В.Н. Сидоренко

СИСТЕМНО-ДИНАМИЧЕСКОЕ МОДЕЛИРОВАНИЕ в среде POWERSIM

Справочник
по интерфейсу
и функциям

МАКС-ПРЕСС

ББК 65.9(2)28

Рецензенты:
Доктор экономических наук,
профессор кафедры математических методов анализа экономики
экономического факультета МГУ им. М.В. Ломоносова
Черемных Ю.Н.

Сидоренко В.Н.

С347 Системно-динамическое моделирование в среде POWERSIM: Справочник по интерфейсу и функциям. – М.: МАКС-ПРЕСС, 2001. – 159 с.

ISBN 5-317-00168-4

В книге описываются возможности системно-динамического визуального моделирования с использованием пакета POWERSIM (версия 2.5c). Приводится описание основных этапов построения системно-динамических моделей, проведения имитационных экспериментов. Рассматриваются преимущественно примеры моделей, используемых в экономике. Дается подробный перечень и описание встроенных в POWERSIM функций, а также их сравнение с функциями, используемыми в других пакетах системно-динамического моделирования, таких как Dynamo, Vensim, STELLA/iThink.

Книга рассчитана на читателей, знакомых с имитационным моделированием и использующих системно-динамические модели в управленческом консалтинге (реинжиниринге) и стратегическом менеджменте.

Напечатано с оригинал-макета автора

ISBN 5-317-00168-4

Содержание

Введение

Данная книга представляет собой, прежде всего, справочник по функциям языка Powersim (версия 2.5c), созданного Нидерландской корпорацией "Powersim" в 1993-1997 гг. специально для визуального системно-динамического моделирования на IBM-совместимых компьютерах в среде Windows 3.1x, NT, 95 и др. Кроме того, в книге приводится в сжатом виде необходимая информация по интерфейсу Powersim, основным инструментам и приемам работы с ними, используемым при системно-динамическом моделировании, а также сравнение Powersim с некоторыми другими распространенными средами разработки системно-динамических моделей (Dynamo/ИМИТАК, Stella/iThink, Vensim) с точки зрения используемого диаграммного языка и синтаксических конструкций. Данная книга не является прямым переводом весьма объемной фирменной документации пакета Powersim, однако содержание раздела, связанного с описанием набора и синтаксиса встроенных функций максимально приближено к исходной документации. Книга содержит необходимую информацию для успешного изучения пакета и работы с ним.

Powersim предназначен для построения непрерывных и частично дискретных моделей. Он относится к семейству языков имитационного моделирования, получивших наибольшее распространение в мире с середины 1990-х гг. наряду с такими языками имитационного моделирования как Process Charter, Extend+BPR, Arena (SIMAN), ProModel, Rethink, Piligrim и др.

Основная цель языка Powersim заключается в построении описания или математической модели воображаемой или реальной системы. Любая модель состоит из множества взаимосвязанных элементов, описываемых переменными. Элементы модели и связи между ними определяют структуру модели. Говорят, что модель построена, если определены все переменные и связи между ними, т.е. если задана структура модели.

Для определения имитационных моделей в Powersim существует редактор диаграмм, в котором все переменные представляются графическими объектами (пиктограммами), соединенными между собой при помощи стрелок, обозначающих потоки и связи (см. модель экономического роста Р. Солоу, "паутинообразную" модель установления рыночного равновесия, демографическую модель "передвижки возрастов" и другие модели, приведенные в книге). Каждая связь отражает некоторую зависимость между переменными, соединенными данной связью. Точное определение вида зависимости определяется уравнением, записанном на языке Powersim.

Powersim дает возможность видеть на одной и той же диаграмме структуру и уравнения модели, а также ее поведение. Для отображения поведения модели в ходе моделирования существуют анимационные средства и динамические объекты, которые можно размещать на диаграмме произвольным образом. Поведение модели определяется из имитационных экспериментов (имитаций) с моделью и может быть использовано не только для анализа самой модели, но и для улучшения понимания поведения моделируемой системы в различных ситуациях.

Язык имитационного моделирования Powersim может быть использован для построения моделей как простых, так и сложных систем. Известно, что для сложных систем характерна множественность описания. Поэтому для них нельзя построить единственную,

истинно верную модель, а можно лишь описать их поведение при помощи тех или иных моделей, отражающих характерное поведение моделируемых систем в конкретных ситуациях. Тем не менее, Powersim является достаточно мощным инструментом (это зависит от компоновки: профессиональная, стандартная, рантайм, демонстрационная), позволяющим не только быстро и наглядно строить и анализировать системно-динамические модели, но и демонстрировать в доступной форме результаты моделирования широкому кругу людей, не обязательно являющихся специалистами в области математического моделирования.

Powersim относится к тому семейству языков имитационного моделирования (Dynamo, Stella/iThink, Vensim, Rusim), который достаточно быстро и эффективно позволяет овладеть техникой имитационного моделирования представителям не только естественных, но и гуманитарных наук. Благодаря этому он получил широкое распространение среди бизнесменов и даже школьников (пока только за пределами России) и в этом смысле является более доступным в освоении, нежели чисто технические среды разработки имитационных моделей, такие как GPSS, GASP, SIMSCRIPT, SIMULA, SLAM, SIMULINK (MATLAB), РДО и др., обладающие более широкими возможностями, но предполагающие наличие у пользователя не только хорошей подготовки в области математического моделирования, но и программирования.

К.ф.-м.н., к.э.н. Сидоренко В.Н.

Глава 1
Краткое описание интерфейса Powersim

1.1 Операционная среда

В окне приложения Powersim отображается меню и окна используемых документов с моделями (рис 1.1). Поскольку большинство команд меню дублируются соответствующими кнопками [кроме всех команд меню Color (Цвет), Window (Окно), Help (Справка)], расположенными на панелях команд и инструментов, то в дальнейшем остановимся только на функциональном назначении кнопок.

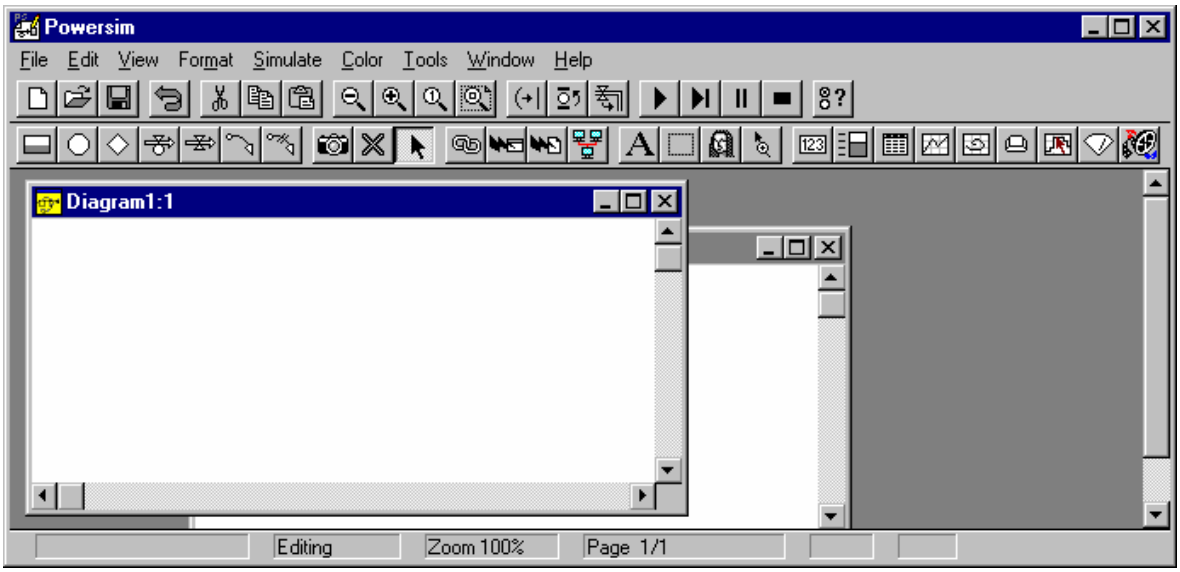


Рисунок 1.1 Главное окно приложения Powersim

Состав команд и соответствующих кнопок зависит от выбранного режима работы: полный или упрощенный интерфейс. Кроме того, существуют панель состояния, расположенная внизу окна приложения и дающая информацию об основных режимах работы с имитационной моделью.

1.1.1 Описание кнопок панели команд

Ниже приводится описание назначения основных кнопок панели команд (см. рис. 1.1), многие из которых характерны для большинства Windows-приложений.

Кнопки для работы с файлами и редактирования диаграмм

Создать модель	Открыть модель	Сохранить модель	Отменить послед-нее действие	Вырезать	Копировать	Вставить

Данные кнопки соотвествуют командам меню File (Файл) и Edit (Правка).

Кнопки для масштабирования и форматирования имитационной модели

						
Уменьшить масштаб	Увеличить масштаб	100 %-й масштаб	Масштабировать по размеру окна	Спрямить связь	Повернуть название	Переместить вентиль

Данные кнопки соответствуют командам меню View (Вид) и Format (Формат). В режиме упрощенного интерфейса кнопки масштабирования и форматирования имитационной модели не отображаются.

Кнопки для управления имитацией и установки опций интерфейса Powersim

				
Запуск имитации	Пошаговая имитация	Вкл./выкл. паузу во время имитации	Остановка имитации	Настойки интерфейса

Первые четыре кнопки соответствуют командам меню Simulate (Имитация), а последняя команде Options... (Настройки интерфейса) в меню Format (Формат), которая позволяет устанавливать различные опции для режимов отображения и правки диаграмм, отображения уравнений, а также устанавливать общие свойства главного и дочерних окон и опции сохранения.

1.1.2 Кнопки панели инструментов

Ниже приводится описание назначения основных кнопок панели инструментов (см. рис. 1.1). Когда нажимается кнопка на панели инструментов или производится соответствующий выбор в меню Tools (Инструменты), изображение курсора при перемещении мыши над рабочей областью окна изменяется на пиктограмму выбранного средства. Это говорит о том, что нажатие левой кнопки мыши где-либо на диаграмме создаст новый образец объекта, связанного с данным инструментом.

Инструменты для работы с объектами модели

						
Уровень	Вспомогательная переменная	Константа	Поток с темпом	Поток	Связь	Связь с запаздыванием

Инструменты редактирования модели

		
Переменная кадра ("теневая переменная")	Ластик	Указатель (курсор)

Данные инструменты для работы с объектами модели являются стандартными для большинства системно-динамических языков (Dynamo, Stella/iThink, Vensim, Rusim). Они необходимы для построения потоковых диаграмм при разработке системно-динамических моделей. Каждому объекту (уровню, темпу и др.) соответствует свой математический объект (их алгоритмическое и математические описание для различных системно-динамических языков приведено ниже в п. 1.3). В режиме упрощенного интерфейса кнопки "Поток" и "Ластик" не отображаются.

Инструменты для работы с передаточными объектами

			
Цепь	DDE	Архив	Сетевая игра

Данная группа инструментов предназначена для работы с моделями нижнего иерархического уровня (моделями подсистем) через объект "Цепь", с другими приложениями (с таблицами MS Excel, Lotus 1-2-3 и документами MS Word) через установку DDE-связи, с файлами текстовых форматов (*.txt) при записи и считывания данных через объект "Архив", а также для создания сетевых игр с использованием объекта "Сетевая игра". В режиме упрощенного интерфейса данная группа инструментов не отображается.

Инструменты для работы со статическими объектами

Текст	Рамка	Рисунок	Линия

Указанные инструменты, в первую очередь, предназначены для оформления причинно-следственных и потоковых диаграмм модели. Кроме того, они могут быть использованы при разработке концептуальной модели и обрисовке структурных блоков модели в общих чертах (например, при определении подсистем в рамках системы или при изображении причинно-следственных диаграмм).

Инструменты для работы с динамическими объектами

Число	Ползунок	Таблица со временем	График со временем	Диаграмма рассеяния	Кнопка	График массива	Спидометр	Мульти-медиа

Инструменты "Ползунок/линейка", "Кнопка" позволяют работать с моделью в интерактивном режиме, задавая начальные условия, коэффициенты в уравнениях модели и реакцию на те или иные события со стороны пользователя. Инструменты "Число", "График массива", "Спидометр" позволяют не только отслеживать изменения выбранных переменных в режиме имитация, но и задавать в интерактивном режиме (при включенной паузе) начальные условия и коэффициенты в уравнениях модели. Остальные инструменты позволяют пользователю наблюдать за ходом имитационного эксперимента, отслеживая изменение выбранных переменных либо графически, либо численно при помощи таблиц, либо событийно при помощи запуска мультимедийных презентаций.

1.1.3 Панель состояния

Панель состояния (см. рис. 1.2), расположенная внизу окна приложения, дает следующую информацию об основных режимах работы с имитационной моделью (слева направо): 1) контекстная справка по основным командам меню / время моделирования в процентах (в режиме имитации); 2) режим правки модели / режим имитации / режим паузы; 3) масштаб диаграммы для имитационной модели; 4) текущий номер страницы в многостраничном документе (в Powersim поддерживается автоматическая разбивка документа на страницы); 5) состояние кнопки Caps Lock (Вкл./выкл. ввод прописных букв) на основной клавиатуре; 6) состояние кнопки Num Lock (Вкл./выкл. дополнительную цифровую клавиатуру) на дополнительной клавиатуре.

Рисунок 1.2 Панель состояния

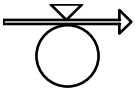
1.2 Диаграммная техника

1.2.1 Стандартные объекты и их обозначения

Рассмотрим типы стандартных объектов, которые используются при построении любой модели в Powersim. В основном, имеются пять типов объектов: уровни (запасы), потоки (материальные связи), вспомогательные переменные, константы и связи. Кроме того, вспомогательные переменные можно комбинировать с потоками для создания "потоков с темпами", а связи можно подразделить на информационные связи, связи с запаздыванием, и инициализирующие связи. Ниже показаны пиктограммы, используемые для обозначения каждого из стандартных объектов, и соответствующие им кнопки панели инструментов.

Level (Уровень или переменная уровня) – переменная, накапливающая изменения. Ее значение изменяется за счет потоков.

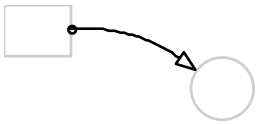
Flow (Поток или материальная связь) – переменная, влияющая на уровни и изменяющая их значения.



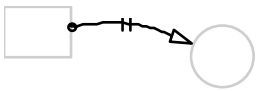
Flow with rate (Поток с темпом или переменная темпа) – переменная, влияющая на уровни. Поток регулируется соединенной с ним переменной темпа, которая обычно является вспомогательной переменной (и может быть отделена от потока).

Auxiliary (Вспомогательная переменная) – переменная, содержащая вычисления с другими переменными.

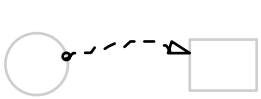
Constant (Константа) – переменная с фиксированным значением, используемым при вычислениях значений вспомогательных переменных или потоков.



Link (Связь) – поставляет информацию вспомогательным переменным относительно значения других переменных.



Delayed Link (Связь с запаздыванием) – используется только если вспомогательная переменная содержит специальные функции задержки (материальной или информационной, см. п. 3.1).



Initialization Link (Инициализирующая связь) – передает начальное значение переменной уровня (образуется автоматически при соединении пиктограммы вспомогательной переменной или константы с пиктограммой уровня).

Кроме того, существуют объекты, возникающие при создании стандартных объектов и являющиеся потомками стандартных объектов:

Cloud (Облако) – неопределенный источник или выход потока, входящего или исходящего из переменной уровня. Обозначает внешние источники / стоки в модели. Возникает, если поток или поток с темпом начинается или заканчивается в любом месте окна, кроме места, где расположен какой-либо другой уровень.

Camera (Переменная кадра или "Теневая переменная") – обозначение, являющееся изображением или псевдонимом для "первоначальной" переменной. Кадры полезны для соединения переменных, размещенных в различных частях модели. Кадр можно создать, используя инструмент "Переменная кадра".

Table lookup function (Табличная функция) – определяется не аналитически, а алгоритмически или графически на основе экспериментальных данных или экспертных оценок. Кроме табличных функций, особые обозначения (символы в круге) предусмотрены и для различных встроенных функций (см. п. 3.1).

Supplementary (Дополнительная переменная) – не является частью структуры системы, т.е. не влияют на другие переменные модели. Дополнительные переменные необходимы для вычисления индикаторных величин, которые отображаются либо графически, либо при помощи таблиц и др. На диаграммах они изображаются обычно так же, как и вспомогательные переменные и поэтому в Powersim особо не выделяются.

Если переменная становится векторной, то контур ее пиктограммы изображается двойной линией, при этом контуры стрелок связей не изменяются.

1.2.2 Сравнение стандартных обозначений объектов в некоторых языках системно-динамического моделирования

В качестве примера, приведем таблицу сравнения основных обозначений стандартных объектов, используемых в таких языках как Dynamo, Stella/iThink, Vensim, Powersim.

Таблица 1.1 Стандартные обозначения в Dynamo, Stella/iThink, Vensim, Powersim

Dynamo	Stella/iThink	Vensim	Powersim	Элемент
Уровень	Уровень	Уровень	Уровень	Уровень
	Темп	Темп		Темп
Вспом. перем.	Вспом. перем.	Вспомогательная переменная	Вспом. перем.	Вспомогательная переменная
Доп переменная	Доп. переменная	Дополнительная переменная	Доп. переменная	Дополнительная переменная
Константа	Константа	Константа	Константа	Константа
Табл функц	Табл. функция	Табличная функция	Табл. функция	Табличная функция
нет	нет	Начальное условие	Начальное условие	Начальное условие
				Материальные связи
				Информационные связи

1.3 Режимы просмотра диаграмм и уравнений модели

В Powersim реализованы два основных режима просмотра документа на экране: просмотр диаграмм и просмотр уравнений. Эта опция выбирается через меню View (Вид), содержащем команды Diagram (Показ диаграмм) и Equations (Показ уравнений), которые работают подобно паре радио-кнопок. Это означает, что единовременно можно выбрать только одну из них. Точка справа от выбранной команды меню указывает текущий режим просмотра активного окна.

1.3.1 Просмотр диаграмм

При просмотре диаграмм в окне отображается структура модели с использованием определенных для потоковой диаграммы символов (см. п. 1.3). Также отображаются динамические, статические и передаточные объекты согласно установкам Diagram View (Просмотр диаграмм) в диалоговом окне Options... (Настройки интерфейса), задаваемым при нажатии кнопки "Настройки интерфейса" или выборе соответствующего пункта меню Format (Формат). Эти установки определяют, какие категории объектов отображаются, а какие нет.

Режим просмотра диаграмм выбирается по умолчанию при создании новых документов. В данном режиме возможно перемещение, редактирование, выделение (при помощи нажатия и удерживания правой кнопки мыши) и копирование объектов для их последующей вставки в другие приложения, а также создание новых динамических и статических объектов. А, нажав кнопку Enter (Ввод) на клавиатуре или дважды щелкнув левой кнопкой мыши на выделенном объекте, можно открыть диалоговое окно Define Variable (Определение переменной) и редактировать уравнение для выбранной переменной.

1.3.2 Просмотр уравнений

Режим просмотра уравнений дает возможность просмотреть уравнения, лежащие в основе модели. Возможность копирования уравнений и их вставки в другие приложения (такие как, например, текстовый редактор MS Word) делает этот режим просмотра полезным при документировании моделей. Окно просмотра уравнений показывает в текстовой форме структуру уравнений текущей модели. Немодельные объекты, такие как графики, зависящие от времени, рисунки и т.д. не представимы в текстовой форме и, следовательно, в данном режиме не отображаются. Отображение диапазонов, размерностей и единиц измерения переменных, в также показ настроек моделирования определяются установками Equations View (Просмотр уравнений), в диалоговом окне Options... (Настройки интерфейса).

Окна при просмотре уравнений доступны только для чтения. Это означает, что в них невозможно добавлять или удалять переменные, потоки или связи во время просмотра. Однако, нажав Enter на клавиатуре или дважды щелкнув левой кнопкой мыши на выделенном уравнении, можно открыть диалоговое окно Define Variable (Определение переменной) и переопределить уравнение для выбранной переменной.

1.3.3 Просмотр модели в обоих режимах одновременно

Для отображения модели в обоих режимах просмотра одновременно необходимо:

1. Открыть модель в режиме просмотра диаграммы.
2. Выбрать в меню Window (Окно) команду New Window (Новое окно).
3. Выбрать в меню "Окно" команду Cascade (Каскад).
4. Активизировать второе окно диаграммы.
5. Включить режим "Просмотр уравнений".

Все уравнения модели теперь будут выданы в новом окне. Если и окно диаграммы, и окно уравнений открыты в одном и том же документе, то окно уравнений автоматически модифицируется, когда происходят изменения в окне диаграммы.

1.4 Создание и редактирование имитационной модели

Весь процесс создания и редактирования модели можно разделить на три этапа:

1) *создание* новых динамических, статических, передаточных и др. объектов, соответствующих элементам и ассоциированным с ними переменным системно-динамической модели, а также средствам интерактивного управления имитационными экспериментами и отображения, анализа и сохранения их результатов и др.;

2) *редактирование* свойств уже существующих объектов (изменение расположения, цвета контура и шрифта названия динамического объекта, а также изменение расположения, размера, формы и цвета статических и др. объектов) путем использования технологии Drag & Drop (перетащить и отпустить), путем выделения объектов и двойного щелчка на них левой кнопкой мыши или нажатия на клавиатуре кнопки Enter (Ввод), а также использования команды меню и кнопки форматирования, расположенные на панели команд;

3) *удаление* выделенных объектов при помощи ластика или кнопки Delete (Удаление) на клавиатуре (при удалении объектов автоматически удаляются и их связи с другими объектами).

Поскольку второй и третий этапы представляются наиболее знакомыми раннее работавшему с Windows-приложениями читателю, то остановимся детально на первом и частично на втором этапе.

1.4.1 Создание объектов, соответствующих элементам системы

Для создания новых объектов данного типа необходимо:

1. Выбрать желаемый инструмент путем нажатия левой кнопки мыши соответствующей кнопки на панели инструментов или путем выбора соответствующего пункта в меню "Инструменты".
2. Нажать левую кнопку мыши в нужной области на диаграмме для того, чтобы создать и разместить новый объект.

При этом новые объекты, соответствующие отдельным элементам системы, примут вид (рис. 1.3):

Constant_1 Auxiliary_1 Level_1

Рисунок 1.3 Пиктограммы вновь созданных объектов

с автоматическим названием, расположенным под объектом [например, Constant_1 (Константа_1)]. Автоназвание изменяется на нужное название переменной путем нажатия левой кнопки мыши на названии объекта и ввода нового (буквенно-цифрового) названия. Вопросительный знак (?) внутри пиктограммы указывает на то, что переменная не определена (не задано ее значение или определяющее ее уравнение) или что допущена ошибка в ее определении. (Отображение вопросительных знаков для неопределенных переменных можно отключить, используя опции "Просмотра диаграмм" диалогового окна "Настройки интерфейса".)

1.4.2 Создание и редактирование нового потока (материальной связи) и потока с темпом

Для создания нового потока необходимо:

1. Выбрать желаемый инструмент путем нажатия левой кнопки мыши на кнопке "Поток" или "Поток с темпом" на панели инструментов или путем выбора соответствующего пункта в меню "Инструменты".
2. Нажать левую кнопку мыши с курсором потока в том месте, откуда будет начинаться поток (начальная точка). Если поток должен исходить из некоторого уровня, то следует нажать кнопку, когда курсор находится над центром данного уровня. В противном случае подразумевается внешний источник, который обозначается облаком (см. п. 1.2.1).
3. Не отпуская левую кнопку мыши, переместить стрелку потока в выходную (конечную) точку и отпустить кнопку. Если поток должен входить в некоторый уровень, то кнопку следует отпускать, когда курсор находится над данным уровнем, цвет которого при этом изменится. В противном случае подразумевается внешний сток, который обозначается облаком (см. п. 1.2.1).

