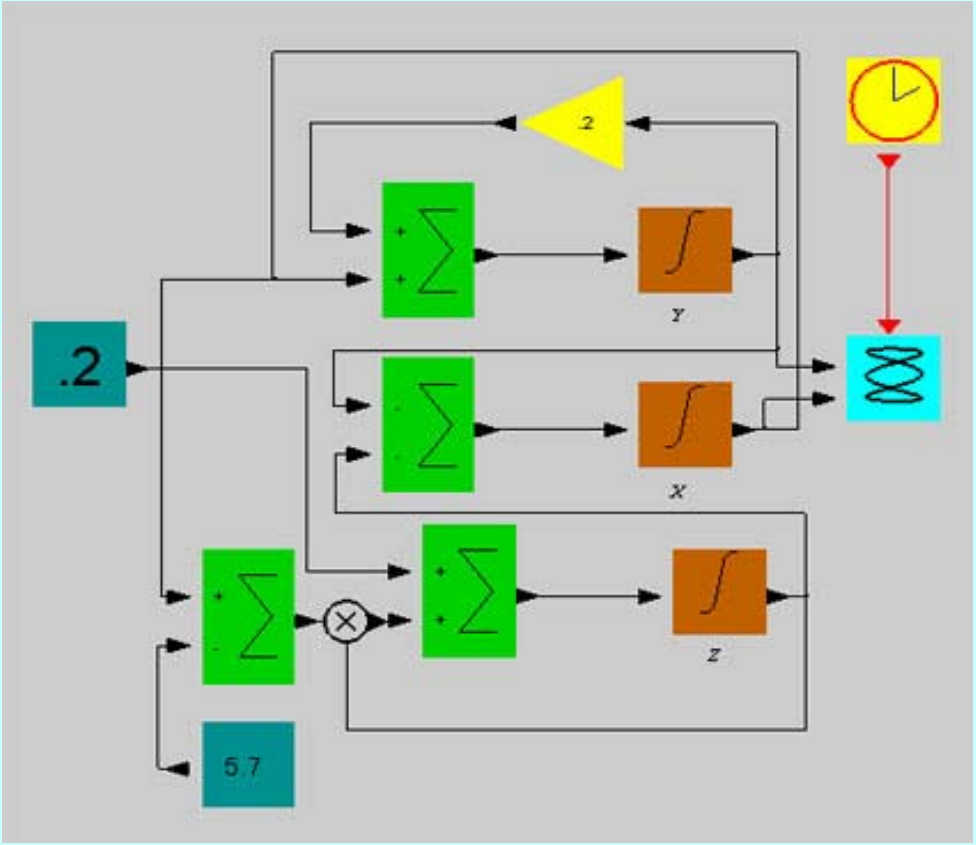
С. Данилов

SCICOS
ПАКЕТ SCILAB ДЛЯ МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ

РУКОВОДСТВО



2011 г

Министерство образования и науки Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования
«Тамбовский государственный технический университет»

С.Н. Данилов

SCICOS
ПАКЕТ SCILAB ДЛЯ МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ

РУКОВОДСТВО

Учебное пособие по дисциплине «Основы управления РЭСБН» и «Информационные технологии».
Утверждено Редакционно-издательским советом ТГТУ для студентов дневной и заочной форм обучения специальности 210303 – «Бытовая радиоэлектронная аппаратура» направления 210400 – «Радиотехника»

Тамбов
Издательство ТГТУ
2011

УДК 519.768.2(075.8)
ББК з973.26-018.3я73
Д183

Рецензенты: д.т.н., профессор Чернышова Т.И.,
заведующий заочным отделением
Тамбовского политехнического техникума, к.т.н., доцент Князев И.Н.

Д183 Данилов С.Н.
SCICOS. Пакет Scilab для моделирования динамических систем. Руководство: учебное пособие для студентов дневной и заочной форм обучения специальности 210303 – «Бытовая радиоэлектронная аппаратура» и направления 210400 – «Радиотехника».. – Тамбов: ТГТУ, 2011. – 74 с.

Учебное пособие по Scicos – составной части свободно распространяемого пакета Scilab предназначено для первого знакомства и приобретения навыков работы. В настоящее время в нашей стране нет подобного руководства, что затрудняет распространение этого мощного и эффективного средства. Знание этого инструмента моделирования необходимо специалистам во всех областях науки и техники. Учебное пособие предназначено для студентов дневной и заочной форм обучения специальности 210303 – «Бытовая радиоэлектронная аппаратура» и направления 210400 – «Радиотехника». Список использованной литературы приведен в конце руководства.

УДК 519.768.2(075.8)
ББК з973.26-018.3я73

© ГОУ ВПО «Тамбовский государственный технический университет», 2011

Введение

Почти за полтора десятка лет массового использования средств вычислительной техники для решения задач моделирования сложилось представление о наборе подходящих программных средств. Эти программные продукты достаточно дорогие и их использование наносит серьезный ущерб бюджету любого проекта. Использование этих программ в учебном процессе (несмотря на наличие специальных цен) также ощутимо для бюджета. Использование пиратских копий является неэтичным, если ориентироваться на легально приобретаемые копии, или преступным, если использовать пиратские копии и ориентировать обучаемых на использование пиратских копий.

На фоне возрастающего интереса пользователей к свободному программному обеспечению имеет смысл рассмотреть альтернативные свободные проекты, связанные с обработкой данных и научными расчетами. Не нужно забывать, что свободное распространение программ является старой академической традицией. Зародилась она в учебных и научных организациях, где передавали друг другу программы, созданные «для себя» с целью решения конкретных учебных и научных задач.

Свободно распространяемый пакет Scilab был разработан в 1994 году во Франции, в Национальном исследовательском институте информатики и автоматизации INRIA и Национальной школе дорожного ведомства ENPC. С 2003 года поддержкой Scilab занимается консорциум Scilab Consortium.

Scicos (Scilab Connected Object Simulator) – составная часть пакета Scilab. Scicos в его составе обеспечивает возможность визуального моделирования динамических систем. Эти моделируемые системы могут быть как непрерывными, так и дискретными.

Scicos имеет дружественный графический интерфейс пользователя для редактирования моделей, состоящих из соединенных блоков Scicos. Scicos блоки могут быть найдены в палитрах Scicos или определены пользователем.

В состав пакета Scilab входит большая библиотека математических функций, расширяемая программами, написанными на языках высокого уровня, например, на C или Fortran 77. В состав пакета входит интегрированный язык высокого уровня, отличающийся в некоторой степени от языка C.

Scilab – пакет числовой обработки данных. Он очень хорош для обработки сигналов. Основной режим работы – командный. К основным функциям пакета Scilab можно отнести работу с матрицами (в том числе разреженными), решение обыкновенных дифференциальных уравнений, численное дифференцирование и интегрирование, построение двумерных и трехмерных графиков по формулам и по результатам расчетов, решение задач линейного программи-

рования, а также возможность создания пользовательских программ. Пакет является кроссплатформенным, на сайте проекта (http://www.scilab.org/) можно найти самые последние сборки для различных операционных систем. На русском языке подробную информацию и пособия по работе с пакетом можно получить как на сайте http://www.csa.ru/~zebra/my_scilab/, так и на сайте http://www.scilab.land.ru/.

С системой Scilab доступны следующие комплекты инструментов:

- 2-х и 3-х мерная графика, анимация;
- линейная алгебра, матрицы;
- полиномы и рациональные функции;
- моделирование: ОДУ и ДАУ решатель;
- классическое и робастное управление, LMI оптимизация;
- дифференциальная и недифференциальная оптимизация;
- обработка сигналов;
- графы и сети;
- параллельно Scilab, используемый PVM;
- статистика;
- интерфейс с компьютерной алгеброй: генерация кодов пакетом Maple для Scilab;
- Scicos и др.

Перечислим основные составные части Scicos.

- Графический редактор: Scicos предоставляет иерархическому графическому редактору для строительства моделей динамических систем, используя блок-схемы. Многие предопределенные блоки также предоставлены в различных палитрах. Новые блоки могут быть определены пользователем в C, Fortran’e или Scilab.
- Компилятор: компилятор Scicos использует образцовое описание, обычно собираемое редактором Scicos, построить столы планирования, которые могут тогда использоваться тренажером и кодовой функцией поколения.
- Симулятор: симулятор Scicos использует таблицы планирования и другую информацию, предоставленную компилятором, чтобы управлять моделированиями. Симулятор имеет гибридную природу, при которой он должен иметь дело с дискретными и непрерывными системами времени, и событиями. Для непрерывного времени используется решатель однородных дифференциальных уравнений (ОДУ) LSODAR или решатель дифференциальных алгебраических уравнений (ДАУ) DASKR в зависимости от природы рассматриваемой непрерывной системы.
- Генератор объектного кода: Scicos может генерировать код C для того, чтобы «понять» поведение некоторых подсистем. Эти подсистемы не должны включать непрерывные во времени компоненты.

Scicos – независимое программное приложение для моделирования, но доступ к Scilab и его функциональным возможностям обеспечивает большую гибкость и расширяет диапазон возможностей моделирования.

Например, при обработке сигнала, легче использовать функции Scilab и написать маленькую программку, чем писать код для базовых функций обработки сигнала. Кроме того, Scilab воспринимает модель Scicos как функцию. Это удобно, когда нужно управлять пакетной работой Scicos.

Наличие доступа к функциям Scilab, при создании моделей очень важно.

1. Пользователь Scicos часто имеет необходимость использовать функции Scilab, например предназначенные для создания фильтров обработки сигналов и др.

2. Язык программирования Scilab может использоваться для пакетной обработки множества задач моделирования или обобщая – модели, разработанные Scicos, могут использоваться как функции в Scilab.

3. Графические возможности Scilab могут использоваться для последующей обработки результатов моделирования.

Но интеграция Scicos и Scilab идет дальше. Редактор Scicos полностью написан на языке Scilab и это обеспечивает много преимуществ:

редактор легко настраивается добавлением меню и функциональных возможностей, изменением поведения и т.п.;

гибкость в проектировании формы и изображений блоков и связей (стандартная графика Scilab);

простота разработки и отладки;

Scicos переносим на новую платформу, на которую переносим Scilab;

структура данных Scicos аналогична структуре данных Scilab и может быть обновлена функциями Scilab: легкий способ взаимодействия с блоками.

Есть и неудобства:

Scilab - интерпретируемый язык и для очень больших моделей, некоторые функции редактора могут быть медленными;

ограниченные возможности графического интерфейса.

Обычно, создание простой новой модели Scicos предусматривает выполнение следующих операций:

запуска Scicos с пустым окном;
открытия одной или более палитр;
копирования нужных блоков из палитр в окно;
установки параметров блоков нужной величины;
соединения входов и выходов блоков;
компиляцию и запуск модели;
переименования и сохранения.

Вышеуказанное не включает использование более сложных операций, как например, создание суперблочных конструкций, создание новых базовых блоков, и т.п., а также косметических операций, как например, редактирование иконок, окрашивание или изменение размеров блоков в окне. Эти операции будут рассмотрены ниже.

Ниже, мы последовательно, по операциям выполним простой сеанс Scicos, в котором создадим модель и запустим ее.

Для запуска Scicos необходимо выбрать Scicos в меню Applications или набрать в основном окне Scilab:

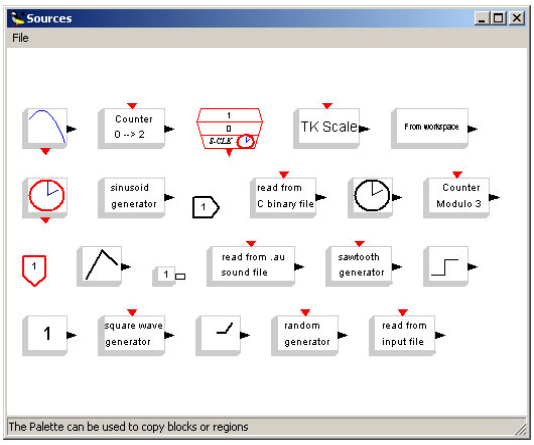
--> scicos();

Открывается основное окно Scicos с пустым полем. Вид основного окна может немного различаться из-за различия операционных систем. Чтобы загрузить уже существующую модель, имя файла модели должно быть использовано как аргумент в одиночных кавычках с расширением. Например:

--> scicos('Flight.sci')

Окно остается открытым в течение всего сеанса Scicos. Откроем палитру блоков. Для того, чтобы открыть ее, выберите Palette в меню Edit и найдите имя нужной палитры среди предлагаемых.

Палитра содержит имеющиеся блоки Scicos. В Scicos есть три типа базовых блоков: обычные базовые блоки, блоки пересечения нуля и блоки синхронизации. Эти блоки могут быть скопированы в окно, которое используется для создания модели Scicos. Эти палитры не могут быть отредактированы. Для редактирования палитры, она должна быть загружена как элемент модели Scicos. Подробнее рассмотрим это ниже. Ниже показан вид палитры источников сигналов (Sources).



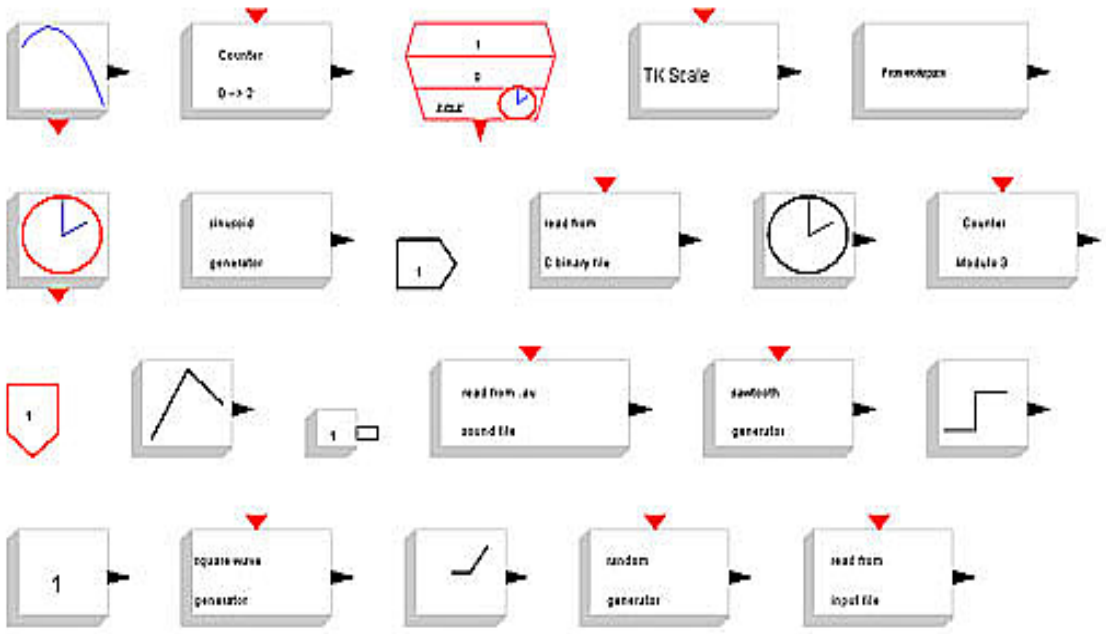
Палитра Sources

Перечень палитр и их состав с краткой характеристикой элементов дан ниже.

Краткое описание палитр и блоков

Палитра по сути просто подобна любой другой модели Scicos; фактически любая модель может быть загружена как палитра, с использованием опции Load как пункта команды Palette в меню. Чтобы заставить Scicos признавать новую палитру, вы должны добавить имя файла, который содержит ее в вектор scicos_pal в Scilab. Это делается автоматически, когда Вы используете пункт Save as Palette в меню Palette. Рекомендуется для более удобной работы набрать в окно Scicos наиболее часто используемые вами блоки и сохранить как новую палитру под именем, которое вы выберете.

Источники (Sources)



Номера в скобках в описаниях блоков соответствуют порядковому номеру блока на рисунке, если считать слева на право.

CLKINV_f – входной порт активизации. Он должен использоваться только внутри суперблоков Scicos в качестве входного порта активации. Если этих входов несколько, они нумеруются подряд, начиная с единицы (12).

CLOCK_c - часы активации. Устанавливается шаг работы и время начала работы (6).

CONST_m – константа (18).

Counter – счетчик. Устанавливается максимальная величина счета и способ счета: на увеличение или на уменьшение (2).

CURV_f – формирует табулированную функцию времени. Между точками используется линейная интерполяция. После окончания времени вывода данных на выходе остается последняя табличная величина. Пользователь может изменить функцию, редактируя кривую (13).

FROMWSB – передача данных из рабочего пространства Scilab в Scicos (5).

GENSIN_f - генератор синусоидального сигнала (7).

GENSQR_f – генератор прямоугольных волновых сигналов. Начальное состояние единица. Второе минус единица. Необходима дискретная активация. Блок CLOCK_c для этого не подходит.

IN_f - входной порт. Он должен использоваться только внутри суперблоков Scicos в качестве входного порта активации. Если этих входов несколько они нумеруются подряд, начиная с единицы (19).

INIMPL_f - входной порт, вход в систему снаружи (по умолчанию) (8).

Modulo_Count - счетчик по модулю. Параметр – величина модуля (11).

RAMP - блок выдает сигнал в установленное время, величина и скорость нарастания которого задаются (20).

RAND_m - генератор равномерно распределенных случайных чисел в диапазоне 0…1 (21).

READAU_f - считывание звуковых файлов AU (15).

READC_f - считывание двоичных данных (9).

RFILE_f - считывание из файла (22).

SampleCLK - Разница между SampleCLK и CLOCK_c заключается в том, что все блоки SampleCLK в схеме являются синхронными. Синхронизм осуществляется за счет двух разных методов вычисления на этапе компиляции. Первый способ основывается на применении часов, которые непосредственно связаны со счетчиком и которые активируют блок выбора событий. Часы работают в соответствии со следующим правилом. Если все блоки имеют одинаковый сдвиг времени, тогда частота часов есть функция периода, и смещение часов равно этому сдвигу. Если смещения являются различными, то частота хода часов это функция периода и смещения, и смещение часов равно нулю. Счетчик

считает от единицы до наименьшего общего кратного периодов (НОК). Число выходов в ESELECT_f блока совпадает с НОК. Второй метод основывается на многочастотном блоке Sources, который генерирует события только для определенного времени. События при этом генерируются не периодически, как и в первом случае (3).

SAWTOOTH_f - генератор пилообразного сигнала в диапазоне 0…1. Установок нет (16).

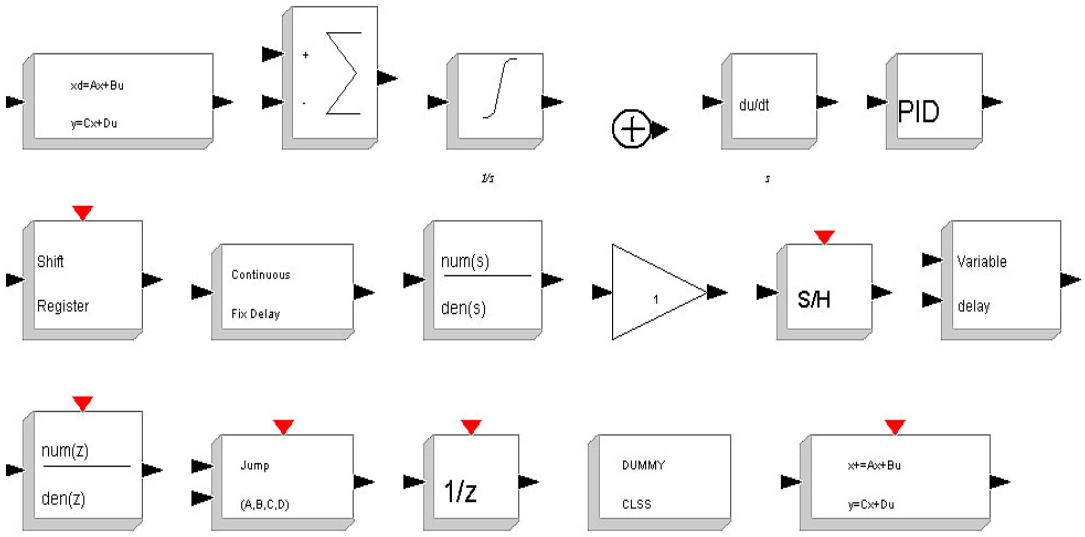
SIGNAL_CREATOR – генератор функции заданной в трех точках с интерполяцией. Он может быть периодическим (установить параметр). Выход активации работает по пересечению нуля от плюса к минусу (1).

STEP_FUNCTION - генератор ступенчатого сигнала (17).

TIME_f – время. Установок нет. Линейно нарастающий сигнал. Крутизна равна единице (10).

TK SCALE – блок выдает постоянный сигнал, величина которого может быть изменена в установленном диапазоне непосредственно во время работы модели с помощью ползункового регулятора с учетом фактора нормализации (для увеличения точности) (4).

Линейные блоки (Linear)



CLINDUMMY_f – блок должен устанавливаться в любой схеме, которая содержит блок с пересечением нуля, но не в непрерывной системе с состоянием, потому что это ode-решатель, который находит нулевое пересечение (16).

CLR - непрерывная передаточная функция. Блок реализует линейную систему, представленную рациональной функцией типа (1+s)/(1+s)*(1+s) и т.п. (9).

CLSS – непрерывная система в пространстве состояний. Должны быть заданы матрицы A, B, C, D и начальное состояние x_0 (1).

DERIV – блок производной. Она вычисляется по входному сигналу $\Delta u/\Delta t$ (5).

DLR - дискретная передаточная функция. Блок реализует линейную дискретную систему, представленную рациональной функцией типа (1+z)/(1+z)*(1+z) и т.п (13).

DLSS - дискретная система в пространстве состояний. Должны быть заданы матрицы A, B, C, D и начальное состояние x_0 (17).

DOLLAR_m - оператор задержки (1/z). Входная величина подается на выход по сигналу активации, а после этого на входе запоминается новая входная величина (15).

GAINBLK – усилитель (10).

INTEGRAL_m – интегратор (3).

PID – PID регулятор (6).

Shift REGISTER – сдвиговый регистр (7).

SAMPHOLD_m – (S/H) экспозиция на выходе входного сигнала и считывание нового входного по сигналу активации (11).

SUM_f – сложение (4).

SUMMATION - блок выполняет суммирование или вычитание входных сигналов. Этот блок может суммировать или вычитать скаляр, вектор, или матрицы. Число входов задается параметром. Этот параметр может быть вектор, состоящий из плюс и минус единиц или это может быть положительная величина (скаляр). В первом случае количество единиц указывает число входов, а знаки указывают, является ли это суммированием или вычитанием. Во втором случае это блок суммирования и величина указывает число входов.

При переполнении результат может иметь различные формы:

1 – нормальный не ограниченный результат.

2 – ограниченный результат.

3 – сообщение об ошибке, предупреждающее пользователя о переполнении.

