function вводится имя функции, входные и выходные переменные, после чего выполняется программирование с использованием графических элементов типа Default Transition и Connective Junction. В результате формируется диаграмма потокового графа функции, представленная на рис.6.35.

Задание входного описания функции автоматически приводит к появлению в окне обозревателя Stateflow, в иерархическом описании модели, соответствующего значка f( ) с указанием имени функции. Активизировав этот значок, в окне получаем таблицу входных и выходных переменных функции, которые требуется описать, указав тип переменных, размеры массивов и т.д. Таблица при необходимости должна быть дополнена списком и описанием внутренних переменных, как это показано на рис. 6.36.

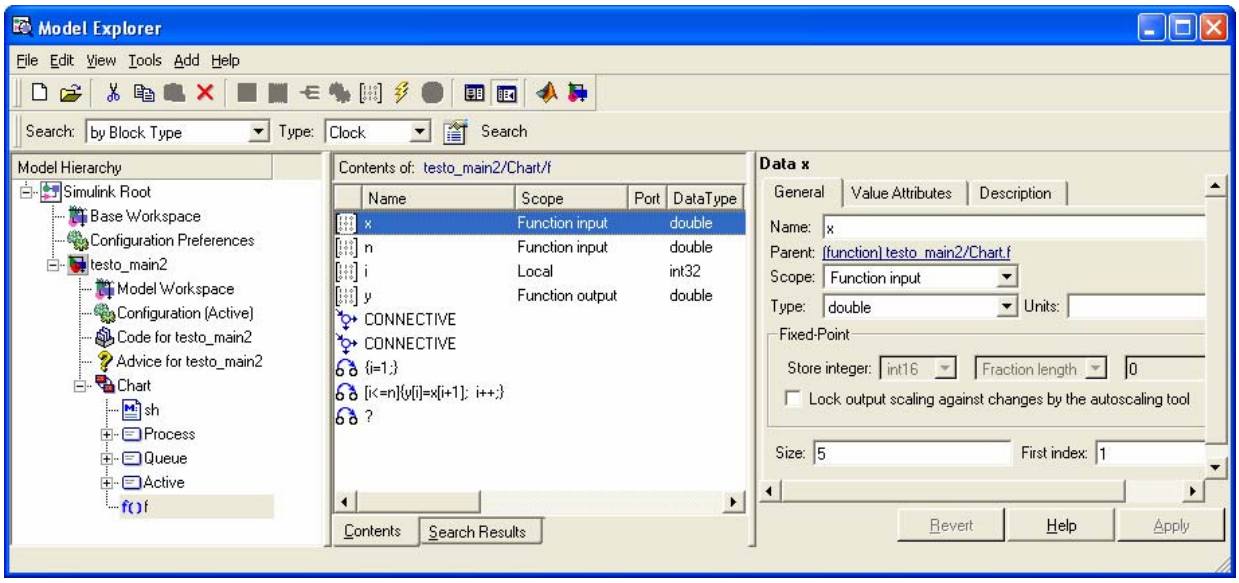


Рис. 6.36

Третий способ выполнения внутреннего программирования процедур обработки данных в SF-модели состоит в использовании графических компонентов Embedded Matlab function, обеспечивающих непосредственное включение в SF-модель функций на языке MATLAB. Эти компоненты перетаскиваются с помощью мыши в окно диаграммы в виде прямоугольников с надписью eM и окошком для указания имени функции, входных и выходных переменных (см. рис 6.35). При активизации этого блока щелчком левой кнопки мыши появляется активное окно, показанное на рис.6.37, в которое вводится текст eM-функции. В иерархическом описании модели рис.6.36 появляется значок, свидетельствующий о появлении соответствующего компонента модели, активизировав который можно получить доступ к таблице определения внешних и внутренних переменных eM-функции. В модели,

представленной на рис. 6.35, подобная функция используется в блоке Queue.full и реализует такую же операцию обновления массива ns, как и графическая функция в блоке Queue.stored. Тем самым иллюстрируется возможность получения одного и того же результата различными способами.