В итоге могут получиться следующие потоковые диаграммы (рис. 1.4):

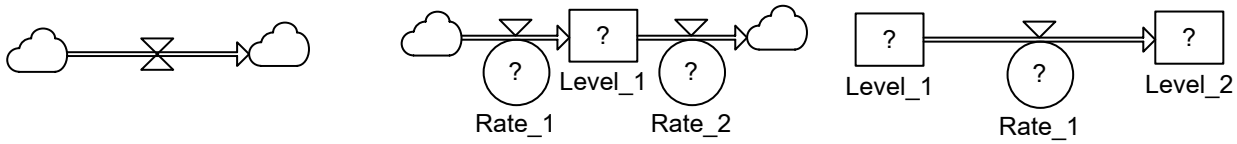


Рисунок 1.4 Возможные потоковые диаграммы

Если при создании потока нужно, чтобы стрелка потока была не в виде отрезка прямой, а в виде ломанной, то для создания угла стрелки потока необходимо проделать следующие действия:

1. При перемещении потока при нажатой левой кнопкой мыши переместить начало стрелки потока в положение, где должен быть угол.
2. Не отпуская левую кнопку мыши, нажать правую кнопку мыши или кнопку Insert (Вставка) на клавиатуре.
3. Продолжить перемещение потока в новом направлении (всего может быть не более 4 изломов стрелки потока) до места назначения (конечная точка).

В итоге может получиться диаграмма следующего вида (рис. 1.5):

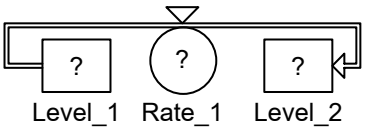


Рисунок 1.5 Потоковая диаграмма с изломанной стрелкой потока

Удаление лишних углов стрелки потока производится в обратном порядке с использованием кнопки Delete (Удаление) на клавиатуре вместо кнопки Insert (Вставка).

Если возникнет необходимость отсоединить поток от уровня, то для этого необходимо выполнить следующие действия:

1. Выделить поток или поток с темпом, нажимая на его пиктограмме левой кнопкой мыши (курсор примет вид руки, а начало и конец потока будут отмечены метками в форме маленьких черных квадратов), как показано на рис. 1.6.

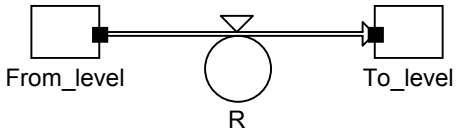


Рисунок 1.6 Выделение потока

2. Нажать левую кнопку мыши на нужной метке (метке уровня To_level) и оттащить начало или конец стрелки темпа от соответствующего уровня, не отпуская левую кнопку мыши.
3. Отпустить левую кнопку мыши на новом месте. При этом появится объект "Облако" (рис. 1.7).

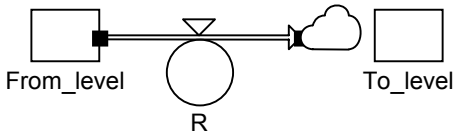


Рисунок 1.7 Отсоединение потока от уровня

Соединение уже существующих стрелок потоков и уровней проводится в обратном порядке. Изменение формы стрелок потоков проводится аналогично, но только при этом используются не метки на концах стрелок потоков, а промежуточные метки.

1.4.3 Создание и редактирование информационной связи

Для установления информационной связи (с запаздыванием или без него) между двумя переменными необходимо:

1. Выбрать желаемый инструмент путем нажатия левой кнопки мыши на кнопке "Связь" или "Связь с запаздыванием" на панели инструментов или путем выбора соответствующего пункта в меню "Инструменты".
2. Поместить курсор внутри первой переменной (From_Var) и нажать левую кнопку мыши.
3. Не отпуская левую кнопку мыши, переместить курсор ко второй переменной (To_Var) и отпустить левую кнопку мыши.

В итоге может получиться диаграмма следующего вида (рис. 1.8):

Рисунок 1.8 Соединение вспомогательных переменных информационной связью

Если информационная связь соединяет константу или вспомогательную переменную с переменной уровня, то эта связь автоматически становится инициализирующей (см. п. 1.2.1).

Если необходимо разделить информационную связь на несколько ветвей, как показано на нижеследующей диаграмме, то для этого необходимо выполнить следующие действия:

1. Переместить стрелку связи в положение, где будет располагаться "точка ветвления".
2. Отпустить левую кнопку мыши при нажатой кнопке Shift (Смещение) на клавиатуре, или нажать правую кнопку мыши, не отпуская левую.
3. Соединить стрелками связи точку ветвления (division point) с каждой из переменных, на которых эта связь оказывает влияние (рис. 1.9).

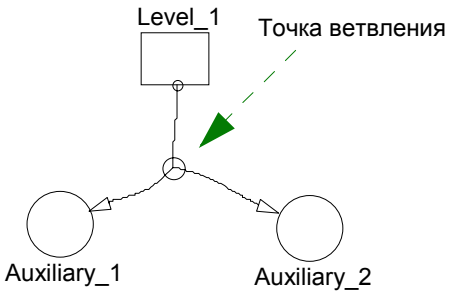


Рисунок 1.9 Разветвленная информационная связь

Для удаления ветвей информационной связи необходимо использовать стандартную процедуру удаления объектов (см. выше п. 1.4). Для изменения формы и размеров стрелок связи следует:

1. Выделить информационную связь, нажимая на ее пиктограмме левой кнопкой мыши (курсор примет вид руки, а начало, конец и касательные к точкам изгиба стрелок связи будут отмечены метками в форме маленьких черных квадратов, как показано на рис. 1.10).

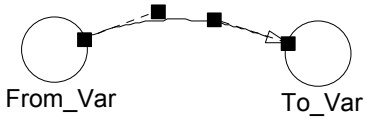


Рисунок 1.10 Выделенная информационная связь

2. Нажать левую кнопку мыши на нужной метке и переместить ее, не отпуская левую кнопку мыши, в нужное место, после чего отпустить левую кнопку мыши.

1.4.4 Создание статических, передаточных и других объектов

Создание статических объектов (текст, рамка, рисунок, линия), передаточных объектов (цепь, DDE, архив, сетевая игра), а также объектов, предназначенных для работы с динамическими объектами (число, ползунок, таблицы, графики, кнопка, спидометр, мультимедиа), которые перечислены в п. 1.1.2, происходит так же, как и в случае с динамическими объектами, соответствующими элементам системы. Единственным отличием является то, что данные объекты не соединяются материальными и информационными связями. Примеры этих объектов см. в приложении 4 и 7.

1.4.5 Создание моделей подсистем

Если модель системы получается слишком громоздкой или используются различные установки для моделирования отдельных частей системы (например, метод и временной интервал моделирования), то удобно упростить ее, выделив подсистемы и построив модели для каждой из них. Далее можно обращаться к моделям подсистем через главную модель (Main model), в которой в качестве элементов будут выступать модели подсистем, т.е. комодели (Co-models), как показано на рис. 1.11.

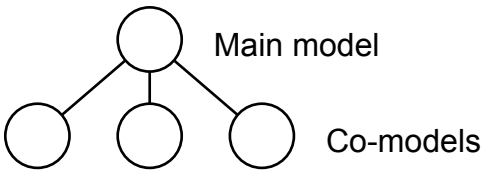


Рисунок 1.11 Связь главной модели с комоделями

Выделение комоделей также оказывается полезным и в тех случаях, когда одна и та же комодель используется для моделирования типовых ситуаций принятия решений (например, в случае, если комодели описывают одинаковый процесс принятия решений различными игроками на рынке ценных бумаг).

Главная модель и комодели создаются стандартным способом, а затем соединяются между собой при помощи передаточного объекта Chain (цепь), создание которого производится также стандартным образом (см. п. 1.4.4). Для этого объекта в диалоговом окне Define Chain (Определение цепи) необходимо задать модель и переменную-источник и переменную-приемник (см. приложение 4). Следует отметить одно важное ограничение, налагаемое на переменную-приемник: в качестве такой переменной может выбираться только константа или переменная уровня, не инициализируемая вспомогательной переменной.

Главная модель управляет процессом моделирования в комоделях при помощи механизма синхронизации, позволяющего, несмотря на возможные различия в интервалах моделирования главной модели и комоделей, начать и закончить процесс моделирования одновременно во всех комоделях и главной модели, как показано ниже в табл. 1.2.

Таблица 1.2 Синхронизация шагов моделирования главной модели и комодели

Шаг моделирования	Главная модель	Комодель
0	0	10,0
1	1	10,1
2	2	10,2
...	...	...
50	50	15,0
...	...	...
100	100	20,0

Включение и исключение комоделей из главной модели, а также установка интервалов и методов моделирования для главной модели и комоделей производится в диалоговом окне Simulation Setup (Установки моделирования) через список Parallel Simulation (Параллельное моделирование) и связанные с ним кнопки.

1.4.6 Определение внутренних свойств объектов

После создания структуры модели и иных объектов (статических, передаточных и др.) и редактирования их внешних свойств (расположения, размера, формы, цвета, заливки и др.) необходимо отредактировать их внутренние свойства, а именно, задать уравнения (темпов и уровней), определяющие соотношения между переменными модели, связать объекты структуры модели с иными объектами, а также задать реакцию модели и интерфейса модели на события, возникающие при взаимодействии пользователя с моделью в интерактивном режиме.

Для этого необходимо использовать диалоговые окна, возникающие при двойном щелчке левой кнопкой мыши на выбранном объекте. Например, отдельный фрагмент диалогового окна, связанного с заданием уравнений для векторной переменной уровня Inventory (Запасы) размерности 3 с нулевыми начальными условиями для каждой компоненты вектора, представлен на рисунке 1.4.

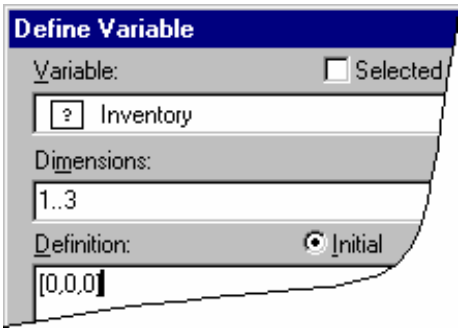


Рисунок 1.12 Фрагмент диалогового окна Define Variable (Определение переменной)

При определении переменной в поле Definition (Определение) необходимо указать уравнение (алгебраическое), которое можно сконструировать из имеющихся влияющих переменных (Linked Variables) и встроенных функций (Functions), списки которых приведены в окне диалога. Кроме того, необходимо указать начальное значение переменной (Initial) (для переменной уровня) размерность переменной (Dimensions) и единицы измерения переменной (Unit of Measure). Следует отметить, что в окне диалога приводятся списки размерностей (Units) или диапазонов (Ranges) определенных до этого переменных и их единицы измерения, так что размерность и единицы измерения определяемой переменной можно просто выбрать из этих списков. При желании, можно также снабдить каждую переменную комментариями, которые вводятся в специальном поле Documentation (Документация) окна диалога.

Рассмотрим, например, модель экономического роста Р. Солоу[1], используемую при макроэкономическом анализе и имеющую следующий вид (рис. 1.13):

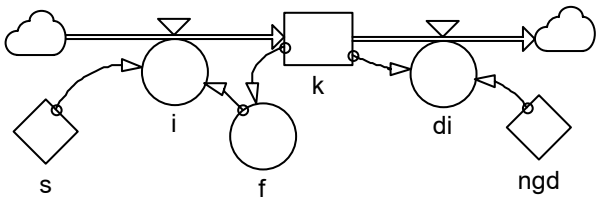


Рисунок 1.13 Потоковая диаграмма для модели Р. Солоу

где *k* – капиталовооруженность одного работника, *i* – удельные фактические инвестиции, *di* – удельные (желаемые) инвестиции, необходимые для сохранения уровня в условиях роста населения с темпом *n*, роста эффективности труда с темпом *g* и выбытием капитала с нормой *d*, *s* – норма сбережения, *f* – производственная функция, *ngd* – сумма *n*, *g* и *d*.

После определения всех переменных модели, в режиме просмотра уравнений (см. п. 1.3.2) можно увидеть следующую систему уравнений:

Уравнения уровней

```
init   k = 10
flow   k = +dt*i - dt*di
doc    k = "Уравнение уровней" {Комментарии}
unit   k = выпуск/работник {с постоянной эффективностью}
```

Уравнения темпов

```
aux    i = s*f
doc    i = "Уравнение темпов"
unit   i = выпуск/(работник*ед.времени)
aux    di = k*ngd
doc    di = "Уравнение темпов"
unit   di = выпуск/(работник*ед.времени)
```

[1] Мэнкью Н.Г. Макроэкономика. Пер. с англ. – М.: Изд-во МГУ, 1994. – С. 144-198.

Вспомогательные уравнения

```
aux    f = SQRT(k)
doc    f = "Вспомогательное уравнение"
unit   f = выпуск/работник {с постоянной эффективностью}
```

Константы

```
const  ngd = 0.1
doc    f = "Константа"
unit   ngd = 1/ед.времени
const  s = 0.1
doc    f = "Константа"
unit   s = 1/ед.времени
```

В данной системе уравнений первые два выражения для k представляют собой задачу Коши вида $dk(t) = (i - di)dt;\ \ k(t_0) = 10.$ В уравнении для f используется встроенная функция SQRT – квадратный корень. Данную зависимость можно было бы задать и через окно диалога [кнопка Graph (График)] с помощью табличной функции, определяемой при помощи статистических методов. При использовании других встроенных функций, например, при изменении нормы сбережений с постоянным шагом, константа автоматически была бы заменена вспомогательной переменной при редактировании уравнения в окне диалога, а на пиктограмме помимо круга появились бы часы.

В качестве примера использования табличных функций рассмотрим "паутинообразную" модель установления рыночного равновесия, используемую в микроэкономике (рис. 1.14).

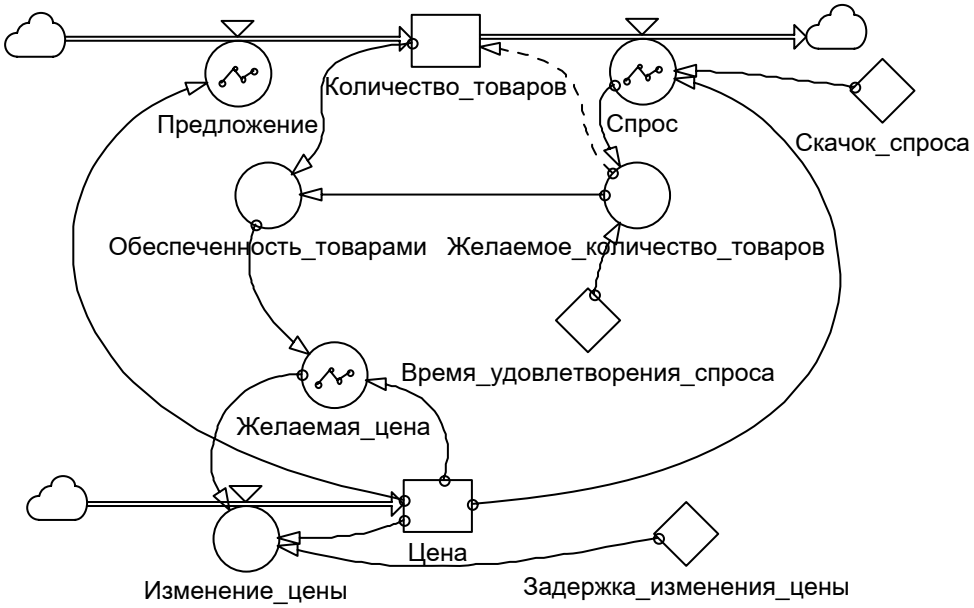


Рисунок 1.14 Потоковая диаграмма для "паутинообразной" модели

Эта модель описывается следующей системой уравнений (без комментариев к переменным):

Уравнения уровней

```
init   Количество_товаров = Желаемое_количество_товаров
flow   Количество_товаров = -dt*Спрос +dt*Предложение
unit   Количество_товаров = шт.
init   Цена = 15
flow   Цена = +dt*Изменение_цены
unit   Цена = руб./шт.
```

Уравнения темпов

```
aux   Изменение_цены = (Желаемая_цена-Цена)/ Задержка_изменения_цены
unit  Изменение_цены = руб./шт./ед.времени
aux   Предложение = GRAPH(Цена,0,5,[0,0,40,57,68,77,84,89,94,
      97,100"Min:0;Max:100"])
unit  Предложение = шт./ед.времени
aux   Спрос = GRAPH(Цена,5,5,[100,73,57,45,35,28,22,18,14,10
      "Min:0;Max:100"])+STEP(Скачок_спроса,10)
unit  Спрос = шт./ед.времени
```

Вспомогательные уравнения

```
aux   Желаемая_цена = GRAPH(Обеспеченность_товарами,0.5,0.1,[2,1.8,
      1.55,1.35,1.15,1,0.87,0.75,0.65,0.55,0.5"Min:0.5;Max:2"])*Цена
unit  Желаемая_цена = руб.
aux   Желаемое_количество_товаров = Спрос* Время_удовлетворения_
      спроса
unit  Желаемое_количество_товаров = шт.
aux   Обеспеченность_товарами = Количество_товаров / Желаемое_
      количество_товаров
unit  Обеспеченность_товарами = безразмерная
```

Константы

```
const Время_удовлетворения_спроса = 4
unit  Время_удовлетворения_спроса = ед.времени
const Задержка_изменения_цены = 15
unit  Задержка_изменения_цены = ед.времени
const Скачок_спроса = 10
unit  Скачок_спроса = шт./ед.времени
```

В данной модели используются три табличных функции GRAPH, две из которых отражают известные законы спроса и предложения (рис. 1.15), а третья – изменение цены (в %), которую готовы платить покупатели, в зависимости от изменения обеспеченности рынка товарами (в %).

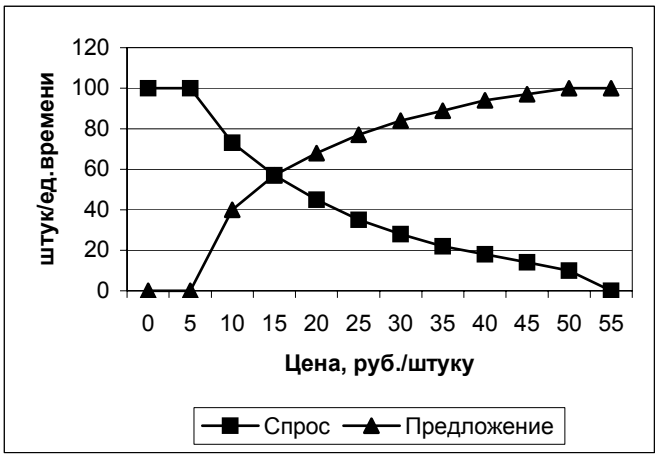


Рисунок 1.15 Функции спроса и предложения

Модель демонстрирует переход от одного устойчивого равновесия к другому в том случае, когда в момент времени t = 10 ед.времени при каждом значении цены увеличивается величина спроса за счет скачка спроса, равного 10 шт./ед.времени. При этом рынок из старого равновесного положения, характеризуемого равновесной ценой 15 руб./шт., равновесным количеством товаров 228 шт. и равновесными спросом и предложением 57 шт./ед.времени, переходит в новое равновесное состояние, характеризуемое новой равновесной ценой, равновесным количеством товаров и равновесными спросом и предложением, равными (с точностью до целых значений) 17 руб./шт., 247 шт., 62 шт./ед.времени соответственно.

Следует отметить, что уравнения уровней в большинстве случаев (если переменные уровня – скаляры) составляются в автоматическом режиме при построении структуры модели. Если же переменные уровня представляют собой вектор, то на этапе построения структуры модели появится диаграмма следующего вида (рис. 1.16):

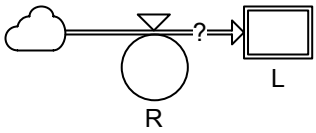


Рисунок 1.16 Потоковая диаграмма при неопределенном уравнении для векторной переменной уровня

В этом случае уравнения уровней придется задавать вручную с использованием векторных функций и ограничений, что приведет следующему изменению диаграммы (появится восклицательный знак, как показано на рис. 1.17):

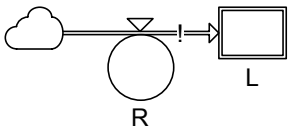


Рисунок 1.17 Потоковая диаграмма при определенном уравнении для векторной переменной уровня

В качестве примера векторной модели рассмотрим модель старения населения, аналогом которой в демографии служит модель "передвижки возрастов", описываемая на языке марковских цепей. В рамках модели предполагается, что все население можно грубо разделить на несколько групп (диапазонов): а) по возрасту (детский, юношеский, средний и пожилой), по месту проживания (север, юг). Изменения численности населения в той или иной возрастной группе в каждом регионе связаны с рождаемостью, смертностью и старением населения (миграция населения между регионами отсутствует). Предполагается, что выполнены следующие условия:

1. Новорожденные пополняют первую возрастную группу. Их родителями могут быть только представители второй и третьей возрастной группы, т.е. лица юношеского и среднего возраста.
2. Старение вызывает переход из предыдущей возрастной группы в следующую (кроме первой). Старение в последней возрастной группе эквивалентно смерти).
3. Стареющие люди выбывают из предыдущей возрастной группы и переходят в последующую (за исключением последней возрастной группы).
4. Умершие вычитаются из соответствующей возрастной группы.