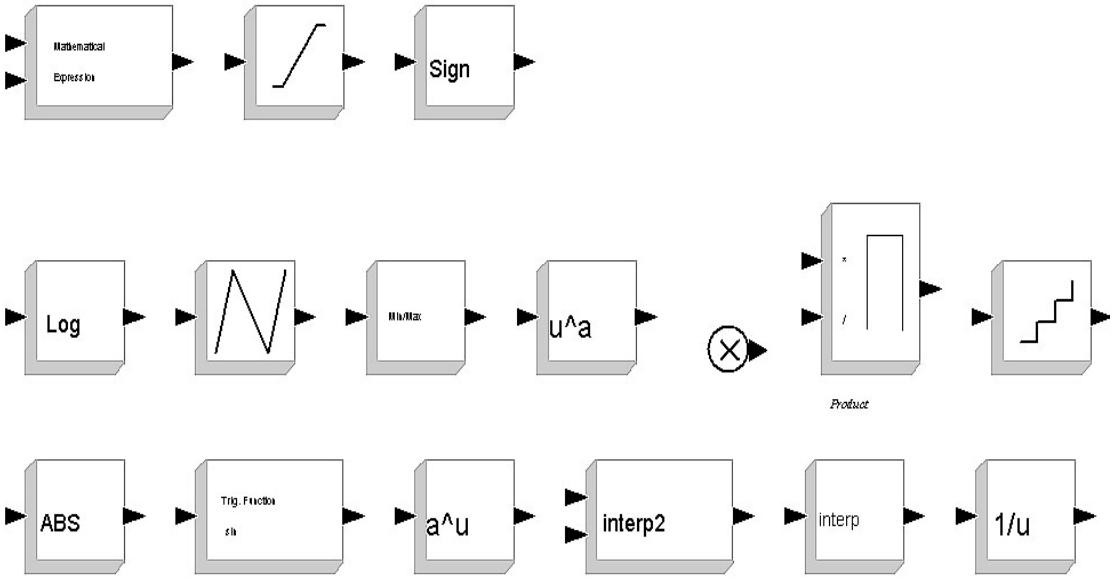
Пользователь может выбирать одну из этих трех форм, установкой в поле "DO ON OVERFLOW": 0, 1 или 2. (2).

TCLSS - непрерывная линейная система со скачком (14).

TIME_DELAY – постоянная задержка по времени (8).

VARIABLE_DELAY – переменная задержка. Первый сигнал – задерживаемый, второй величина задержки (12).

Нелинейные блоки (Non_linear)



ABS_VALUE – абсолютная величина (8).

EXPBLK_m – экспонента (13).

EXPRESSION – математическое выражение (1).

INTRP2BLK_f – 2D интерполяция (11).

INTRPLBLK_f – интерполяция (12).

INVBLK – инвертирование (16).

LOGBLK_f – логарифмирование (4).

LOOKUP_f – блок, заданный таблицей (5).

MAXMIN – блок выдает на выход наибольший или наименьший элемент или элементы входных элементов. Функцию можно выбрать, обращаясь к параметрам (6).

POWBLK_f – степень. Блок реализует операцию y(i) = u(i)^a. Размерность входного и выходного портов определяются при компиляции в соответствии с подсоединенными портами (7).

PROD_f – умножение (8).

PRODUCT – умножение (9).

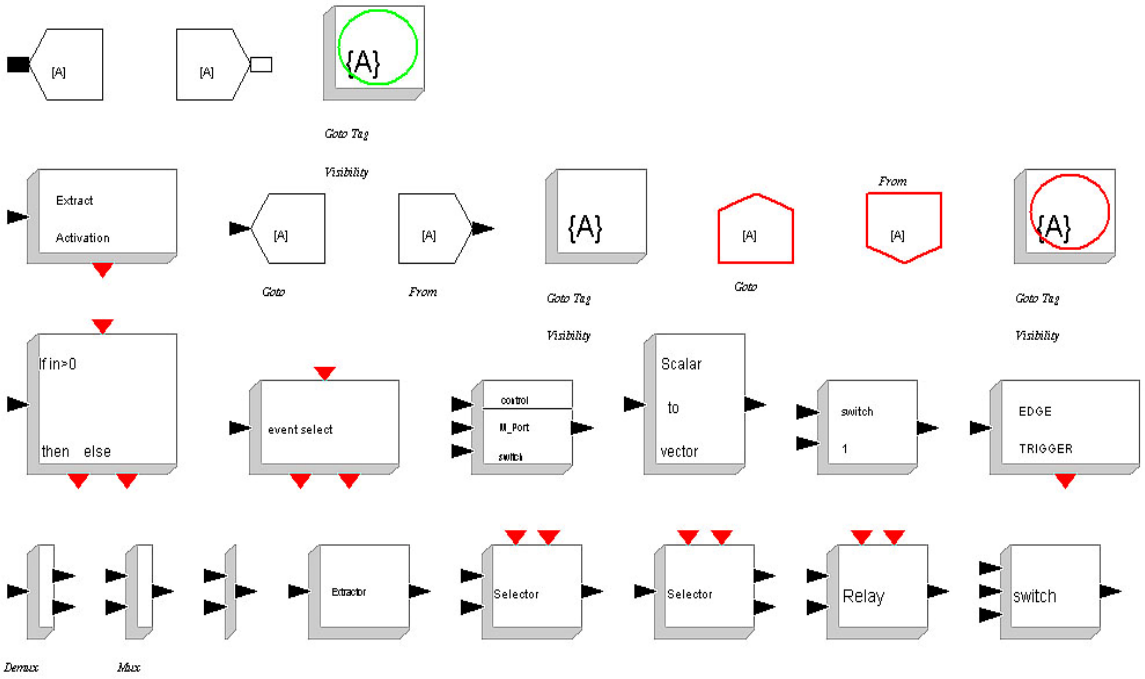
QUANT_f – квантизатор (10).

SATURATION – ограничитель (2).

SIGNUM – функция Signum (3).

TrigFun – тригонометрическая функция (12).

Переход (Branching)



CLKFROM – прием данных от соответствующего CLKGOTO (9).

CLKGOTO – передача данных соответствующему блоку CLKFROM (8).

CLKGotoTagVisibility - определяют область видимости этикетки CLKGOTO (10).

DEMUX – демультиплексор (17).

EDGE_TRIGGER – этот блок генерирует событие на увеличение, уменьшение или и то, и другое (см. установку параметров). Он реагирует только на скачек совпадающий с событием. Выходное событие синхронно с событием, вызвавшим скачек. Непрерывное пересечение нуля не обнаруживает (16).

ESELECT_f – блок синхронизации подобный If-Then-Else. Вход и выход синхронны. Входное событие направляется на тот или иной порт в зависимости от величины сигнала основного входа (12).

Extract_Activation – блок преобразования входного сигнала в сигнал активации (4).

EXTRACTOR – выделяет конкретный заданный входной сигнал из многомерного входного сигнала (20).

FROM – прием данных от соответствующего GOTO (6).

FROMMO – прием данных от соответствующего GOTOMO (2).

GOTO – передача данных блоку FROM (5).

GOTOMO – передача данных блоку FROMMO (1).

GotoTagVisibility – определяют область видимости этикетки (7).

GOTO GotoTagVisibilityMO – определяют область видимости этикетки GOTOMO (3).

IFTHEL_f – блок синхронизации с условием If-Then- Else (11).

ISELECT_m – выбирает сигналы из поступающих событий. Блок имеет один вход (22).

M_SWITCH – многопозиционный ключ (13).

MUX – мультиплексор (18).

NRMSOM_f – слияние данных (19).

RELAY_f – реле (23).

SCALAR2VECTOR – преобразование скаляра в вектор (14).

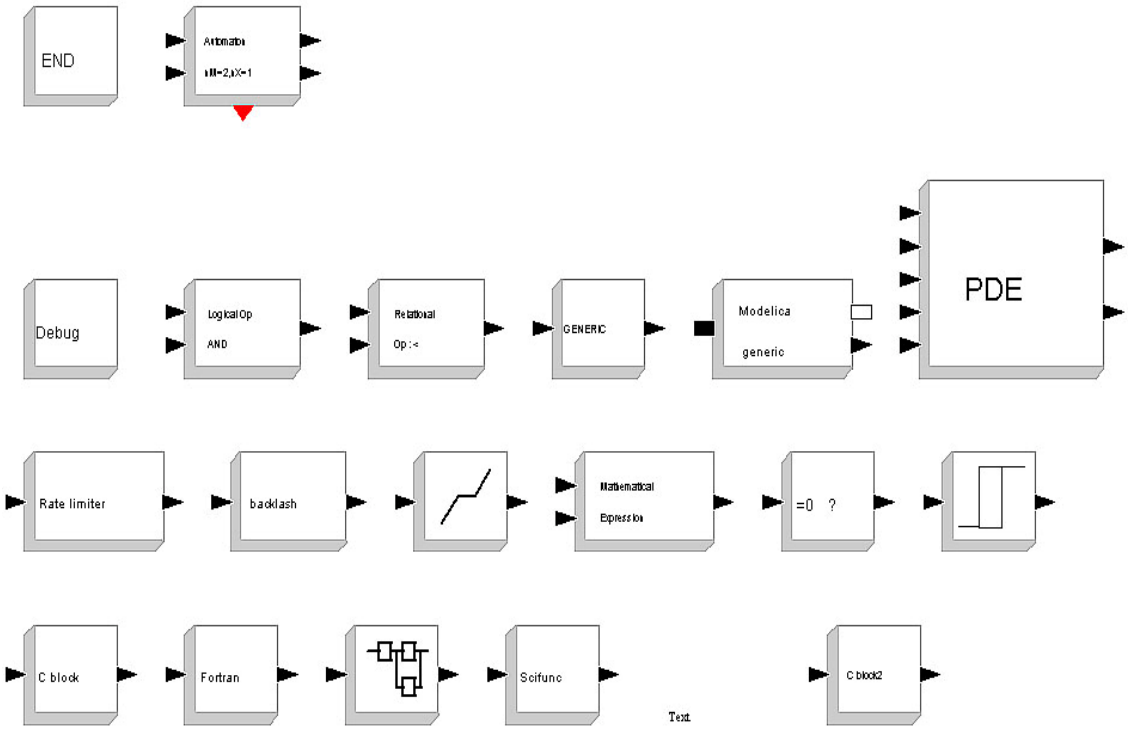
SELECT_m – блок принимает векторные или матричные сигналы. Нужно установить параметр типа сигнала, который будет приниматься – Input Type

(вектор или матрица). Блок решает, какие элементы выбрать в зависимости от типа входа (21).

SWITCH2_m – блок пропускает сигнал через первый (верхний) вход или через третий (нижний) в зависимости от величины сигнала на среднем (втором) управляющем входе. Можно выбрать критерий (порог) подключения к первому входу (24).

SWITCH_f – блок ручного переключения. В параметрах задается количество положений переключателя, и в каком положении переключатель находится (15).

Другие (Others)



AUTOMAT – Этот блок дает возможность строить гибридные автоматы, то есть гибридные системы, чья дискретная часть определена способами и переходами между способами, а непрерывная часть определена через дифференциальные алгебраические уравнения. Блок обеспечивает автоматическое переключение между подсистемами. Подсистемы построены таким способом, что они имеют стационарный входной вектор, и вычисляют плавную и скачкообразную функции (пересечение нуля) и передают их назад к блоку автомата. Стационарные переменные определены в блоке автомата, подсистемы – статические функции (подробнее см. help на панели меню) (2).

BACKLASH – мертвый ход (10).

c_block – блок, задаваемый пользователем на языке C (20).

CBLOCK – блок создает шаблон функции на языке С. Кроме того создает библиотечные и объектные файлы (15).

CONSTRAINT_f – определение имплицитных алгебраических отношений (13).

DEADBAND – обеспечивает заданную область нулевого выходного сигнала (11).

DEBUG_SCICOS – блок отладки - мощный инструмент отладки в Scicos. Когда он помещен в модель Scicos, в течение моделирования он проверяет каждый блок перед активацией и после. Открывая блок отладки, пользователь может писать scilab сценарий, который будет выполнен при его активации. Блок отладки очень похож на стандартный Scilab блок пятого типа, который вызывается двумя аргументами (блок и флаг). Он имеет доступ к функциям типа curblock (), scicos_time (), и т.д. Для каждого блока, в течение моделирования, блок отладки вызывается первым (как будто это сам блок).

Потом работает сам блок, а потом опять блок отладки. Таким образом, он контролирует величины блока до и после его работы. Блок отладки может также изменять эти величины. Например, если в блоке отладки установлена простая пауза, то моделирование идет пошагово. Но объявление паузы может быть условным. Например:

If curblock () == 3 and scicos_time () > 2 and флаг == 2 then pause, end.

Испытание может также быть сделано зависящим от величин блока, например: If block.outptr (1 ()) > 1 9 then disp (scicos_time) (), end.

Этот блок может также использоваться, чтобы создать файлы, с информацией о величинах блока в течение моделирования.

Следует помнить, что для функционирования блока отладки, уровень отладок должен быть установлен два или выше. Уровень отладок изменяется автоматически, когда блок отладки помещен внутрь модели. Можно вернуть уровень отладок к нулю не удаляя его из модели. Это может быть выполнено, используя меню Debug Level, или в Scilab немедленно (часто во время паузы), используя функцию scicos_debug (3).

ENDBLK – этот блок может быть использован для установки конечного времени работы модели. При этом модель будет остановлена по данным этого блока, а не по установке в Simulate/Setup. Этот параметр может быть числом или символической переменной определенной в контексте scicos (2).

EXPRESSION – математическое выражение (12).

fortran_block – блок задаваемый пользователем на языке Fortran (16).

generic_block3 – блок обеспечивает общую функцию интерфейса, но вычислительная функция должна быть определена отдельно или как функция Scilab, или Fortran, или функция С. Помимо имени функции пользователь дол-

жен определить тип и т.п.. Функция, должна быть сохранена вместе с моделью и загружена или динамически связана перед моделированием (6).

HYSTHERESIS – гистерезис (14).

LOGICAL_OP – логическая операция (установка вида операции в параметрах) (4).

MBLOCK – блок обеспечивает легкий способ создания блока Scicos, поведение которого описано программой Modelica. При использовании этого блока, пользователь может писать и собирать Modelica программы в Scicos без создания интерфейса (7).

PDE – этот блок является реализацией нескольких численных методов (конечные элементы, разности и объемы 1 и 2 порядка) решения одномерных дифференциальных уравнений в частных производных (ЧДУ) в пределах Scicos. Математические рамки ограничивают ЧДУ линейным скаляром максимальным порядком два (время и пространство). Цель состоит в том, чтобы обеспечить инженеров и физиков удобным комплектом инструментов в Scicos, который позволит им графически описывать такие решаемые уравнения. Система решения выбирает наиболее эффективный численный способ в зависимости от типа уравнения и управляет решением (8).

RATELIMITER – блок ограничивает первую производную сигнала, проходящего через него. Сигнал на выходе изменяется не быстрее, чем заданный предел (9).

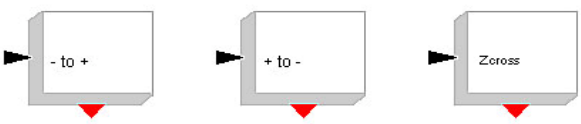
RELATIONALOP - операция логического сравнения. Вид сравнения устанавливается в параметрах блока (5).

scifunc_block_m - функциональный блок Scilab (18).

SUPER_f – создание суперблока (17).

TEXT_f – текстовый блок (19).

Порог (Threshold)

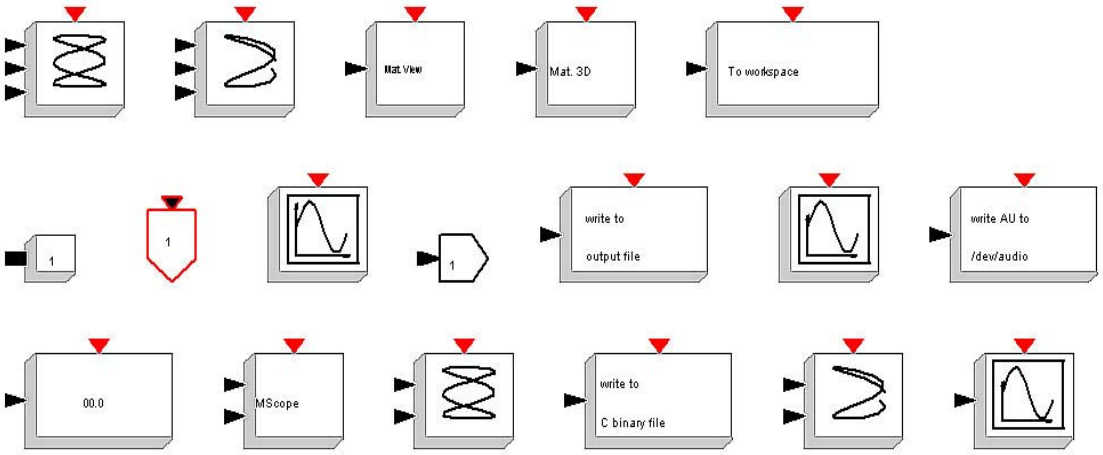


NEGTOPOS_f - порог пересечения нуля от минуса к плюсу (1).

POSTONEG_f - порог пересечения нуля от плюса к минусу (2).

ZCROSS_f - обнаружение пересечения нуля (3).

Средства отображения (Sinks)



AFFICH_m – дисплей (13).

CANIMXY – y = f(x) визуализирует изменение входных сигнала. Можно задать тип линии и ее размер (17).

CANIMXY3D – z = f(x,y) визуализирует изменение входных сигналов. Можно задать тип линии и ее размер (2).

CEVENTSCOPE – просмотр сигнала активации (8).

CFSCOPE - осциллограф с плавающей точкой подключения. Output window number – номер графического окна, используемого для показа. Его нужно задавать большим, чтобы не перепутать с другими окнами. Links to view – номер линии связи, с которой наблюдается сигнал (11).

CLKOUTV_f - выходной порт активизации (11).

CMAT3D – матричный 3D осциллограф (4).

CMATVIEW - матричный осциллограф с цветным отображением (3).

CMSCOPE – многооконный дисплей (14).

CSCOPE – простой осциллограф (18).

CSCOPXY – отображает зависимость y = f(x). Переменные x и y подаются на два входа (15).

CSCOPXY3D – отображает зависимость z = f(x,y). Переменные z, x и y подаются на три входа (1).

OUT_f - выходной порт (9).

OUTIMPL_f - выходной порт внешний (6).

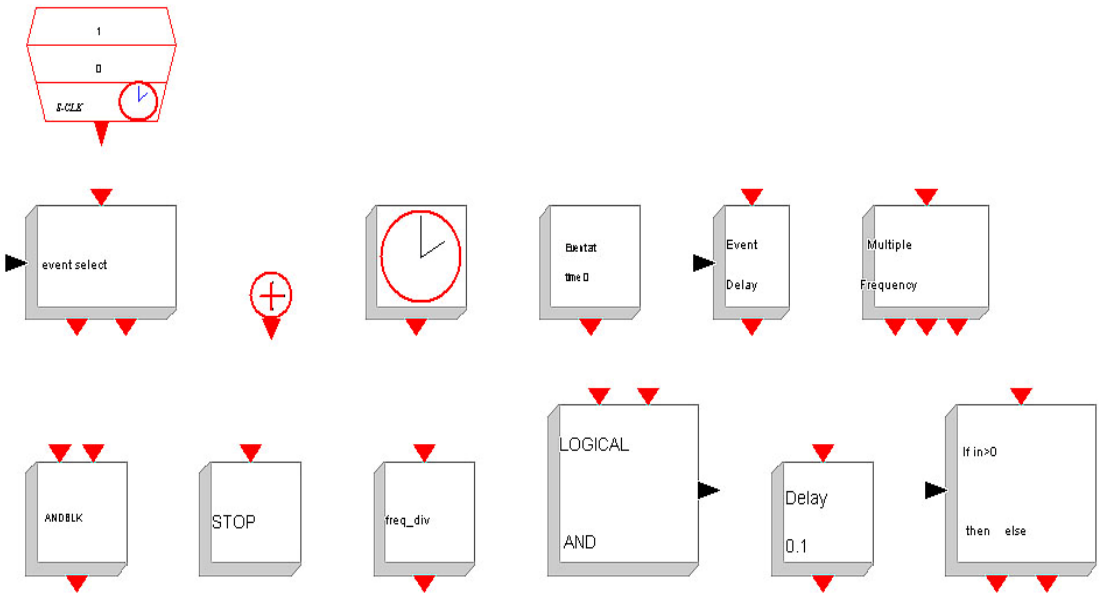
TOWORK_c – передача данных в рабочую область Scilab (5).

WFILE_f - запись в файл (10).

WRITEAU_f – запись звукового файла AU (12).

WRITEC_f - запись двоичных данные (16).

События – (Event)



ANDBLK – блок формирует событие на выходе при наличии на входе двух событий вместе (8).

ANDLOG_f – Блок формирует +1, если события пришли на оба входа вместе и –1, если пришло только одно событие (11).

CLKSOMV_f – этот блок суммирует до трех событий. Выход воспроизводит входные события. Этот блок собственно не блок scicos, т.к. при компиляции он игнорируется. Вход и выход синхронны (3).

CLOCK_c – часы активации (4).

ESELECT_f – синхронный блок Event-Select. Специальный блок, подобный If-Then-Else. Вход и выход синхронизованы. Входящее событие направляется на тот или иной выход в зависимости от управляющего сигнала (2).

EVTDLY_f – задержка события (6).

EVTGEN_f – генератор события. Генерирует одно событие в установленное время (5).

EVTVARDLY – переменная задержка события. После поступления события на вход активации, оно задерживается на величину, определяемую сигналом, поступающим на сигнальный вход. Блок может также генерировать начальное событие на выходе (12).

freq_div – деление частоты (10).

STOP_f – при поступлении события работа модели останавливается и активируется главное окно Scicos. Работа может быть начата вновь или продолжена (кнопка Run) (9).

IFTHEL_f – блок синхронизации If-Then-Else, т.е. синхронизация по первому или второму выходу в зависимости от выполнения или не выполнения условия (13).

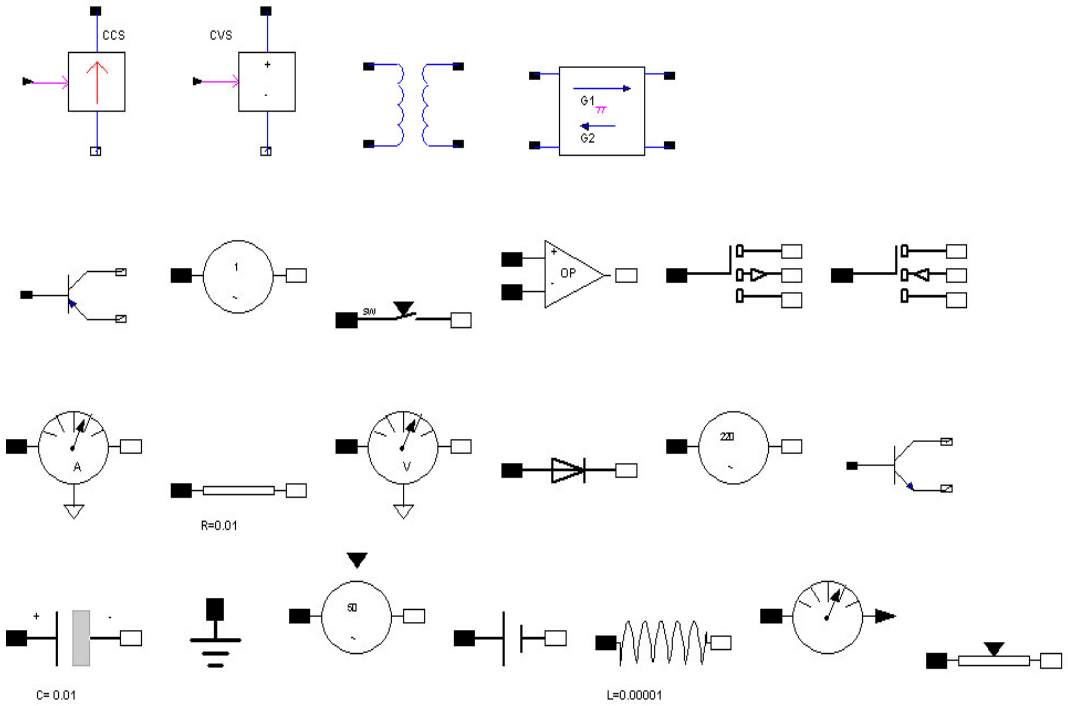
M_freq – блок создает события в определенные моменты времени работы модели. Периоды задаются в поле " Sample Time ", а временные сдвиги в поле "Offset". Блок имеет один вход, а количество выходов зависит от числа моментов задаваемого времени. Например, если вектор времени [1 1 2], и вектор сдвига - [0 0.5 0], тогда блок имеет 7 выходов.