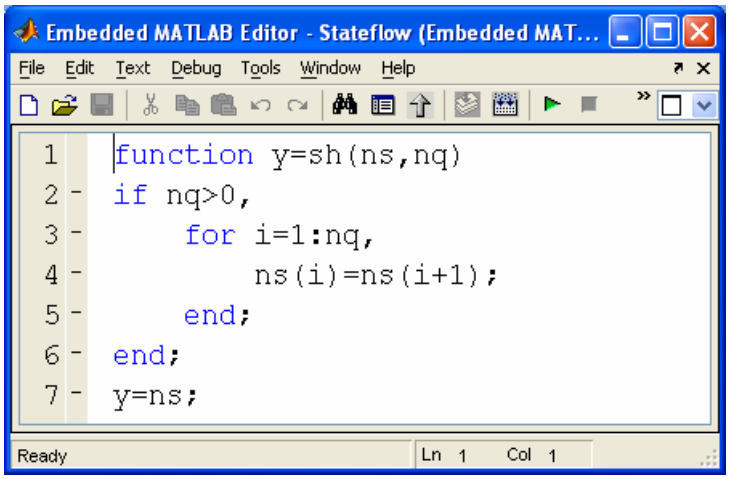


Рис. 6.37

В результате использования рассмотренных внутренних графических компонентов Stateflow появляется возможность определить переменные и массивы внутри самой SF-модели (как это видно на рис 6.35). Это упрощает описание общей интегрированной модели MATLAB + Simulink + Stateflow по сравнению с вариантом, в котором активно используются переменные и функции, определенные в MATLAB (этот вариант ранее представлен текстом m-сценария и SF-моделью рис. 6.33).

Дальнейшим естественным шагом является построение моделей более сложных систем массового обслуживания на основе использования рассмотренных выше базовых конструкций как элементов разрабатываемой модели. Этот процесс можно существенно упростить, если ввести иерархическое описание событий и данных для тиражируемых модулей, то есть выделить те переменные, которые имеют внутреннюю «видимость» и не используются вне данного модуля. Тогда процесс создания модели практически можно свести к копированию и вставке типовых конструкций в общей SF-диаграмме.

На рис. 6.38, 6.39 представлен пример построения модели двухфазной системы массового обслуживания, содержащей в первой фазе

накопитель и один канал обслуживания (Q1) и во второй фазе два параллельных канала обслуживания (Q21 и Q22), предваряемые накопителями. Входной поток заявок является пуассоновским и имеет интенсивность Li. Все накопители имеют одинаковую предельную длину очереди nqm. Канал обслуживания первой фазы имеет интенсивность потока обслуживания Lo, а каналы обслуживания второй фазы – Lo/3 каждый. Все эти переменные устанавливаются и могут варьироваться в управляющей MATLAB-программе. Собственно S-модель состоит из одного блока Chart и блока Scope, обеспечивающего визуализацию результатов работы модели.