Эта модель описывается потоковой диаграммой следующего вида (рис. 1.18):

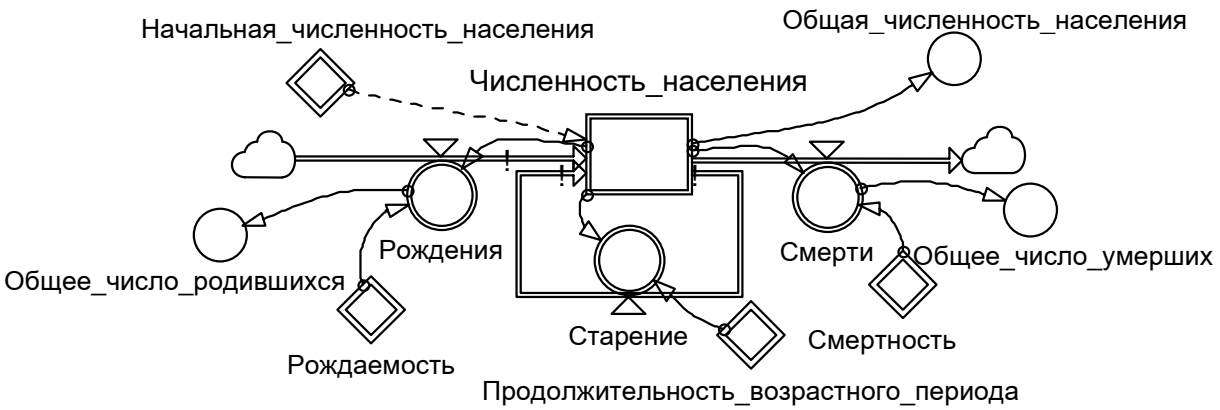


Рисунок 1.18 Потоковая диаграмма для модели "передвижки возрастов"

После определения всех переменных модели, можно увидеть следующую систему уравнений в режиме просмотра уравнений (см. п. 1.3.2):

Диапазоны

```
range  Возраст = Детский, Юношеский, Средний, Пожилой
range  Регион = Север, Юг
```

Уравнения уровней

```
dim    Численность_населения = (A=Возраст,R=Регион)
init   Численность_населения = Начальная_численность_населения
flow   Численность_населения =
       +dt*(ARRSUM(Рождения(*,R))|A=FIRST(Возраст);0)
       +dt*(Старение(A-1,R)|A>FIRST(Возраст);0)
       -dt*(Старение(A,R)|A<LAST(Возраст);0)-dt*Смерти
doc    Численность_населения = "Численность населения по всем
       возрастным группам и регионам"
unit   Численность_населения = чел
```

Уравнения темпов

```
dim    Рождения = (A=Юношеский..Средний,R=Регион)
aux    Рождения = Численность_населения(A,R)*Рождаемость(A)
doc    Рождения = "Число родившихся от родителей, находящихся в
       юношеской и средней возрастной группе по регионам"
unit   Рождения = чел/год

dim    Смерти = (A=Возраст,R=Регион)
aux    Смерти = Численность_населения(A,R)*Смертность(A)
doc    Смерти = "Число умерших по всем возрастным группам и регионам"
unit   Смерти = чел/год

dim    Старение = (A=FIRST(Возраст)..LAST(Возраст)-1, R=Регион)
aux    Старение = Численность_населения(A,R) /
       Продолжительность_возрастного_периода(A)
doc    Старение = "Численность стареющих людей по всем возрастным
       группам и регионам"
unit   Старение = чел/год
```

Дополнительные уравнения

```
aux    Общая_численность_населения = ARRSUM(Численность_населения)
aux    Общее_число_родившихся = ARRSUM(Рождения)
aux    Общее_число_умерших = ARRSUM(Смерти)
```

Константы

```
dim    Начальная_численность_населения = (Возраст,Регион)
const  Начальная_численность_населения = [[1000,800], [950,750],
       [900,700], [400,1000]]
doc    Начальная_численность_населения = "Начальная численность
       населения по всем возрастным группам и регионам"
unit   Начальная_численность_населения = чел

dim    Продолжительность_возрастного_периода =
       (FIRST(Возраст)..LAST(Возраст)-1)
const  Продолжительность_возрастного_периода = [12, 25, 20]
doc    Продолжительность_возрастного_периода = "Продолжительность
       каждого возрастного периода, исключая последний"
unit   Продолжительность_возрастного_периода = год
```

```
dim    Рождаемость = (Юношеский..Средний)
const  Рождаемость = [0.04, 0.01]
doc    Рождаемость = "Коэффициент рождаемости от родителей,
       находящихся в юношеской и средней возрастной группе (одинаков
       для всех регионов)"
unit   Рождаемость = 1/год

dim    Смертность = (Возраст)
const  Смертность = [0.0019, 0.0008, 0.0066, 0.065]
doc    Смертность = "Коэффициент смертности по возрастным группам
       (одинаков для всех регионов)"
unit   Смертность = 1/год
```

В данной системе уравнений используются вектора (массивы) [..,...,...], встроенные функции: ARRSUM (*, R) – сумма элементов массива (по индексу *), FIRST(Возраст) – нижний предел возрастного диапазона, LAST(Возраст) – верхиний предел возрастного диапазона, названия возрастного и регионального диапазонов (Возраст, Регион), переменные индекса (A,R), оператор условного перехода (Выражение1 | Условие; Выражение2), равный значению Выражения1 в случае выполнения Условия, и значению Выражения2 – в противном случае. Более подробную информацию о встроенных функциях можно найти в п. 3.1.

При использовании статических и передаточных объектов, таких как графики, таблицы, ползунки, архив и др. необходимо в соответствующем окне диалога задать свойства этих объектов выбрать те переменные, изменение значений которых будут отражать данные объекты. Например, для вышеуказанной модели Солоу можно построить график зависимости $i(k)$ (при помощи инструмента "Диаграмма рассеяния"), как показано на рис. 1.19

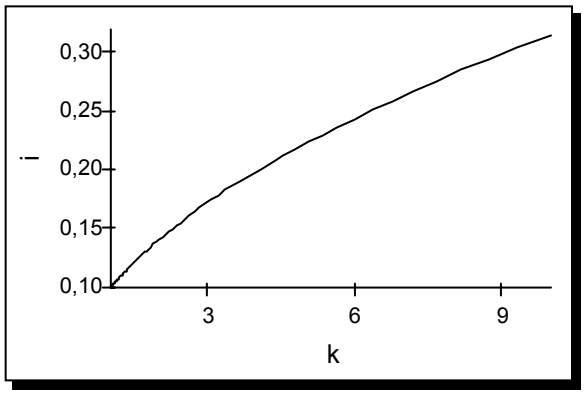


Рисунок 1.19 Зависимость удельных фактических инвестиций от капиталовооруженности в модели Р. Солоу

или вывести в табличном виде текущие значения переменных модели (табл. 1.3).

Таблица 1.3 Зависимость переменных из модели Р. Солоу от времени

Time	k	di	i	f
93	1,04	0,104	0,102	1,02
94	1,04	0,104	0,102	1,02
95	1,04	0,104	0,102	1,02
96	1,04	0,104	0,102	1,02
97	1,03	0,103	0,102	1,02
98	1,03	0,103	0,102	1,02
99	1,03	0,103	0,102	1,02
100	1,03	0,103	0,101	1,01

Кроме того, можно использовать фазовые диаграммы, как, например, в "паутинообразной" модели при демонстрации сходимости к рыночному равновесию (рис. 1.20).

Рисунок 1.20 Фазовая диаграмма для "паутинообразной" модели

Также можно использовать в моделях временные ряды или отдельные значения переменных модели, хранимые в формате MS Excel, при помощи динамического обмена данными с использованием передаточного инструмента DDE (см. п. 1.1.2). Для установки такой связи необходимо:

1. При запущенном приложении Powersim запустить приложение MS EXCEL.
2. Открыть файл с данными или набрать их в новом документе, как, например, в случае с запасами товаров на складе (рис. 1.21). При этом в заголовках строк и столбцов вместо пробелов должны быть символы подчеркивания.

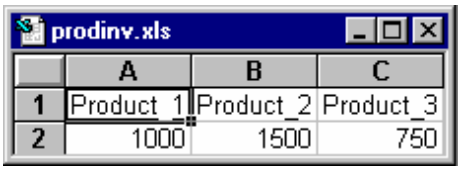


Рисунок 1.21 Таблица импортируемых данных

3. Переключиться (например, при помощи комбинации кнопок Alt+Tab на клавиатуре) в приложение Powersim.
4. Создать при помощи инструмента DDE-объект в любом месте окна модели (см. п.1.4.4).
5. Перетащить и отпустить над выделенным DDE-объектом объекты, соответствующие переменным, предназначенным для динамического обмена, или явно их указать, открыв диалог редактирования свойств DDE-объекта по двойному щелчку на нем левой кнопкой мыши, выделив нужные переменные (Variables) и нажав кнопу "Добавить" (Add->).
6. В открытом окне диалога в выпадающем списке Transfer (Преобразование) выбрать пункт From Application (Данные из другого приложения).
7. Выбрать в выпадающем списке Orientation (Ориентация) пункт Vertical (Данные по столбцам).
8. Выбрать в выпадающем списке Column Separation (Разделитель колонок) тип разделителя колонок, а именно символ табуляции (Tab). В качестве разделителей можно выбирать пробел, символ табуляции, точку с запятой, точку, запятую.
9. Выбрать в выпадающем списке Line Terminator (Разделитель строк) тип разделителя строк, а именно символы CR-LF (Возврат каретки и перевод строки, т.е 13-й и 10-й ASCII-символы).
10. В выпадающем списке Numerical Format (Числовой формат) выбрать пункт Control Panel (Установка как на панели управления).
11. В поле Service (Сервер или приложение) набрать название приложения, т.е. Excel.

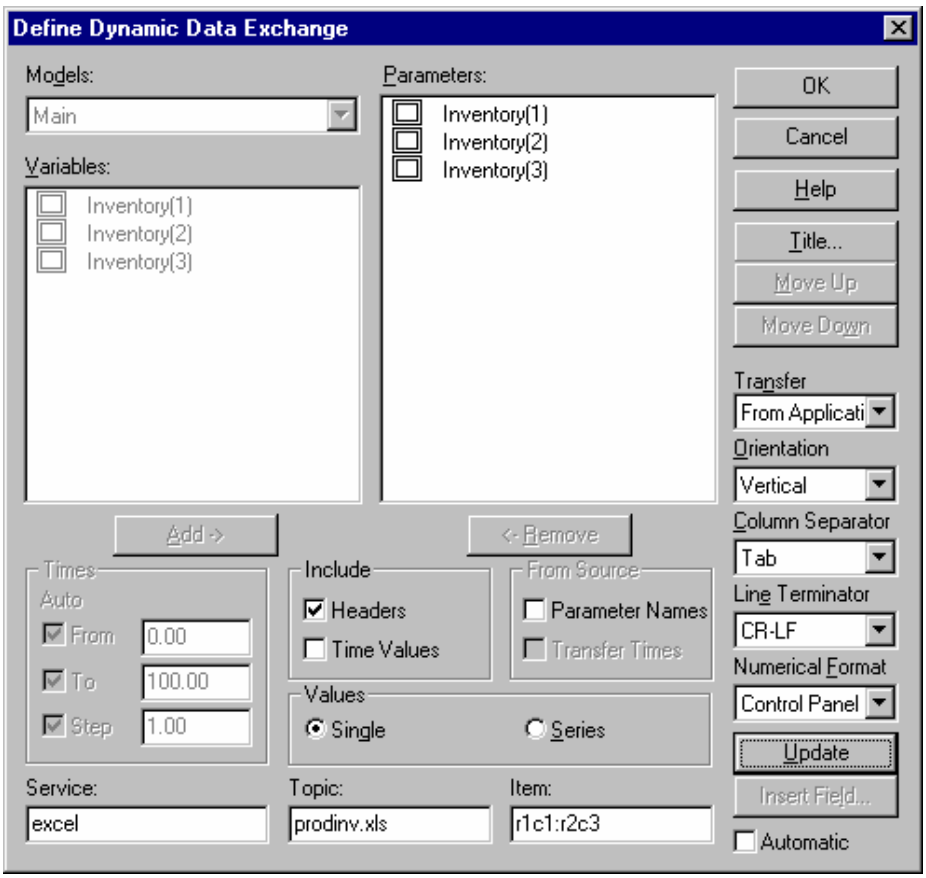


Рисунок 1.22 Окно диалога редактирования свойств DDE-объекта

12. В поле Topic (Название) указать название файла данных, т.е. prodinv.xls.
13. В поле Item (Пункт) ввести диапазон данных в формате R1C1:R2C3 (R – строка, C– столбец), что будет соответствовать получению данный из ячеек A1:C2.
14. Установить опцию Headers (Заголовки), и опцию Single (Одиночное значение), поскольку в колонках таблицы мы указали заголовки и выбираем для обмена всего одно значение каждой из трех компонент A векторной переменной Inventory(A) (запасы товаров на складе). Если при этом будет выбрана опция Parameter Names (Имена параметров), то заголовки переменных будут загружаться из файла Excel. Однако, если они не будут соответствовать названиям связанных с ними переменных модели, то будет выдано сообщение об ошибке. Если отключить опцию Headers, то нужно будет изменить диапазон данных для обмена на R2C1:R2C3, иначе будет выдано сообщение об ошибке.
15. Нажать кнопу Update (Обновление) и закрыть окно диалога.

После создания всех необходимых объектов и определения их свойств можно приступать к проведению имитационных экспериментов.

1.5 Подготовка и проведение имитационных экспериментов

Для подготовки и проведения имитационных экспериментов (имитаций) необходимо:

1. Определить все уравнения для переменных модели.
2. Задать все начальные условия и константы модели.
3. Выбрать период моделирования и размерность единицы времени.
4. Выбрать метод и шаг интегрирования (метод Эйлера, Рунге-Кутта с постоянным или переменным шагом).
5. Выбрать тип имитационного эксперимента (непрерывный, пошаговый, игровой) и провести сам имитационный эксперимент.
6. Путем проведения серии имитационных экспериментов при разных начальных условиях и константах модели провести верификацию, анализ чувствительности модели и обработку результатов имитационных экспериментов.
7. Использовать модель для получения прогнозных оценок относительно динамики интересующих переменных модели и в других целях (например образовательных).

1.5.1 Подготовка имитационного эксперимента

Для проведения имитационных экспериментов необходимо определить все переменные модели, а также задать начальные условия и константы. Это означает, что на структурной (потоковой) диаграмме не должно оставаться никаких пиктограмм, внутри которых были бы вопросительные знаки. Кроме того, никакие динамические объекты для ввода значений переменной (например ползунки) не должны быть соединены со вспомогательными переменными с включенной опцией Allow Input (Разрешить Ввод). Если любое из упомянутых условий не будет выполнено до начала имитации, Powersim выдаст сообщение об ошибке.

Кроме того, важно на данном этапе выбрать общий для всех объектов формат отображения (по умолчанию) числовых данных в меню диалогового окна Options (Настройки), возникающего при выборе одноименной команды из меню Format (Формат). Здесь возможны следующие варианты:

1,234.5	Запятая – разделитель тысяч. Точка – разделитель целой и дробной частей числа.
1.234,5	Точка – разделитель тысяч. Запятая – разделитель целой и дробной частей числа.
1234.5	Нет разделителя тысяч. Точка – разделитель целой и дробной частей числа.
1234,5	Нет разделителя тысяч. Запятая – разделитель целой и дробной частей числа.
С Панели управления	Используются национальные настройки оболочки Windows как для разделителя тысяч, так и для разделителя целой и дробной частей числа. Эти настройки считываются с Панели управления Windows (Язык и стандарты).

Для отдельного объекта формат отображения числовых данных можно менять, редактируя соответствующее свойство этого объекта при помощи диалогового окна Format Number (Формат числа), представленного на рис. 1.23. Данное свойство доступно из диалогового окна Define ... (Определение ...) при нажатии кнопок, имеющих в своем названии слово Format (Формат), или кнопок с названием Axis (Оси координат). При нажатии кнопок последнего типа появляются диалоговые окна, содержащие кнопки с названием Format (Формат).

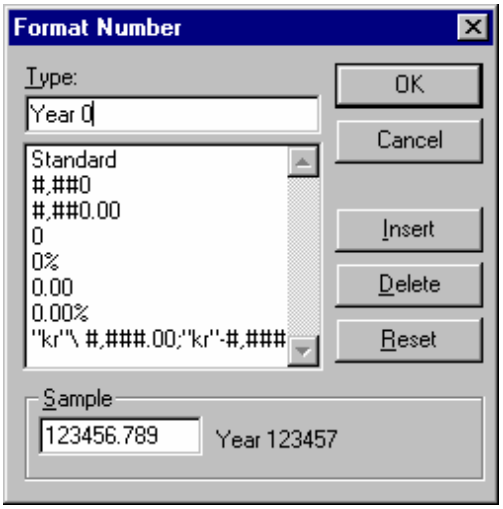


Рисунок 1.23 Окно диалога Format Number (Формат числа)

Такие установки до начала моделирования проводить не обязательно (т.е. не будут выдаваться сообщения об ошибках), но желательно, поскольку это повышает "читаемость" модели (особенно пользователями модели, а не разработчиками).

Выбор периода моделирования и размерности единицы времени

Период моделирования или временной горизонт в моделях Powersim по умолчанию составляет 100 единиц времени. При этом период моделирования начинается в момент времени 0 и заканчивается в момент времени 100. В большинстве случае период моделирования выбирается так, чтобы можно было отследить переход моделируемой системы из одного состояния в другое, а также начальное и конечное состояние. Например, в классической модели Дж. Форрестера "Мировая динамика" период моделирования составляет 200 лет, а начальный и конечный моменты времени равны соответственно 1900 и 2100 лет. За этот период можно отследить не только переходные процессы в моделируемой системе, но начальное и конечное состояния. Установка начального (Start Time) и конечного (Stop Time) моментов времени осуществляется через диалоговое окно Simulation Setup (Установки моделирования), как показано на рис. 1.24.

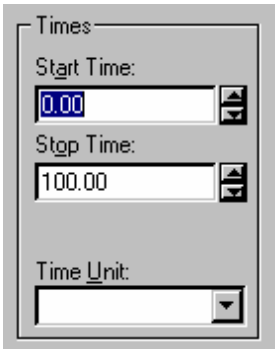


Рисунок 1.24 Часть окна диалога Simulation Setup (Установки моделирования) для выбора начального и конечного моментов интервала моделирования

Выбор размерности единицы времени (час, день, неделя, месяц, год и др.) производится сначала через диалоговое окно Edit Define Unit (Правка единиц размерности) путем добавления новой единицы размерности, а затем – через диалоговое окно Simulation Setup (Установки моделирования) путем выбора соответствующей размерности из выпадающего списка Time Unit (Единица времени). Выбранная единица времени по умолчанию будет отражаться на всех графиках со временем по оси абсцисс.

Выбор метода и шага интегрирования (моделирования)

При выборе метода и шага интегрирования (моделирования) важно правильно оценить соотношение производительности компьютера и точности вычислений. Так, если точность вычислений не важна или модель довольно большая, а вычислительной мощности компьютера недостаточно, то лучше воспользоваться самым простым методом Эйлера, точность которого пропорциональна шагу интегрирования. Следует отметить, что при наличии в системе незатухающих колебаний метод Эйлера может сильно исказить поведение модели (вместо незатухающих получатся возрастающие колебания).

Если требуется более высокая точность вычислений, то следует использовать методы Рунге-Кутта 2-го, 3-го и 4-го порядка с постоянным или переменным шагом. В Powersim переменный шаг интегрирования можно установить только для метода Рунге-Кутта 4-го порядка. Точность этих методов пропорциональна соответственно шагу моделирования в квадрате, кубе и в четвертой степени, а переменный шаг позволяет уменьшать шаг на интервалах, где переменная меняется быстро, и увеличивать его на интервалах, где переменная меняется медленно. Алгоритмы работы указанных методов интегрирования можно пояснить при помощи следующей простой модели:

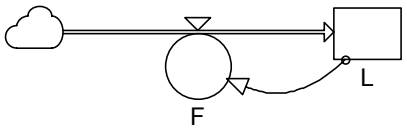


Рисунок 1.25 Потоковая диаграмма для иллюстрации методов интегрирования

описываемой следующим образом:

```
init   L = 0
flow   L = +dt*F
aux    F = f(L)
```

В данной системе уравнений L, F – соответственно значения переменных уровня и темпа в момент времени t, f – функция, определяющая зависимость F от L. Тогда имеем следующие алгоритмы интегрирования:

Метод Эйлера

Шаг 1: Вычисление значения переменной темпа F в момент времени $t = T$

$$F_T = f(L_T),$$

где f – функция, определяющая зависимость F_T от L_T.

Шаг 2: Вычисление значения переменной уровня L в момент времени $t = T + \Delta t$ на основе имеющихся значений переменных уровня и темпа в предыдущий момент времени

$$L_{T+\Delta t} = L_T + \Delta t * F_T,$$

где Δt – шаг интегрирования. Таким образом, в методе Эйлера используется информация о значении переменной темпа (т.е. производной функции L по времени) в одной, а именно, начальной точке временного интервала $[T, T+\Delta t]$.

Метод Рунге-Кутта 2-го порядка с постоянным шагом

Шаг 1: Вычисление значения переменной темпа F в двух точках интервала $[T, T + \Delta t]$ в моменты времени $t = T$ и $t = T + 3/4*\Delta t$

$$F1_T = f(L_T)*\Delta t, \quad F2_{T+3/4*\Delta t} = f(L_T + F1_T)*\Delta t,$$

где f – функция, определяющая зависимость F от L, а Δt – шаг интегрирования.

Шаг 2: Получение оценки 2-го порядка значения переменной уровня L в момент времени $t = T + \Delta t$ на основе средневзвешенного значений $F1$ и $F2$ (временные индексы для простоты записи опущены).

$$L_{T+\Delta t} = L_T + 1/3*F1 + 2/3*F2.$$

Метод Рунге-Кутта 3-го порядка с постоянным шагом

Шаг 1: Вычисление значения переменной темпа F в трех точках интервала $[T,\ T + \Delta t]$ в моменты времени $t = T$, $t = T + 1/2*\Delta t$ и $t = T + 3/4*\Delta t$

$$F1_T = f(L_T)*\Delta t,\quad F2_{T+1/2*\Delta t} = f(L_T + 1/2*F1)*\Delta t,\quad F3_{T+3/4*\Delta t} = f(L_T + 3/4*F2)*\Delta t,$$

где f – функция, определяющая зависимость F от L, а Δt – шаг интегрирования.

Шаг 2: Получение оценки 3-го порядка значения переменной уровня L в момент времени $t = T + \Delta t$ на основе средневзвешенного значений $F1$, $F2$ и $F3$.

$$L_{T+\Delta t} = L_T + 2/9*F1 + 3/9*F2 + 4/9*F3.$$

Метод Рунге-Кутта 4-го порядка с постоянным шагом

Шаг 1: Вычисление значения переменной темпа F в четырех точках интервала $[T,\ T + \Delta t]$, а именно по одному разу в моменты времени $t = T$, $t = T + \Delta t$ и два раза в точке $t = T + 1/2*\Delta t$

$$F1_T = f(L_T)*\Delta t,\quad F2_{T+1/2*\Delta t} = f(L_T + 1/2*F1)*\Delta t,$$

$$F3_{T+1/2*\Delta t} = f(L_T + 1/2*F2)*\Delta t,\quad F4_{T+\Delta t} = f(L_T + F3)*\Delta t,$$

где f – функция, определяющая зависимость F от L, а Δt – шаг интегрирования.

Шаг 2: Получение оценки 4-го порядка значения переменной уровня L в момент времени $t = T + \Delta t$ на основе средневзвешенного значений $F1$, $F2$, $F3$ и $F4$.

$$L_{T+\Delta t} = L_T + 1/6*F1 + 2/6*F2 + 2/6*F3 + 1/6*F4.$$

Метод Рунге-Кутта 4-го порядка с переменным шагом

Шаг 1: Вычисление значения переменной темпа F в четырех точках интервала $[T,\ T + \Delta t]$, а именно по одному разу в моменты времени $t = T$, $t = T + \Delta t$ и два раза в точке $t = T + 1/2*\Delta t$

$$F1_T = f(L_T)*\Delta t,\quad F2_{T+1/2*\Delta t} = f(L_T + 1/2*F1)*\Delta t,$$

$$F3_{T+1/2*\Delta t} = f(L_T + 1/2*F2)*\Delta t,\quad F4_{T+\Delta t} = f(L_T + F3)*\Delta t,$$

где f – функция, определяющая зависимость F от L, а Δt – шаг интегрирования.

Шаг 2: Получение оценки 4-го порядка значения переменной темпа F на основе средневзвешенного значений $F1$, $F2$, $F3$ и $F4$.

$$C4 = 1/6*(F1 + 2*F2 + 2*F3 + F4).$$

Шаг 3: Получение оценки 3-го порядка значения переменной темпа F на основе средневзвешенного значений $F2$ и $F3$.

$$C3 = 1/3*(F2 + 2*F3).$$

Шаг 4: Вычисление ошибки оценок 3-го и 4-го порядка значения переменной темпа F

$$E = ABS(C4 - C3),$$

где ABS – функция, значение которой равно абсолютному значению выражения $C4 - C3$.