- первый активизирован, когда время моделирования кратно первому периоду плюс первый сдвиг;
- второй активизирован, когда время моделирования кратно второму периоду плюс второй сдвиг;
- третий выход активизирован, когда произошло первое и второе событие;
- четвертый, когда время моделирования кратно третьему периоду плюс третий сдвиг;
- пятый, когда произошло первое и четвертое событие;
- шестой, когда произошло второе и четвертое событие;
- седьмое, когда произошло третье и четвертое событие и т.д.

Так что число выходов равно (2 ** (количество периодов)-1) (7).

SampleCLK – см. выше (1).

Электрический – (Electrical)



Capacitor – конденсатор электрический (17).

CCS – источник тока управляемый (1).

ConstantVoltage – источник постоянного напряжения (20).

CurrentSensor – амперметр (11).

CVS – управляемый генератор напряжения (2).

Diode – диод (14).

Ground – заземление (точка нулевого потенциала) (18).

Gyrator – фазовращатель (4).

IdealTransformer – идеальный трансформатор (3).

Indutor – индуктор электрический (21).

NMOS – NMOS транзистор (10).

NPN – транзистор n-p-n (16).

OpAmp – норатор-нуллатор пары (8).

PMOS – PMOS транзистор (9).

PNP – транзистор p-n-p (5).

PotentialSensor – потенциалоскоп (22).

Resistor – резистор (12).

SineVoltage – источник синусоидального напряжения (6).

Switch – неидеальный электрический ключ (7).

VariableResistor – электрический переменный резистор (23).

VoltageSensor – вольтметр (13).

VsourceAC – электрический источник переменного напряжения (15).

VVsourceAC – регулируемый источник переменного напряжения (19).

Инструментарий термогидравлики – (ThermoHydraulics)

Thermal-Hydraulics Bache –термогидравлический резервуар (1).

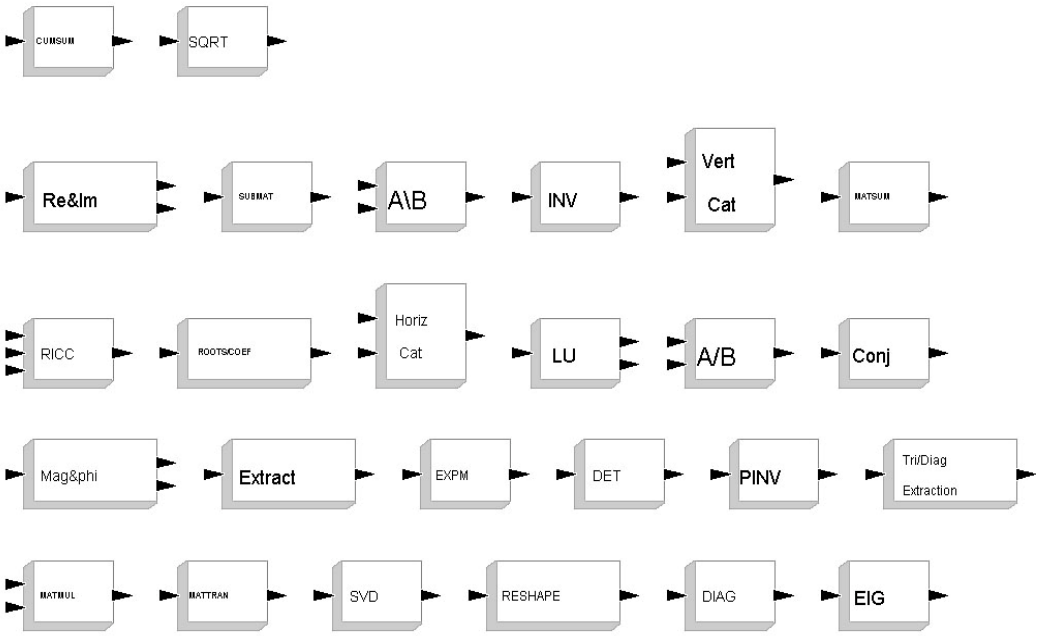
PerteDP – термогидравлическая труба (3).

PuitsP – термогидравлическая утечка (5).

SourceP – термогидравлический источник постоянного давления (4).

VanneReglante – термогидравлический управляющий клапан (2).

Матрицы – (Matrics)



CUMSUM – кумулятивная сумма (1).

EXTRACT – экстракция матрицы (16).

EXTTRI – треугольная или диагональная экстракция (20).

MATBKSL – левое матричное деление (5).

MATCATH – горизонтальная конкатенация (11).

MATCATV – вертикальная конкатенация (8).

MATDET – детерминант матрицы (18).

MATDIAG – создание диагональной матрицы (25).

MATDIV – матричное деление (13).

MATEIG – собственная матрица (26).

MATEXPM – экспоненциальная матрица (17).

MATINV – обращение матрицы (6).

MATLU – факторизация (12).

MATMAGPHI – блок производит два типа преобразования.

1. Блок преобразовывает комплексное число к модулю и углу в радианах, в этом случае вход комплексный, а выход два вещественных числа. Если вход два вещественных числа, то выход по углу ноль или π, а по величине модуля – модуль входного числа.

2. Обратное преобразование (15).

MATMUL - матричное умножение (21).

MATPINV - псевдообращение матрицы (19).

MATRESH – блок изменяет размерность матрицы или вектора на заданную в поле "output size desired". Выходная размерность может быть меньше или равна входной (24).

MATSING SVD (23).

MATSUM – сумма матриц (8).

MATTRAN – транспонирование матрицы (22).

MATZCONJ – объединение матриц (14).

MATZREIM – комплексная декомпозиция (3).

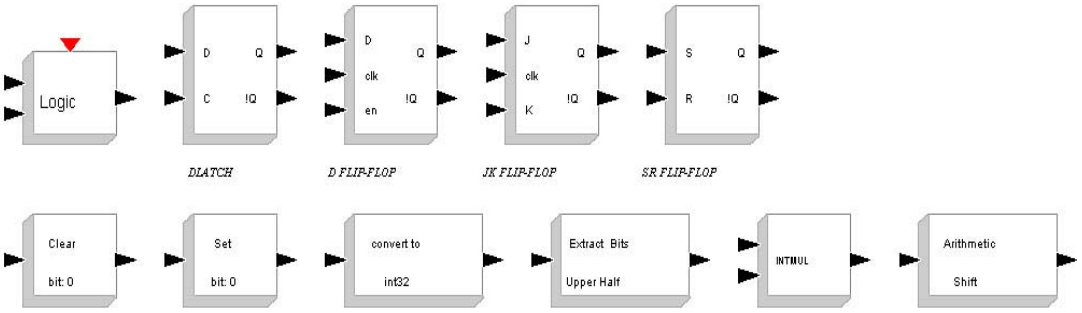
RICC – решение уравнения Риккати (9).

ROOTCOEF - вычисление коэффициента (10).

SQRT - квадратный корень (2).

SQRT SUBMAT - субматричное извлечение SUBMAT (4).

Цифровые блоки (Integer)



BITCLEAR - очистка битов (6).

BITSET – установка битов (7).

CONVERT – преобразование типа данных (8).

DFLIPFLOP – D-триггер (3).

DLATCH – D блокированный триггер (2).

EXTRACTBITS – EXTRACTBITS (9).

INTMUL – матричное умножение целого INTMUL (10).

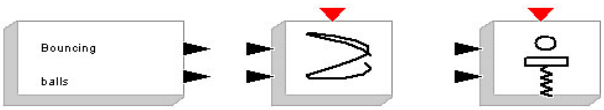
JKFLIPFLOP – JK триггер (4).

LOGIC – блок комбинационной логики (1).

SHIFT – сдвиговый регистр (11).

SRFLIPFLOP – SR триггер (5).

Примеры (DemoBlocks)

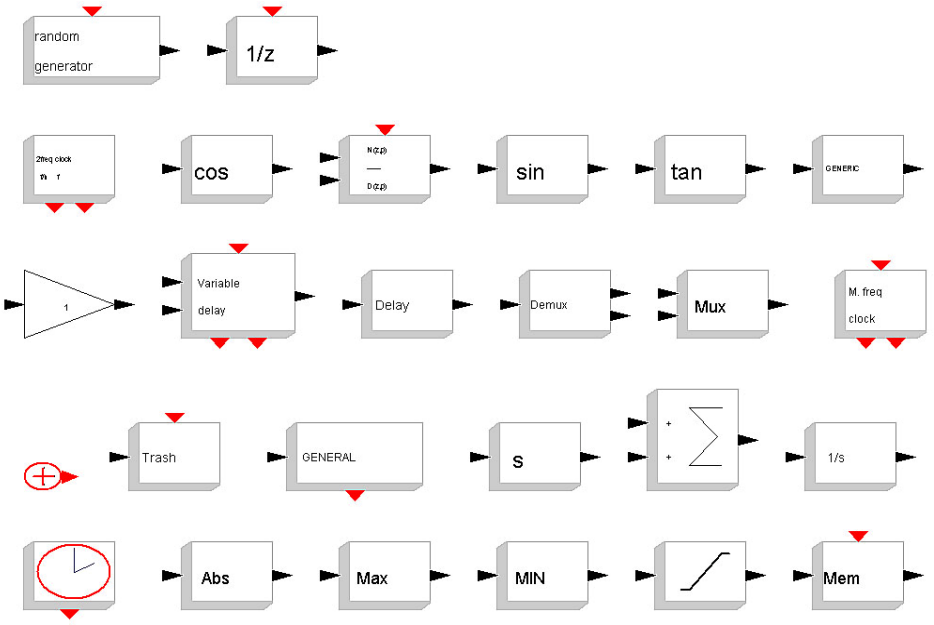


BOUNCE – генератор координат шаров (1).

BOUNCEXY – подпрыгивающие шары (2).

BPLATFORM – шар на платформе (3).

Старые блоки



Первая простая модель

Скопируйте с палитр: sinusoid generator, Cloc_c (палитра Sources), Csscope (палитра Scope), и Gain (палитра Linear).

Для того, чтобы скопировать блок, его можно перетащить мышью или, выделив блок (регион блоков) и щелкнув правой клавишей мыши (далее по тексту ЩПК), выбрать сору, а потом щелкнув ЩПК в окне, вставить – paste.

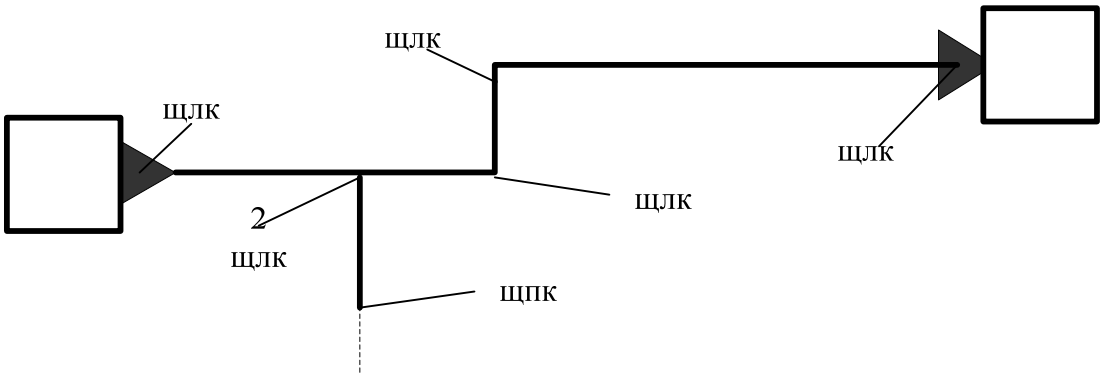
Генераторы Event (они выделены красным цветом) задают временную последовательность событий. Эти события, доступные как сигнал активизации с выхода блоков Event, могут быть использованы, чтобы активизировать другие блоки (задать моменты, когда блоки производят какие-то действия, например, выдают выходной сигнал) через свой входной порт активизации. Входные и выходные порты активизации находятся соответственно на верхней и нижней сторонах блоков. Основные входные и выходные порты установлены на боковых сторонах.

Позиции блоков могут быть выбраны так, что входные и выходные порты (подсоединенные), будут выровнены в линию, используя опцию Align (выстраивать в линию) в меню Edit.

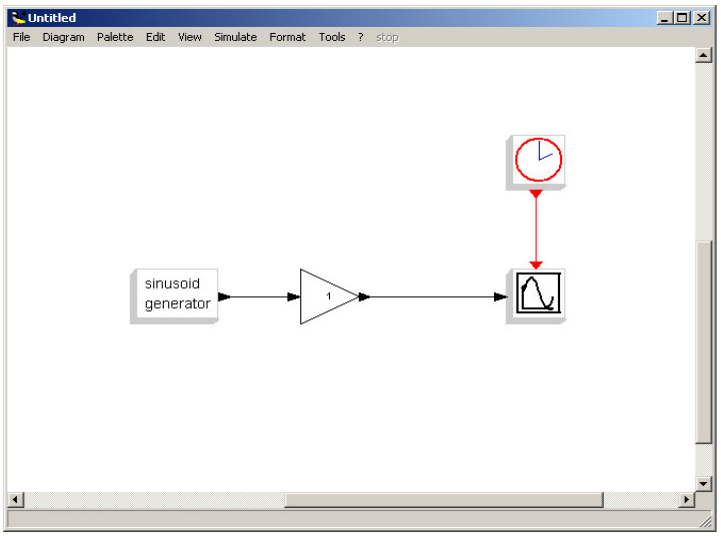
Соединение блоков

Для соединения блоков делается один щелчок левой клавишей мыши (ЩЛК) на порте одного блока и затем один на порте второго блока. Если в процессе протягивания связи сделать еще один ЩЛК, то линию можно повернуть под прямым углом. Для удаления связи ее выделяют одним ЩЛК и Delete.

Для создания ответвления от линии делают на нужном месте два ЩЛК и тянут линию в нужное место. Или сделав ЩПК, выбирают в появившемся меню Linke. На уже выделенной связи можно сделать ЩЛК один раз. Для удаления еще не завершенной линии связи сделайте один ЩПК.

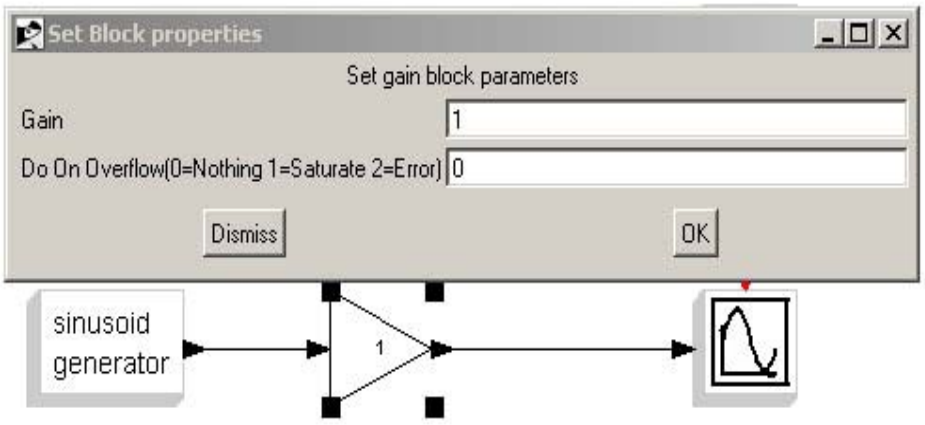


Следует помнить, что линии сигналов активизации, по умолчанию, красные, а основных сигналов черные. Эти цвета можно изменить, используя Default Link Color в меню Misc (прочее) или Default Link Color в меню Format.



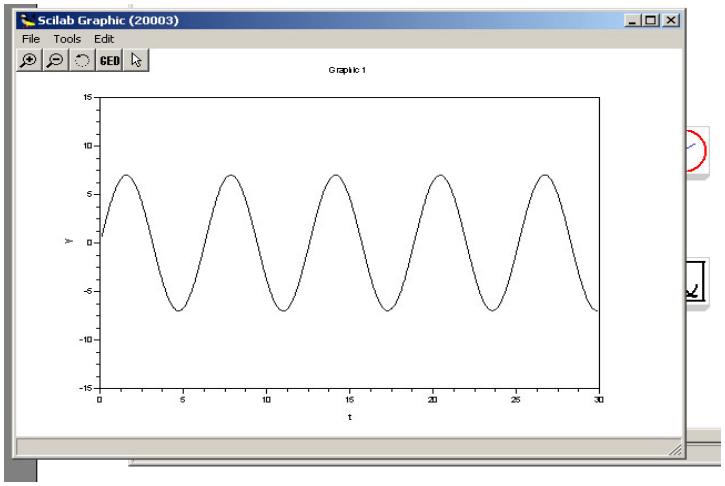
Как только все входы и выходы будут связаны, построение модели Scicos завершено и она может быть запущена. Перед этим параметры блоков, если это необходимо, должны быть изменены. Чтобы изменить параметры блока – два ЩЛК мыши на объекте. Это открывает диалоговое меню, которое позволяет модифицировать блочные параметры. Эти параметры могут быть заданы, используя правильные выражения Scilab. Эти выражения запоминаются символически, и затем оцениваются.

И так два ЩЛК мыши на иконке Gain. Появляется диалоговое окно.



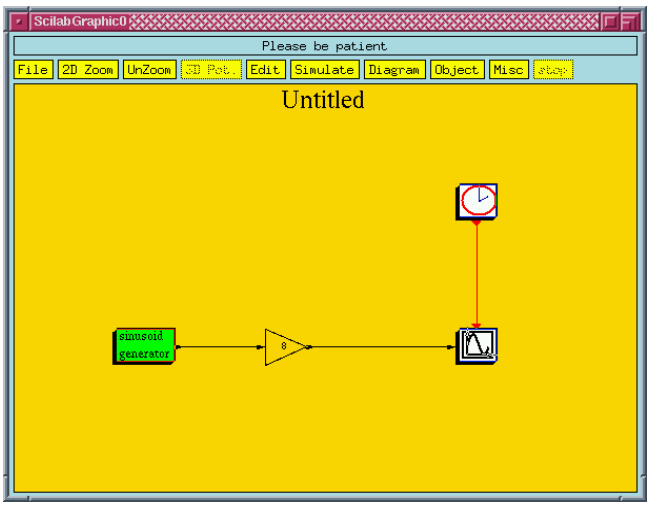
Установим параметр Gain равным 7 и нажмем кнопку Ok. На иконке блока Gain появилась цифра 7.

Запустим модель. Для этого в меню Simulation выберем опцию Setap и в появившемся окне установим конечное время работы модели (параметр Final integration) time равным 30. Далее в меню Simulation выбираем опцию Ran. Появляется графическое окно. Оно представлено на следующем рисунке.



Моделирование может быть остановлено с помощью кнопки Stop в правой части панели меню в основном окне Scicos. Далее работа модели может быть продолжена, перезапущена или прекращена: при нажатии кнопки Ran выпадает панелька с кнопками Continue, Restart, End.

Если желательно, то можно проделать косметические изменения (цвета линий, цвет фона, 3D аспект, и т.п.). Тогда окно Scicos может выглядеть приблизительно так.



При необходимости можно сохранить результат моделирования (графическое окно) выбрав File/Save в окне диаграммы с расширением .scg. Для того, чтобы загрузить сохраненный графический результат используйте Load в меню File диаграммы. График будет загружен поверх имеющегося.

Модели Scicos могут быть сохранены в файлах, с расширением .cos или .cosf. Например, если имя модели Untitled, создается двоичный файл Untitled.cos. Также возможно сохранение модели Scicos в формате ASCII с расширением .cosf. Преимущество сохранения в формате ASCII в том, что оно машинонезависимое.

Для запуска модели извне, просто запустите команду Scicos с именем файла, который нужно загрузить, в качестве аргумента. Пример:

--> scicos('Untitled.cos');

Установка параметров моделирования

Модель Scicos может работать от t = 0 до указанного времени окончания работы. В зависимости от модели используется решатель для обычного дифференциального уравнения (ОДУ) или решатель для дифференциального алгебраического уравнения (ДАУ), а если модель является дискретной, ни один из них не используется. У каждой задачи есть свои собственные специфические особенности. Это требует некоторой гибкости при моделировании и анализе моделей. Рассмотрим параметры моделирования, которые пользователь должен задать прежде, чем начать моделирование.

1. Время окончания работы **(Final Simulation Time)**: это время по умолчанию составляет 10000 секунд, время начала работы всегда равно нулю.

2. Вычисление в режиме реального времени **(Realtime Scaling)**: время моделирования и фактическое время не одинаково. Например, моделирование десяти секунд обычно не занимает десять секунд реального счета. Реальное время зависит от многих факторов, включая сложность модели, размеры шага выбранного решателя и быстродействие компьютера. В основном время моде-

лирования меньше, чем реальное время. Эта опция увеличивает время реального моделирование, устанавливая единицу времени Scicos в одну секунду.

• Отклонение **(Tolerances)**: относительное допустимое отклонение задает ошибку относительной величины каждого состояния, то есть, процент от величины. Например, 10^{-6} означает, что точность вычисленного состояния в пределах 10^{-4} %. Абсолютное отклонение, представляет приемлемую ошибку. Его величина по умолчанию 10^{-4}.

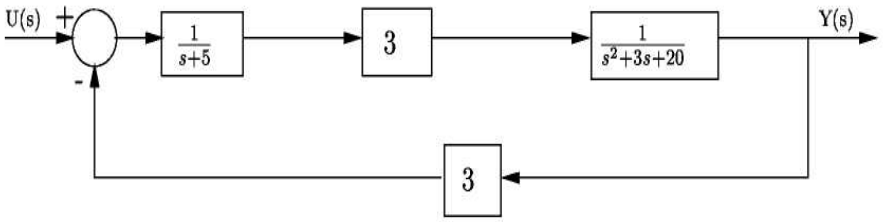
• Максимальный шаг (**Maximum Step-size)**: максимальный шаг задает наибольший шаг интегрирования, который может выбрать решатель. Величина по умолчанию 100001 (inf). Эта установка важна тем, что ограничивает шаг интеграции, препятствуя тому, чтобы решатель не выбрал слишком крупный шаг.

• Максимальный временной интервал интегрирования **(Maximum Integration Time Interval)**: максимальный временной интервал для каждого вызова решателя. Он должно быть уменьшен, если поступает сообщение «too many calls» (слишком много запросов).

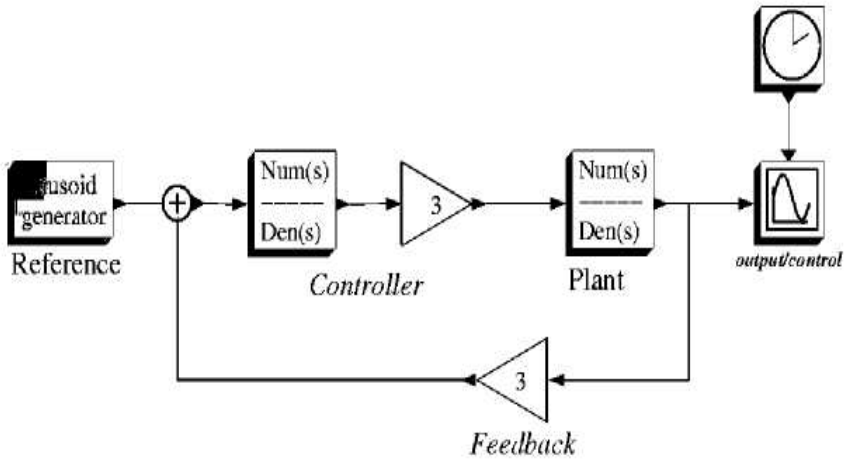
• Допустимое время (**Tolerance on Time**): наименьший временной интервал, для которого используется численный решатель, чтобы обновить непрерывные величины.