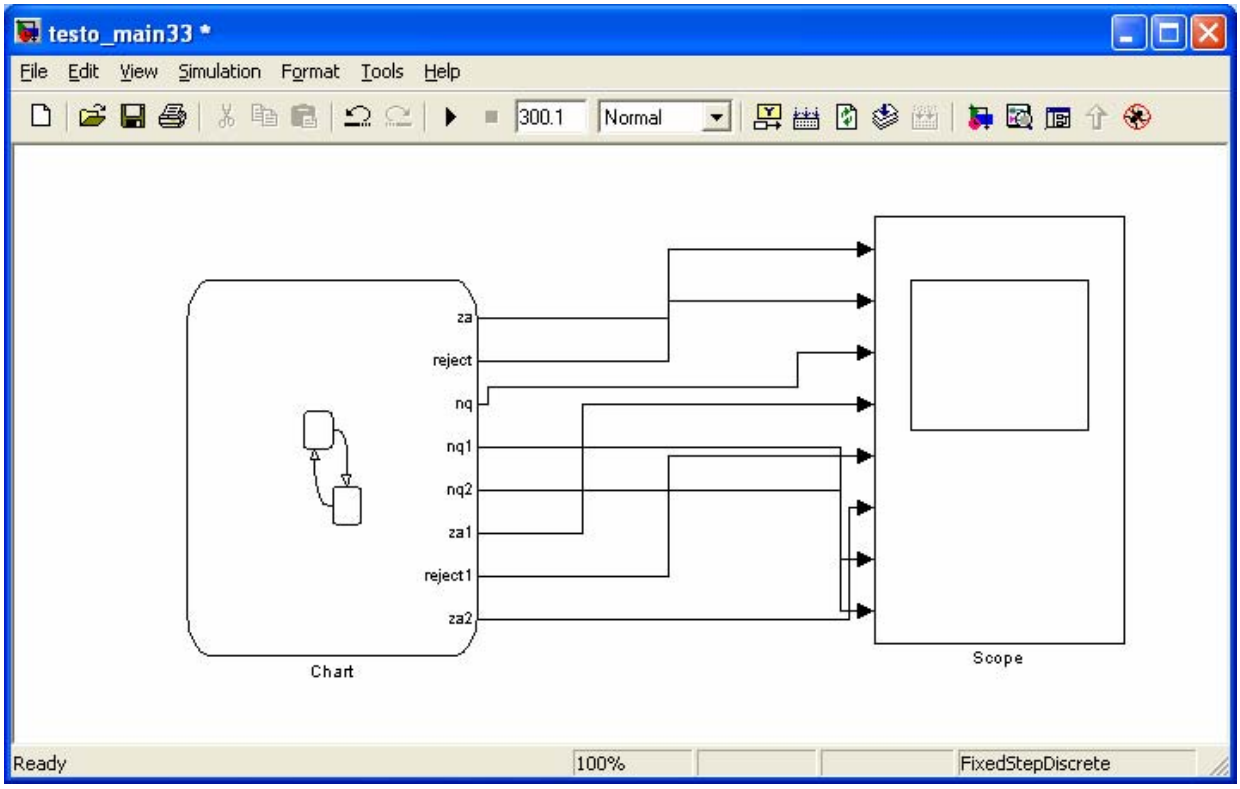


Рис.6.38

Наибольший интерес представляет построение собственно SF-модели, представленной на рис. 6.39. В ней блок Activ, как и ранее, является генератором потока заявок (событий zayvka) на входе прибора