Шаг 5: Вычисление пределов приемлемой ошибки

$$A = AbsError + ABS(RelError*C3),$$

где $AbsError$, $RelError$ – соответственно величины абсолютной и относительной ошибок, задаваемые пользователем в диалоговом окне Simulation Setup (Установки моделирования).

Шаг 6: Повторение шагов 1-5 с уменьшающимся шагом интегрирования Δt, если хотя бы для одной из переменных уровня выполнено условие $E > A$. Максимальное число, определяющее количество этапов дробления шага интегрирования, по умолчанию равно 3 и записано в файле POWERSIM.INI в строке VariableStepLimit=3 раздела [SimulationEngine].

Шаг 7: Получение оценки 4-го порядка значения переменной уровня L в момент времени $t = T + \Delta t$.

$$L_{T+\Delta t} = L_T + C4.$$

Постоянный шаг интегрирования Δt выбирается, как правило, достаточно малый для того, чтобы отследить интересующие пользователя самые небольшие изменения переменных в модели. Как уже говорилось выше, с уменьшением шага моделирования увеличивается точность вычислений, но одновременно с этим возрастает и время интегрирования уравнений модели. Как правило, шаг интегрирования должен быть меньше 1/2, но больше 1/5 длительности самой короткой задержки первого порядка, имеющейся в модели.

Определение метода и шага интегрирования, а также величин абсолютной и относительной ошибки (в случае выбора метода Рунге-Кутта 4-го порядка с переменным шагом) осуществляется в диалоговом окне Simulation Setup (Установки моделирования), как показано на рисунке 1.26.

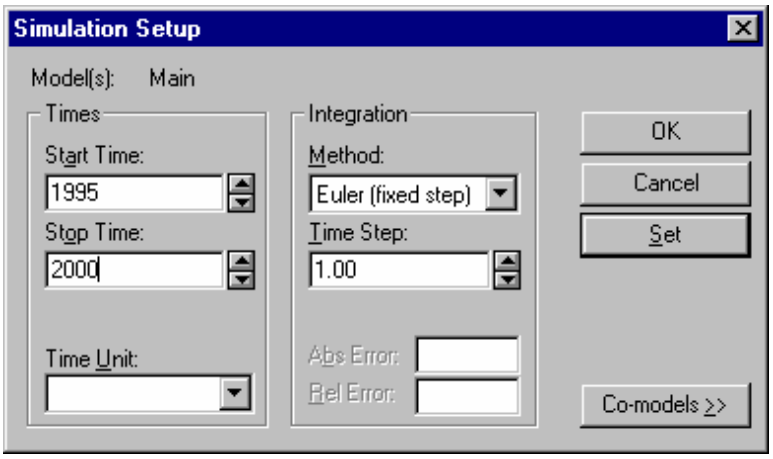


Рисунок 1.26 Окно диалога Simulation Setup (Установки моделирования)

Сравнение результатов применения различных методов и шагов интегрирования можно провести при помощи проведения нескольких имитационных экспериментов и вывода результатов на зависящий от времени график или в таблицу. Для этого указанные объекты имеют специальную опцию Simulation Starts New Generation (Моделирование начала нового поколения) в меню, вызываемом при нажатии кнопки Generations (Поколения) в диалоговом окне определения объекта. Далее необходимо, закрыв соответствующее диалоговое окно, установить новый метод и шаг интегрирования в диалоговом окне Simulation Setup (Установки моделирования) и провести имитацию.

Для одновременного просмотра результатов проведения имитационных экспериментов при использовании разных методов и шагов интегрирования на одном графике или в таблице необходимо установить опцию Add New Generation to Display List (Добавить новое поколение в список отображаемых переменных) в меню, вызываемом при нажатии кнопки Display (Показ) в диалоговом окне определения объекта. При этом каждая новая имитация будет добавлять новый график или колонку таблицы к уже существовавшим ранее.

Следует отметить, что аналогичная процедура используется при проведении анализа чувствительности модели по отношению к изменению экзогенных (т.е. внешних по отношению к модели) параметров (см. п. 1.5.3).

1.5.2 Выбор типа и проведение имитационного эксперимента

В Powersim возможно проводить несколько типов имитационных экспериментов: непрерывный, пошаговый, игровой.

В непрерывном имитационном эксперименте можно изменять установки моделирования, начальные условия и константы модели только в начальный момент времени, в то время как в пошаговом имитационном эксперименте эти изменения можно вносить в начале каждого шага.

Игровой имитационный эксперимент является модификацией пошагового имитационного эксперимента, который можно осуществлять как на локальном компьютере, так и через сеть, однако, в виду своей сложности, он подробно описывается в разделе 1.6.

Для управления имитационными экспериментами служат следующие кнопки панели команд

Запуск имитации	Пошаговая имитация	Вкл./выкл. паузу во время имитации (Пауза)	Остановка имитации

Если при проведении имитационного эксперимента в определенный момент времени или через определенный временной интервал потребуется сделать паузу, то это можно сделать до начала проведения имитационного эксперимента при помощи опций в диалоговом окне Run Setup (Настройки имитации), вызываемом соответствующей командой меню Simulate (Имитация).

Иногда может оказаться полезным наблюдение динамики (при помощи имеющихся средств анимации) в замедленном режиме. Для этого можно в диалоговом окне Run Setup (Настройки имитации) определить задержку (в миллисекундах) при проведении вычислений на каждом шаге. Максимальная величина задержки на каждом шаге не может превышать 100 мс/шаг.

Кроме того, можно проводить за один раз несколько имитаций, предварительно указав их количество в том же диалоговом окне Run Setup (Настройки имитации).

Проведение имитационного эксперимента

Для проведения непрерывного имитационного эксперимента необходимо нажать кнопку . При этом можно сделать паузу вышеописанным способом или вручную, нажав кнопку . Далее можно

1) продолжить имитационный эксперимент в непрерывном режиме, нажав повторно кнопку ;
2) продолжить имитационный эксперимент в пошаговом режиме, нажимая кнопку ;
3) прекратить имитационный эксперимент, нажав кнопку .

Для проведения пошагового имитационного эксперимента необходимо нажать сначала кнопку Пауза и кнопку Запуск имитации, как показано ниже

Затем, нажимая кнопку , нужно провести пошаговый эксперимент, модифицируя во время пауз необходимые константы модели при помощи таких объектов, как число, ползунок, график массива, кнопка, спидометр.

Следует отметить, что изменение констант модели при помощи вышеупомянутых объектов возможно только при нажатых кнопках "Пауза" и "Запуск имитации". В противном случае визуальное изменение свойств этих объектов (например, перемещение ползунка посредством мыши) блокируется, а вокруг объекта появляется рамка, как при выделении объекта в случае копирования.

1.5.3 Верификация, анализ чувствительности модели и обработка результатов имитационных экспериментов

Верификация модели проводится с целью улучшения модели и проверки ее обоснованности (валидности), т.е. того, насколько хорошо полученная модель описывает поведение моделируемой системы. При верификации модели может уточняться структура и параметры модели (например, начальные условия и константы), т.е. производиться калибровка модели.

Анализ чувствительности, в свою очередь, позволяет выявить переменные модели, наиболее чувствительные по отношению к изменению констант и начальных условий, полученных на этапе калибровки модели, а также определить области изменения эндогенных (внутренних) параметров уже верифицированной модели (т.е. линии поведения модели) и исследовать различные режимы поведения модели на устойчивость. При анализе чувствительности структура модели, как правило, не меняется.

Поскольку верификация тесно связана с анализом чувствительности модели и отличается лишь критериями оценки (см. ниже), то рассмотрим более подробно проведения анализа чувствительности модели. Анализ чувствительности модели в Powersim можно проводить как по отношению к изменению начальных условий и констант модели, так и по отношению к изменению метода и шага интегрирования. Кроме того, в особых случаях можно изменять период моделирования.

Как уже говорилось в п. 1.5.1, при анализе чувствительности можно провести несколько имитационных экспериментов и вывести результаты на зависящий от времени график или в таблицу. Для этого указанные объекты (график или таблица) имеют специальную опцию Simulation Starts New Generation (Моделирование начала нового поколения) в меню, вызываемом при нажатии кнопки Generations (Поколения) в диалоговом окне определения объекта. Далее необходимо, закрыв соответствующее диалоговое окно, задать новые начальные условия, константы модели, изменить, если необходимо, метод и шаг интегрирования и провести имитацию.

Для одновременного просмотра результатов проведения нескольких имитационных экспериментов на одном графике или в таблице необходимо установить опцию Add New Generation to Display List (Добавить новое поколение в список отображаемых переменных) в меню, вызываемом при нажатии кнопки Display (Показ) в диалоговом окне определения объекта. При этом каждая новая имитация будет добавлять новый график (рис. 1.27) или колонку (рис. 1.28) таблицы (слева направо) к уже существовавшим ранее, как показано ниже для макроэкономической модели Р. Солоу (см. п. 1.4.6).

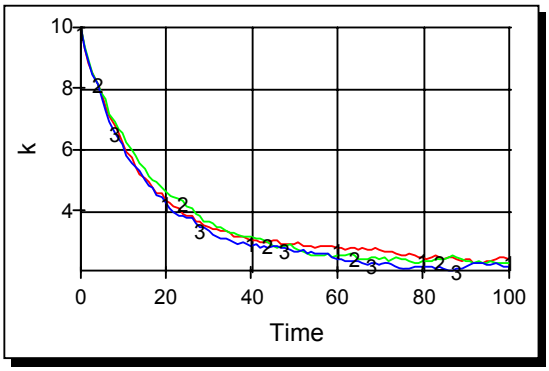


Рисунок 1.27 Графическое представление результатов нескольких имитаций в модели Р. Солоу

Time	k	k	k
93	2.28	2.00	2.42
94	2.26	1.99	2.45
95	2.21	2.03	2.40
96	2.25	2.01	2.34
97	2.26	1.95	2.31
98	2.31	2.02	2.37
99	2.25	2.08	2.40
100	2.25	2.02	2.36

Рисунок 1.28 Табличное представление результатов нескольких имитаций в модели Р. Солоу

Кроме ручного управления проведением серии имитационных экспериментов, когда вручную определяются отдельные константы, значения констант модели можно задавать и автоматически при помощи использования специальных тестовых функций. В этом случае константы становятся переменными, равными инициализирующим функциям INIT(Выражение), в которых выражения представляют собой детерминированные или табличные функции от номера (RUN) и количества имитаций (RUNCOUNT) или являются стохастическим функциями (см. справочник по функциям в гл. 2).

Примеры инициализирующих функций

1) Изменение согласно выбранному функциональному закону. При проведении имитаций переменная A принимает линейно растущие значения 0,25, 0,5, 0,75, 1.

```
aux    A = INIT(RUN/RUNCOUNT)
```

2) Изменение согласно априори выбранному закону, заданному в виде табличной функции. При количестве имитаций, равном 4, переменная A принимает значения 1, 2, 5, 10 при проведении 1-й, 2-й, 3-й и 4-й имитации.

```
aux    A = INIT(GRAPH(4*(RUN-1)/(RUNCOUNT-1),0,1,[1,2,5,10]))
```

3) Изменение согласно выбранному стохастическому закону. При проведении имитаций переменная A принимает случайные значения, равномерно распределенные в интервале [1, 5].

```
aux    A = INIT(RANDOM(1,5))
```

Следует отметить, что серию имитационных экспериментов можно проводить и без использования функции INIT(Выражение), приравняв соответствующую переменную Выражению, зависящему от номера и количества имитаций. Однако в каждом случае при обращении к такой переменной данное Выражение будет пересчитываться, в то время как при использовании функции INIT оно вычисляется один раз и хранится в памяти компьютера в течение всего имитационного эксперимента.

Результаты имитационных экспериментов можно анализировать как в Powersim, так и в других приложениях, таких как Excel, используя механизм динамического обмена данными (DDE) для передачи результатов имитационных экспериментов из Powersim в другие приложения. При этом можно воспользоваться большим набором встроенных в другие приложения функций для проведения статистического анализа (поиск размаха выборки по имитациям, вычисление средних значений переменных и стандартных отклонений по имитациям и др.).

Критерии, используемые при верификации и анализе чувствительности модели

Вернемся к критериям анализа результатов имитационных экспериментов для проведения верификации и анализа чувствительности модели.

В первом случае, в ходе проведения серии имитационных экспериментов случайным или иным способом из всего набора начальных условий и констант, которые можно варьировать, выбираются те, которые позволяют добиться минимального отклонения модельной линии поведения системы от реальной и, таким образом, откалибровать модель. В качестве критерия S, который необходимо минимизировать по всему множеству экзогенных параметров C (констант и начальных условий), например, можно использовать сумму квадратичных отклонений наблюдаемых Y_t и моделируемых $\hat{Y}_t$ значений переменной Y за весь период моделирования:

$$S = \sum_t (Y_t - \hat{Y}_t)^2 \to \min_C .$$

Во втором случае, в качестве одного из критериев чувствительности имитационной модели можно использовать точечную или дуговую эластичность, т.е. процентное изменение

параметра Y (например, переменной уровня, темпа вспомогательной или дополнительной переменной) по отношению к процентному изменению параметра X (например, начального условия или константы модели при проведении серии имитационных экспериментов):

$$E = \frac{\Delta Y(\%)}{\Delta X(\%)},$$

где в случае точечной эластичности

$$\Delta Y(\%) = \frac{|Y(X_{\max}, t_0) - Y(X_{\min}, t_0)|}{Y(X_{\min}, t_0)} 100\%, \quad \Delta X(\%) = \frac{X_{\max} - X_{\min}}{X_{\min}} 100\%,$$

а в случае дуговой эластичности

$$\Delta Y(\%) = \frac{2|Y(X_{\max}, t_0) - Y(X_{\min}, t_0)|}{Y(X_{\max}, t_0) + Y(X_{\min}, t_0)} 100\%, \quad \Delta X(\%) = \frac{2(X_{\max} - X_{\min})}{X_{\max} + X_{\min}} 100\%.$$

При этом если значения $X_{\min}$, $X_{\max}$ задаются вручную, а момент времени $t = t_0$ выбирается из условия:

$$\max_t |Y(X_{\max}, t) - Y(X_{\min}, t)|.$$

При верификации и анализе чувствительности модели можно также использовать и иные критерии (интегральные и др.).

1.5.4 Использование имитационной модели

Далее полученную модель можно использовать как в целях прогнозирования динамики моделируемой системы, так и в иных целях (например, образовательных в качестве локального или сетевого игрового обучающего модуля-симулятора).

Для того чтобы использовать потоковые и причинно-следственный диаграммы, а также графики, таблицы и иные элементы интерфейса Powersim, необходимо выполнить следующие действия:

1. Нажать правую кнопку мыши в том месте, где будет находиться правый верхний угол выделительной рамки.
2. Не отпуская правую кнопку мыши, выделить желаемый фрагмент окна документа.
3. Скопировать его в буфер обмена, используя команду Copy (Копировать) меню Edit (Правка) или кнопку  .
4. Вставить скопированный фрагмент в документ другого приложения (например, MS Word), команду Paste (Вставить) меню Edit (Правка) или кнопку  .

При этом скопированный в буфер обмена фрагмент будет вставлен в документ другого приложения как активно связанный объект (OLE-объект), который, в отличие от DDE-связи, представляет собой не копию выделенного в Powersim фрагмента окна документа, а регулярно обновляемую ссылку на этот фрагмент, представляющий собой Powersim-объект.

При нажатии правой кнопки мыши на вставленном фрагменте появляется всплывающее меню, список команд которого будет дополнен командами Edit (Правка) и Play (Запуск имитации) из подменю Объект Diagram (Объект диаграммы Powersim). Кроме того, двойной щелчок левой кнопкой мыши на вставленном фрагменте диаграммы Powersim приведет к открытию приложения Powersim и соответствующей модели.

1.6 Расширенные возможности Powersim

1.6.1 Игровое моделирование в Powersim

В Powersim также можно проводить игровые имитационные эксперименты с одной или несколькими имитационными моделями, в том числе с использованием локальных и глобальных сетей. Возможно проведение игровых экспериментов типа "человек-машина" и "человек-человек" (игра в команде). При этом при проведении игровых экспериментов всех участников можно разделить на три группы:

1) администратор игры (1 человек);
2) активные (до 10 человек) и эмулируемые компьютером (не ограниченное число) игроки;
3) наблюдатели (не ограниченное число).

При этом администратор определяет полномочия игроков и процесс принятия решений, игроки на каждом шаге моделирования изменяют значения определенных администратором игры параметров, а наблюдатели, выступающие в роли болельщиков, пассивно наблюдают за ходом игры (например, через Интернет). Существует два режима принятия решений (*стратегический* – для апробации решений без их сохранения, *операционный* – для принятия и сохранения окончательного решения). Синхронизация работы всех участников игры осуществляется через локальную или глобальную сеть, как показано на рис. 1.29

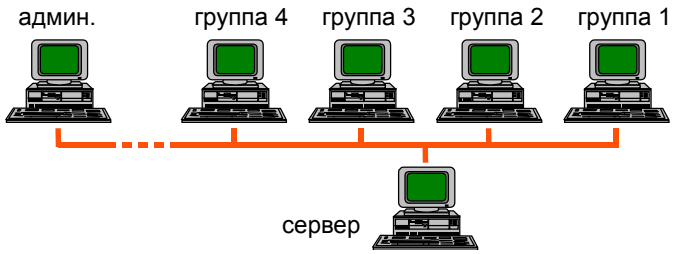


Рисунок 1.29 Синхронизация игрового имитационного эксперимента

при помощи трех групп текстовых файлов:

1) файлы настроек игроков (*.PSN – Powersim Node), размещаемые, как правило, на локальных компьютерах;
2) файлы настроек игры (*.PSS – Powersim Server), размещаемые на сервере (т.е. компьютере, выполняющем роль сетевого сервера);
3) файлы принятия решений (*.TXT), размещаемые на сервере.

Файл настроек игроков содержит уникальный порядковый номер игрока, его персональные игровые настройки и ссылки на файлы настройки игры (например, File1=2,C:\...\MODEL.PSS для игрока под вторым номером). Файл настроек игры содержит название и профиль игры ("человек-машина", "человек-человек"), число игроков, число эмулируемых игроков (моделируются компьютером), расположение и названия файлов принятия решений, период принятия решений, список изменяемых при принятии решений переменных, начальные установки игры. Файл принятия решений имеет следующую структуру (табл. 1.4):

Таблица 1.4 Структура файла принятия решений

Время	**Название переменной**	**Название переменной**	**...**
t1	V11	V21	
t2	V12	V22	...
...	...	...	...

Для полной реинициализации игры с обнулением всех ранее принятых решений файлы принятия решений необходимо удалить с локальных компьютеров, так как в противном случае игра будет продолжена, а не начата сначала.

Кроме того, каждый игрок должен иметь доступ к файлу с моделью-симулятором, который может быть расположен как на локальном компьютере, так и на сервере. Если каждый игрок имеет свой собственный файл с моделью-симулятором [этот файл может быть как файлом главной модели, так и файлом какой-либо комодели (см. п.1.4.5)], то в нем можно определить как одинаковый, так и разный интерфейс для всех игроков.

В первом случае можно говорить ***симметричной игре***, когда процесс принятия решений и интерфейс модели одинаков для всех игроков. Такое возможно при моделировании конкурентного рынка, а также в теории массового обслуживания. В этом случае каждый из игроков управляет какими-либо компонентами векторных параметров, определенных в модели-симуляторе, как, например, в случае с четырьмя игроками, продавцами на рынке, устанавливающими каждый свою цену

```
range  PlayerRange = 1..4
aux    Price(p=PlayerRange)=SELECTDECISION(INDEX(p),Price_Decided(p),
       Price_Assumed(p),Price_Simulated(p),Price_Dummy),
```

где параметры функции SELECTDECISION обозначают следующее (табл. 1.5):

Таблица 1.5 Параметры функции принятия решений SELECTDECISION

INDEX(p)	**Локальный игрок № P**
Price_Decided(p)	Цена, определяемая игроком № P
Price_Assumed(p)	Цена, предполагаемая данным игроком относительно локального игрока № P
Price_Simulated(p)	Цена, определяемая имитационной моделью для игрока № P
Price_Dummy	Цена, определяемая не участвующим игроком

Следует отметить, что в симметричной игре есть глобальная и локальная нумерация игроков. Глобальный номер игрока (G) отражается на панели принятия решений (Game Control, см. рис. 1.31), в то время как локальный номер (L) используется моделью при расчетах. Так, текущий игрок всегда имеет локальный номер, равный 1. Соотношение между локальным и глобальным номером игрока следующее:

```
L = (G - PLAYERNUMBER + GETTOTPLAYERS) MOD GETTOTPLAYERS + 1
G = (L + PLAYERNUMBER - 1) MOD GETTOTPLAYERS,
```

где PLAYERNUMBER – номер текущего игрока из файла его настроек, GETTOTPLAYERS – функция, определяющая общее количество игроков.

Во втором случае речь идет об ***асимметричной игре***, когда процедура принятия решений и интерфейс модели у всех игроков различен. Примером этого может быть игровое моделирование технологических и внутрифирменных процессов, когда за каждым участником игры закреплен свой участок работ (производство, распределение, оптовая продажа готовой продукции и т.д.). В этом случае каждый игрок, как правило, управляет скалярными параметрами, определенными в модели-симуляторе.

```
aux    Price=SELECTDECISION(PLAYERNUMBER,Price_Decided,Price_Assumed,
       Price_Simulated,Price_Dummy),
```

где параметры функции SELECTDECISION обозначают следующее (табл. 1.6):

Таблица 1.6 Параметры функции принятия решений SELECTDECISION

PLAYERNUMBER	Номер данного игрока
Price_Decided	Цена, определяемая данным игроком
Price_Assumed	Цена, предлагаемая другими игроками
Price_Simulated	Цена, определяемая имитационной моделью
Price_Dummy	Цена, определяемая не участвующим игроком

Если модель не поддерживает стратегический режим принятия решений, то значение Price_Assumed может быть любым. Если же в игре все игроки активные, то значение Price_Dummy также может быть любым.

В зависимости от режима принятия решений функция SELECTDECISION может принимать следующие значения (табл. 1.7):

Таблица 1.7 Значение функции принятия решений SELECTDECISION

Игрок/режим	Операционный	Стратегический
Данный игрок	Price_Decided	Price_Decided
Другие игроки	Price_Decided	Price_Assumed
Эмулируемые другие игроки	Price_Simulated	Price_Assumed
Не участвующий игрок	Price_Dummy	Price_Dummy

Во время проведения игрового имитационного эксперимента при загрузке игроком своей локальной модели-симулятора, вначале считываются файлы настроек игры и инициализируются файлы принятия решений. При этом возникает диалоговое окно Select game to be played (Выбор профиля игры), показанное на рис. 1.30.

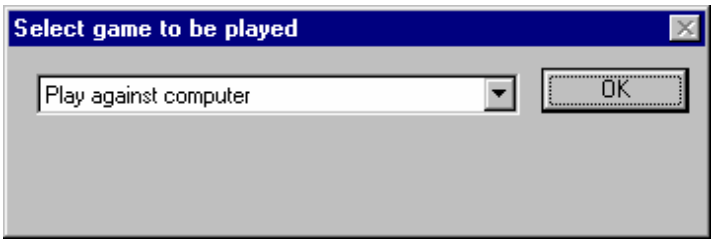


Рисунок 1.30 Окно диалога Select game to be played (Выбор профиля игры)

Затем при запуске моделирования при помощи команды Run (Имитация) считываются файлы принятия решений и игрокам предлагаются на выбор следующие действия: Try Decision (Испытать решение) или Revert to Game (Отменить решение) в стратегическом режиме и Accept Decision (Принять решение) в операционном режиме при помощи панели принятия решений (Game Control), изображенной на рис. 1.31.