Блок-схема в Scicos

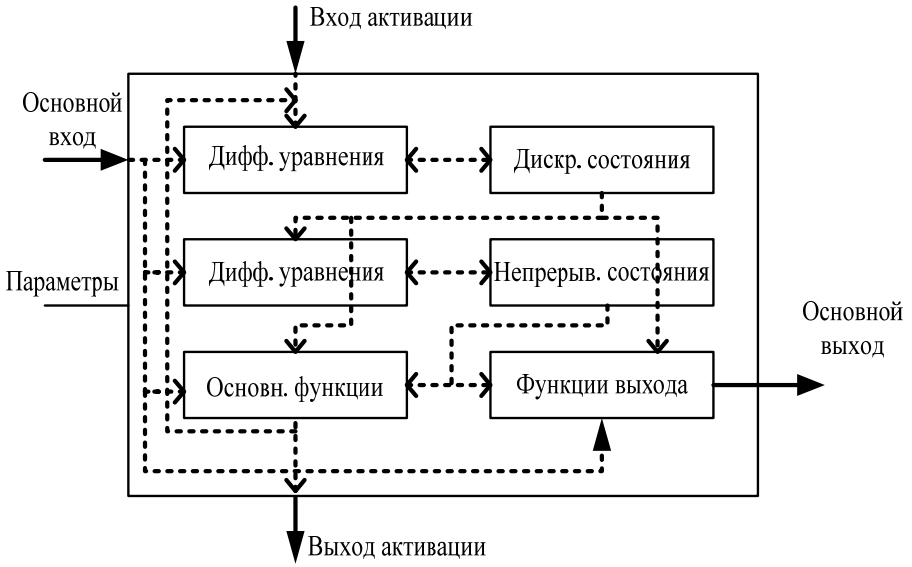
Самое фундаментальное понятие для разработки систем управления - блок-схема (модель). Классическая модель блок-схемы динамической системы графически состоит из блоков и связей, которые представляют непрерывные сигналы и дискретные события.



Отношения между каждой элементарной динамической системой в блок-схеме иллюстрированы при помощи сигналов, соединяющих блоки. Все вместе блоки и линии в блок-схеме описывают полную динамическую систему.



Самый простой элемент модели – блок. На блок поступает входной сигнал и он, с учетом своих динамических свойств, формирует выходной сигнал. Детали внутреннего устройства каждого блока скрыты от внешней части модели, т.е. он похож на черный ящик. Другими словами модель «интересуется» только отношениями в блоке между входом и выходом. В общем, каждый блок имеет структуру, показанную на следующем рисунке.



Каждый блок связан с другими блоками линиями передачи сигналов, как дискретных или непрерывных. Непрерывный сигнал представляет собой величину, которая изменяется в течение некоторого времени и определена для всех моментов времени в течение моделирования, в то время как дискретный сигнал является кусочно-непрерывным и только событие (Event, сигнал активации) изменяет его величину.

Отношения между входом и выходом блока определены рядом непрерывных и дискретных уравнений. Эти уравнения определяют отношения между сигналами, формируют сигналы и переменные внутренних состояний.

Дискретные события играют важную роль в моделях гибридных систем. Чтобы явно определить дискретные события, в Scicos используют специальные связи. Их называют связями активации, в отличие от основных связей. Источник активации может активизировать несколько непрерывных или дискретных блоков. Модель Scicos может содержать несколько источников событий. В связи с событиями в моделях Scicos нужно учитывать два важных момента: существуют одновременные события и синхронные события.

Два события одновременны, только если они активизированы в одно и то же время.

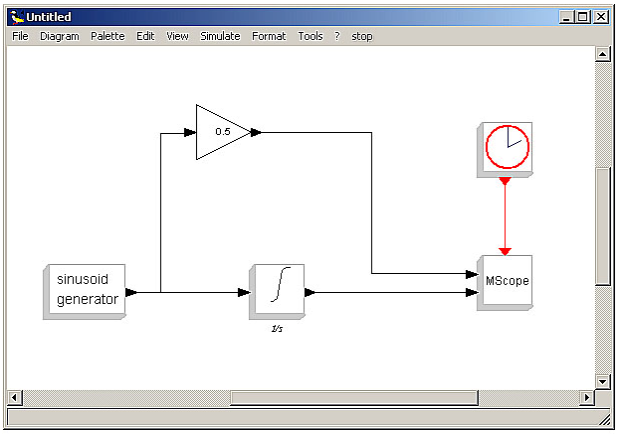
Два события синхронны, только если они активизированы одним и тем же блоком.

Два одновременных события не обязательно синхронны, и два синхронных события не обязательно одновременны.

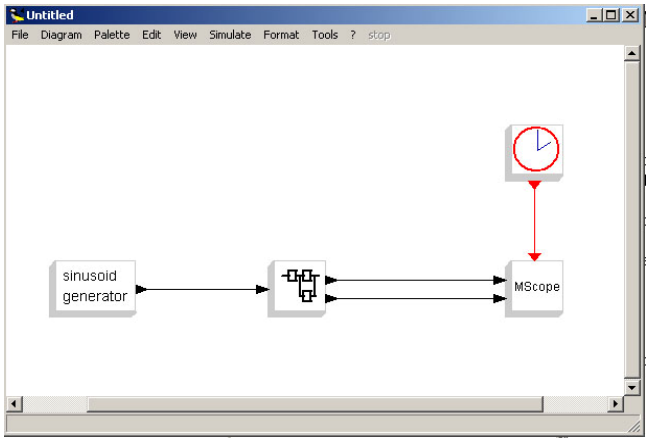
Иерархия (суперблоки)

Модель, состоящая из очень большого количества блоков не удобна, ее трудно запомнить, работа с ней и ее отладка затруднены. Для больших систем полезно использовать средства создания иерархической модели.

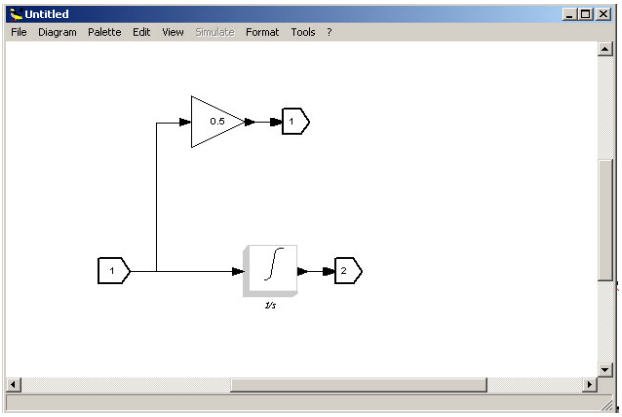
Создадим модель, копируя блоки из палитр, в соответствии с нижеприведенным рисунком.



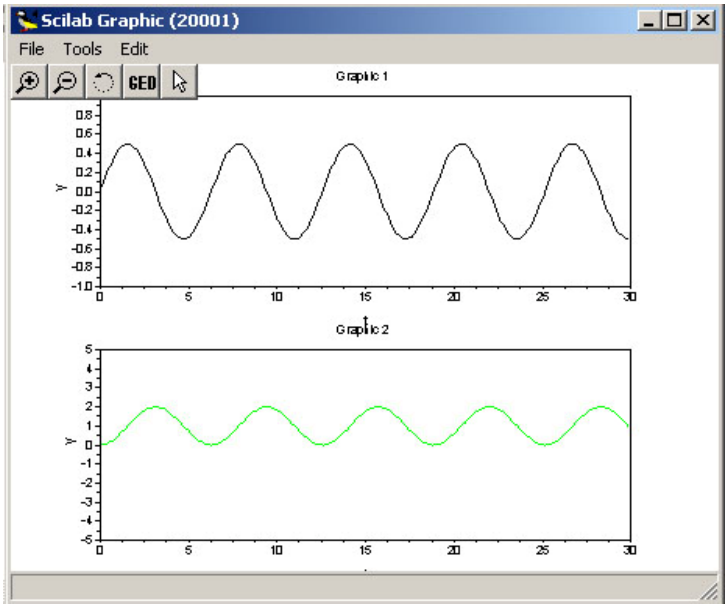
Разместим интегратор и усилитель в суперблоке. Для этого можно выбрать Region to super blok в меню Diagram и затем выбрать область, которую нужно поместить в суперблок (удерживая левую клавишу мыши, образовать охватывающий нужные блоки прямоугольник). Суперблок создан. Он заменяет два блока.



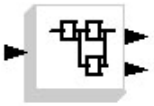
Проверим состав суперблока. Для этого сделаем двойной ЩЛК на суперблоке. Он открывается в новом окне.

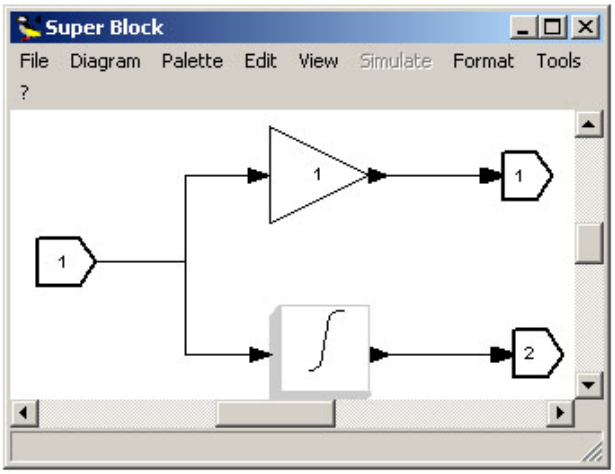


Выйдем из суперблока (меню Diagram/To main diagram) и запустим модель Scicos. Получаем такой результат.



Возможен другой путь создания суперблока. Откроем пустое окно Scicos и копируйте в него пустой суперблок из палитры Others. Откройте суперблок щелкая на нем и заполните его необходимыми блоками. Ниже на рисунке показан суперблок, созданный вторым способом и его содержимое.





Созданный суперблок имеет два выходных сигнальных порта и один входной порт. Теперь суперблок может быть использован подобно любому другому блоку. Подключите его порты к другим блокам и Ran!

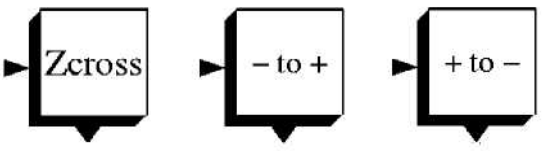
Переход через нулевой уровень

Пересекающие нулевой уровень события (и соответствующие блоки палитры Treshold) введены, чтобы преодолеть трудность в моделировании непрерывных систем, когда есть некоторые «прерывания», т.е. резкие изменения (скачки) сигналов. Эти явления могут вызвать ошибки при использовании используемых методов интеграции. Численный решатель Scicos, благодаря механизму обнаружения моментов перехода через ноль, может обнаружить такие события. Для обычных случаев в Scicos используется блок пересечения нуля. Цель этого блока состоит в том, чтобы при обнаружении «разрывов» (скачков) сигналов производить переключения уравнений системы.

Применение блоков пересечения нуля чаще всего необходимо в гибридных моделях для непредсказуемых событий. Например, при моделировании системы контроля уровня жидкости в резервуаре в случае прекращения притока, как только уровень превысит определенную величину.

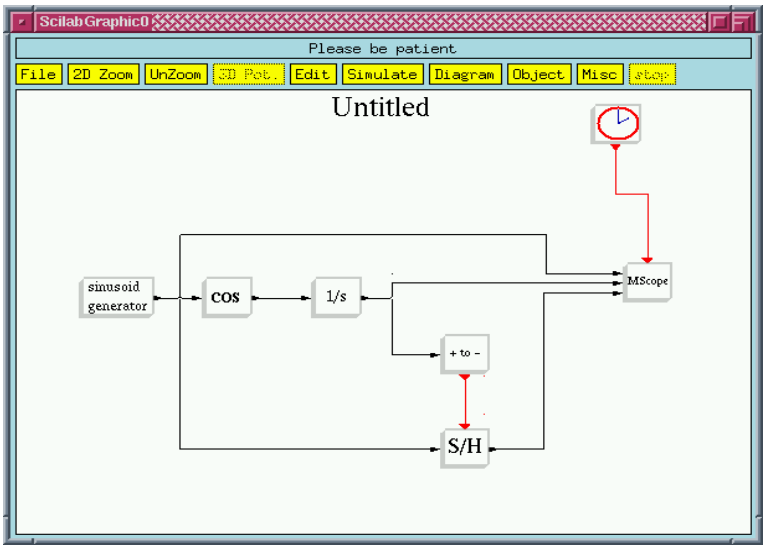
В некоторых приложениях, в дополнение к установлению факта пересечения, мы должны знать направление пересечения.

В этих целях в Scicos имеются два других блока: блок «+ to –» и блок «– to +», которые фиксируют не только факт пересечения, но и учитывают направление этого пересечения.



Покажем простой сеанс Scicos, чтобы проиллюстрировать использование блоков нулевого пересечения.

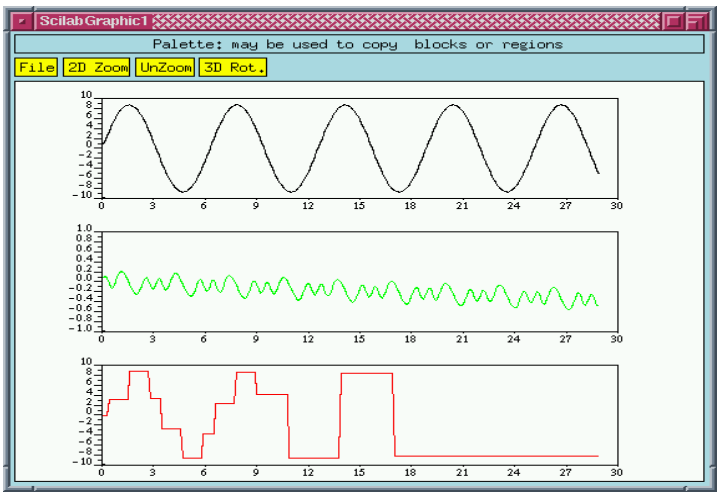
Мы уже столкнулись с часами активации Event Clock, которые генерировали последовательность равномерно расположенных во времени событий. Они были использованы, чтобы активизировать Scope. Откроем пустое окно Scicos. Создадим следующую модель, копируя блоки из палитр.



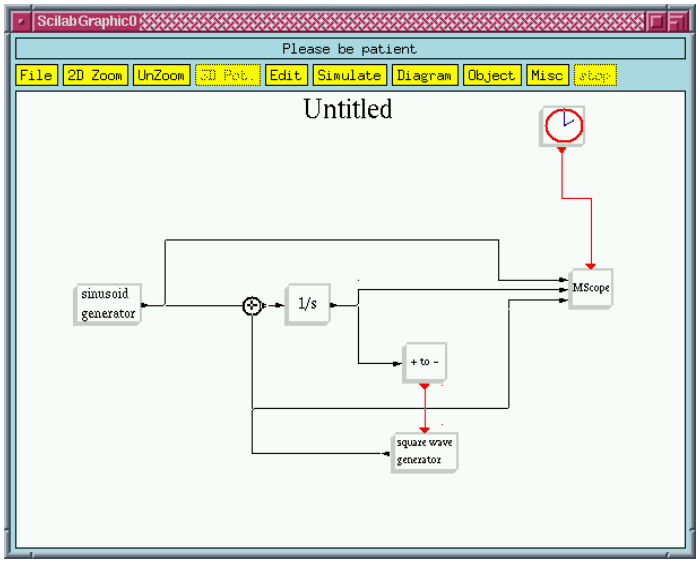
Блок «**+to-**» находим в палитре Treshold, S/H-блок (sample and hold, образец и сохранение) – в палитре линейных блоков и cos-блок – в нелинейной палитре. Отметим, что блок MScope имеет 3 входа. Количество входов MScope является параметром блока; его нужно установить прежде, чем входы будут подсоединены.

Блок «**+to-**» генерирует событие каждый раз, когда входной сигнал пересекает нулевой уровень, изменяясь от плюса к минусу. Входной сигнал должен быть непрерывной функцией времени. Эти события активизируют блок S/H, который копирует входной сигнал (образец) на свой выход. Сигнал на выходе не изменяется до следующей активации блока.

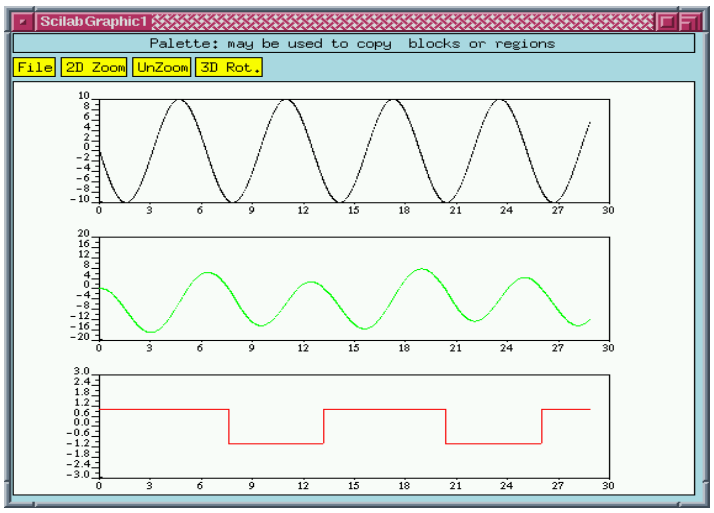
Результат счета показан на рисунке.



Параметр амплитуды блока генератора sinusoid установлен 8.7. События, с выхода блока пересечения нулевого уровня, были использованы для генерации дискретных сигналов. Эти дискретные сигналы могут быть использованы в свою очередь для управления непрерывными компонентами. Это часто делается в управляющей системе, где дискретные контроллеры управляют непрерывными системами. Простой пример:

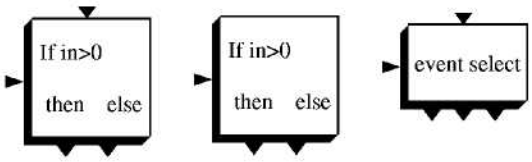


Дискретный сигнал возвращается в непрерывную часть схемы. Отметим, что у генератора прямоугольных волн выходной порт слева (в линейной палитре, выход у этого блока справа). Это сделано посредством использования команды Flip в меню Editor. Этот блок выдает на выход 0 или 1, переключение производится сигналом активации. Результат моделирования показан ниже.



Блоки с условием

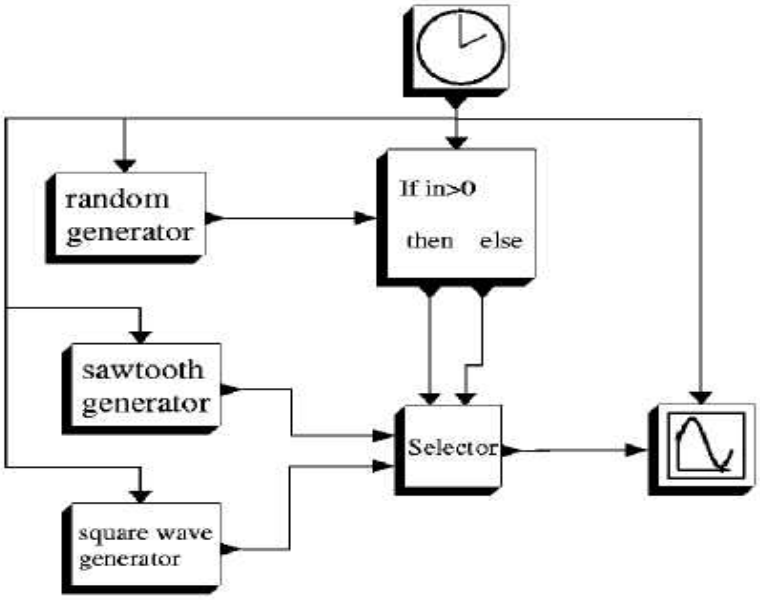
Основные функциональные блоки выполняют заданные вычислительные функции, тогда как блоки с условием управляют порядком вычислений в модели. Эти блоки могут активизировать другие блоки в модели Scicos при выполнении некоторых условий.



Всякий раз, когда эти блоки получают сигнал активации, они в этот же момент формируют синхронный сигнал активации на выходном порту как функцию величины сигнала на их основном входе.

Сигналы на выходах этих блоков взаимоисключающи.

Чтобы показать применение блока "Then-If-Else" рассмотрим следующую схему.



У селектора есть два входных порта активации. Блок может быть активизирован в данный момент только одним из них. Всякий раз, когда селектор активизирован по первому или второму входу входной сигнал принимается на вход также с первого или второго входа основного сигнала соответственно. Таким образом, на выходе селектора присутствует пилообразный сигнал, если сигнал random generator положительный и прямоугольный волновой сигнал, если сигнал random generator отрицательный.

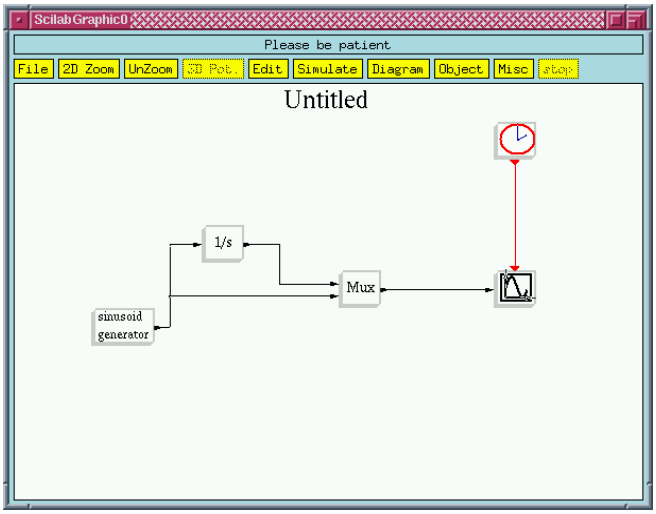
Если блок, "Then-If-Else" не имеет входного порта активации, он управляет блоками непрерывного сигнала, которые получают активацию по другим цепям. В этом случае, численный решатель видит только те блоки, которые находятся в активной ветви блока "If-then-Else".

Блоки «Event select» также считаются синхронными блоками. У него есть встроенный индикатор пересечения нуля. Блок имеет один вход и в любой момент активизирован только один выход.

Векторный вход-выход

Мы уже видели, что блоки могут иметь более чем один вход. Они могут также иметь более чем один выход и, кроме того, каждый вход и выход может быть распараллелен (векторизован). Таким образом, каждый порт ассоциируется с размерностью. При соединении выходного порта с входным, размерности портов должны быть равными.