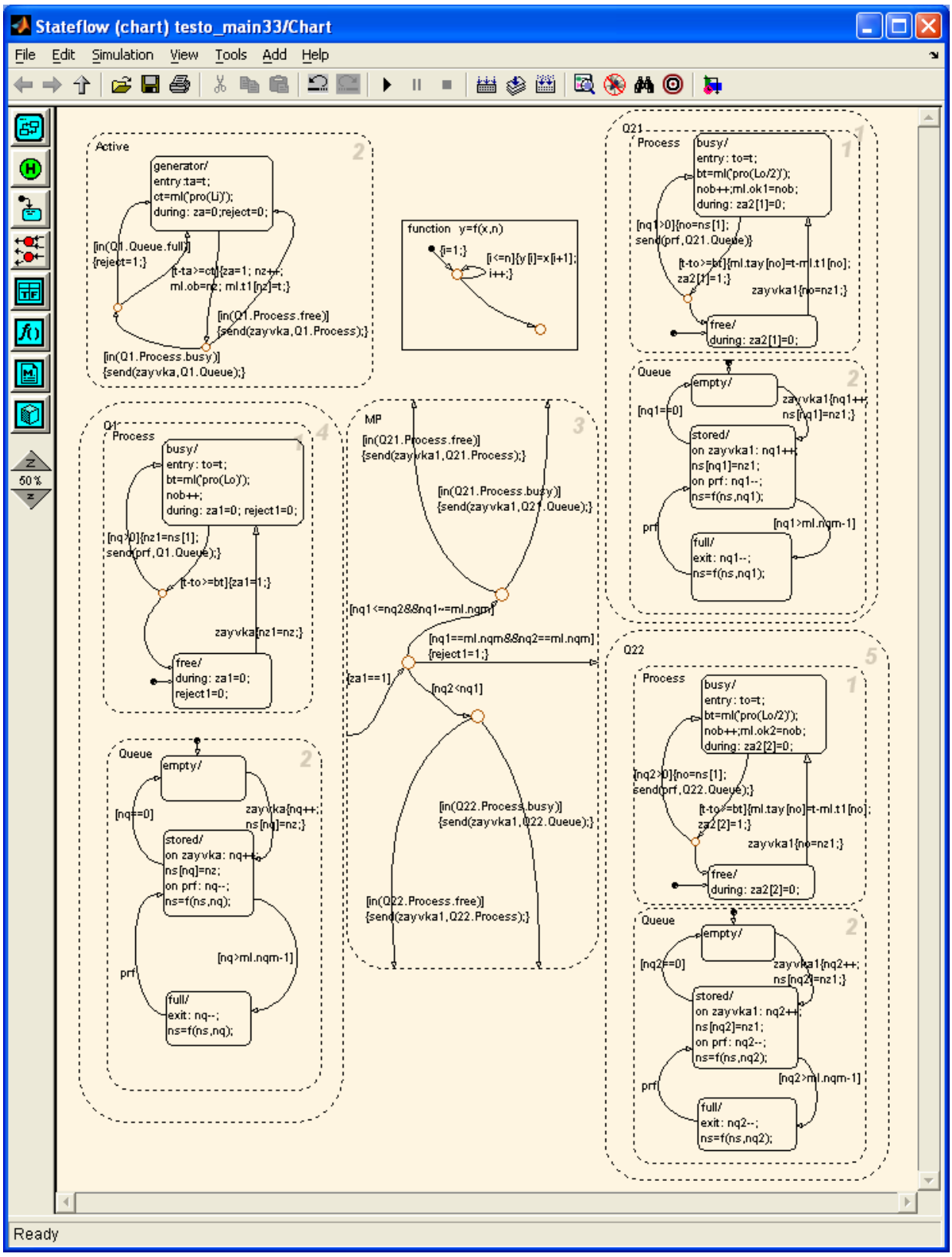

Stateflow (chart) testo_main33/Chart
File Edit Simulation View Tools Add Help
Active
generator/
entry:ta=t;
ct=ml('pro(Li)');
during: za=0;reject=0;
[in(Q1.Queue.full)]
{reject=1;}
[t-ta>=ct]{za=1; nz++;
ml.ob=nz; ml.t1[nz]=t;}
[in(Q1.Process.free)]
{send(zayvka,Q1.Process);}
[in(Q1.Process.busy)]
{send(zayvka,Q1.Queue);}
function y=f(x,n)
{i=1;}
[i<=n]{y[i]=x[i+1];
i++;}
Q21
Process
busy/
entry: to=t;
bt=ml('pro(Lo/2)');
nob++;ml.ok1=nob;
during: za2[1]=0;
[nq1>0]{no=ns[1];
send(prf,Q21.Queue)}
[t-to>=bt]{ml.tay[no]=t-ml.t1[no];
za2[1]=1;}
zayvka1{no=nz1;}
free/
during: za2[1]=0;
Queue
empty/
[nq1==0]
zayvka1{nq1++;
ns[nq1]=nz1;}
stored/
on zayvka1: nq1++;
ns[nq1]=nz1;
on prf: nq1--;
ns=f(ns,nq1);
prf
[nq1>ml.nqm-1]
full/
exit: nq1--;
ns=f(ns,nq1);
Q1
Process
busy/
entry: to=t;
bt=ml('pro(Lo)');
nob++;
during: za1=0; reject1=0;
[nq>0]{nz1=ns[1];
send(prf,Q1.Queue);}
[t-to>=bt]{za1=1;}
zayvka{nz1=nz;}
free/
during: za1=0;
reject1=0;
Queue
empty/
[nq==0]
zayvka{nq++;
ns[nq]=nz;}
stored/
on zayvka: nq++;
ns[nq]=nz;
on prf: nq--;
ns=f(ns,nq);
prf
[nq>ml.nqm-1]
full/
exit: nq--;
ns=f(ns,nq);
MP
[in(Q21.Process.free)]
{send(zayvka1,Q21.Process);}
[in(Q21.Process.busy)]
{send(zayvka1,Q21.Queue);}
[nq1<=nq2&&nq1~=ml.nqm]
[nq1==ml.nqm&&nq2==ml.nqm]
{reject1=1;}
[za1==1]
[nq2<nq1]
[in(Q22.Process.busy)]
{send(zayvka1,Q22.Queue);}
[in(Q22.Process.free)]
{send(zayvka1,Q22.Process);}
Q22
Process
busy/
entry: to=t;
bt=ml('pro(Lo/2)');
nob++;ml.ok2=nob;
during: za2[2]=0;
[nq2>0]{no=ns[1];
send(prf,Q22.Queue);}
[t-to>=bt]{ml.tay[no]=t-ml.t1[no];
za2[2]=1;}
zayvka1{no=nz1;}
free/
during: za2[2]=0;
Queue
empty/
[nq2==0]
zayvka1{nq2++;
ns[nq2]=nz1;}
stored/
on zayvka1: nq2++;
ns[nq2]=nz1;
on prf: nq2--;
ns=f(ns,nq2);
prf
[nq2>ml.nqm-1]
full/
exit: nq2--;
ns=f(ns,nq2);
50 %
Ready