Рисунок 1.31 Панель принятия решений

Следует отметить, при инициализации панели принятия решений игроку задается вопрос "Желает ли он присоединиться предыдущему варианту игры и принятым до этого решениям?". В случае утвердительного ответа на поставленный вопрос, компьютер учитывает ранее принятые решения, считанные из файла принятия решений. В противном

случае ранее принятые решения не учитываются. Напомним, что настройки процесса принятия решений находятся в файлах настроек игры.

Для того чтобы все вышеперечисленные возможности принятия решений были доступны игроку, в его файле с имитационной моделью должен присутствовать объект Network Game (Сетевая игра)  , который создается стандартным образом (см. п. 1.4.4). Далее необходимо определить свойства этого объекта, для чего необходимо проделать следующие действия (в случае создания новой сетевой игры):

1. Дважды щелкнуть по объекту Сетевая игра левой кнопкой мыши. При этом появится диалоговое окно Define Network Game (Определение Сетевой игры), как показано на рис. 1.32.

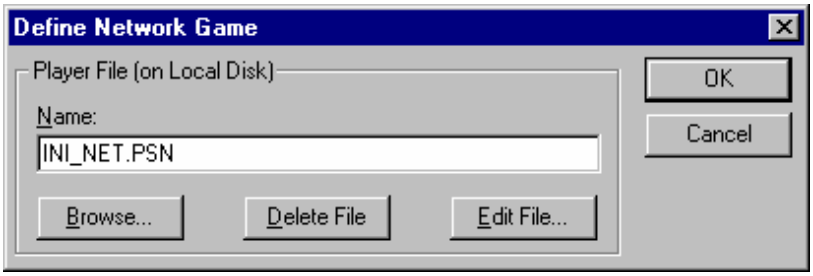


Рисунок 1.32 Окно диалога Define Network Game (Определение Сетевой игры)

2. Ввести название файла настроек игрока в поле Name (Название), после чего нажать кнопку Edit File... (Правка файла...). Это приведет к появлению диалогового окна Edit Player File (Правка файла настроек игрока), как показано на рис. 1.33.

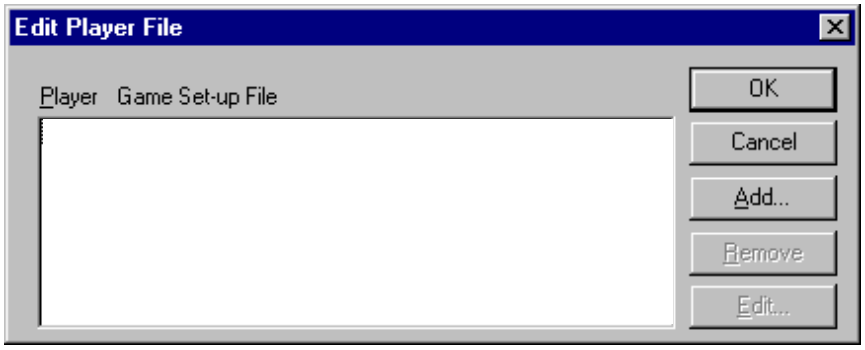


Рисунок 1.33 Окно диалога Edit Player File (Правка файла настроек игрока)

3. Нажать кнопку Add... (Добавить...) Это приведет к появлению диалогового окна Game File and Player Number (Файл настроек игры и номер игрока), как показано на рис. 1.34.

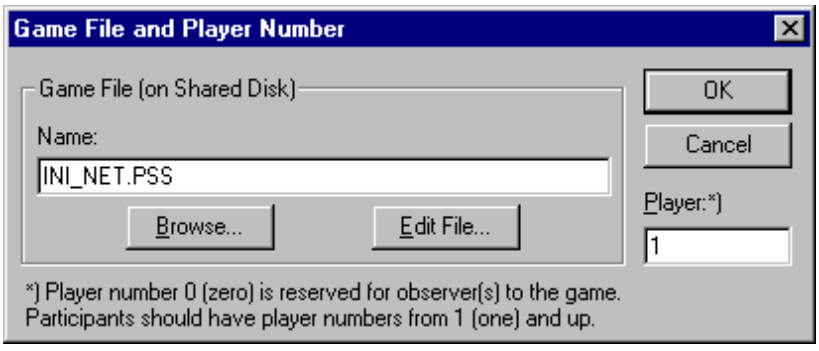


Рисунок 1.34 Окно диалога Game File and Player Number (Файл настроек игры и номер игрока)

4. Ввести название файла настроек игры Name (Название), номер игрока (например 1) в поле Player (Игрок) и нажать кнопку Edit File... (Правка файла...). При этом откроется диалоговое окно Edit Game File (Правка файла настроек игры), рис. 1.35.

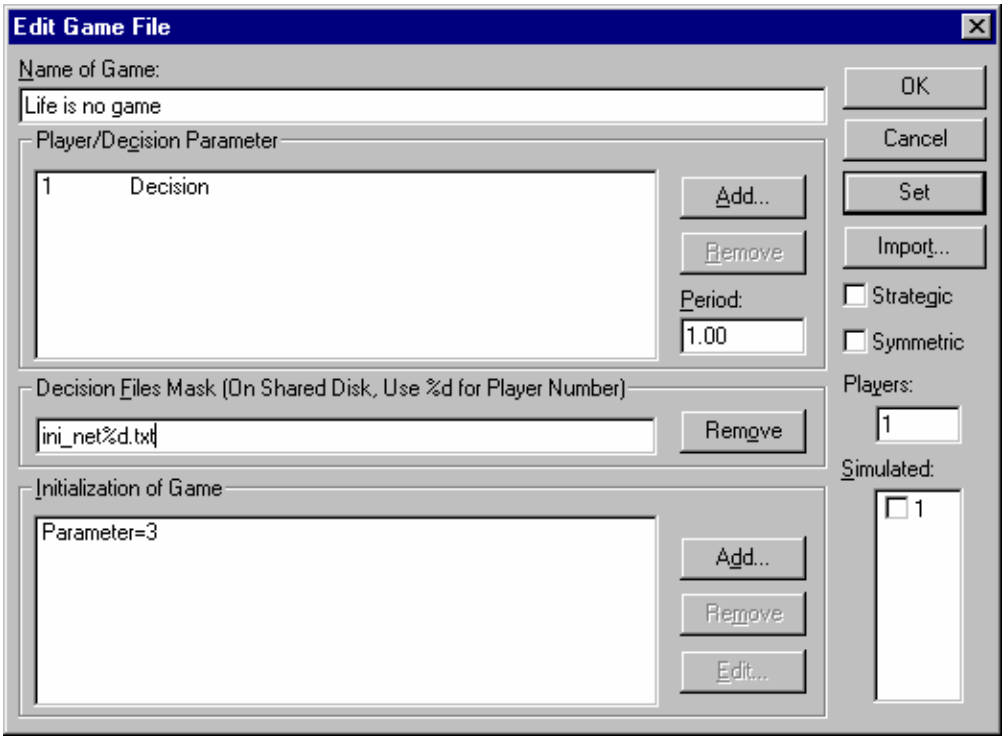


Рисунок 1.35 Окно диалога Edit Game File (Правка файла настроек игры)

5. Ввести название профиля игры в поле Name of Game (Название игры) и название файла принятия решений в поле Decision Files Mask (Маска файлов принятия решений). Также можно импортировать имеющиеся профили игры из сохраненных ранее файлов настроек игры, нажав кнопку Import (Импорт).
6. Определить период принятия решений в поле Period (Период), а также выбрать тип игры (симметричный или несимметричный) и включить, если необходимо, возможность принятия решений в стратегическом режиме.
7. Ввести общее число игроков в игре в поле Players (Игроки), нажав после этого кнопу Set (Установить). Отметить галочками в поле Simulated (Эмулируемые) тех игроков, которые будут моделироваться компьютером для данного игрока.
8. Добавить параметры (как правило, константы) и их начальные значения в поле Initialization of Game (Инициализация игры), а также переменные, которые будут изменяться в процессе принятия решений в поле Player/Decision Parameter (Игрок/Параметр решения), нажав соответствующую кнопку Add... (Добавить...). При этом, например, в последнем случае появится диалоговое окно Decision Parameter (Параметр решения), представленное на рис. 1.36.

Рисунок 1.36 Окно диалога Decision Parameter (Параметр решения)

9. В данном диалоговом окне следует выбрать номер игрока и параметр из списка доступных параметров и нажать кнопку OK (Утвердить).
10. Утвердить все сделанные изменения, нажимая в окнах соответствующих диалогов кнопку OK.

Далее всем параметрам модели, которые изменяются в ходе принятия решений, необходимо сопоставить активные элементы управления (например, объект "Число" или "Ползунок") и провести имитационный эксперимент, нажав кнопку Run (Имитация). При этом у каждого игрока экзогенные параметры модели-симулятора будут инициализироваться теми значениями, которые были определены ранее в поле Initialization of Game (Инициализация игры). Пример игровой модели см. в приложении 7.

1.6.2 Управление Powersim из других приложений при помощи динамического обмена данными (*DDE*)

В качестве примера рассмотрим работу Powersim под управлением такими популярными приложениями, как MS Excel (5.0 и др.) и Lotus 1-2-3. Такое управление осуществляется при помощи макросов, реализующих механизм динамического обмена данными (DDE). Рассмотрим более подробно случай совместной работы Excel и Powersim.

Передача данных из MS Excel в Powersim

При помощи данного макроса реализуется передача данных из Excel в документ Powersim с названием PSDoc$ для инициализации переменной PSVar$ значением Value. До запуска данного макроса необходимо запустить Powersim.

```
Sub SetPSValue(PSDoc$, PSVar$, Value)
On Error GoTo Err_Label
    Ch = DDEInitiate("Powersim", PSDoc$)
    DDEPoke Ch, PSVar$, Value
    DDETerminate Ch
Exit Sub
Err_Label:
    DDETerminate Ch
    MsgBox "Не могу присвоить значение переменной Powersim"+PSVar$
End Sub
```

Следует отметить, что в качестве третьего аргумента в подпрограмме DDEPoke ожидается ссылка на ячейку таблицы Excel. Так, нижеследующий макрос инициализирует переменную Price (Цена) в модели рыночного равновесия MarketEq.SIM значением, записанным в ячейке A1 таблицы MS Excel, обозначаемой в Powersim как Cells(1,1).

```
Sub SetPS()
    SetPSValue "C:\Powersim\Mod\MarketEq.SIM", "Price", Cells(1,1)
End Sub
```

При этом предполагается, что модель находится в папке "C:\Powersim\Mod\", а ячейка A1 таблицы Excel – на активном листе. Если последнее предположение не выполнено, то необходимо в макросе SetPS явно указать название листа таблицы Excel и ячейки, т.е. заменить Cells(1,1) на Worksheets("Sheet1").Range("A1").

Передача данных из Powersim в Excel

При помощи данного макроса реализуется передача данных в Excel из документа Powersim PSDoc$ от переменной PSVar$, значение которой присваивается значению функции GetPSValue. Перед запуском данного макроса необходимо запустить Powersim.

```
Function GetPSValue(PSDoc$, PSVar$)
On Error GoTo Err_Label
    Ch = DDEInitiate("Powersim", PSDoc$)
    returnlist = DDERequest(Ch, PSVar$)
    GetPSValue = returnlist(1)
    DDETerminate Ch
Exit Function
```

```
Err_Label:
    DDETerminate Ch
    MsgBox "Невозможно получить значение переменной Powersim"+PSVar$
    GetPSValue = 0
End Function
```

Следует отметить, что функция DDERequest возвращает массив Excel, в то время как следующая функция returnlist присваивает значению функции GetPSValue значение первого элемента полученного массива. Функцию GetPSValue можно использовать, например, для инициализации ячейки A1 таблицы Excel, определяемой в Powersim как Cells(1, 1), значением

```
Sub GetPS()
    Cells(1,1) = GetPSValue("C:\Powersim\Mod\MarketEq.SIM", "Price")
End Sub
```

Передача текущих значений переменной A (скалярной) возможна также и без использования макроса при загруженной модели Powersim (например, с названием M.SIM). Для этого в соответствующую ячейку таблицы Excel необходимо записать следующую формулу:

```
=Powersim|M.SIM!A
```

Управление приложением Powersim из Excel

При помощи данного макроса реализуется запуск приложения Powersim из Excel, загрузка и проведение имитационного эксперимента для модели, хранящейся в документе PSDoc$.

```
Sub RunPSDoc(PSDoc$)
On Error GoTo Err_Label
    Ch = DDEInitiate("Powersim", "System")
    DDEExecute Ch, "[Open(" + Chr$(34) + PSDoc$ + Chr$(34) + ")]"
    DDEExecute Ch, "[Run]"
    DDETerminate Ch
Exit Sub
Err_Label:
    DDETerminate Ch
    MsgBox "Не могу открыть и провести имитацию для модели Powersim"
    + PSDoc$
End Sub
```

Отметим, что вместо команды DDEExecute Ch,"[Open("+Chr$(34)+PSDoc$+Chr$(34) +")]" можно также использовать команду DDEExecute Ch,"[Open("""+ PSDoc$+""")]". Данный макрос можно использовать для загрузки модели MarketEq.SIM и проведения с ней имитационного эксперимента.

```
Sub RunPSMarketEq()
    RunPSDoc("C:\\Powersim\\Mod\\MarketEq.SIM")
End Sub
```

Сложный пример управления приложением Powersim из Excel

При помощи следующего макроса реализуются следующие действия:

1. Загружается потоковая диаграмма Powersim из документа с названием M.SIM.
2. Проводится одношаговая имитация.
3. Ожидается, когда завершится одношаговая имитация или когда завершится интервал моделирования. В последнем случае производится переход к шагу 8.
4. Из Powersim в Excel передается значение переменной A.

5. Проводятся некоторые вычисления в Excel, результаты которых записываются в ячейку A1.
6. Переменная B из модели Powersim инициализируется значением из ячейки A1 таблицы Excel.
7. Производится переход к шагу 2.
8. Завершается работа.

При этом макрос будет выглядеть следующим образом:

```
Sub InteractPS()
Rem Шаг 1
    Ch = DDEInitiate("Powersim", "M.SIM")
Rem Шаг 2
NextStep:
    DDEExecute Ch, "[RUN.STEP]"
Rem Шаг 3
    While GetValueOf(Ch, "Control.Pause") <> 1
       If GetValueOf(Ch, "Control.Play") = 0 Then
          GoTo Done
       End If
    Wend
Rem Шаг 4
    A = GetValueOf(Ch, "A")
Rem Шаг 5 (Умножение значения переменной A на 10)
    Cells(1,1).Value = A*10
Rem Шаг 6
    SetValueOf Ch, "B", Cells(1,1)
Rem Шаг 7
    GoTo NextStep
Done:
    DDETerminate Ch
End Sub
```

В двух вышеперечисленных примерах управления приложением Powersim использовалась подпрограмма DDEExecute, выполнявшая определенные действия над загруженной моделью Ch, в соответствии с командами, указанными после нее, например "[OPEN(M.SIM)][RUN][QUIT]". Кавычки внутри команд используются, только если параметры команд (строковые переменные) помимо букв и цифр содержат другие символы, а также символы "_", "$", ".".

Перечислим основные типы команд, которые используются для управления приложением Powersim из других приложений.

Команды для управления приложением:
APP.MAXIMIZE() – развернуть окно приложения
APP.MINIMIZE() – свернуть окно приложения
APP.MOVE(X,Y) – переместить окно приложения в заданную позицию
APP.RESTORE() – восстановить окно приложения
APP.SIZE(Ширина, Высота) – изменить размер окна приложения
QUIT() – закрыть приложение

Команды для управления документами:
FILE.CLOSE(True/False) – закрыть все активные документы
NEW() – создать новый документ
OPEN(Имя файла) – открыть существующий файл
PAUSE(True/False) – включить/выключить паузу

RUN() – запустить имитацию
RUN.STEP(Шаг) – шаг моделирования
SAVE() – сохранить активный документ
SAVE.AS(Новое имя документа) – сохранить активный документ под другим именем
STOP() – остановить имитацию

Команды для управления окном документа:
ACTIVATE(Имя окна) – активировать окно документа
CLOSE(True/False) – закрыть активное окно
CLOSE.ALL() – закрыть все окна
NEW.WINDOW() – создать новое окно для активного документа
WINDOW.MAXIMIZE(Имя окна) – развернуть активное или указанное по имени окно
WINDOW.MINIMIZE(Имя окна) – свернуть активное или указанное по имени окно
WINDOW.MOVE(X, Y, Имя окна) – переместить активное или указанное по имени окно
WINDOW.RESTORE(Имя окна) – восстановить активное или указанное по имени окно
WINDOW.SIZE(Ширина, Высота, Имя окна) – изменить размер активного или указанного по имени окна

Следует отметить, что в вышеперечисленных командах значения координат X, Y, ширина и высота окна указываются в точках (1/72 дюйма), а имя окна указывается в кавычках. Если же имя окна не указано, то действия выполняются по отношению к активному окну. Кроме того, используемый в некоторых командах параметр True/False должен принимать только одно значение (True или False).

Глава 2
Справочник по функциям Powersim

2.1 Структура описания функций

Список функций содержит унарные и бинарные операторы, а также нормальные функции с нулевым или большим количеством параметров.

2.1.1 Унарные операторы

В табл. 2.1 приведены все унарные операторы с информацией об их положении относительно операнда (до или после).

Таблица 2.1 Таблица унарных операторов

Оператор	Положение	Назначение
!	после	факториал
%	после	процент
+	до	унарный плюс
–	до	унарный минус
NOT	до	логическое НЕ

Унарные операторы имеют более высокий приоритет по сравнению с бинарными операторами. Операторы, помещаемые после операнда, имеют самый высокий приоритет при выполнении.

2.1.2 Бинарные операторы

Двоичные операторы обрабатывают два операнда, один до и один после оператора. Когда в выражении происходит смешивание операторов, связи между оператором и операндом (аргументом) могут становиться неоднозначными, как в следующем примере:

10 + 3 * 2.

Что является результатом: 26 или 16? Ответ на данный вопрос зависит от относительного приоритета операций сложения и умножения, т.е. от того, какой из операторов является "самым сильным" в борьбе за операнды (аргументы).

Обычно постулируется, в том числе и в POWERSIM, что умножение выполняется до сложения, т.е. умножение обладает более высоким приоритетом по сравнению со сложением. Это означает, что вышеупомянутое выражение может быть переписано как:

10 + (3 * 2),

что в результате равно 16.

Круглые скобки можно всегда использовать для прояснения порядка вычислений или изменения порядка вычисления выражения, как, например, в случае:

(10 + 3) * 2.

Для некоторых операторов важно знать, с какого конца начать вычисление выражения, возводящего в степень тот же самый оператор несколько раз на том же самом уровне (т.е. без круглых скобок). Для операторов, подобных сложению и умножению, порядок вычисления не имеет значения, за исключением перемножения матриц. Однако, это не так в случае операторов деления, вычитания и оператора возведения в степень. Выражения ниже поясняют эти случаи:

1 / (2 / 3) = 3 / 2,
(1 / 2) / 3 = 1 / 6,
(1 - 2) - 3 = - 4,
1 - (2 - 3) = 2,
4 ^ (3 ^ 2) = 262144,
(4 ^ 3) ^ 2 = 4096.

Порядок, который нужно использовать, когда никакие круглые скобки не присутствуют, определяется ассоциативностью оператора, который может быть или слева, или справа. Все операторы, за исключением оператора возведения в степень (^), ассоциативны слева, т.е. они группируются слева направо.

В табл. 2.2 бинарных операторов приводятся все бинарные операторы, их приоритет и ассоциативность.

Таблица 2.2 Таблица бинарных операторов

Оператор	Приоритет	Ассоциативность	Назначение
^	7	справа	возведение в степень
*	6	слева	умножение
/	6	слева	деление
MOD	6	слева	остаточный член от деления
+	5	слева	сложение
–	5	слева	вычитание
<	4	слева	меньше
< =	4	слева	меньше или равно
>	4	слева	больше
> =	4	слева	больше или равно
=	3	слева	равно
< >	3	слева	не равно
AND	2	слева	логическое И
XOR	1	слева	логическое исключающее ИЛИ
OR	1	слева	логическое ИЛИ

2.1.3 Руководящие принципы описания функций

Каждая функция описывается согласно следующим руководящим принципам (табл. 2.3):

Таблица 2.3 Структура описания встроенных в Powersim функций

Структура	Описание
Синтаксис	описание функции и параметров
Вход	определение каждого входного параметра
Выход	определение каждого выходного параметра
Результат	результат действия функции
Замечание	специальные случаи, побочные эффекты и т.д.
Пример	примеры использования функции
График	схематическое изображение функции или график функции
Уравнения	уравнения, которые можно использовать для реализации функции
Демо-версия	демонстрационные модели, использующие функцию
Группа(ы)	группа(ы), к которой принадлежит данная функция
Родственные функции	список похожих по структуре и результатам исполнения функций

В описании синтаксиса функций используются следующие обозначения:

Название(...)	название произвольного массива.
Название(N)	название вектора с N элементами.
Название(M, N)	название массива с N строками и M столбцами.
[, Название = V]	название произвольного параметра, равного по умолчанию величине V.
X1, ... XN	функция N переменных.

Для логических величин при обозначении ненулевых и нулевых значений используются слова True (ИСТИНА) и False (ЛОЖЬ).*

2.1.4 Входные и выходные параметры функций и их типы

Аргументы функций называются параметрами (или операндами в случае таких операторов, как (+) или (*) и др.). Обычно параметры можно определить как выражения, в которые входят функции, операторы, круглые скобки, и т.д. Однако некоторые параметры определяются как выходные параметры, т.е. параметры, которые изменяются функцией. Такие параметры следует определять или как переменные уровня или как константы.** Кроме того, функция вычисляет и возвращает свое значение (см. ниже возвращаемые значения).

В описании функции входные параметры могут быть отнесены к одному из следующих типов:

1) начальные параметры,
2) вычисляемые параметры,
3) произвольные параметры,
4) параметры с запаздыванием.

Кроме того, в зависимости от выбранной функции параметры могут быть скалярами (числами) или массивами.

Начальные параметры

Некоторым функциям, например INIT, необходимы параметры, которые вычисляются только во время инициализации моделирования. Результат функции будет зависеть от начальных значений таких параметров, причем любые изменения параметров игнорируются после инициализации моделирования.

Входные связи, которые используются только как начальные параметры, рассматриваются как инициализирующие связи, т.е. они будут обозначены пунктирными линиями, как только Powersim обнаружит, что они идут к начальным параметрам (см. рис. 2.1).***

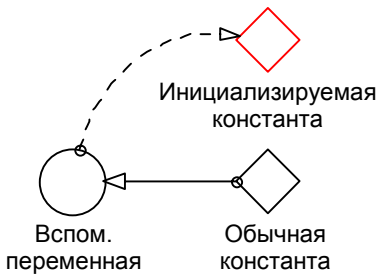


Рисунок 2.1 Инициализирующие связи и переменные

* Параметры можно определять как выражения, если они не используются в качестве выходных параметров, которые должны быть или уровнями или константами.