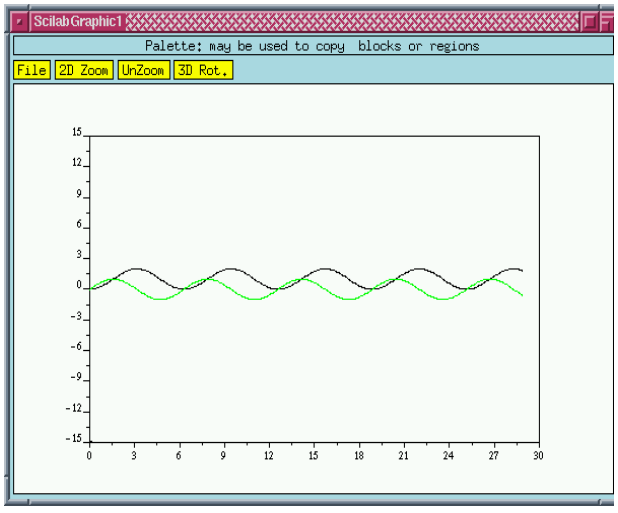
Создадим следующую схему, копируя блоки из палитр.



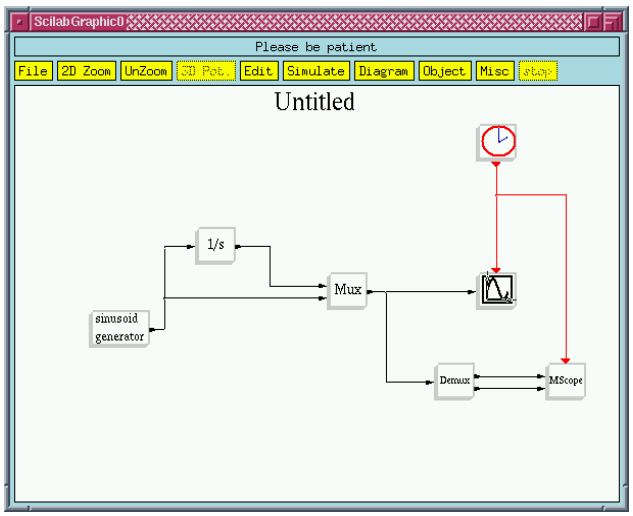
Mux (мультиплексор) находится в палитре Branching (расширение). Этот блок может иметь любое количество входов, в том числе и векторных; выход является вектором, полученным как конкатенация входов.

Единственный параметр этого блока – количество входов; нет необходимости определять размерность каждого входа. Scicos определяет эту размерность по блоку, с которым связан входной порт.

В вышерассмотренном примере, входы размерностью единица и, следовательно, выход является двумерным вектором. Аналогично для Scope (здесь - нет необходимости определять размер входного вектора, это делается автоматически.



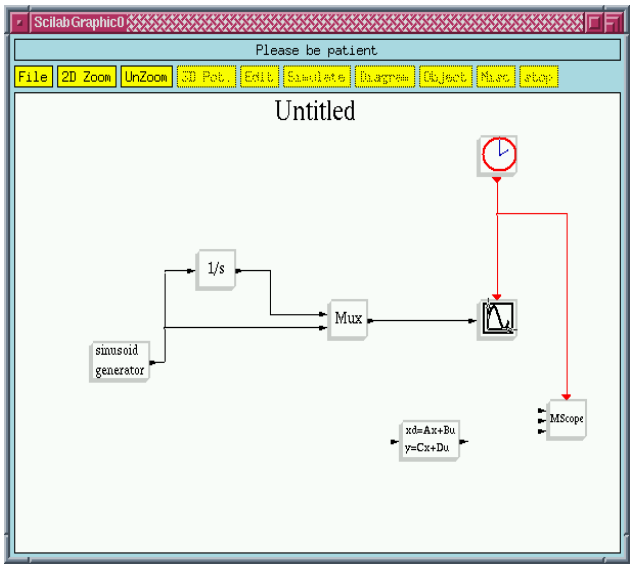
Изменим схему следующим образом.



Блок Demux производит действие, противоположное блоку Mux, то есть разделяет входной вектор на векторы меньшего размера. Demux отменяет то, что сделал Mux.

Чтобы избежать наложения сигналов, убедитесь, что были устанавлены параметры MScope – два входа. При этом он будет использовать два графических окна.

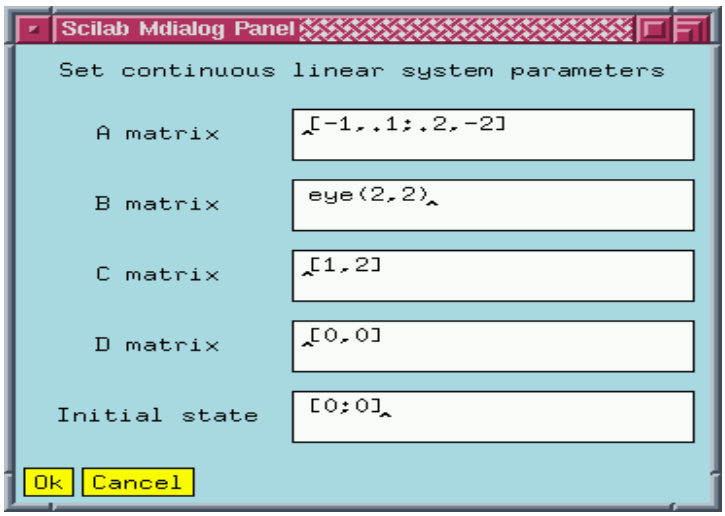
Изменим схему.



Новый блок является линейной динамической системой.

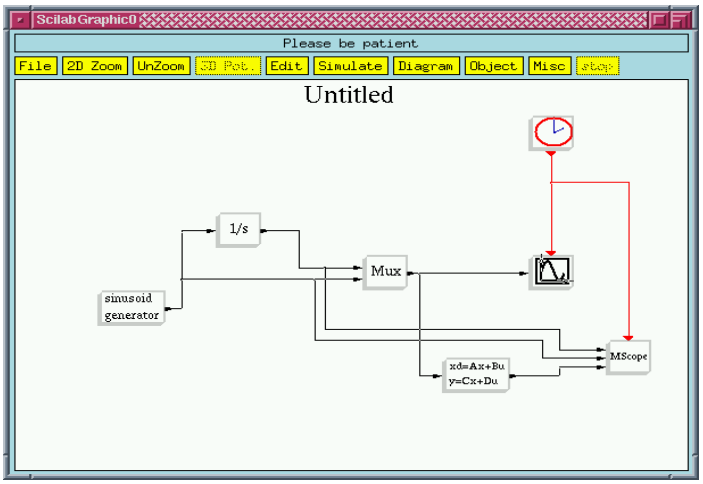
dx/dt = Ax + Bu; u входной вектор, x текущее состояние
y = Cx + Du; y выход

Изменим параметры нового блока следующим образом:



Заметим, что блочные параметры: A, B, C и D матрицы. Они определяют размерности входных и выходных векторов.

Параметры задаются, используя синтаксис Scilab. Завершим схему следующим образом:

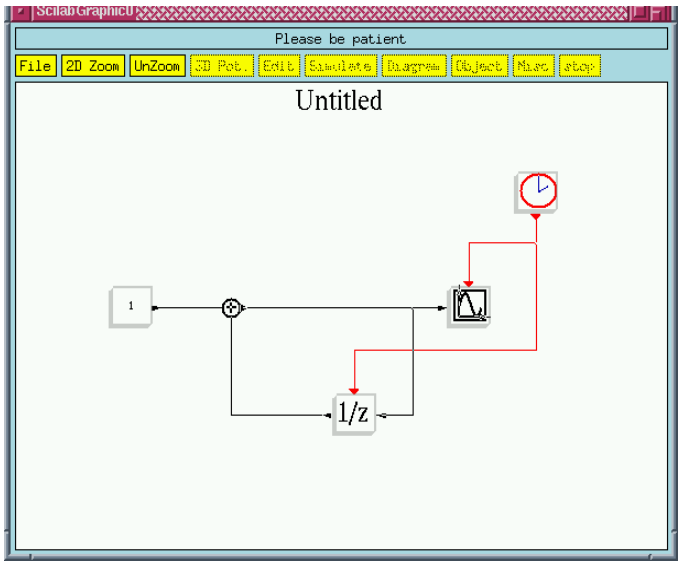


Размерности входных выходных векторов совместимы.

Дискретные системы и наследование

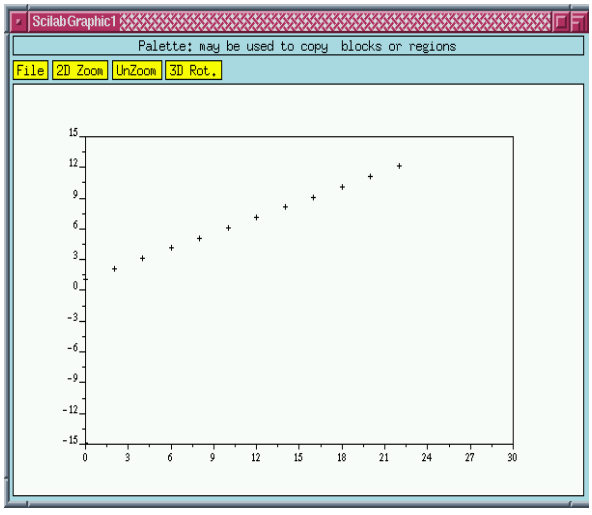
Мы уже столкнулись с блоками дискретного времени. Блок является дискретным, если он активизирован событиями и таким образом генерирует кусочно-постоянный сигнал. Активизирующие события не обязательно должны быть периодичными. Схема может быть сформирована и исключительно на дискретных блоках.

Рассмотрим пример.



Здесь 1/z (дискретная задержка) – регистр сдвига. Каждый раз, получая сигнал активации, он отображает свое внутреннее состояние на выходе и изменяет свое внутреннее состояние в соответствии с входным сигналом. Вышеприведенная схема является счетчиком. По каждому сигналу часов состояние увеличивается на единицу. В этом случае естественно, что Scope отображает сигнал в виде точек, а не линии. Это может быть сделано посредством изменения параметров Scope.

Результат счета для периода часов активации две секунды показан на рисунке.



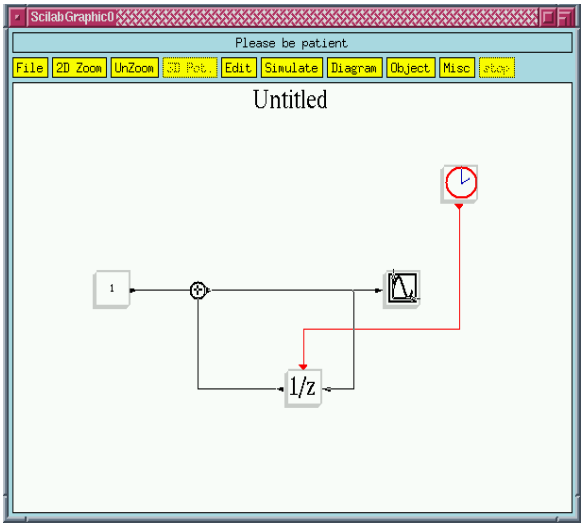
Отметим, что блоки этой схемы активизируются только во время события за исключением блока константы, который активизирован постоянно. Блок ⊕ не активируется сигналом активизации, но наследует время активизации от входных сигналов.

В общем случае, блок без входных портов активации – или активен всегда или наследует время активации от входных сигналов. В этом последнем

случае, время активизации является объединением времени активизации входных сигналов.

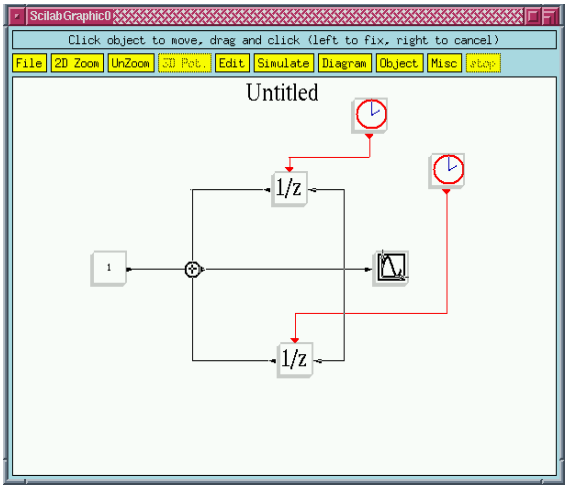
Блоки с изменяющимся во времени выходным сигналом (например, generator sinusoid), даже если у них есть входы, не наследуют время активации; они просто активны всегда.

Создадим новую схему.



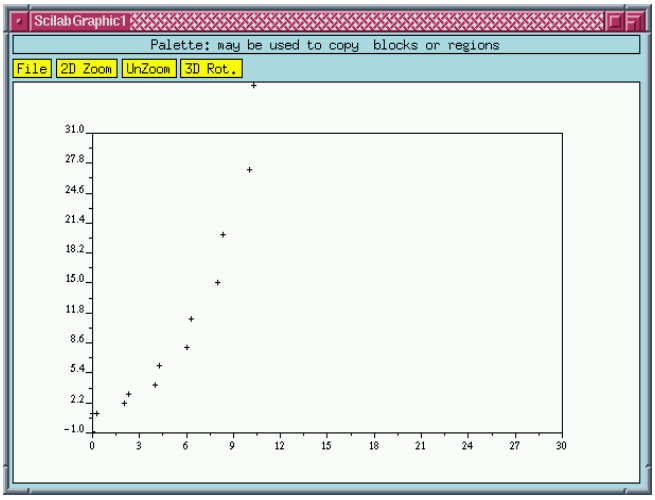
Блок Scope может наследовать время активации. *Входной порт активизации этого блока поэтому может быть удален установкой соответствующего блочного параметра*. Результат счета в этом случае идентичен предшествующему.

Создадим новую схему. Для того, чтобы увидеть механизм наследования в действии, в схему введены два независимых источника событий. Период обоих часов равен 2 с, но начальное время вторых часов установлено 0.3 с (первых – 0 с). Таким образом, в схеме имеются два независимых генератора событий.

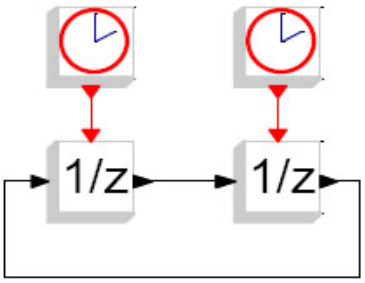


В этом случае, нет риска возникновения неопределенности, поскольку два независимых события никогда не происходят в одно и то же время. Это обусловлено таким выбором параметров часов. При другом выборе может возникнуть неопределенность. Поэтому этот тип схем применять не следует. Аналогичный результат может быть получен иначе, как это показано ниже.

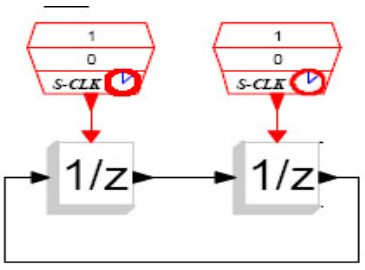
Запустим модель. Отметим, что время активации Scope является совмещением времени активации двух регистров. Scope активизируется в моменты 2n и 2n+0.3 секунд, n = 0,1,2...



Рассмотрим еще несколько простых схем. Как было сказано выше, если каждый блок активизируется собственным генератором событий, например, так как это показано на следующей схеме, то блоки не синхронны, даже если оба генератора событий настроены одинаково.

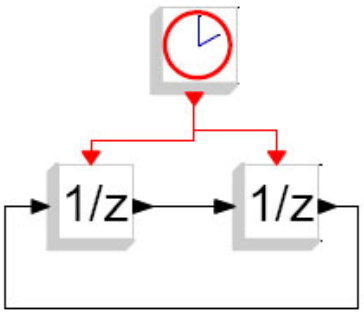


Результат моделирования для этой блок-схемы в Scicos не предсказуем. Эта же схема при использовании SampleCLK выглядит так:

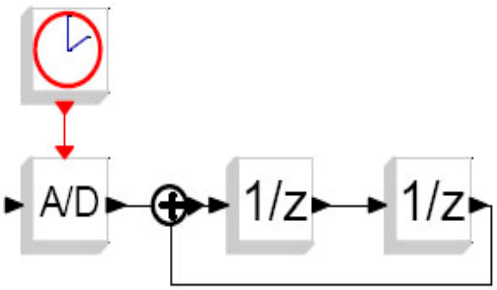


В этом случае, компилятор Scicos начинает выполнять необходимые преобразования на блоках SampleCLK так, чтобы найти самые медленные часы, которые с учетом поддискретизации могут заменить активизацию этих блоков.

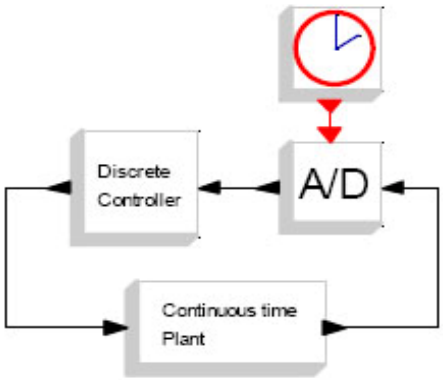
В схеме, показанной на рисунке, вычисления тривиальны, поскольку оба блока имеют идентичные периоды. Эквивалентная схема будет такая:



В Scicos блок 1/z, подобно любому другому блоку, может работать без входного порта активизации, наследуя ее со своего основного входа. Например, так:



Следующая блок-схема является обобщением этого принципа:



Непрерывная часть не наследует никакой активации, поскольку она и так всегда активна. Кроме того, следует отметить, что D/A блок здесь не нужен так как выход дискретной части и без этого хранит выходную величину до следующей активации. Здесь такой блок использован, чтобы ясно указать точку активации.

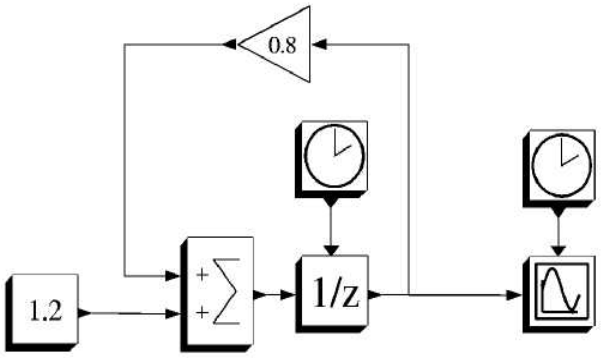
Моделирование гибридных динамических систем в Scicos

Моделирование дискретных систем

В библиотеках Scicos есть несколько блоков для того, чтобы моделировать дискретные системы, например, такие как задержка (1/z). Эти системы должны быть активизированы событиями. Предположим, что мы хотим моделировать дискретную систему с периодом T = 0.5 секунд:

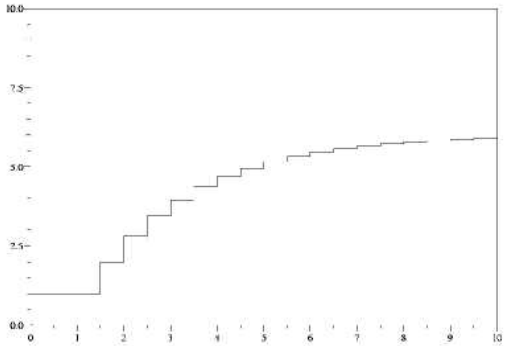
$$y(k + l) = 0.8\, y(k) + 1.2$$

Она будет иметь вид:



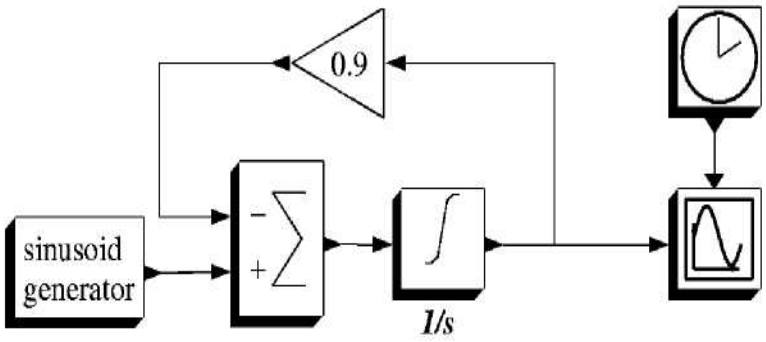
В этой модели дискретное состояние y (k) представлено блоком 1/z, а блок часов активации используется, для организации создания дискретных событий. Параметры блока заданы так, чтобы первое событие произошло при t = 1, а далее формировалась последовательность событий с периодом Ts = 0.5.

Результат моделирования для начального условия y (0) = 1 и времени окончания моделирования Tf = 10 с показан ниже.

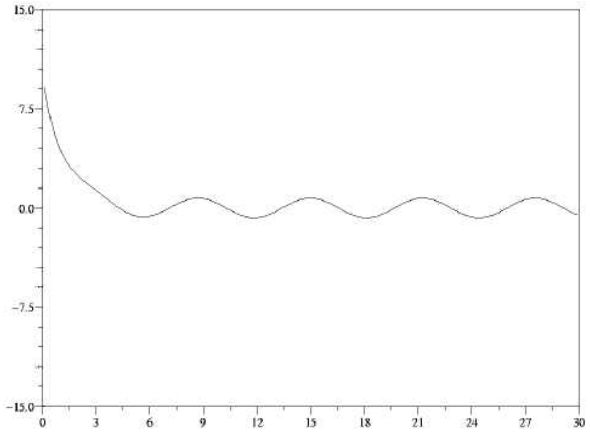


Моделирование непрерывных систем

Модель Scicos – графическое представление математической модели динамической системы. Математическая модель динамической системы задается дифференциальным уравнением. Например, таким: $\dot{x} = \sin(t) — 0.9\, x$. Тогда модель будет иметь следующий вид.

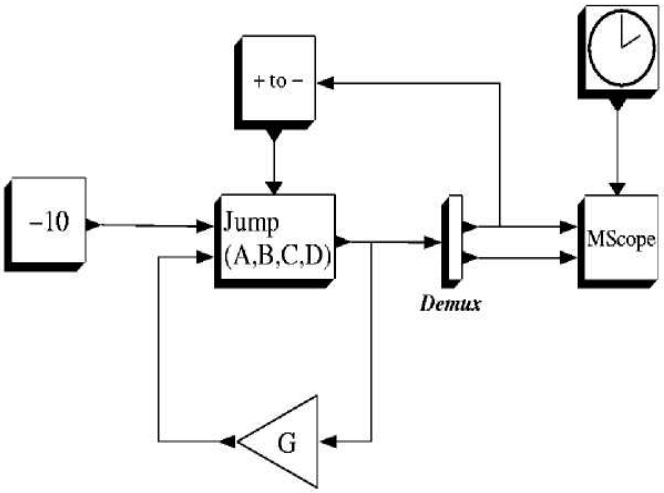


В этой модели использовались часы для периодической активации блока Scope. Таким образом, все связи передают непрерывные сигналы, за исключением связи между часами и Scope. Результат моделирования для начального условия x (0) = 10 имеет следующий вид.