Рис. 6.39

Q1 с интенсивностью Li. В этом блоке также фиксируется количество генерируемых заявок, передаваемое в УП переменной ob, и запоминаются моменты времени появления заявок на входе системы. Эти заявки обрабатываются в Q1. В структуре данного модуля объединяются ранее подробно рассмотренные компоненты Queue и Process, имитирующие работу накопителя и канала обслуживания первой фазы. Текущее количество заявок в очереди здесь, как и ранее, определено переменной nq. В момент окончания процесса обслуживания очередной заявки на выходе блока фиксируется значение переменной za1=1, которой во все остальные моменты времени присваивается нулевое значение.

Для распределения заявок, поступающих на вторую фазу обслуживания, в модели введен специальный модуль MP, иллюстрирующий возможности Stateflow по использованию внутренних переходов для формирования логически управляемых ветвей действий. Выполнение условия [za1=1] здесь порождает генерацию события zayvka1 – появления новой заявки на входе приборов второй фазы обслуживания, которая в зависимости от загруженности очередей Q21 и Q22, определяемых переменными nq1 и nq2, и текущих состояний блоков Q21.Process и Q22.Process (free или busy), пересылается в тот или иной модуль. В случае, когда оба накопителя второй фазы заполнены, в модуле MP фиксируется отказ путем присвоения значения переменной reject1=1, которой во все остальные моменты времени присваивается нулевое значение. По сути, принцип работы данного модуля повторяет принцип работы модуля Active, то есть реализует генерацию и пересылку заявок, поступающих на вторую фазу обслуживания, в необходимые блоки модулей Q21 и Q22.

Внутренняя структура Q21 и Q22 практически полностью повторяет структуру модуля Q1, с тем только отличием, что в них осуществляется фиксация интервалов времени нахождения заявок в системе, которые записываются в массив tay, определенный в УП, подсчитывается количество обслуженных каждым модулем заявок (переменные ok1 и ok2). Для фиксации моментов окончания обслуживания заявок используются компоненты массива za2[1] и za2[2]. Переменные nz и nz1, используемые в Q1, Q21,Q22, служат для запоминания номеров заявок, поступающих на обслуживание.

На рис.6.40 представлено окно обозревателя Stateflow, иллюстрирующее иерархию состояний, а также перечень событий и данных, имеющих видимость в пределах всей SF-диаграммы. Все остальные переменные: события типа prf, массивы типа ns, используемые для хранения номеров заявок, находящихся в очередях, вспомогательные данные типа ct, ta, bt, to, nob и т.п. определяются и имеют область видимости в пределах использующих их блоков.