** Значения выходных параметров, определяемые функцией, присваиваются данным параметрам в конце шага моделирования.

*** Начальные параметры вычисляются всегда.

Вычисляемые параметры

Параметры обычно вычисляются до вызова функции. Однако есть некоторые функции, которые условно вычисляют параметры в зависимости от значения других параметров, текущего времени, и т.д. Функция IF (ЕСЛИ) – наиболее важный пример функции, использующей вычисляемые параметры. Эта функция требует три параметра, где первый – условие, используемое для выбора одного из двух остающихся параметров в качестве результата функции, как, например, в случае:

```
IF(Условие, Значение1_если_условие_истинно, Значение2_если_условие_
ложно)
```

Эта функция работает следующим образом:

1. Проверяется Условие и результат проверки передается с выражениями "Значение1_если_условие_истинно" и "Значение2_если_условие_ложно" на вход IF-функции.
2. IF-функция проверяет значение Условия, и выбирает одно из выражений "Значение1_если_условие_истинно" или "Значение2_если_условие_ложно". Первое выбирается, если Условие выполнено; последнее – в противном случае (если условие не выполнено).
3. IF-функция вычисляет выбранное выражение и использует значение этого выражения как значение функции.

Произвольные параметры

В отдельных функциях присутствуют произвольные параметры, т.е. параметры, которые можно не учитывать при использовании функции. Если произвольный параметр не определен явно пользователем, то он принимает значение по умолчанию, которое затем и используется данной функцией. Произвольные параметры можно не определять только справа налево в списке параметров, т.е. нельзя задать произвольный последующий параметр, не определив предыдущего.

Например, функция DELAYINF использует параметры Вход, Время_запаздывания, Порядок_задержки и Начальное_значение. Параметры Порядок_задержки и Начальное_значение являются произвольными. Следовательно, данную функцию можно вызвать тремя следующими способами:

```
DELAYINF(Вход, Время_запаздывания, Порядок_задержки, Начальное_
значение)
DELAYINF(Вход, Время_запаздывания, Порядок_задержки)
DELAYINF(Вход, Время_запаздывания)
```

Однако невозможно задать только Начальное_значение, не определив Порядок_задержки.

Параметры с запаздыванием

Первый параметр Функции задержки – параметр с запаздыванием. Этот параметр используется как входящий поток во внутренний уровень функции. Это означает, что значение параметра влияет на результат функции не непосредственно, а косвенно через внутренний уровень. Как следствие, такие параметры могут принимать участие в круговом определении модели точно так же, как если бы данная функция была задана в виде явной структуры, состоящей из уровней и темпов. Например, рис. 2.2 демонстрирует структуру информационной задержки 1-го порядка.

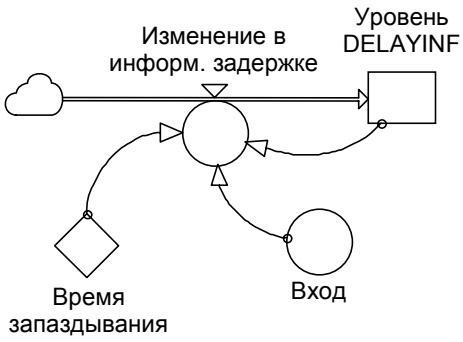


Рисунок 2.2 Структура информационной задержки первого порядка

В этой структуре можно было бы добавить связь, идущую от переменной Уровень_ DELAYINF к вспомогательной переменной Вход, даже если это привело бы к круговому определению модели. Это возможно, так как переменная Уровень_DELAYINF является частью цикла. Рисунок 2.3 иллюстрирует использование функции DELAYINF.

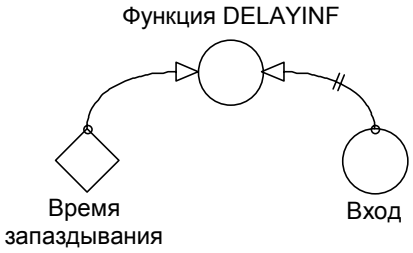


Рисунок 2.3 Использование функции DELAYINF

Здесь также есть возможность добавить связь, идущую от переменной Функция_ DELAYINF к переменной Вход, так как одна из связей в создаваемом цикле является связью с запаздыванием.

Данная особенность полезна во многих случаях, поскольку функции задержки часто участвуют в круговых связях. Явное использование параметров задержки также упрощает график для понимания, так как запаздывание информации явно описывается на графике сдвоенной линией, пересекающей стрелку связи. Отметим также, что значок функции содержит символ, отражающий наличие внутреннего уровня в Функции_DELAYINF*, как показано на рис. 2.4.

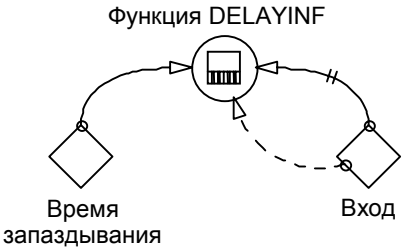


Рисунок 2.4 Пиктограмма, отражающая наличие внутреннего уровня в функции информационной задержки DELAYINF

* Если функция задержки инициализируется входным параметром, то, в дополнение к связи с запаздыванием, следует обеспечить инициализирующую связь. Это обусловлено тем, что при инициализации функция непосредственно использует инициализирующий параметр (без задержки). Если не задано ни одного инициализирующего параметра, то начальное значение по умолчанию вычисляется на основе входного параметра. Это изменение точно соответствует изменению, которое следует внести в вариант диаграммы для той же самой модели, добавляя инициализирующую связь, идущую от переменной Вход к переменной Уровень_DELAYINF (см. рис. 2.4).

Описав вышеуказанные типы входных параметров, следует отметить, что некоторые функции используют переменное число параметров (не следует путать с произвольными параметрами). Так, функции MIN и MAX, могут использовать от 1 до 255 параметров. Например, возможны все последующие типы записей (при надлежащем определении параметров A, B, и S):

```
MAX(1,3,5,2,3,1)
MAX(A, B)
MAX(3, A, 2, B, S(1))
```

Возвращаемые значения

Функции обычно возвращают скалярные значения, за исключением нескольких функций для работы с массивами, которые возвращают массивы. Возвращаемое значение скалярной функции можно использовать в выражении в любом месте, где может использоваться буквенный скаляр. Соответственно, возвращаемое значение функции для работы с массивами можно использовать в любом месте, где может использоваться буквенный массив.

Недопустимые параметры функции

Во время моделирования функция может получать параметры, при которых ее возвращаемое значение становится неопределенным или может выходить за пределы принятого числового диапазона (переполнение). Подобные ошибки не останавливают моделирование, но приводят к специальным аномальным числам, таким как бесконечности ($+\infty$ и $-\infty$) или к выражениям вида "не число". В отчетах такие значения отображаются соответственно как +?, -? и =?.

Например, выражение 10/0 на выходе даст неопределенность, что будет отражено в отчете как +?. Аномальные числа трактуются по-своему каждой функцией, если они используются как аргументы функции.

2.2 Основные группы встроенных в Powersim функций

В данном разделе дается краткий обзор всех функций, имеющихся в Powersim с необходимыми комментариями. В следующей главе каждая функция описывается подробно по форме, указанной в п. 2.1.3. Все функции условно можно разделить на 17 групп, причем тем или иным функциям или группам функций соответствуют особые пиктограммы, как показано ниже.*

1. *Функции для работы с массивами* – функции, в которых используется один или большее количество аргументов, являющихся массивами.
2. *Встроенные функции* – функции, являющиеся частью языка моделирования Powersim.
3. *Комплексные функции* – функции для работы с комплексными числами.
4. *Условные функции* – функции, работа которых зависит от условий, входящих в функции в качестве дополнительных аргументов.

5. *Функции управления* – функции, используемые для управления процессом моделирования.
6. *Функции преобразования типа* – функции, преобразующие значения из одного типа в другой.

7. *Функции задержки (запаздывания)* – функции с замедленным откликом на изменение аргументов.
8. *Финансовые функции* – функции, предназначенные для вычисления текущей и будущей ценности, а так же размера частичных платежей.

9. *Графические функции* – функции, в которых зависимость между значением функции и значениями ее аргументов задается в табличном или графическом виде.

10. *Функции с памятью* – функции, в которых для вычисления текущих значений функции используются прошлые значения аргументов.
11. *Логические (булевы) функции* – функции, возвращающие логическое значение ("Истина" или "Ложь").
12. *Математические функции* – функции для всех видов математических вычислений.

13. *Смешанные функции* – функции, которые хорошо не вписываются в другие группы.

14. *Случайные (стохастические) функции* – функции, значения аргументов которых генерируются генератором случайных чисел по определенному закону распределения.
15. *Статистические функции* – функции для проведения вычислений с аргументами, представляющими собой выборки на множестве данных.

16. *Функции, зависящие от времени* – функции, в которых либо значение функции, либо аргументы имеют размерность единицы времени моделирования.
17. *Тригонометрические функции* – функции для выполнения тригонометрических вычислений.

---

* Некоторые функции присутствуют более, чем в одной группе. Это сделано для того, чтобы упростить правильный выбор группы при поиске функции, например, в списке функций диалогового окна Define Variable (Определение переменной).

2.2.1 Функции для работы с массивами

В данной группе функций используется один или большее количество аргументов, являющихся таблицами.

ABSC	абсолютное значение комплексного числа или массива
ADD	сумма массивов
ADDCR	сумма комплексных и вещественных чисел или массивов
ADDRC	сумма вещественных и комплексных чисел или массивов
ANGLEC	угол комплексного числа или массива
ARRAVG	среднее значение элементов массива
ARRMAX	максимальный элемент массива
ARRMIN	минимальный элемент массива
ARRPROD	произведение элементов массива
ARRSTDDEV	стандартное отклонение элементов массива
ARRSUM	сумма элементов массива
CARTOPOL	преобразование комплексного числа или массива из декартовой формы в полярную
DELAYPPLINFV	векторная конвейерная информационная задержка с переменным временем запаздывания
DELAYPPLMTRV	векторная конвейерная материальная задержка с переменным временем запаздывания
ELEMCOUNT	число элементов массива
IMAGC	мнимая часть комплексного числа или таблицы
INVERT	обращение квадратной матрицы
INVERTC	обращение комплексного числа или квадратной матрицы
LOOKUP	поиск по индексу элемента массива
MATRIXPROD	произведение матриц
POLTOCAR	преобразование комплексного числа или массива из полярной формы в декартову
PROD	произведение выражений по индексам
PRODC	произведение комплексных чисел, векторов или массивов
PRODCR	произведение комплексных и вещественных чисел, векторов или массивов
PRODRC	произведение вещественных и комплексных чисел, векторов или массивов
REALC	вещественная часть комплексного числа
SCANEQ	поиск равного компонента вектора
SCANGT	поиск большего компонента вектора
SCANGTEQ	поиск большего или равного компонента вектора
SCANLT	поиск меньшего компонента вектора
SCANLTEQ	поиск меньшего или равного компонента вектора
SCANNEQ	поиск неравного компонента вектора
SHIFTCIF	условный циклический сдвиг компонент вектора
SHIFTLCNT	линейный сдвиг компонент вектора
SHIFTLIF	условный линейный сдвиг компонент вектора
SPROD	скалярное произведение векторов
TRANSPOSE	транспонирование матрицы
TRANSPOSEC	транспонирование матрицы комплексных чисел
VECTLEN	длина вектора
VECTOR	создание вектора
VECTPROD3D	векторное произведение трехмерных векторов

2.2.2 Встроенные функции

К данной группе функций относятся функции, являющиеся частью языка моделирования Powersim.

|	оператор защиты
;	оператор ограничения
AND	логическое И
BUT	оператор ограничения
COUNT	число элементов диапазона или главного индекса диапазона
DEFAULT	выбор выражения ограничения по умолчанию
FIRST	нижний предел диапазона или главного индекса диапазона
INDEX	преобразование переменной индекса в скаляр
LAST	верхний предел диапазона или главного индекса диапазона
OR	логическое ИЛИ
OTHERWISE	оператор защиты по умолчанию
SUM	суммирование выражений по индексам
WHEN	оператор защиты

2.2.3 Комплексные функции

Эта группа содержит функции для работы с комплексными числами.

ABSC	абсолютное значение комплексного числа или массива
ADD	сумма массивов
ADDCR	сумма комплексных и вещественных чисел или массивов
ADDRC	сумма реальных и комплексных чисел или массивов
ANGLEC	угол комплексного числа или массива
CARTOPOL	преобразование комплексного числа или массива из декартовой формы в полярную
IMAGC	мнимая часть комплексного числа или массива
INVERTC	обращение комплексного числа или квадратной матрицы
POLTOCAR	преобразование комплексного числа или массива из полярной формы в декартову
PRODC	произведение комплексных чисел, векторов или массивов
PRODCR	произведение комплексных и вещественных чисел, векторов или массивов
PRODRC	произведение вещественных и комплексных чисел, векторов или массивов
REALC	вещественная часть комплексного числа
TRANSPOSEC	транспонированная матрица комплексных чисел

Краткое руководство по использованию комплексных функций

Для улучшения восприятия можно определить комплексное число (`Complex`) как именованный диапазон с двумя элементами. Можно использовать оба из нижеприведенных определений диапазона:

```
range    Complex = 1..2
range    Complex = (Re, Im)
```

Комплексное число *C* с вещественной частью, равной 3, и мнимой частью, равной 4, определяется так:

```
range    Complex = 1..2
const    C(Complex) = [3, 4]
```

Постоянный вектор *V* с тремя комплексными числами (компонентами) определяется следующим образом:

```
const    V(1..3,Complex) = [[1,4],[2,0],[0,1]]
```

Матрица *M* размерности 2x2, состоящая из комплексных чисел, определяется так:

```
const    M(1..2,1..2,Complex) = [[[1,4],[0,1]],[[3,4],[-8,7]]]
```

2.2.4 Условные функции

В данную группу входят функции, работа которых зависит от условий, входящих в функции в качестве дополнительных аргументов.

IF	арифметическое ЕСЛИ
PULSEIF	условный импульс
SAMPLEIF	условная выборка
SHIFTCIF	условный циклический сдвиг компонент вектора
SHIFTLIF	условный линейный сдвиг компонент вектора

2.2.5 Функции управления

Функции управления используются для управления процессом моделирования.

PAUSEIF	условная пауза
PAUSEWHILE	условная длящаяся пауза
STOPIF	условная остановка всех имитаций
STOPRUNIF	условная остановка текущей имитации

2.2.6 Функции преобразования типа

Данная группа функции преобразовывают значение из одного типа в другой.

%	процент
ABS	абсолютное значение
BOOL	преобразование числа к булевой форме
CARTOPOL	преобразование комплексного числа или массива из декартовой формы в полярную
CEIL	округление вверх до ближайшего целого
DEGTOGRAD	преобразование градусов в грады
DEGTORAD	преобразование градусов в радианы
FLOOR	округление вниз до ближайшего целого
FRAC	дробная часть числа
GRADTODEG	преобразование градов в градусы
GRADTORAD	преобразование градов в радианы
INDEX	преобразование переменной индекса в скаляр
INT	целая часть числа
PCT	преобразование числа в процент
POLTOCAR	преобразование комплексного числа или массива из полярной формы в декартову
RADTODEG	преобразование радиан в градусы
RADTOGRAD	преобразование радиан в грады
ROUND	округление до ближайшего целого
SIGN	знак числа

В таблице 2.4 представлены результаты использования некоторых функций преобразования типа.

Таблица 2.4 Таблица связи некоторых функций преобразования типа

X	FLOOR(X)	ROUND(X)	CEIL(X)	INT(X)	FRAC(X)
-2.2	-3	-2	-2	-2	-0.2
-1.8	-2	-2	-1	-1	-0.8
0.2	0	0	1	0	0.2
2.6	2	3	3	2	0.6

2.2.7 Функции задержки

Данная группа содержит функции с замедленным откликом на изменение аргументов.

DELAYINF	информационная задержка *n*-го порядка
DELAYMTR	материальная задержка *n*-го порядка
DELAYPPL	конвейерная (канальная, дискретная) задержка
DELAYPPLINF	конвейерная информационная задержка с переменным временем запаздывания
DELAYPPLINFV	векторная конвейерная информационная задержка с переменным временем запаздывания
DELAYPPLMTR	конвейерная материальная задержка с переменным временем запаздывания
DELAYPPLMTRV	векторная конвейерная материальная задержка с переменным временем запаздывания

Задержки (запаздывания) происходят часто в реальных системах. Они возникают на каждой стадии функционирования системы – в решениях, при перевозке, в усреднении данных и запасов всех видов. Например, заказ какого-либо товара не обязательно приводит к его непосредственному получению. Или же какое-либо заболевание дает о себе знать только спустя определенный период времени. И, наконец, информация относительно ежедневного сбыта должна накопиться прежде, чем будет получено значение среднемесячного сбыта. Кроме того, такое накопление само по себе требует определенного времени.

Задержки удобно разделять на два типа:

1) задержки во времени, следующие из протекания материальных процессов;
2) задержки во времени, следующие из запаздываний в процессах получения, анализа информации и реакции на нее.

Эти два типа задержек называются соответственно материальными и информационными задержками. Рассмотрим примеры задержек разных типов.

Материальная задержка 1-го порядка

Рассмотрим простую эпидемиологическую модель, которая поясняет работу данной задержки в системе. В данной модели определенная группа людей становится инфицированной и проходит инкубационный период прежде, чем начинают проявляться признаки заболевания. Это отражено на рис. 2.5. В системе присутствует некоторое запаздывание между моментом времени, когда человек становится инфицированным, и моментом времени, когда начинают проявляться признаки заболевания и человек становится больным.

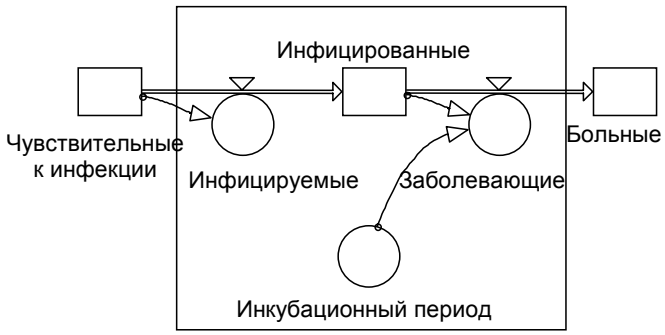


Рисунок 2.5 Потоковая диаграмма для эпидемиологической модели

Задержка определяется (средней) длительностью инкубационного периода. Часть модели, которая представлена на рис. 2.5, имеет структуру материальной задержки 1-го порядка. Мы имеем 1-ый порядок задержки, поскольку есть только один уровень, вовлеченный в задержку. Общая структура материальной задержки 1-го порядка дана на рис. 2.6:

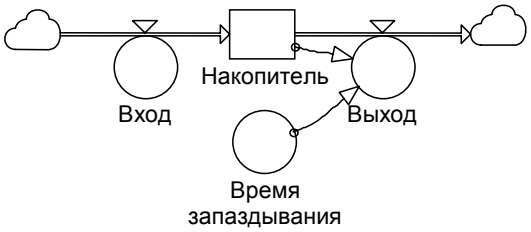


Рисунок 2.6 Структура материальной задержки 1-го порядка

Данная потоковая диаграмма описывается следующими уравнениями[*]:

```
level    Накопитель = "Начальное значение" - dt*Выход + dt*Вход
aux      Вход = "Входной темп, на который распространяется задержка"
aux      Выход = Накопитель/Время_запаздывания
aux      Время_запаздывания = "Среднее время запаздывания"
```

Начальное значение переменной Накопитель, значение переменных Вход и Время_запаздывания могут принимать различные значения. При использовании функции DELAYMTR, этот класс задержек может быть смоделирован следующим образом (рис. 2.7):

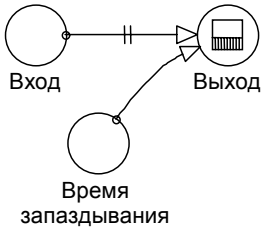


Рисунок 2.7 Использование функции DELAYMTR(, ,1,)[**]

Данной диаграмме соответствует уравнение:

```
aux      Выход = DELAYMTR(Вход, Время_запаздывания, 1, "Начальное
         значение")
```

Если Начальное значение для DELAYMTR не задано, то по умолчанию оно будет равно значению Входа. Это означает то, что Вход является одновременно и запаздывающим и не запаздывающим параметром для Выхода. Следовательно, еще нужно добавить обычную стрелку связи от Входа к Выходу. Эта связь станет инициализирующей связью, как показано на рис. 2.8.

---

[*] Далее по тексту уравнение для переменной типа level будет обозначать сокращенную запись двух уравнений для переменных типа init и flow.

[**] Объяснение пиктограммы для переменной Выход см. в п. 2.1.4

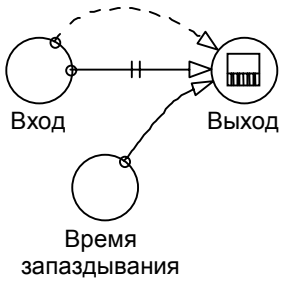


Рисунок 2.8 Инициализация DELAYMTR при помощи значения Входа

Материальная задержка 3-го порядка

Если мы предполагаем, что время инкубационного периода, необходимое для проявления признаков, составляет три месяца, то мы можем разделить всю группу инфицированных на три подгруппы (рис. 2.9): 1) те, кто находится на первом месяце инкубационного периода (Стадия_1), 2) те, кто находится на втором месяце (Стадия_2), 3) те, кто находится на третьем месяце (Стадия_3).

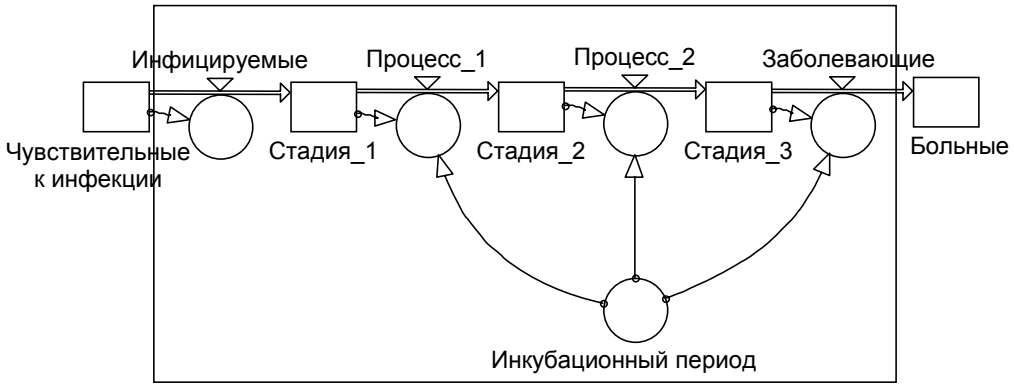


Рисунок 2.9 Потоковая диаграмма для материальной задержки 3-го порядка

Часть диаграммы внутри прямоугольной рамки (рис. 2.9) представляет собой материальную задержку 3-го порядка. Порядок три возникает из предложения, что поток проходит три уровня. Структура материальной задержки 3-го порядка показана далее на рис. 2.10:

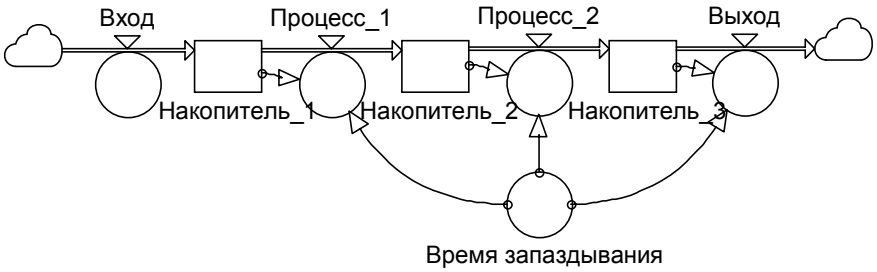


Рисунок 2.10 Структура материальной задержки 3-го порядка

Уравнения материальной задержки 3-го порядка задаются следующим образом:

```
level    Накопитель_1 ="Начальное значение"+dt*Вход-dt*Процесс_1
level    Накопитель_2 ="Начальное значение"+dt*Процесс_1-dt*Процесс_2
level    Накопитель_3 ="Начальное значение"+dt*Процесс_2-dt*Выход
aux      Вход = "Входной темп, на который распространяется задержка"
aux      Процесс_1 = Накопитель_1/(Время_запаздывания/3)
aux      Процесс_2 = Накопитель_2/(Время_запаздывания/3)
aux      Выход = Накопитель_3/(Время_запаздывания/3)
aux      Время_запаздывания  = "Среднее время запаздывания"
```

Начальные значения промежуточных уровней, значения Входа и Времени_ запаздывания, являются параметрами в структуре задержки, так же как и в случае материальной задержки 1-го порядка (см. рис. 2.7). Точно как в случае материальной задержки 1-го порядка, материальная задержка 3-го порядка может быть смоделирована при помощи функции DELAYMTR.

```
aux      Выход = DELAYMTR(Вход, Время_запаздывания, 3, "Начальное
         значение")
```

Материальная задержка бесконечного порядка (конвейерная, канальная, дискретная)

В гипотетической задержке бесконечного порядка выход никак не меняется до тех пор, пока не истечет время запаздывания. Как только это происходит, на выходе точно воспроизво-дится переменная входа. Данная задержка может рассматриваться как "лестница эскалатора" (в метро) или конвейерная лента, на одном конце которой помещаются предметы, а с другого конца они снимаются по прошествии фиксированного временного интервала.