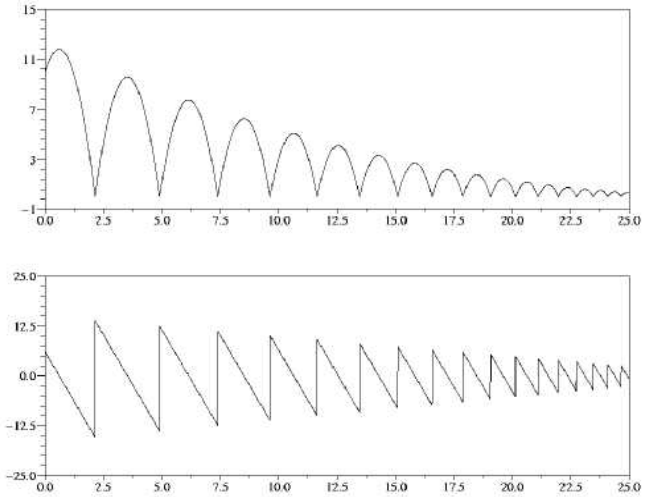


Моделирование гибридных систем

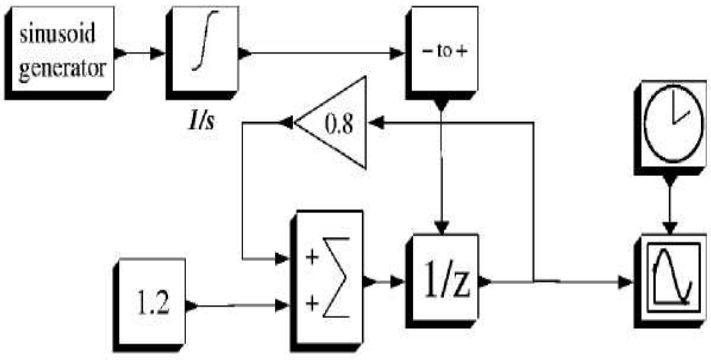
Непрерывные и дискретные сигналы, зависящие от событий могут взаимодействовать по-разному. Во-первых, они могут быть входами одного блока. По существу, нет никакого различия между дискретным и непрерывным сигналом. Фактически сигнал Scicos может быть дискретным в течение какого-то времени, а потом непрерывным. Это означает, что в Scicos можно выполнять совместные операции с непрерывными и дискретными сигналами. Непрерывные сигналы могут создавать события через блоки пересечения нуля. Наконец, события могут создать скачки в непрерывных состояниях так же как в дискретных. Рассмотрим модель прыгающего мяча.



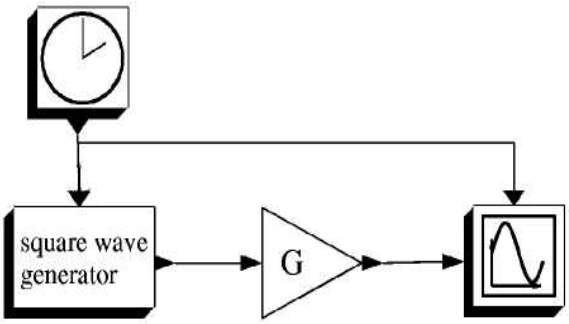
В этой модели использована линейная система с блоком задания скачка, блок усиления в цепи обратной связи по скорости, блок ускорения (-10) и пороговый блок, чтобы создавать событие и изменять скорость. Этот последний блок контролирует высоту мяча и всякий раз, когда он пересекает ноль, генерирует случай, который подается для управления скачком. Результат моделирования показан на рисунке.



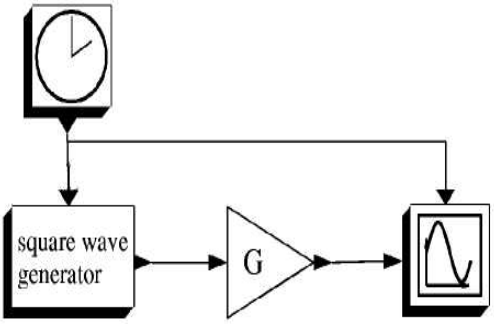
Другая модель активизирует дискретную систему пороговым блоком «–to +» так, как показано на рисунке.



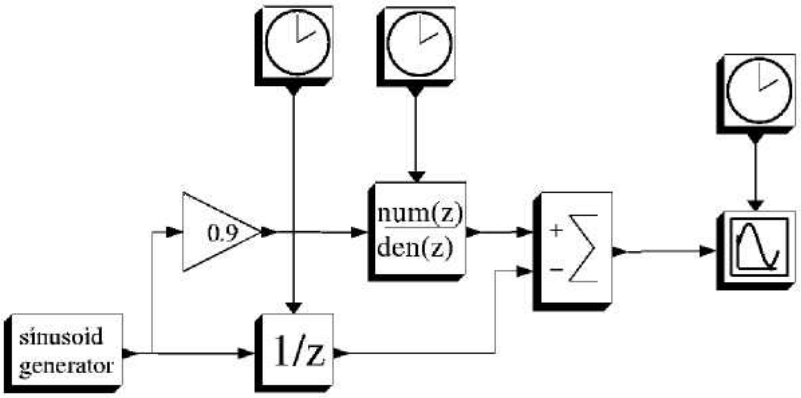
Рассмотрим следующую модель.



Рассмотрите диаграмму Scicos в рис.

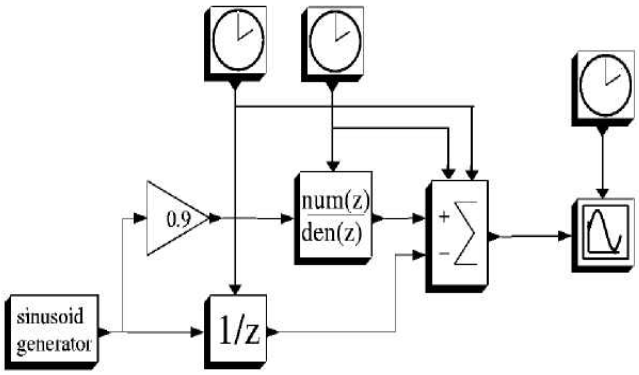


Здесь есть блок, у которого нет входных портов активации, он активизирован наследованием. Блок без активации наследует активацию через его регулярные входы. Активация, связанная с основным сигналом, производится в моменты активации блока, который формирует сигнал. В этом примере есть только один источник активации, часы событий. Блок усиления наследует свою активацию от основного входного сигнала, т.е. с выхода генератора прямоугольных волн. Фактически ничего не изменилось бы, если бы у блока усиления был входной порт активации, связанный с выходом часов активации. Рассмотрим пример.



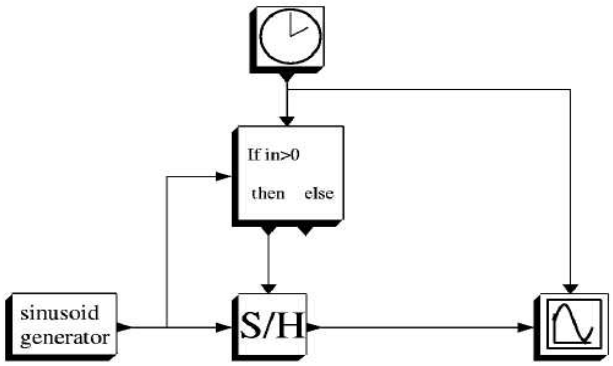
В этом примере сумматор работает с наследованием. Активация осуществляется через входные порты блока, который наследует активацию. Блок суммирования наследует два входных сигнала, и механизм наследования создает два входных порта активации для блока сумматора.

Способ использования наследования, которым Scicos с активизирует сумматор, полностью эквивалент способу, показанному на следующей модели, где все активации явно введены.

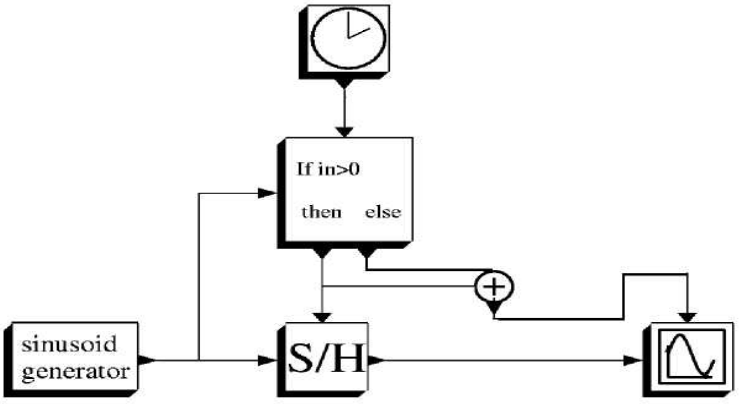


Механизм наследования также работает и для случая условных активаций и в асинхронных случаях.

Например, следующая блок-схема Scicos, после учета наследования,

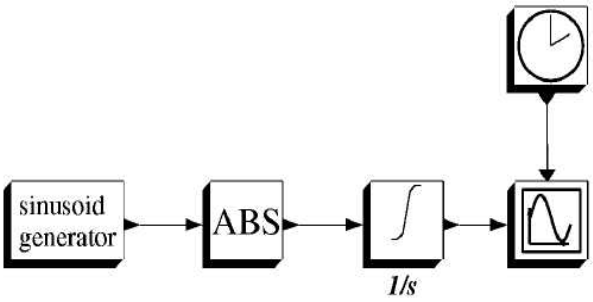


может быть преобразована так, как показано ниже.



Непрерывные и постоянно активные блоки

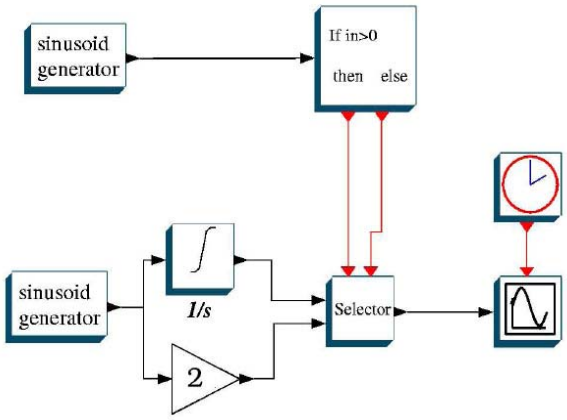
Блок может быть всегда объявлен активным, заданием параметра dep_t = True. Это означает, в частности, что блок непрерывный, то есть, его выходные сигналы и его состояния могут изменяться непрерывно во времени. Механизм наследования также относится и к непрерывному блоку. Если следовать формализму Scicos, активность всегда должна определяться сигналом, полученным на входном порту активации. Однако, чтобы избежать излишней сложности, это свойство просто определено как свойство блока. Есть множество блоков в палитрах Scicos, которые являются постоянно активными. Эти блоки могут использоваться с другими блоками, чтобы строить гибридные схемы.



В этом примере генератор синусоиды и блок интегратора (1/s) всегда активны. Блок ABS (вычисление абсолютной величины) наследует активность от генератора синусоиды.

Блок активации с условием также работает с непрерывными сигналами.

Рассмотрим схему.



В этом примере Scope показывает или интеграл от синусоиды или саму синусоиду после усиления. Это сделано, посредством выбора соответствующего сигнала селектором в зависимости от величины сигнала другого генератора синусоиды.

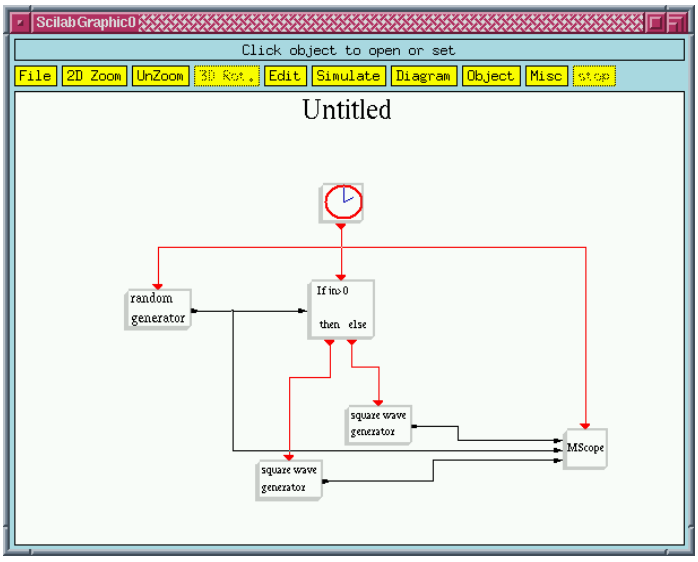
Использование Synchro-блоков

Мы уже видели, что не все блоки модели Scicos должны активизироваться. Мы также знаем, что модель может иметь два независимых источника активации, управляющих разными блоками. Другая ситуация возникает, когда активация необходима во многих точках схемы.

Когда два блока активизированы одним источником активации (например одним генератором событий), мы говорим, что они синхронизированы. В этом случае, у них одинаковые времена активации, и если выход каждого подключен к входу другого, то компилятор знает, что блоки исполняются в правильном порядке. Два блока, активизированные двумя независимыми часами в одно и то же время не синхронизированы. Даже при том, что у этих часов одинаковые параметры. Дело в том, что они могут быть активизированы симулятором в любом порядке.

С другой стороны, два активирующих сигнала могут быть синхронными, но не быть одновременными. Например, моменты одной активация могут быть подмножеством моментов другой активации (активация с кратными частотами). В этом случае, часть событий, сформированных тактовым генератором с более высокой частотой, одновременна с событиями низкочастотного тактового генератора. Но и в этом случае оба сигнала (с высокой и низкой частотами) должны быть порождены одним генератором, иначе они будут не синхронны. Т.е. необходимо убедиться, что низкочастотная активация получена из высокочастотной.

Рассмотрим еще один пример. Здесь блок синхронизации маршрутизирует события, поступившие на входной порт активации между выходными портами активизации. Выбор выходного порта активации зависит от величины сигнала на основном входе блока.

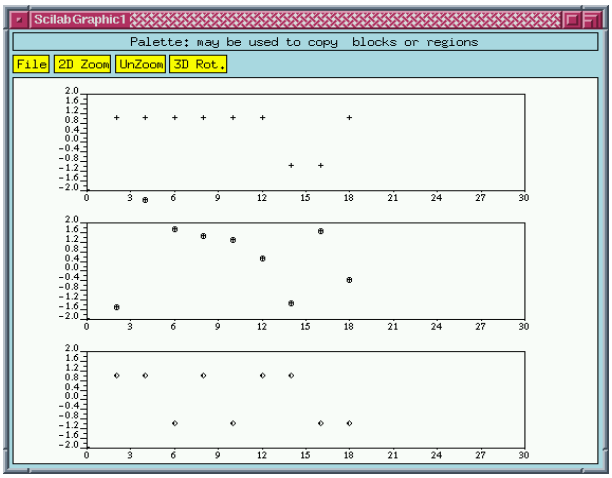


«If-then-else» направляет поступающий сигнал активизации (в этом случае он поступает с часов) на один из своих выходных портов активизации. Если входной сигнал (то есть выходной сигнал блока random generator), положительный, сигнал активации от часов поступает в первый выходной порт активизации, в противном случае – во второй. Генератор шума формирует случайную последовательность. Параметры блока определяют статистические свойства

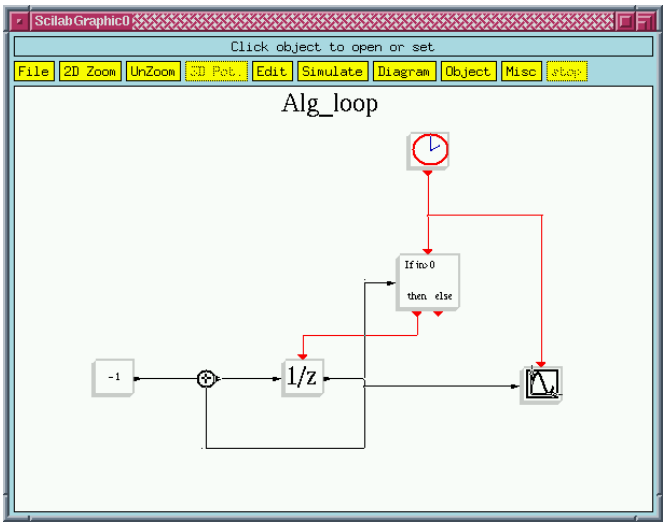
случайной переменной. Блок генератора прямоугольных волн выдает свой сигнал на выход, изменяя свое внутреннее состояние в момент активации.

Выберем опцию Gaussian для шумового генератора. Модифицируем параметры MScope, чтобы получать 3 входа.

Результат для периода часов 2 с показан ниже.

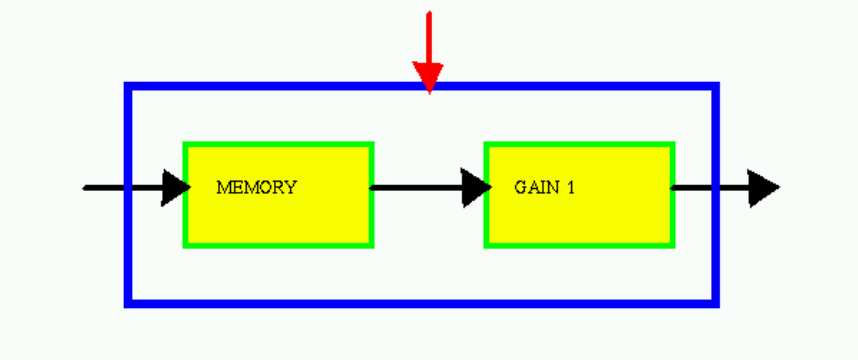


Создадим новую схему. Рассмотрим теперь цикл событий с условием, например счетчик, который останавливается при нулевом результате. Здесь есть специфические проблемы.



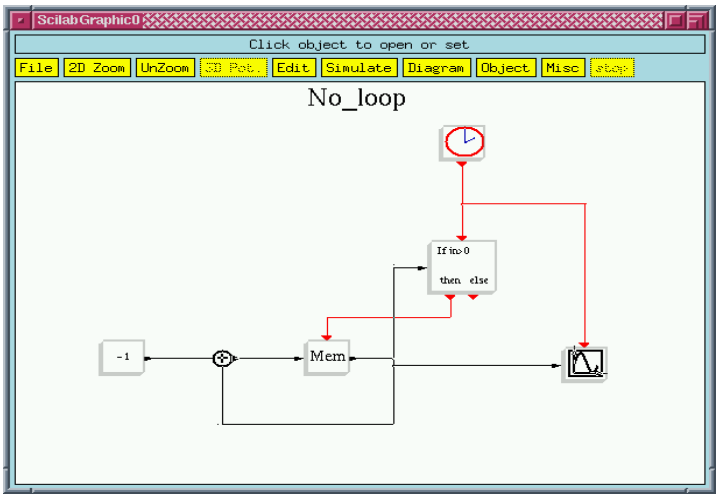
Компилятор выдает сообщение: algebraic loop. Эта модель неприемлема. Проблема здесь в том, что для решения, на какой порт If-then-else-блок должен направлять поступающее активационное событие, нам нужно знать величину выхода 1/z-блока. Но этот выход зависит от поступления события (или не поступления) из If-then-else-блока. Возникает неопределенность, которая называется алгебраический цикл (algebraic loop).

Может показаться, что 1/z-блок – это просто в некотором отношении блок задержки и он должен прервать цикл. Но это не так. Причина - в том, что функции 1/z-блока реализуются следующим образом.



Структура 1/z-блока

Из схемы видно, что когда 1/z-блок активизирован, содержимое памяти подается на выход, а затем входной сигнал копируется в память. Таким образом, нет непосредственной связи входом и выходом, а есть между входным сигналом активизации и выходом. Но для того, чтобы прервать цикл, нам нужно иметь выход MEMORY доступным за пределами блока. Это не возможно в данном блоке. Поэтому в Scicos есть специфический блок Mem (в палитре Others), который позволяет это сделать. Создадим новую схему.

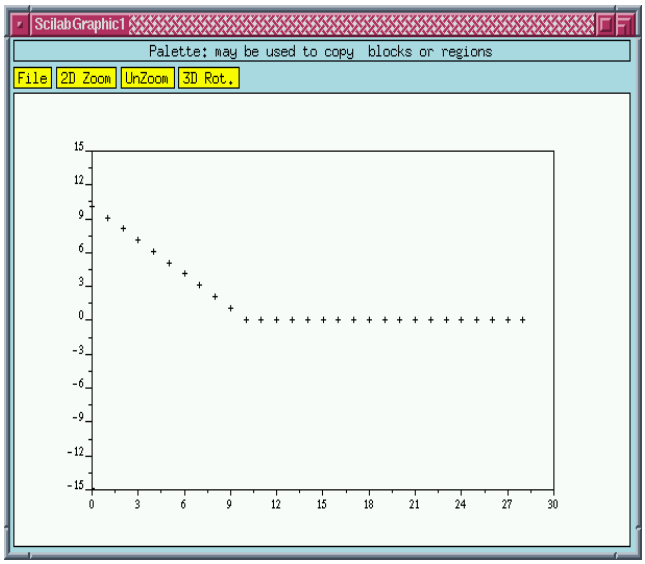


Эта схема успешно проходит компиляцию.

Если мы в начале запишем в блок Mem положительное число, то при каждом такте часов активации эта величина уменьшается на единицу пока не станет равна нулю. При этом сигналы часов переключаются в неподключенный выходной порт активизации синхронизатора. Счетчик останавливается.

Запустим модель.

В Mem записано 10. Работа продолжается до конечного времени моделирования. Можно остановить моделирование, когда достигнут ноль, соединением порта else с блоком Stop (палитра Event).

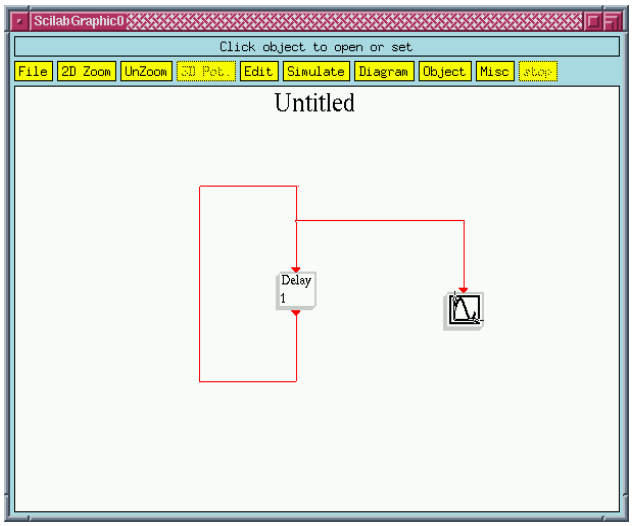


Что такое Event Clock?

Может показаться, что Event Clock (часы активации) не являются основным блоком Scicos. Event Clock являются существенным элементом любой модели Scicos. Это скомпилированный суперблок. Во многих отношениях он подобен основному блоку, вот почему для него используется термин блок или блок Scicos.

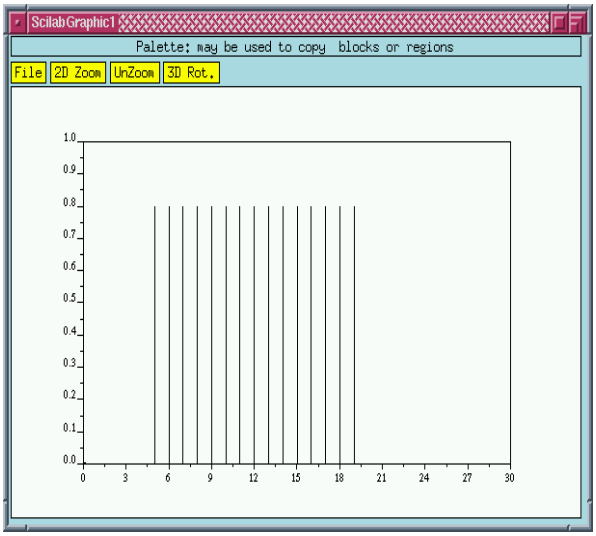
Выходной сигнал активации блока Synchro не задержан по времени. Но блок Baisc в Scicos может генерировать задержанные события. Это та особенность, которая использована в Event Clock.

Создадим следующую схему. Блок Delay (или точнее говоря, event delay block) находится в палитре Events.