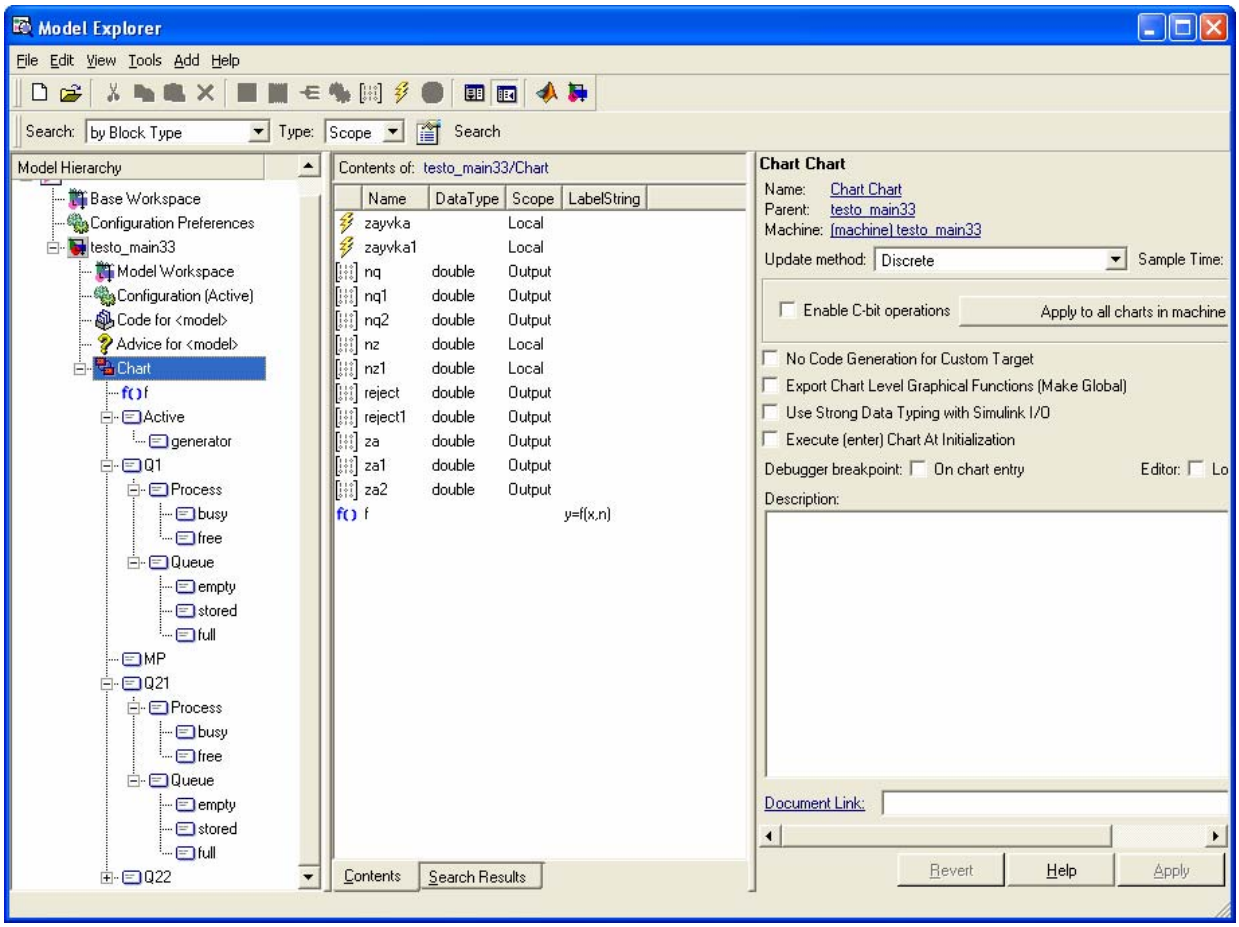


Рис. 6.40

В качестве иллюстрации работоспособности разработанной модели СМО на рис. 6.41 представлено визуальное отображение всех основных процессов, реализуемых в системе, получаемое в блоке Scope S-модели. Значения интенсивности потока заявок при этом равно Li=1 Гц, а интенсивностей потоков обслуживания в первой фазе Lo=1 Гц и во второй

фазе, соответственно, Lo/3=0.33 Гц. Время моделирования одной реализации равно 300 с.