Эта задержка может быть смоделирована при помощи ряда уровней, число которых равно Времени_запаздывания / Шаг моделирования. За один шаг материал перемещается от предыдущего уровня к последующему, пока не достигнет конечного уровня, где он удаляется. В Powersim данная задержка может быть смоделирована при помощи векторного уровня с применением функция SHIFTLIF на каждом временном шаге для перемещения элементов из одного положения в следующее (рис. 2.12).

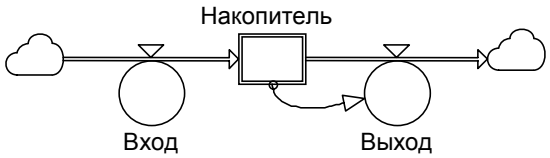


Рисунок 2.11 Потоковая диаграмма для конвейерной задержки

Полагая, что во Времени запаздывания укладывается десять шагов моделирования, получим следующие уравнения:

```
dim      Накопитель = 1 .. 10
level    Накопитель(i) = "Начальное значение" + dt*(Вход|i = 1;0) -
         dt*(Выход|i = LAST(i);0)
aux      Вход = "Входной темп, на который распространяется задержка"
aux      Выход = SHIFTLIF (ИСТИНА, Накопитель)
```

Для непосредственного задания этого вида задержки существует функция DELAYPPL.

```
aux     Выход = DELAYPPL(Вход, Время_запаздывания,0)
```

Как и в случае с функцией DELAYMTR, Вход должен быть соединен с Выходом при помощи дополнительной незапаздывающей связи, если значение Входа используется для инициализации задержки (см. рис. 2.7). Такое поведение функции DELAYPPL задается по умолчанию, если не задан последний аргумент данной функции.

Информационная задержка 1-го порядка

Информации так же, как и материалу, требуется время для перехода из одной точки системы в другую. Например, экономистам, как правило, не известен текущий ВВП (валовой внутренний продукт). Также требуется время для получения информации относительно рынка факторов производства. Кроме того, восприятие людей отстает от истинного отражения действительности. Такие запаздывания часто являются неизбежной частью структуры системы и, следовательно, должны фиксироваться в моделях. Многие из данных запаздываний связаны с трансакционными издержками.

Рассмотрим пример, в котором решения о темпах производства основаны на информации о сбыте. Однако, когда организации принимают решения, они имеют тенденцию полагаться на более свежую информацию, чем на ранее накопленные данные. Следующий рисунок показывает структуру так называемого экспоненциального усреднения информации о сбыте (продажах) с заданным временем усреднения.

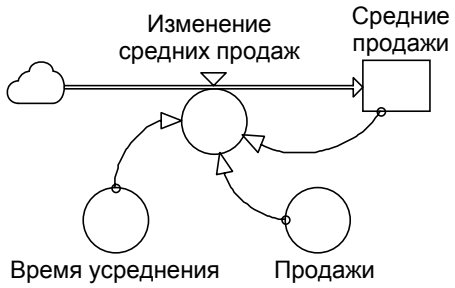


Рисунок 2.12 Потоковая диаграмма экспоненциального усреднения информации

Средний сбыт может быть также использован как критерий для оценки ожидаемого сбыта. Информационная задержка иногда также упоминается как сглаживание входа. На рис. 3.13 показана общая структура информационной задержки 1-го порядка:

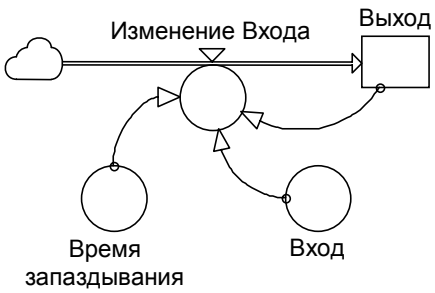


Рисунок 2.13 Структура информационной задержки 1-го порядка

Выход можно считать как "ощущаемое" значение входа. Структура информационной задержки 1-го порядка описывается следующими уравнениями:

```
level     Выход = "Начальное значение" + dt*Изменение_Входа
aux       Вход = "Входной темп, на который распространяется задержка"
aux       Изменение_Входа = (Вход - Выход)/Время_запаздывания
aux       Время_запаздывания = "Время запаздывания"
```

Данный тип задержки может быть получен при использовании функции DELAYINF.

```
aux       Выход = DELAYINF(Вход, Время_запаздывания, 1, "Начальное
          значение")
```

При этом потоковая диаграмма примет вид, представленный на рис. 2.14:

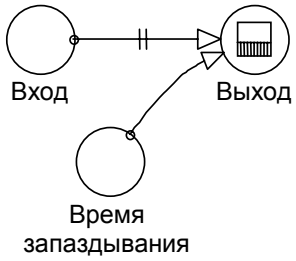


Рисунок 2.14 Использование функции DELAYINF(, ,1,)*

* Объяснение пиктограммы для переменной Выход см. в п. 2.1.4

Информационная задержка 3-го порядка

В качестве другого примера рассмотрим систему производства / распределения, в которой поток информации течет в форме заказов, перемещающихся от потребителей к поставщику. Структура системы показана ниже на рис. 2.15. Информация, посылаемая от рынка к поставщику, корректируется каждым сектором с заданным временем усреднения.

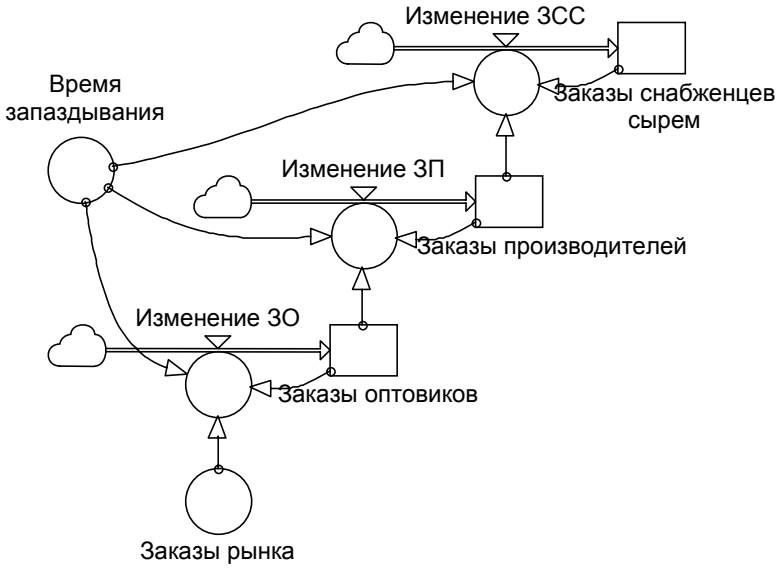


Рисунок 2.15 Потоковая диаграмма производства / распределения товаров

Структура информационной задержки 3-го порядка в общев виде дана ниже на рис. 2.16.

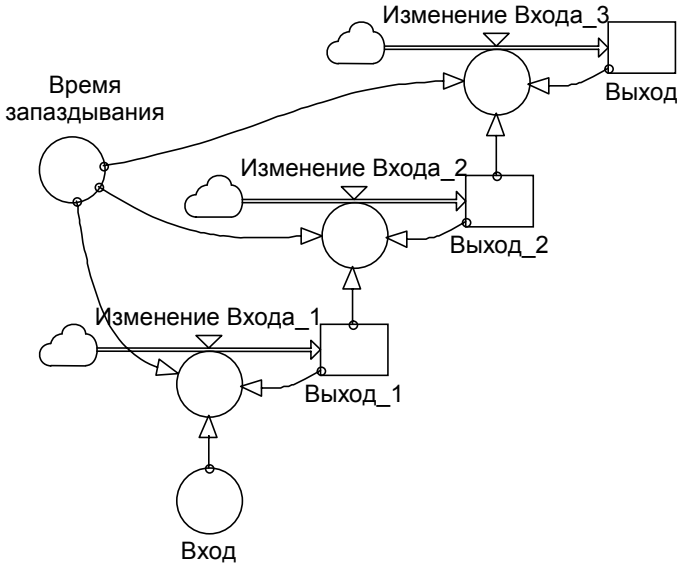


Рисунок 2.16 Структура информационной задержки 3-го порядка

Данная структура описывается следующими уравнениями:

```
level    Выход_1 = "Начальное значение" + dt*Изменение_Входа_1
level    Выход_2 = "Начальное значение" + dt*Изменение_Входа_2
level    Выход = "Начальное значение" + dt*Изменение_Входа_3
aux      Вход = "Входной темп, на который распространяется задержка"
aux      Изменение_Входа_1 = (Вход-Выход_1)/(Время_запаздывания/3)
aux      Изменение_Входа_2 = (Выход_1-Выход_2)/(Время_запаздывания/3)
aux      Изменение_Входа_3 = (Выход_2-Выход)/(Время_запаздывания/3)
aux      Время_запаздывания = "Время_запаздывания"
```

Информационная задержка 3-го порядка может быть реализована при помощи функции DELAYINF:

```
aux      Выход = DELAYINF(Вход, Время_запаздывания, 3, "Начальное
         значение")
```

Среднее время запаздывания

Во всех пяти примерах задержек присутствовало среднее Время_запаздывания, которое является важной характеристикой любой задержки, и определяет результат действия установившегося состояния задержки. В неизменных условиях, Вход и Выход должны быть равны. В установившемся состоянии Вход, умноженный на среднее Время_запаздывания, дает количество материала или информации, проходящего через задержку.

Передаточная характеристика (отклик) задержки

Передаточная характеристика (отклик) задержки говорит о том, как поведение Выхода связано с поведением Входа. Различные виды задержек могут иметь весьма разнообразные передаточные характеристики (отклики) на изменение Входа даже при одном и том же среднем Времени задержки. При построении модели выбор задержки с той или иной передаточной характеристикой является очень ответственным этапом. Если передаточная характеристика изначально выбрана неправильно, то это может существенно повлиять на качественное поведение модели.

Для создания задержки в каком-либо потоке внутри математической модели могут быть использованы различные вычислительные процессы. Мы рассматриваем здесь функцию задержки, моделируемую только единственным классом процессов, а именно, экспоненциальными задержками. Экспоненциальные задержки просты по форме и адекватно, с точки зрения современных знаний о сложных системах, описывают большую часть запаздываний, присутствующих в реальных системах, которые мы моделируем. Отклик экспоненциальной задержки с ростом ее порядка (т.е. числа уровней) увеличивается.

На рис. 2.17 показаны передаточные характеристики для задержек различного порядка (1-го, 3-го, 6-го порядка и канальной задержки) при ступенчатом входном воздействии (Step Input). Данный график отражает отклик материальных задержек на входное изменение с увеличением порядка задержки при Времени_запаздывания, равном 10 для всех задержек.

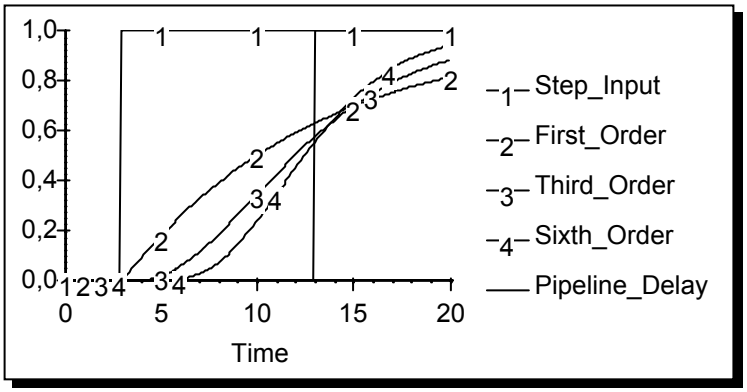


Рисунок 2.17 Передаточная характеристика (отклик) задержек различного типа на ступенчатое входное воздействие

Отклики тех же самых задержек (1-го, 3-го, 6-го порядка и канальной задержки) на импульсный входной сигнал показаны ниже на рис. 2.18:

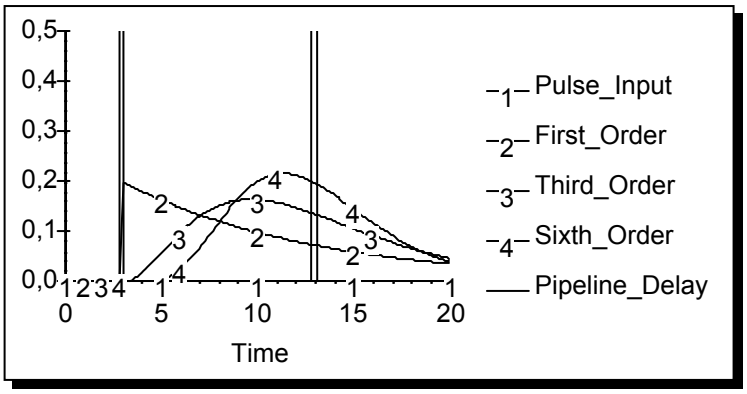


Рисунок 2.18 Передаточная характеристика (отклик) задержек различного типа на импульсное входное воздействие

Передаточные характеристики информационной и материальной задержек одного порядка одинаковы, если Время_запаздывания неизменно и одинаково для обеих задержек. Однако существует и ряд отличий, присущих данным типам задержек.

Материальные задержки относятся к материальным потокам и не должны, согласно закону сохранения материи, уничтожать или создавать дополнительные единицы материи в проходящих через них потоках. Это означает, что в материальной задержке с постоянным входящим потоком, изменение Времени_запаздывания должно привести к изменению выходящего потока. Например, если Время_запаздывания сократилось, то выходящий из задержки материальный поток на короткое время увеличится в связи с уменьшением длины "конвейера" и выталкиванием излишков материалов за пределы "конвейера". Впоследствии количество проходимого через задержку материала приспособится к условиям нового установившегося равновесного состояния.

Информационная задержка, в свою очередь, ведет себя по-иному. Изменение Времени_ запаздывания никак не отражается на изменении выходящего потока при постоянном информационном входящем потоке, поскольку в информационных потоках нет соответствующего закона сохранения. В установившемся состоянии (при постоянном входящем потоке) усредненное значение входящего информационного потока будет совпадать с мгновенным значением данного потока. Следовательно, значение Времени_ запаздывания можно изменить, не оказывая воздействия на значение усредненного выходящего информационного потока. (Выход задержки изменяется только за счет потока, а величина (Вход – Выход)/Время_запаздывания обращается в нуль в установившемся состоянии и не зависит от значения Времени_запаздывания. Более подробно об этом см. уравнения информационной задержки 1-го порядка, приведенные выше.)

2.2.8 Финансовые функции

Данная группа функций предназначена для вычисления текущей ценности, будущей ценности и частичных (периодических) платежей.

FV	будущая ценность (будущая стоимость)
NPV	чистая приведенная ценность (стоимость)*
PMT	частичный платеж (взнос или поток платежей)
PV	текущая (настоящая) ценность (текущая / современная стоимость)

Powersim использует следующее уравнение для выражения одного финансового показателя через другие при фиксированных (ежегодных) частичных платежах:

$$PV(1+r)^n + \frac{PMT}{r}[(1+r)^n - 1] + FV = 0\,.$$

* Так же чистый приведенный или дисконтированный доход.

Если ставка процента $r = 0$, то уравнение приводится к виду:

$$PV + PMT \cdot n + FV = 0,$$

где PV – текущая ценность, PMT – частичный платеж (поток платежей), r – ставка процента (например, % в год), n – число периодов (например, годы), FV – будущая ценность. Для PV, PMT, и FV используется следующее "*соглашение о знаках*": полученный частичный платеж описывается положительным числом, а выплаченный – отрицательным.

В формулах (функциях) предполагается, что первая оплата происходит в конце первого периода, т.е. мы имеем дело с рентой постнумерандо. Следует иметь в виду, что ставки процента даны относительно выбранного шага моделирования. Например, если один шаг равен году, то ставка процента должна быть определена как ежегодная величина.

Простой случай – работа с одним периодом

Текущая ценность это дисконтированная ценность будущих потоков платежей, поскольку деньги сегодня стоят больше, чем завтра, так как сегодня их можно вложить в дело и начать получать с них процент немедленно. Это первый основной принцип финансов.

Текущая и будущая ценность

Таким образом, текущая ценность отложенного платежа может быть найдена путем умножения платежа на дисконтный множитель, который меньше единицы. Если FV_1 обозначает ожидаемый размер платежа через один период, то текущая ценность может быть найдена по формуле:

$$PV = -D \cdot FV_1,$$

где D – дисконтный множитель. Отрицательный знак в формуле появился из-за соглашения о знаках (см. выше), используемого нами в финансовых формулах. При этом исходная формула примет вид:

$$PV \cdot (1 + r) + FV_1 = 0.$$

Если, например, будущий платеж (FV_1) должен быть положительным, то мы должны произвести выплату сейчас, т.е. текущая ценность (PV) будет отрицательна. Если ставка процента равна r, то дисконтный множитель для одного периода определяется как:

$$D = \frac{1}{1 + r}.$$

Предположим, что в результате пожара сгорел ваш завод и вы остались с участком земли стоимостью в 50 тыс. $ и чеком на 200 тыс. $, полученным от страховой компании. Вы предполагаете восстановить завод, но ваш советник по недвижимости вместо этого предлагает вам построить офисное здание. С одной стороны, стоимость монтажных работ составила бы 300 тыс. $, плюс к этому добавилась бы стоимость земли, которую в противном случае можно было бы продать за 50 тыс. $. С другой стороны, ваш советник предвидит нехватку офисных помещений в будущем и предсказывает, что через год от продажи нового офисного сооружения можно будет получить 400 тыс. $. Таким образом, вы можете вложить 350 тыс. $ в настоящий момент времени с ожиданием получить 400 тыс. $ через год (В данном примере знак потока платежей определяется из соглашения о знаках. Например, "капиталовложение в размере 350 тыс. $" следует представить как -350). Следует также учитывать данное соглашение о знаках и в том случае, если текущая ценность ожидаемого платежа 400 тыс. $ сравнивается с капиталовложением в размере 350 тыс. $. Следовательно, необходимо определить текущую ценность 400 тыс. $, которые будут получены через год, отсчитывая от текущего момента времени.

Рассмотрим инвестиции в недвижимость, полагая на некоторое время, что имеется полная уверенность в получении 400 тыс. $ через год. В этом случае текущая ценность капиталовложения составит 350 тыс. $. Однако, офисное сооружение – не единственный способ получить 400 тыс. $ через год. Вы можете вложить имеющиеся средства в краткосрочные государственные ценные бумаги со сроком погашения 1 год. Предположим, что эти ценные бумаги обладают годовой доходностью 7%. Каков должен быть размер капиталовложения, чтобы получить 400 тыс. $ в конце года? Согласно нашей формуле получим:

$$PV = -D \cdot FV_1 = -\frac{FV_1}{1+r} = -\frac{400}{1+0.07} = -373.832 .$$

В табл. 2.5. представлены два альтернативных варианта капиталовложений.

Таблица 2.5 Варианты капиталовложений и их текущая ценность

Варианты	**Капиталовложения, тыс. $**	$\boldsymbol{FV_1}$**, тыс. $**
В офисное сооружение	350	400
В ценные бумаги	373.832	400

Альтернативные капиталовложения

Из анализа двух вариантов капиталовложений видно, что вложение капитала в офисное сооружение будет стоить меньше, чем вложение капитала в ценные бумаги, т.е. следует выбрать первый вариант, который вам рекомендовал ваш советник по недвижимости. Текущая ценность капиталовложения в ценные бумаги легко получается при помощи функции *PV*:

$$PV\ (7\%, 1, 0, 400) = - 373.832.$$

Предположим, что как только вы привели в порядок землю и начали строительство на ней офисного здания, вы решаетесь продать ваш проект. За сколько его можно продать?

Так как собственность дает 400 тыс. $, то инвесторы желали бы заплатить за него 373.832 тыс. $. Это именно то, что потребуется от них для того, чтобы получить 400 тыс. $ от вложения капитала в государственные ценные бумаги. Поэтому текущая ценность в данном случае будет определять единственно возможную цену, которая удовлетворяет и покупателя и продавца. Следовательно, текущая ценность собственности является также рыночной ценой.

Обобщение на несколько периодов

Есть наиболее быстрый способ, позволяющий очень просто вычислить текущую ценность потока платежей за несколько периодов. Рассмотрим ряд примеров.

Бесконечная (вечная или пожизненная) рента

Пусть имеются ценные бумаги, которые выпущены государством и представляют собой бесконечную (вечную или пожизненную) ренту. Это могут быть рентные облигации, которые государство не обязано выкупать, но по которым оно обязано ежегодно выплачивать фиксированную сумму их владельцу. Норма доходности рентных облигаций равна размеру обещанного ежегодного частичного платежа (взноса), деленному на текущую ценность облигации:

$$r = -\frac{PMT}{PV} .$$

Очевидно, что из этой формулы можно найти текущую ценность облигации по заданной учетной ставке *r* и размеру частичного платежа *PMT*.

Предположим, что некоторый человек хочет обеспечить себя постоянным пожизненным доходом (пенсией), равным 100 тыс. $ в год. Таким образом, он должен приобрести вечный аннуитет. Если годовая ставка процента равна 10% и цель состоит в обеспечении себя указанным постоянным годовым доходом, то сумма денег, которую следует отложить для этой цели сегодня, вычисляется следующим образом:

$$PV = -\frac{PMT}{r} = -\frac{100}{0{,}1} = -1000\,.$$

В этом примере мы хотим, чтобы абсолютное значение *PV* было бы таким же, как и *FV*, что получается путем использования следующей функции:

$$PV\,(10\%,\ 1,\ 100,\ 1000) = -\ 1000.$$

Ипотека

Смысл ипотеки (ренты с ежегодными выплатами) заключается в том, что клиент выплачивает некоторой организации фиксированную сумму (частичный платеж) ежегодно в течение точно установленного периода времени. Ипотечная ссуда или соглашение о покупке с оплатой в рассрочку – наглядные примеры ежегодных рент.