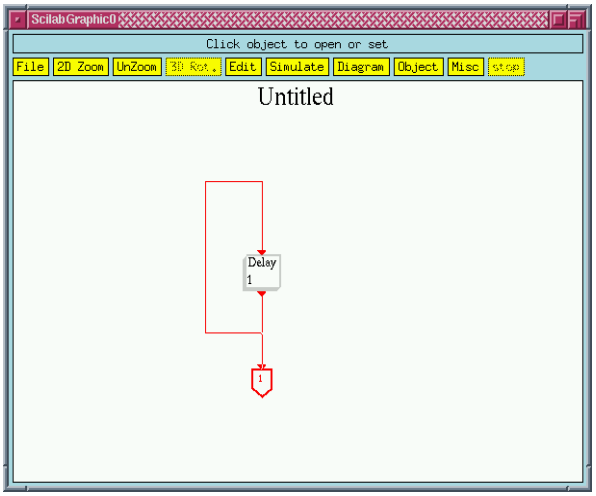


Блок Delay имеет два параметра. Первый определяет задержку между поступающим и исходящим событием, второй – время начала генерации событий. Если это время отрицательно, то блок первоначально не генерирует событие. Это означает, что в этой ситуации модель остается замороженной во времени. Во многих ситуациях установка времени начала генерации не нужна.

Установим параметры блока Delay (delay = 1 и initial firing time = 5) и запустим модель. Видно, что зацикленный блок задержки ведет себя как an Event Clock.



Теперь создадим блок из полученных часов. Для этого сформируем следующую схему.



Эта схема не может быть запущена. Она не является моделью Scicos, это скорее, внутренность суперблока. Наша задача преобразование этой схемы в блок, то есть компиляция суперблока.

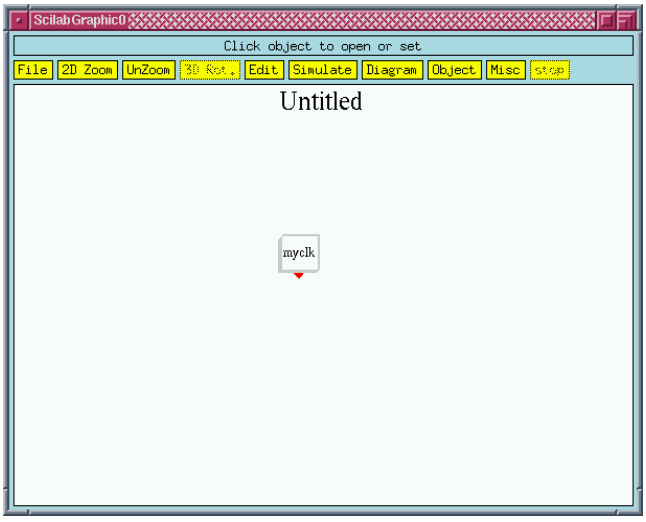
Сохраним его как функцию сопряжения (Interfacing или GUI – графического интерфейса пользователя) под именем myclk.

Это можно сделать, используя команду Save as Interfacing function в меню File.

Каждый блок в Scicos имеет функцию сопряжения (или графического интерфейса пользователя). Эта функция Scilab определяет все свойства блока (размер, цвет, параметры, интерфейс пользавателя, начальные условия и т.д.) и имя вычислительной функции, которая определяет математические свойства блока (например, вычисляет выходной сигнал по входному сигналу). Вычислительная функция обычно пишется на C или Fortran'е для обеспечения вычислительной эффективности, но может также быть записана в Scilab. Связывающая функция является всегда функцией Scilab.

Сохранение схемы в форме связывающей функции, ассоциирует ее с блоком, который должен включать схему. Количество входов и выходов блока определяется количеством входных выходных портов на схеме (как основных, так и активационных) и т.п.

Загрузим новый блок в пустое окно. Воспользуемся кнопкой Add new block меню Edit.



Это созданные часы события. Параметры скомпилированного суперблока просто объединение всех параметров блоков в пределах этого суперблока. При компиляции суперблока, связывающая функция сгенерирована таким образом, что когда блок щелчком мыши открыт для установки (open/set), параметры блоков в пределах суперблока могут быть установлены один за другим. В некоторых случаях это не удобно. Для того, чтобы модифицировать интерфейс пользователя, Interfacing function может быть отредактирована вручную.

Создание новых блоков Scicos

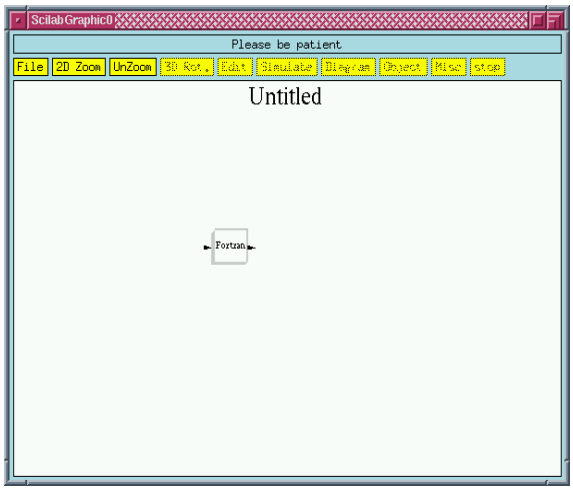
Для того чтобы создать новый блок, нам потребовались две функции. Во-первых, функция сопряжения, которая определяет графические свойства, параметры и т.п. и имя функции, которая производит вычисления необходимые при

моделировании. Функции сопряжения должна быть функцией Scilab, вычислительная функция может быть написана на C, Fortran'e или языке Scilab.

Мы рассмотрели пока один способ создания новых блоков: компиляция суперблока.

Если рассматривается блок простого (непосредственного) действия (у него нет состояния) и он не имеет входных или выходных портов активации, то он может быть реализован с использованием C или Fortran-блока в палитре Others. Эти блоки содержат описание вычислительной функции соответственно на C или Fortran'e.

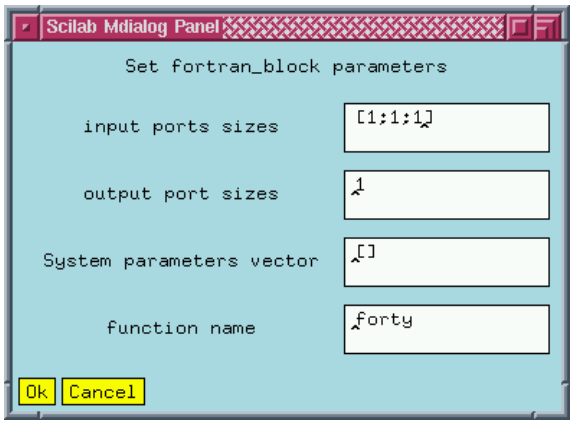
Скопируем блок Fortran из палитры Others.



Блок Fortran допускает внешнее задание вычислительной функции. Для того, чтобы использовать этот блок, вы должны иметь fortran-компилятор или f2c. Для MC Windows, вам необходим Visual C++. Компилятор f2c включен в пакет Scilab.

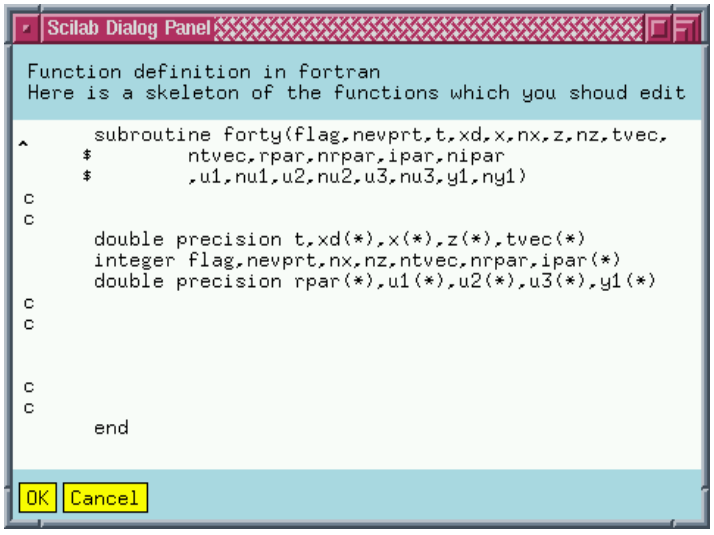
Щелкните на блоке после выбора Open/Set в меню Object.

Вы можете теперь задать размерность входных и выходных портов, параметры и имя вычислительной функции.

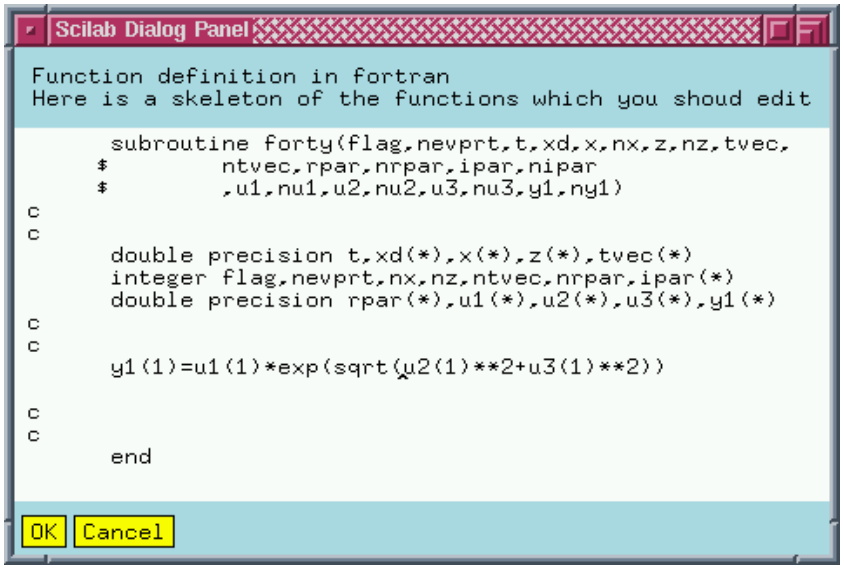


Показанный на рисунке блок будет иметь три скалярных входа и один скалярный выход. Именем вычислительной функции будет forty.

Теперь Scicos автоматически генерирует шаблон функции forty (вызовы и объявления).

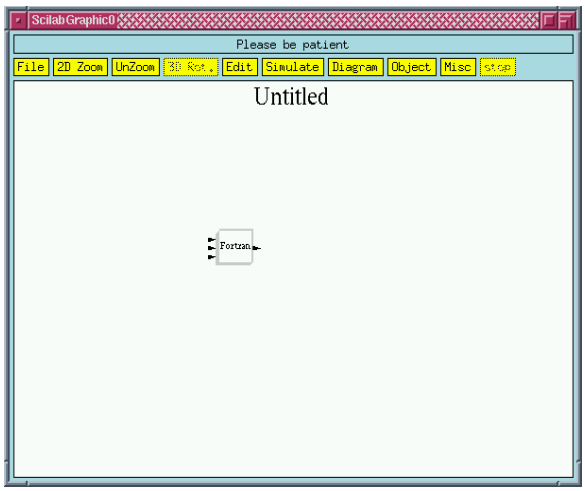


Теперь функция может быть задана записью любого числа Fortran-выражений на этой диалоговой панели. Добавим на панель следующий текст.

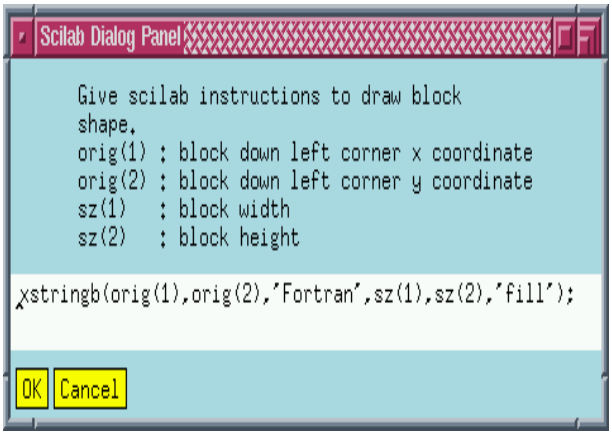


Получилась записанная на Fortran’е вычислительная функция. После щелчка на кнопке OK, forty компилируется и линкуется автоматически, и мы возвращаемся в основное окно Scicos.

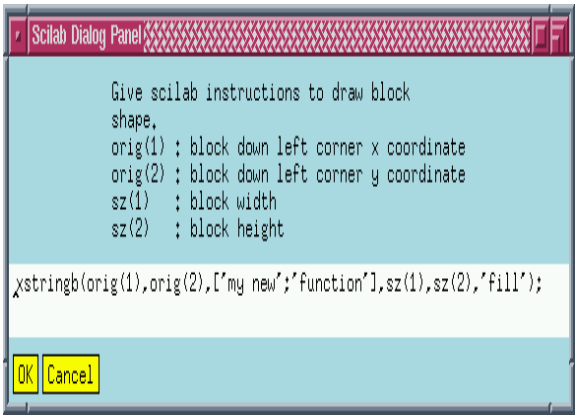
Блок теперь имеет нужное число входных и выходных портов. Он может быть использован при создании моделей.



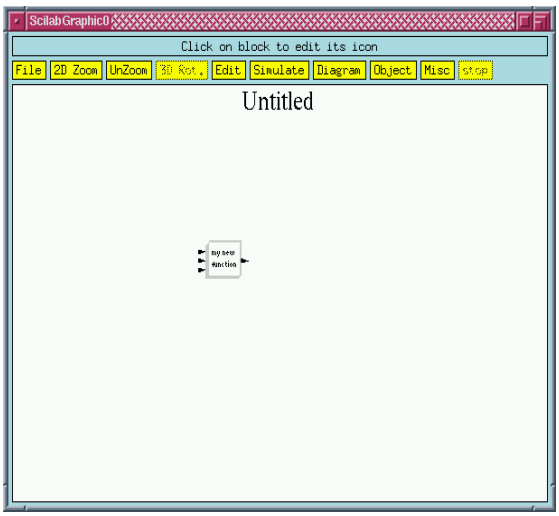
При сохранении модели, которая содержит этот блок сохраняется и Fortran-функция. Когда эта модель в последующем загружается в Scicos, компиляция и линковка производится автоматически. Иконка блока может быть изменена командой Icon в меню Format.



Старая иконка.



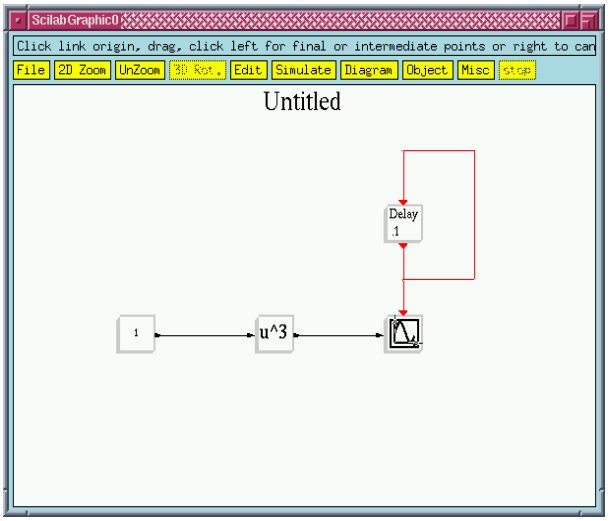
Новая иконка.



Блок на языке C создается точно так же, только вместо Fortran'а используется язык C. Также и для блока Scifunc, за исключением того, что используются выражения Scilab. В случае Scifunc блок может иметь как дискретные, так и непрерывные состояния.

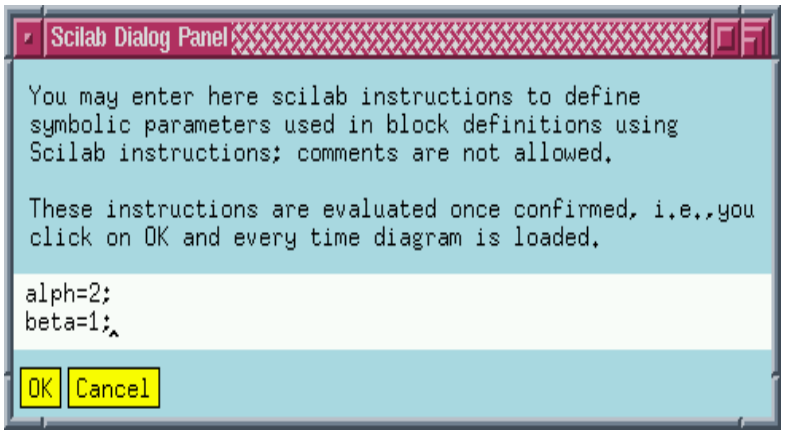
Символические переменные и контекст

Для того, чтобы установить параметры блока, могут быть использованы любая переменная или выражение Scilab. Например, частота генератора синусоиды может быть установлена в 2 * π/10 %. Единственное ограничение в том, чтобы переменные определялись в текущей среде до использования. Команда Scilab может также включать переменные Scilab, но эти переменные должны быть были ранее определены в "контексте" модели. Такие переменные называют *символическими параметрами.* Создадим схему.



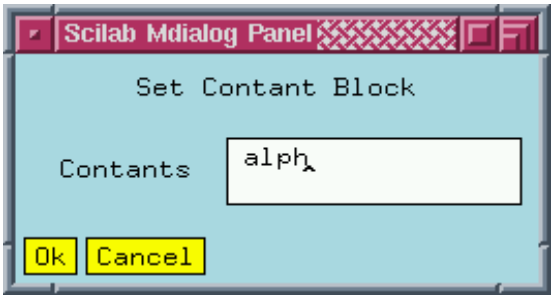
Блок u^3 вычисляет третью степень входного сигнала. В данном случае «3» это параметр блока.

Уточним контекст модели следующим образом. Context в меню Edit позволяет получить доступ к контексту схемы.

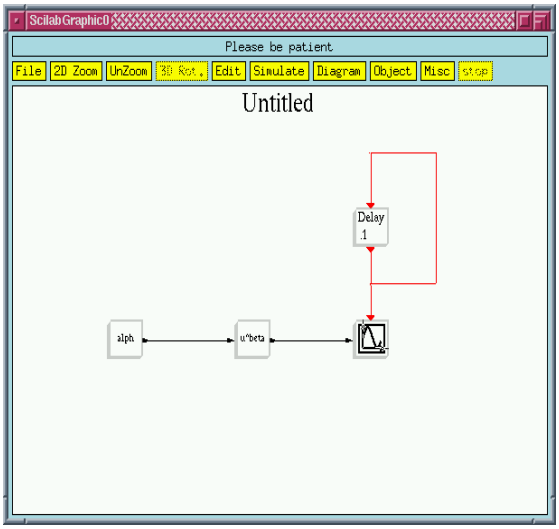


Переменные alph и beta теперь заданы и могут быть использованы в качестве параметра.

Установим в качестве параметров блоков Constant и возведения в степень alph и beta соответственно. Например, для блока Constant это можно сделать следующим образом.



Этот параметр запоминается в символической форме. Схема после изменений параметров выглядит так.



Величины alph и beta определены в контексте. Чтобы изменять величину этих параметров, достаточно изменить контекст. Символическое задание параметров особенно полезно, когда переменная используется во многих точках схемы. Изменение величины этой переменной в контексте изменяет величины параметра всех блоков, в которых эта переменная используется.

Каждый суперблок имеет собственный контекст. Переменная определенная в контексте модели (схемы) может быть использована, чтобы устанавливать блочные параметры блоков модели.

Если переменная не определена в контексте суперблока, она используется с определением в контексте всей модели, включающей суперблок или в контексте суперблока, содержащего рассматриваемый супербока и так далее.

В случае множественного определения, область определена в контексте отдельной части модели.

Многомодельная система и дискретный решатель ДАУ

Моделирование физической системы очень часто приводит к ДАУ с дискретностью или многомодельности. Многорежимная формулировка - способ описать негладкие мультимодельные системы в терминах конечного числа гладких систем. Идея состоит в том, чтобы разделить состояние системы на различные области и определить способ перехода к нужной области. Предполагается, что система описана в терминах единственной гладкой модели в пределах каждой области. Простой пример многорежимного ДАУ:

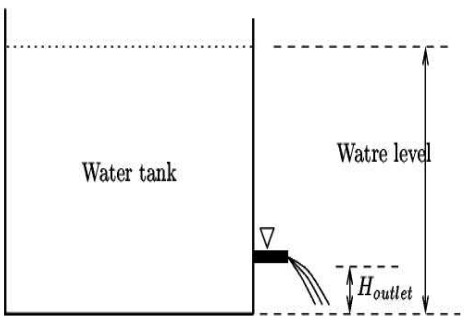
If $(x > 0)$ then

$f1(x, \dot{x}) = 0$ else

$f2(x, \dot{x}) = 0$.

В уравнениях для простоты опущен параметр t.

Т.о., гибридная модель может быть составлена из нескольких моделей таким образом, что каждая модель действительна в определенной области. Как физический пример, рассмотрим водный резервуар с открытым выходом.



В этой системе x, уровень воды в резервуаре, может быть выражен как функция Q, вытекающего потока воды:

$$\dot{x} = k1Q,$$

где k1является постоянным. Q может также быть выражено как функция x:

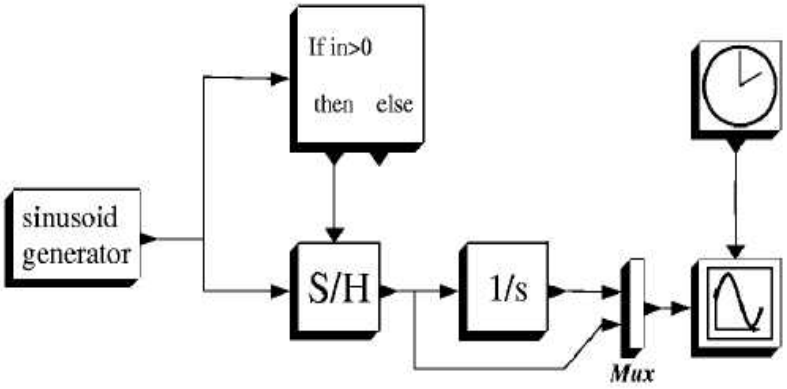
$$Q = \begin{cases} k_2\sqrt{x - H_{outlet}} & \text{if } x \geq H_{outlet} \\ 0 & \text{if } x < H_{outlet}. \end{cases}$$

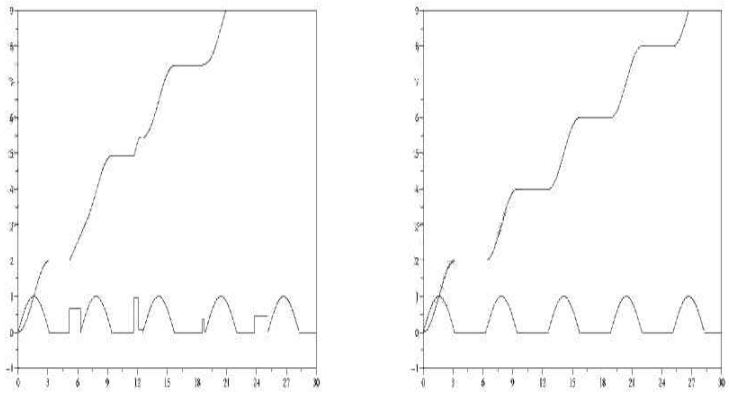
Следовательно, поведение этой простой модели описано мультимодельной системой, составленной из двух уравнений. Это разделение с предположением гладкости в каждой области очень важно, потому что негладкие системы не могут обращаться непосредственно к решателю. Численные решатели предполагают, что переменные и их первые производные непрерывны. Если это предположение не верно, в точках разрыва должны быть предприняты специальные меры.