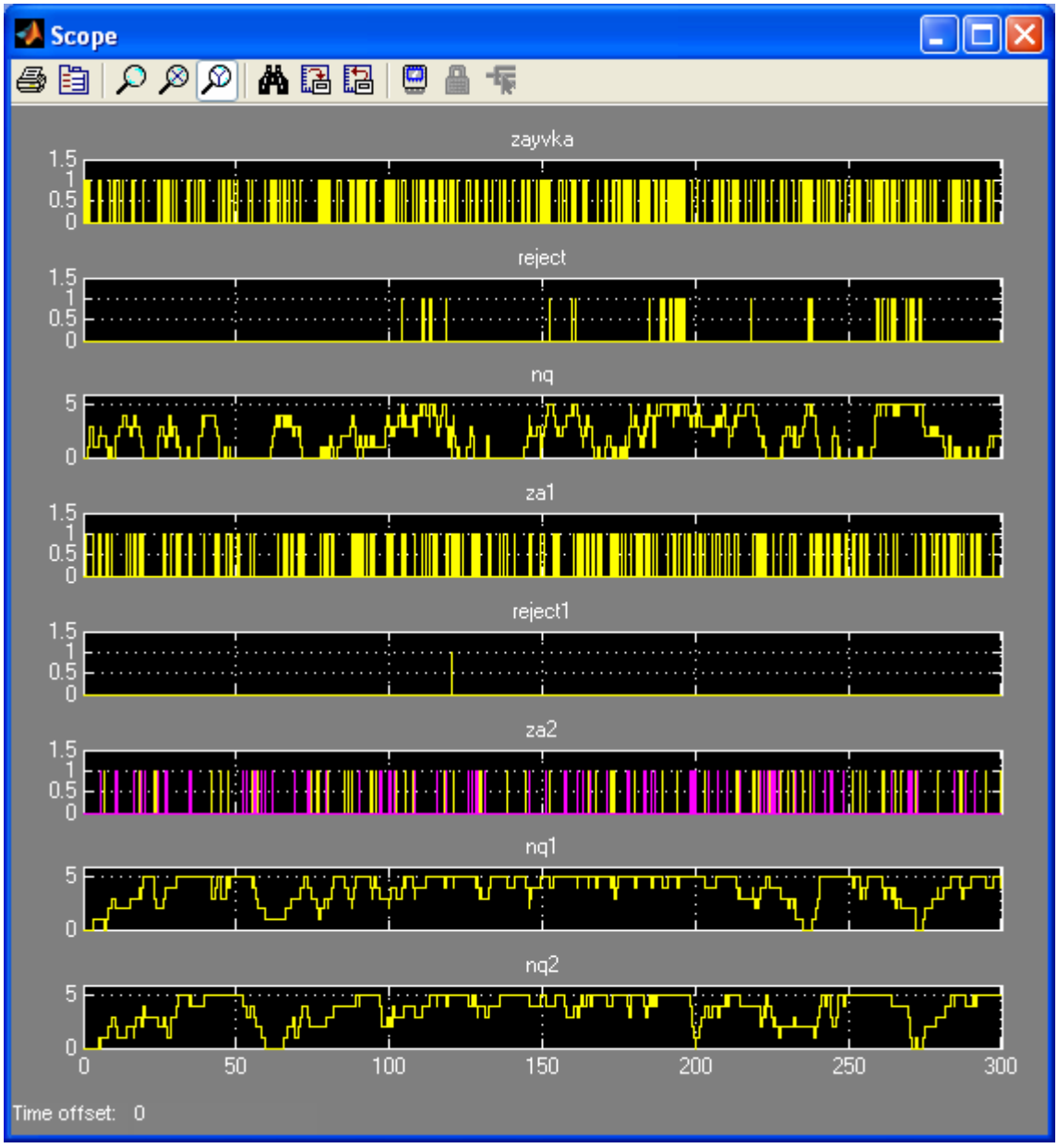


Рис. 6.41

Таким образом, представленные примеры иллюстрируют развитые возможности по созданию имитационных моделей СМО и любых событийно управляемых систем с использованием интегрированной среды MATLAB + Simulink + Stateflow.