Чтобы избежать этой проблемы, числовой решатель должен использовать ОДУ/ДАУ до пункта разрыва и затем отдельно после. В большинстве случаев точка разрыва непредсказуема, и она должна быть обнаружена. Чтобы обнаружить и ограничить область разрыва, решатели используют блоки пересечения нуля, которые пересекают его в точке разрыва. Например, для примера резервуара мы можем использовать уравнение:

$$G_{zc}(x) = x - H_{outlet}.$$

Благодаря этой функции решатель может найти точный момент появления разрыва. В некоторых случаях он останавливается с сообщением об ошибке, когда шаг становится слишком маленьким, в других случаях счет может продолжиться с ошибочным результатом. В качестве примера рассмотрим простую систему.





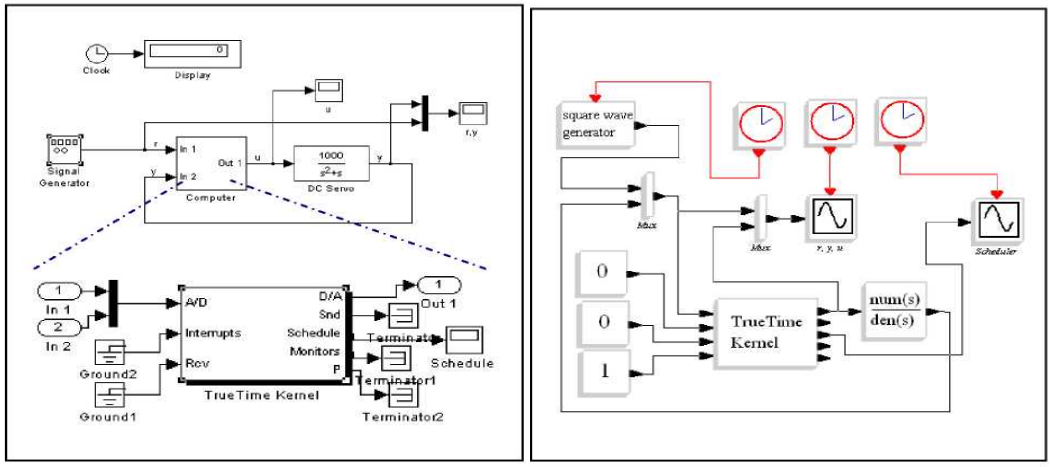
На левом рисунке, представлен результат моделирования, без учета неоднородности и нулевого пересечения. Справа правильный результат моделирования.

Три основных отличия между Simulink и Scicos:

1. Наличие сигналов активации блоков: модель Scicos должна быть конфигурирована по другому, чтобы получить те же результаты.

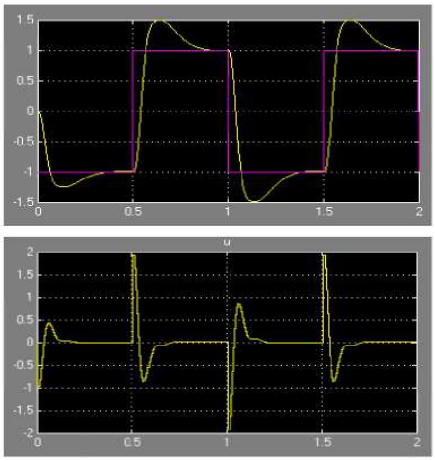
2. Размерность входов и выходов: Scicos требует установки этих параметров для блоков, в то время как в Simulink’е они устанавливаются в соответствии с вычислительным кодом.

Иллюстрация 1 показывает пример PID моделей в исполнении Simulink и Scicos.

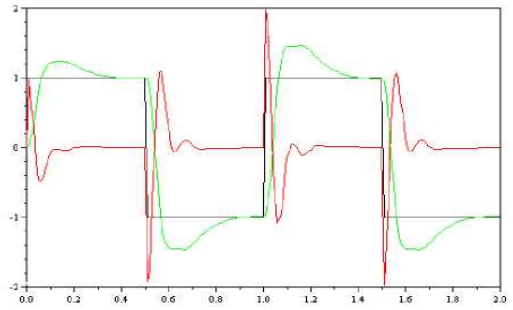


Из графиков видно, что оба имитатора приводят к одинаковым результатам.

Simulink



Scicos



Примеры

Изучать программирование на наглядных примерах намного легче, чем читать учебники, как бы хорошо они ни были написаны. Хороший пример заставляет быстро поверить в свои силы и подвигает на создание своих собственных моделей.

Пример 1. Рассмотрим структуру системы оценивания в пространстве состояний, получающую выборку данных от линейной системы. Система смоделирована как линейная непрерывная:

$\dot{x} = Ax + Bu,$

$y = Cx,$

где A, B, C - постоянные матрицы.

Наблюдению доступен вектор y, поступающий на вход без шумов. Кроме того, на вход поступает управление u и начальное значение вектора x: x_0. На выходе формируется оценка €. Переходный процесс в такой системе должен быть по возможности коротким, но при этом флуктуационная составляющая ошибки не должна быть велика.

Уравнение оценивания в таком случае имеет вид:

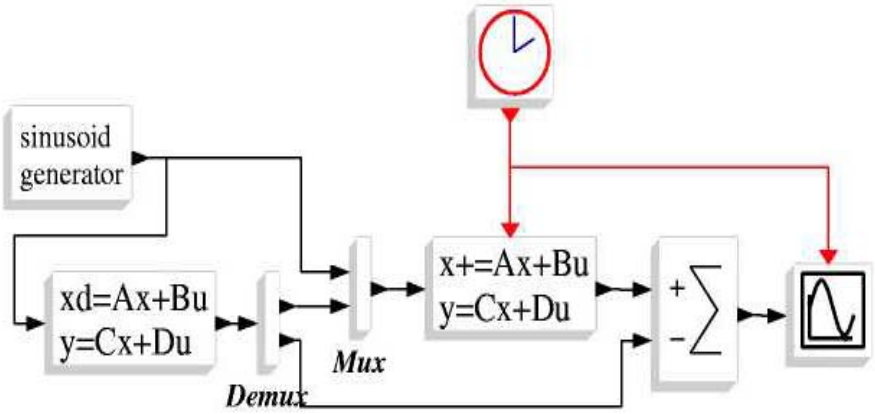
$\dot{x} = A$€ $+ Bu + K(y - C$€$)$.

Матрица K должна быть выбрана так, чтобы у собственных значений $-KC$ были отрицательные вещественные части.

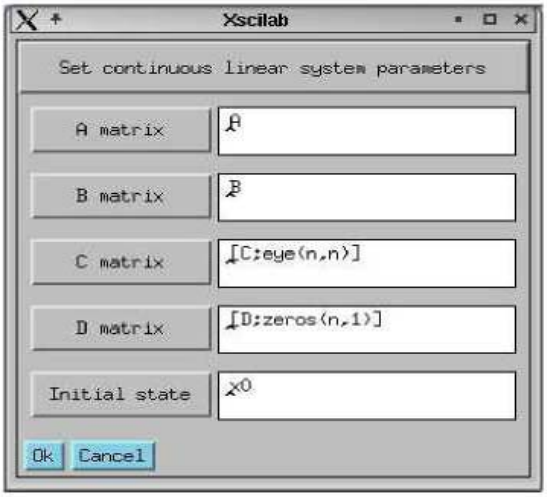
Текст программы Scilab для выполнения этой процедуры с неопределенными заранее матрицами, помещенными в контекст модели, является следующим:

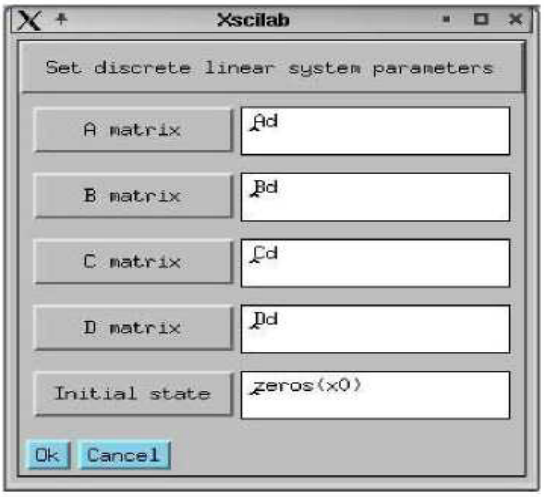
```
n=5;m=2;dt=.2;
A=rand(n,n);A=A-max(real(spec(A)))*eye()
B=rand(n,1);C=rand(m,n);D=zeros(m,1);
x0=rand(n,1);
K=ppol(A',C',-ones(x0))';
Ctr=syslin('c',A-K*C,[B,K],eye(A),zeros([B,K]))
Ctrd=dscr(Ctr,dt)
[Ad,Bd,Cd,Dd]=abcd(Ctrd)
```

Функция ppol используется, чтобы получить матрицу усиления K. На вход u подается sin (t).



Использование символических параметров полезно, потому что оно позволяет нам создавать универсальную модель. Чтобы изменить матрицы A, B, и C и размерность системы или время дискретизации, мы должны изменить только определение m, n и матрицы. Никаких изменений в схеме не требуется.





На верхней панели показаны параметры до их изменения. На нижней после изменения.

Фактически, перезаписывая контекст, мы можем сделать диаграмму полностью универсальной.

Сделаем загрузку:

```
load('datafile')
K=ppol(A',C',-ones(x0))';
Ctr=syslin('c',A-K*C,[B,K],eye(A),zeros([B,K]))
Ctrd=dscr(Ctr,dt)
[Ad,Bd,Cd,Dd]=abcd(Ctrd)
```

Мы только должны удостовериться, что файл данных datafile содержит переменные A, B, C, D, x0, и dt прежде, чем начать сеанс Scicos. Этот файл может быть создан в Scilab следующим образом:

→save('datafile",A,B,C,x0,dt)

Комментарий.

Если контекст содержит много строк кода Scilab, удобно поместить код в отдельный файл скрипта и выполнить его единственной командой **exec** в контексте. Однако, если файл, выполняемый командой **exec,** изменен, когда модель уже открыта и пользователь хочет, чтобы изменения были приняты во внимание, он должен сделать оценку, потому что у Scicos нет никакого способа узнать, были ли сделаны изменения.

Нужно также отметить, что использование отдельного файла скрипта подразумевает, что диаграмма Scicos не является модульной, и этот файл скрипта должен всегда сопровождать модель.

Пример 2. Решить дифференциальное уравнение

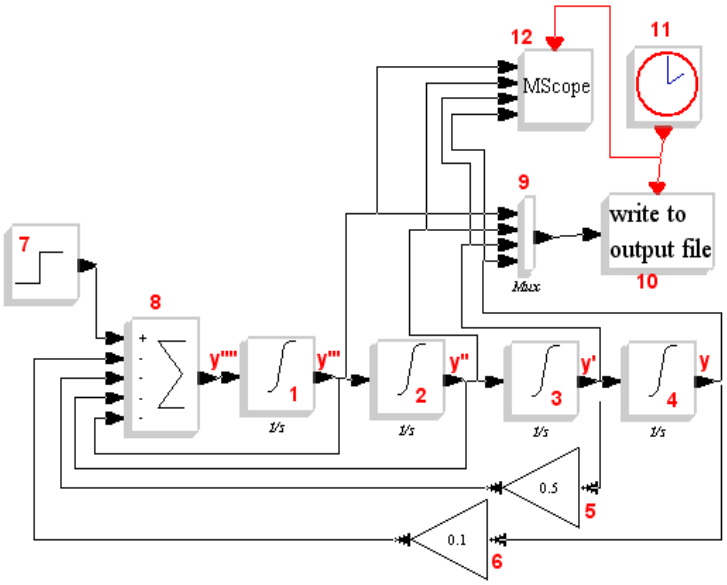
$$y'''' + y'''+ y'' + 0.5y' + 0.1y = 0.1,$$

при нулевых начальных условиях.

Для самой высокой производной запишем

$y'''' = - y'''- y'' - 0.5y´- 0.1y + 0.1.$

Предположим, что y'''' известна. Тогда, после четырехкратного интегрирования получим y. Самая высокая производная, y'''' существует между блоками 8 и 1.



Постоянное слагаемое 0.1 формируется блоком 7 как шаг в момент t=0. Производные y'''', y''', y'' и y отображаются в MScope 12.

Результаты решения через мультиплексор 9 записаны в файл в ASCII-коде блоком 10.

Начальные условия у всех интеграторов нулевые, в соответствии с условиями задачи.

У интеграторов есть верхние и нижние пределы. Необходимо не забывать задать их перед пуском.

Блок 7 по умолчанию имеет начальное время ступени равное 1. Его необходимо заменить на 0.

Блоки 5 и 6 не обязательны, так как можно установить входной множитель в блоке 8. Здесь они включены для наглядности.

Часы активации имеют начальное время равное нулю, а период 0.1. Это не имеет никакого отношения к точности решения, а только задает шаг выхода.

MScope имеет размерность входных портов портов {1 1 1 1}. Величины Ymin и Ymax были установлены после первого прогона. Период обновления (refresh-period) такой же как окончательное время интегрирования.

Запись в файл (блок 10) удобна тем, что данные могут быть импортированы в другие программы. Необходимо задать имя файла и задать строку формата в стиле Fortran. Это значит:

формат пишется в круглых скобках и включает:

/ – новая линия;

iw – область целого числа с шириной w знаков;

fw.d – область числа с плавающей запятой шириной w знаков, включая d цифры после десятичной запятой. Здесь w> = d+3, чтобы включить по крайней мере одну цифру целого числа, знак и точку;

ew.d – область числа с плавающей запятой, степень десяти шириной w знаков, включая d цифры после десятичной запятой. Здесь w> = d+7, чтобы включить по крайней мере одну цифру целого числа, знак для мантиссы, трех знаков для экспоненты, знака экспоненты и одного для точки;

nx n – пробелы;

tn – перемещает курсор принтера в положение n.

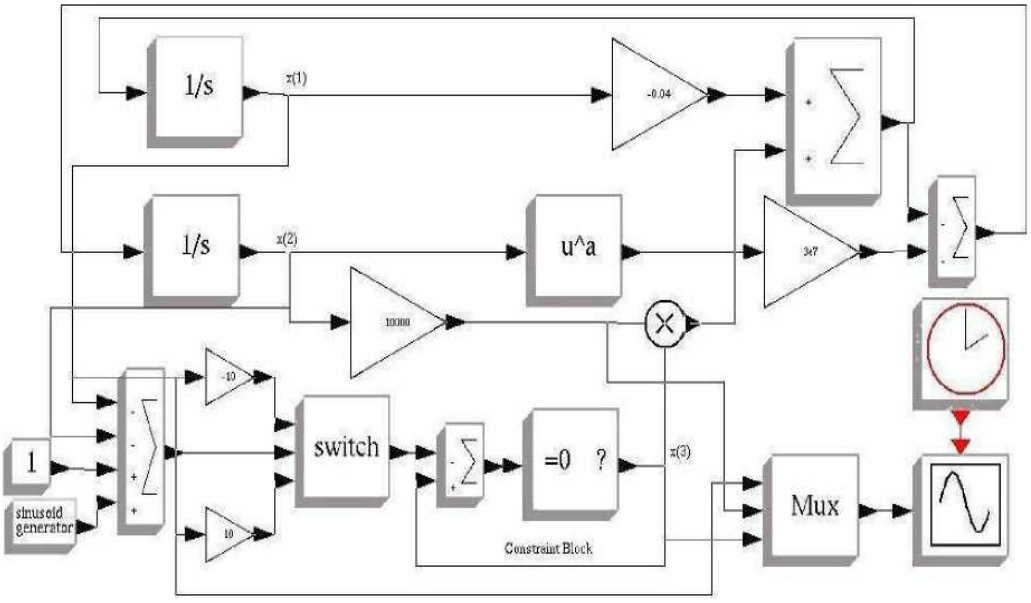
В этом примере выбрано (7 (e16.8,1x)) для строки формата (8 десятичных чисел), так как величины переменных малы.

Пример 3. Рассмотрим решение еще одного ДУ.

Продемонстрируем простым примером, как моделирование некоторых уравнений может стать сложным (можно показать как эту задачу можно облегчить с помощью компонентов Modelica). Рассмотрим систему ДУ:

$$\begin{cases} \dot{x_1} = -.04x_1 + 10^4 x_2 x_3 \\ \dot{x_2} = .04x_1 - 10^4 x_2 x_3 - 3 \times 10^7 {x_2}^2 \\ \text{if } (1 + sin(0.1t) - x_2 - x_1 > 0.5) \\ \text{then } x_3 = -10x_1 \\ \text{else } x_3 = 10x_1 \end{cases}$$

Теперь покажем реализацию этих уравнений в модели.

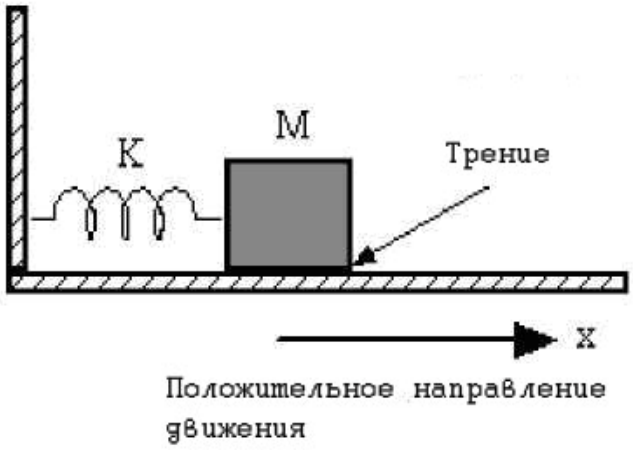


В блочном подходе есть несколько недостатков. Для моделирования часто необходимо слишком много блоков. Это делает модель неудобочитаемой. Также в такой сложной блок-схеме трудно отразить наглядно структуру физической или технической задачи.

Пример 4.

Блок с массой M перемещается по грубой поверхности с коэффициентом трения f. Постоянная сила сопротивления – f·M·g, где g – сила тяготения, и K – коэффициент упругости пружины.

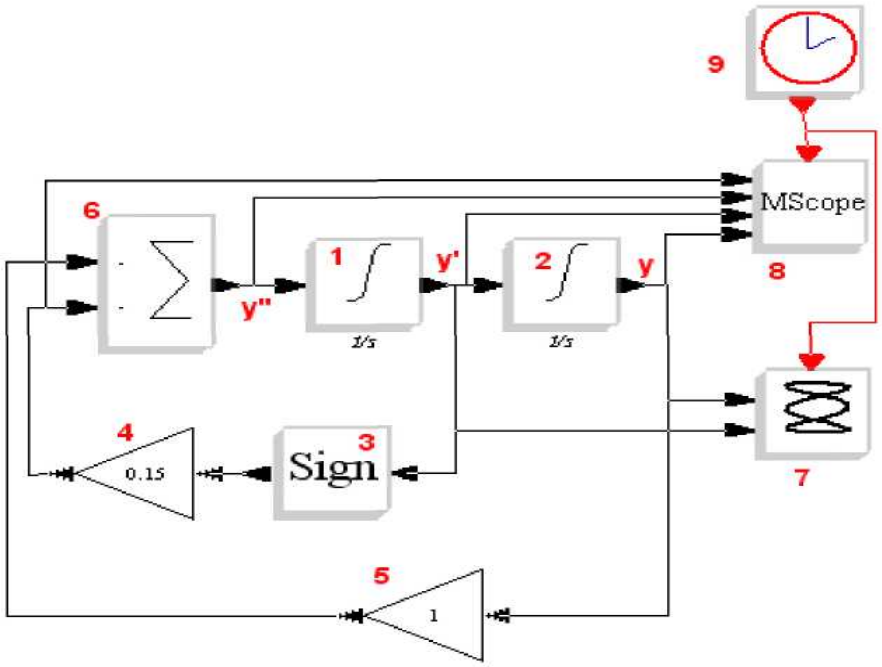
Покажем движение блока на графике, если его перемещают на 1 метр направо и отпускают.

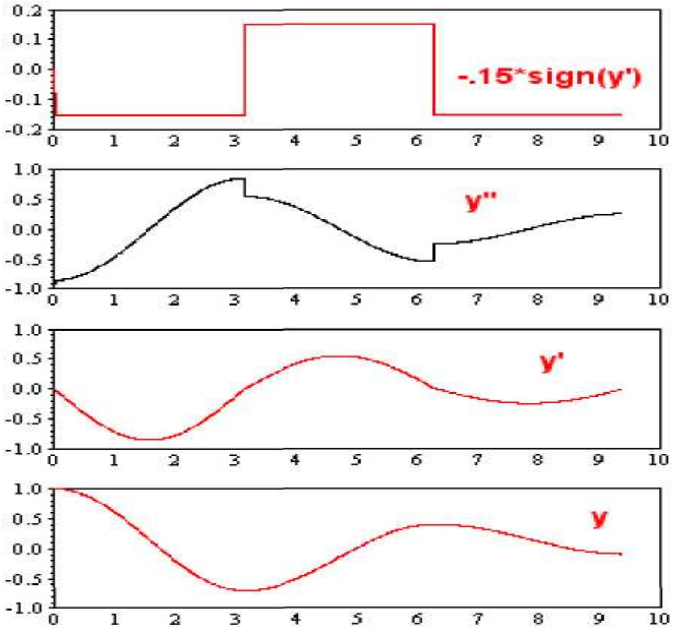


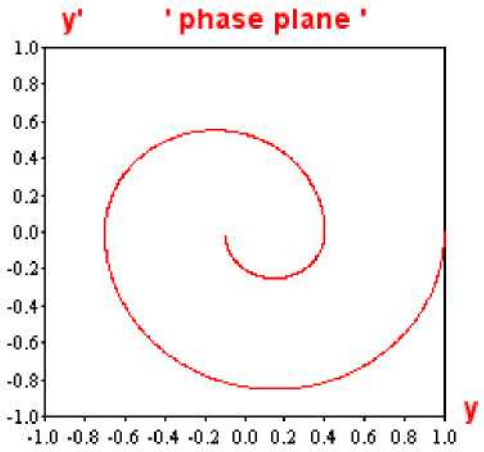
Такое движение можно описать уравнением:

$$y'' = -y - 0.15*\text{sign}(y').$$

Начальное условие для положения установлено в 1 метр (в блоке интегратора 2), время окончания интеграции 9.4, время обновления MScope 10. Период часов активации 0.01, а начальное время равно нулю.







Комментарии.

Фазовый портрет отражает начальные величины y = 1 и y' = 0. Скорость в начале равна нулю, а ускорение отрицательно, таким образом, блок начинает перемещаться влево. Это означает, что постоянная сила трения, задаваемая блоком 6, должна быть положительна и это так и есть, так как она отрицательна на входе (знак минус на входе блока 6).

Если установить время интеграции больше, чем физически возможное время (здесь $3 \cdot \pi$ секунды) то Scicos входит в бесконечный цикл и после этого придется остановить модель вручную.

Литература

1. Ramine Nikoukhah. Scicos: a dynamic systems modeler and simulator. INRIA-Rocquencourt. Domaine de Voluceau, France.
2. Masoud Najafi. The numerical solver for the simulation of the hybrid dynamical systems. Universite Paris, 2005, 237 c.
3. Jean-Marie Zogg. Arbeiten mit Scilab und Scicos. Fachhochschule Ostschweitz, 2007, 125 c.
4. Finn Haugen. Master Scicos. 8. July 2008.
5. Grudlagen digitale Regelung und Mechatronik Simulation mit Scilab und Scicos. www.ebokaktiv.de
6. Ramine Nikoukhah. Scicos: a dynamic systems modeler and simulator. INRIA, Franse.
7. Материалы сайта www.scilab.org