Уравнения ниже поясняют простой прием для вычисления ежегодных рент. Сначала берем бесконечную ренту, которая производит поток платежей *PMT* каждый период, начиная с первого периода. Она имеет текущую ценность, равную

$$PV = -\frac{PMT}{r}\,.$$

Затем берем другую бесконечную ренту, которая производит поток платежей *PMT* каждый период, начиная с периода $n + 1$. Ее текущая ценность равна

$$PV = -\frac{PMT}{r}\frac{1}{(1+r)^n} = -\frac{PMT}{r(1+r)^n}\,.$$

Обе бесконечные ренты обеспечивают бесконечный поток платежей, начиная с периода $n + 1$. Единственное отличие между ними состоит в том, что первая рента также обеспечивает поток платежей в каждом из периодов с 1 по *n*. Иными словами, разность между первой и второй рентой дает ежегодную ренту с потоком платежей *PMT* в течение *n* периодов. Следовательно, текущая ценность ежегодной ренты есть разность между текущими ценностями двух указанных бесконечных рент.

Выражение в скобках – коэффициент ежегодной ренты, который является текущей ценностью (при учетной ставке *r*) ежегодной ренты постнумерандо в размере 1 ден. единицы (например $ 1), оплачиваемой в конце каждого периода. Переписывая уравнение в ином виде, получим

$$PV(1+r)^n + PMT[(1+r)^n - 1] = 0\,.$$

Предположим, что ваши спонсоры задались вопросом, какая сумма им потребуется для обеспечения дохода в размере 100 тыс. $ в год в течение 20 лет. Ответ на этот вопрос следует из вышеприведенной формулы. Тот же самый результат получается при помощи функции *PV*:

$$PV\,(10\ \%,\ 20,\ 100,\ 0) = -851.356.$$

Допустим, что вы решили получить ссуду на покупку автомобиля в размере 10 тыс. $ при эффективной годовой ставке процента 10% и должны вернуть ее посредством

5 ежегодных частичных платежей, осуществляемых в конце каждого года. Каков размер вашего частичного платежа? Применяя функцию *PMT*, получим:

$$PMT\ (10\ \%, 5, 10, 0) = -2.638.$$

Следующая модель (рис. 2.19) поясняет процесс погашения ссуды:

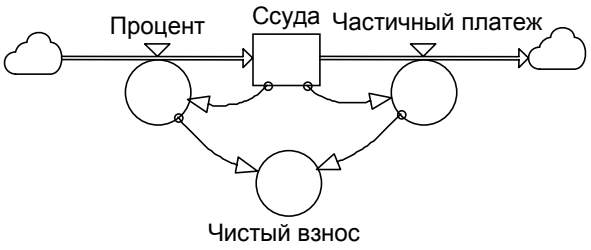


Рисунок 2.19 Потоковая диаграмма погашения ссуды

Данная диаграмма описывается следующими уравнениями:

```
level   Ссуда = 10 + dt*Процент - dt*Частичный_платеж
aux     Частичный_платеж = -PMT (10%, 5, INIT (Ссуда), 0)
aux     Процент = Ссуда*10%
aux     Чистый_взнос = Частичный_платеж - Процент
```

В табл. 2.6 показаны следующие величины: размер Частичного_платежа, накопленный Процент и ежегодный Чистый_ взнос:

Таблица 2.6 Параметры погашения ссуды по годам

Год	Ссуда, тыс. $	Частичный платеж, тыс. $	Процент, тыс. $	Чистый взнос, тыс. $
1	10.000	2.638	1.000	1.638
2	8.362	2.638	0.836	1.802
3	6.560	2.638	0.656	1.982
4	4.578	2.638	0.458	2.180
5	2.398	2.638	0.240	2.398

Будущая ценность

Будущая ценность (*FV*) в терминах текущей ценности (*PV*), ставки процента (*r*) и числа периодов (*n*) определяется (с учетом соглашения о знаках) следующим образом:

$$FV = -PV(1+r)^n.$$

Иногда требуется найти, сколько ежегодных платежей при фиксированной годовой ставке процента накопилось бы к концу *n* периодов. В этом случае будущую ценность проще всего вычислить, найдя текущую ценность и умножив ее на $(1 + r)^n$.

Предполагая, что ваши спонсоры желают знать, какова будет отдача от ежегодных инвестиций в размере 100 тыс. $ (см. пример выше), следует дать такой ответ:

$$FV = -PV(1+r)^n = -(-851.356)(1+0.1)^{20} = 5727.5.$$

Тот же самый результат может быть получен с использованием функции *FV*:

$$FV\ (10\%, 20, 0, -851.356) = 5727.5.$$

Вместо того, чтобы использовать текущую ценность, можно использовать ежегодную ренту для получения того же самого результата:

$$FV\,(10\ \%,\ 20,\ -100,\ 0) = 5727.5.$$

Чистая приведенная ценность

Периодические платежи при постоянной ставке процента

Когда ставка процента постоянна, то для оценки желательности капиталовложений используется чистая приведенная ценность (NPV), которая может быть определена следующим образом:

$$NPV = PMT_0 + \sum_{t=1}^{n} \frac{PMT_t}{(1+r)^t}.$$

В вышеупомянутых формулах, PMT_t – поток платежей в конце периода t, PMT_0 – начальное (стартовое) капиталовложение (имеет отрицательное значение) и r – ставка процента.

Периодические платежи при плавающей ставке процента

Рассмотрим чистую приведенную ценность в случае с переменной ставкой процента и переменными значениями частичных платежей:

$$NPV = PMT_0 + \sum_{t=1}^{n} \frac{PMT_t}{\prod_{k=0}^{t-1}(1+r_k)},$$

где PMT_t – поток платежей в конце периода t, PMT_0 – начальное (стартовое) капиталовложение (имеет отрицательное значение) и $r_{t\text{-}1}$ – ставка процента в течение периода от t-1 до t.

Вышеупомянутая формула реализована в Powersim функцией *NPV*. Рассмотрим, например, следующий проект: первоначально имеется начальное (стартовое) капиталовложение в размере 1 тыс. $, которое приносит доход в 0.2 тыс. $ последующие десять лет. Первые три года – ставка процента (InterestRate) – 7%, в течение остающихся лет – 9%. Чистая приведенная ценность (NetPresentValue) для этого проекта может быть вычислена согласно следующей модели (рис. 2.20):

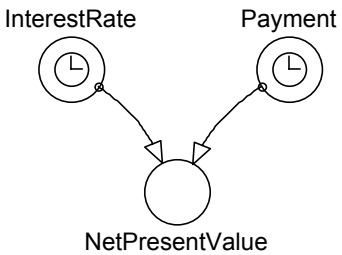


Рисунок 2.20 Структурная схема для расчета чистой приведенной ценности (NPV)

Соответствующие уравнения:

```
aux     NetPresentValue = NPV(Payment, InterestRate)
aux     Payment = IF(TIMEIS (STARTTIME), -1, 0.2)
aux     r = IF(TIME < STARTTIME+3, 7%, 9%)
```

В таблице на рис 2.21 показано изменение в течение 10 лет чистой приведенной ценности (NetPresentValue), потока платежей (Payment) и ставки процента (InterestRate):

TIME	Payment	InterestRate	NetPresentValue
0	-1.000,00	0,07	-1.000,00
1	200,00	0,07	-813,08
2	200,00	0,07	-638,40
3	200,00	0,09	-475,14
4	200,00	0,09	-325,36
5	200,00	0,09	-187,95
6	200,00	0,09	-61,88
7	200,00	0,09	53,78
8	200,00	0,09	159,89
9	200,00	0,09	257,23
10	200,00	0,09	346,54

Рисунок 2.21 Динамика чистой приведенной ценности, потока платежей и ставки процента по годам

Так как чистая приведенная ценность проекта положительна, то проект должен быть профинансирован. Однако проект, с теми же самыми потоками платежей, рассчитанный на пятилетний период, следует отвергнуть из-за отрицательной чистой приведенной ценности.

2.2.9 Графические функции

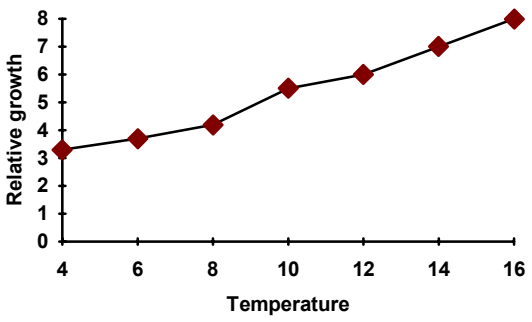
Это функции, в которых зависимость между значением функции и значениями ее аргументов задается в табличном или графическом виде*.

GRAPH	кусочно-линейный график с горизонтальными асимптотами
GRAPHCURVE	полиномиальный график с линейными асимптотами
GRAPHLINAS	кусочно-линейный график с линейными асимптотами
GRAPHSTEP	ступенчатый график с горизонтальными асимптотами

Графические функции полезны, когда мы не знаем точную математическую формулу для функциональной зависимости между двумя переменными, или если мы хотим использовать статистические или эмпирические данные для выражения этой зависимости.

Допустим, мы имеем эмпирические данные, показывающие ежедневный относительный темп прироста лосося на 1 кг (Relative growth) при различных температурах (Temperature), которые можно представить в табличном и графическом виде (табл. 2.7):

Таблица 2.7 Табличное и графическое задание функции

Температура, ˚C	**Относительный темп прироста, %**	**Данные, показанные графически**
4	3,3	
6	3,7	
8	4,2	
10	5,5	
12	6,0	
14	7,0	
16	8,0	

* Графические функции можно редактировать, используя окно диалога Edit Graph/Vector (Правка графика / вектора), вызываемое из диалогового окна Define Variable (Определение переменной) путем нажатия кнопки Graph (График...). Данные могут переноситься из таблиц или других приложений, использующих буфер обмена (операции "Копировать" и "Вставить").

Следующее уравнение выражает ту же самую связь между температурой и относительным темпом прироста (в %):

```
Рост = GRAPH(Температура, 4.00,2.00, [3.3,3.7,4.2,5.5,6,7,8])
```

Различные виды графических функций характеризуются способами вычисления значений, лежащих как между фиксированными точками, так и вне заданного диапазона значений переменной. Иными словами, графические функции характеризуются заложенными в них способами интерполяции (между точками) и экстраполяции (вне диапазона). Ниже приводится краткое объяснение этих терминов.

Интерполяция

В Powersim реализованы три способа определения значений, которые находятся между фиксированными точками графика, а именно горизонтальная, линейная, и полиномиальная интерполяция.

В первом случае в функции используется значение переменной в фиксированной точке слева от данного входного значения. В линейном случае проводится воображаемая линия между двумя фиксированными точками, прилагающими к входному значению, а значение функции есть точка пересечения входного значения с этой линией. В полиномиальном случае по всем фиксированным точкам задается многочлен третьего порядка и решается уравнение третьего порядка для данного входного значения. Рис. 2.22 поясняет горизонтальную, линейную и полиномиальную интерполяцию (крестиками показаны фиксированные точки).

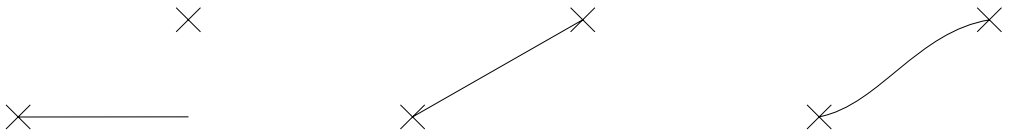


Рисунок 2.22 Горизонтальная, линейная, полиномиальная интерполяция

Экстраполяция

Когда входное значение графической функции лежит ниже первой или выше последней фиксированной точки, выходное значение вычисляется на основе либо горизонтальной, либо линейной экстраполяции. Линии, расширяющие с обоих концов графика за пределы последней фиксированной точки, называются асимптотами. В случае горизонтальных асимптот, значение в первой и последней точках продолжаются по горизонтальной линии в оба конца графика. Другой случай – линейная экстраполяция, когда в каждом из концов графика линии асимптот проходят через две близлежащие к концам графика точки. Рис. 2.23 поясняет эти два случая.

Рисунок 2.23 Графики с горизонтальными и линейными асимптотами

2.2.10 Функции с памятью (*хронологические*)

Это функции, в которых для вычисления текущих значений функции используются прошлые значения аргументов.

DELAYINF	информационная задержка
DELAYMTR	материальная задержка *n*-го порядка
DELAYPPL	конвейерная задержка
DELAYPPLINF	конвейерная информационная задержка с переменным временем
DELAYPPLINFV	векторная конвейерная информационная задержка с переменным временем
DELAYPPLMTR	конвейерная материальная задержка с переменным временем
DELAYPPLMTRV	векторная конвейерная материальная задержка с переменным временем
DERIVN	производная *n*-го порядка по времени
EULER	выборка в начале шага моделирования
FORCAST	прогнозирование значения
HIVAL	самое большое моделируемое значение
INIT	начальное значение
INTEGRATE	интегрирование
LOVAL	самое маленькое моделируемое значение
NPV	чистая приведенная ценность
QUEUE	очередь со многими клиентами и серверами
SAMPLE	периодическая выборка
SAMPLEIF	условная выборка
TREND	тренд

Очередь со многими клиентам и серверами (система массового обслуживания)

Рассмотрим пример очереди с несколькими клиентами, посылающими запрос на обслуживание серверам. Функцию QUEUE (Очередь) в Powersim обеспечивает механизм моделирования процесса "первым вошел, первым вышел" (ФИФО). Например, будем моделировать поток людей (клиентов), приходящих в банк для того, чтобы оплатить счет у одного из кассиров банка (серверов). Клиент, приходящий в банк вынужден будет стоять в очереди, если все кассиры заняты. Как только какой-либо кассир (сервер) освободится, будет обслужен первый по очереди клиент.

Прежде всего, определим число клиентов и число серверов как два именованных диапазона: rКлиенты и rКассиры. Для этого используем команду Define Range (Определение диапазона) в меню Edit (Правка). Определим один поток клиентов и два потока кассиров (серверов). Тогда получим:

```
range   rКлиенты = 1..1
range   rКасссиры = 1..2
```

Теперь построим следующую диаграмму (рис. 2.24):

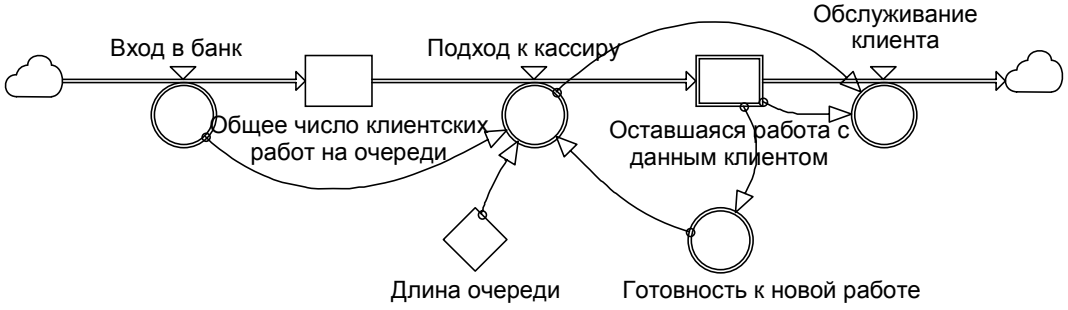


Рисунок 2.24 Потоковая диаграмма для системы массового обслуживания

Темп Вход_в_банк определяется как время обслуживания, запрошенное клиентом, входящим в настоящий момент времени в банк (в данном примере, этот темп имеет только одну компоненту, следовательно, Вход в банк можно было бы определить как скаляр без использования диапазона rКлиенты).

Будем считать, что темп потока входящей в банк клиентуры описывается распределением Пуассона с заданным средним значением. Выберем шаг моделирования TIMESTEP, равный одной минуте. Полагая, что один человек (клиент) входит в банк каждую последующую минуту, можно написать данную зависимость в виде:

```
aux     Вход_в_банк = POISSON(1/2)
```

Если мы хотим проводить имитационные эксперименты, не зависящие от шага моделирования, то необходимо изменить данную зависимость следующим образом:

```
aux     Вход_в_банк = POISSON(1/2*TIMESTEP)/TIMESTEP
```

Это означает, что в среднем один человек входит в банк каждую последующую минуту. Деление на TIMESTEP нужно для того, чтобы трактовать Вход_в_банк как поток импульса, поскольку потоки неявно умножаются на dt.

Клиенты могут нуждаться в разных количествах времени обслуживания кассирами. Например, можно предположить, что время обслуживания распределено нормальным образом (по Гауссу) со средним значением, равным четырем минутам, и единичным стандартным отклонением. Это можно записать как MAX(0, NORMAL (4, 1)). Функция MAX используется для того, чтобы избежать отрицательных времен обслуживания.

Если мы умножим вышеупомянутое время обслуживания на распределение клиентуры во времени, то получим переменную, которая генерирует последовательность времен обслуживания:

```
aux     Вход_в_банк=POISSON(1/2*TIMESTEP)/TIMESTEP*MAX(0,NORMAL(4,1))
```

Темп Обслуживание_клиента определяет, как быстро каждый кассир обслуживает клиента. Каждый раз один кассир выполняет одно обслуживание, что выражается следующим образом:

```
aux     Обслуживание_клиента = MAX(Подход_к_кассиру + Оставшаяся_
        работа_с_данным_клиентом/TIMESTEP, 1)
```

Отметим, что кассир не будет выполнять больший объем работы, чем необходим для завершения текущей работы. Так как исходящий из уровня поток Оставшаяся_работа_с_данным_клиентом равен Обслуживание_клиента*dt, то в качестве верхнего предела колебаний темпа Обслуживание_клиентамы будем использовать выражение Оставшаяся_работа_с_данным_клиентом / TIMESTEP + Подход_к_кассиру.

Кассир (сервер) готов к работе, если ему нечего делать, или если он выполнит текущую работу внутри данного временного шага. Это условие задается переменной Готовность_к_новой_ работе:

```
aux     Готовность_к_новой_работе = Оставшаяся_работа_с_данным_
        клиентом <= TIMESTEP
```

Полное описание модели имеет следующий вид:

Уравнения уровней

```
range   rКлиенты = 1..1
range   rКассиры = 1..2
dim     Оставшаяся_работа_с_данным_клиентом = (rКассиры)
level   Общее_число_клиентских_работ_на_очереди = 0 - dt*ARRSUM
        (Подход_к_кассиру) + dt*ARRSUM (Вход_в_банк)
level   Оставшаяся_работа_с_данным_клиентом = 0 -
        dt*Обслуживание_клиента + dt*Подход_к_кассиру
```

Уравнения темпов

```
dim     Вход_в_банк = (rКлиенты)
aux     Вход_в_банк=POISSON(1/2*TIMESTEP)*TIMESTEP*MAX(0,NORMAL(4,1))
dim     Подход_к_кассиру = (rКассиры)
aux     Подход_к_кассиру = QUEUE(Вход_в_банк,100,0,Готовность_к_
        новой_работе, Длина_очереди)
dim     Обслуживание_клиента = (rКассиры))
aux     Обслуживание_клиента =
        MIN(Оставшаяся_работа_с_данным_клиентом /TIMESTEP +
        Подход_к_кассиру,1)
dim     Готовность_к_новой_работе = (rКассиры)
aux     Готовность_к_новой_работе= Оставшаяся_работа_с_данным_
        клиентом <= TIMESTEP
```

Константы

```
const   Длина_очереди = 0
```

На рис. 2.25 показана зависимость длины очереди от времени моделирования за 20-минутный период. Моделирование велось с шагом, равным 0.125.

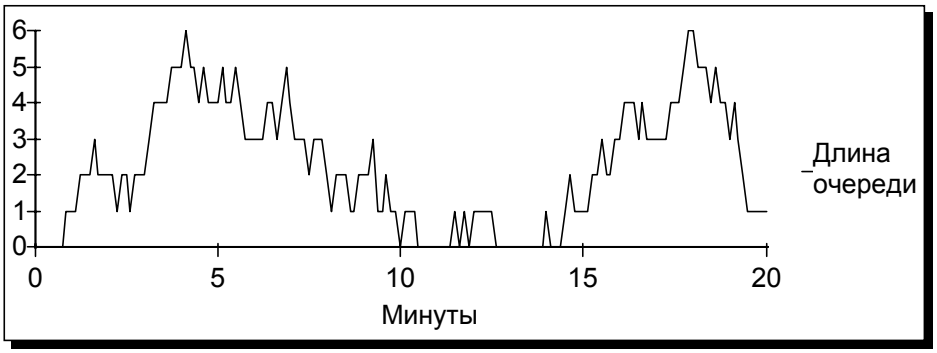


Рисунок 2.25 Зависимость длины очереди от времени

Модель очереди с одним клиентом и одним сервером также описана в п. 2.3.4 и в п. 2.4.7.

2.2.11 Логические функции

Это функции, возвращающие логическое значение True (Истина) или False (Ложь).

<	меньше
<=	меньше или равно
<>	не равно
=	равно
>	больше
>=	больше или равно
AND	логическое И
BOOL	преобразование числа к булевой форме
FALSE	логическая ЛОЖЬ
NOT	логическое НЕ
OR	логическое ИЛИ
TRUE	логическая ИСТИНА
XOR	логическое исключающее ИЛИ

2.2.12 Математические функции

Это функции для всех видов математических вычислений.

^	возведение в степень
!	факториал
%	процент
*	умножение
+	сложение
-	вычитание
/	деление
ABS	абсолютное значение
ADD	сумма массивов
ARRPROD	произведение элементов массива
ARRSUM	сумма элементов массива
DERIVN	производная *n*-го порядка по времени
DIVZ0	деление с нулевым результатом при обращении знаменателя в нуль
DIVZ1	деление с единичным результатом при обращении знаменателя в нуль
DIVZX	деление с заданным результатом при обращении знаменателя в нуль
EXP	экспонента числа
HYPOT	гипотенуза
INFINITY	бесконечно большое положительное число
INTEGRATE	интегрирование
INVERT	обращение квадратной матрицы
LN	натуральный логарифм числа
LOG	логарифм числа по основанию N
MATRIXPROD	произведение матриц
MOD	остаточный член от деления
NAN	не число
PCT	преобразование числа в процент
POLY	полином (многочлен)
SIGN	знак числа
SPROD	скалярное произведение векторов
SQRT	квадратный корень
TRANSPOSE	транспонирование матрицы
VECTLEN	длина вектора
VECTPROD3D	векторное произведение трехмерных векторов

2.2.13 Смешанные функции

Это функции, которые хорошо не вписываются в другие группы.

ASSIGN	присвоение значения переменной
GETPLAYERS	количество игроков
GETSIMPLAYERS	количество моделируемых игроков
GETTOTPLAYERS	общее количество игроков
LIMIT	ограничение переменной заданным диапазоном
PLAYERNUMBER	номер текущего игрока
RUN	номер текущей имитации
RUNCOUNT	количество имитаций
SELECTDECISION	выбор решения
SOUND	звук
STRATEGICMODE	проверка нахождения игры в стратегическом режиме

2.2.14 Случайные (стохастические) функции

Это функции, значения аргументов которых генерируются генератором случайных чисел по определенному закону.

EXPRND	экспоненциальное распределение
NORMAL	нормальное (гауссово) распределение
POISSON	распространение Пуассона
RANDOM	равномерное распределение

2.2.15 Статистические функции

Это функции для проведения вычислений с аргументами, представляющими собой выборки на множестве данных.

ARRAVG	среднее число элементов массива
ARRMAX	максимальный элемент массива
ARRMIN	минимальный элемент массива
ARRSTDDEV	стандартное отклонение элементов массива
AVG	среднее значение (среднее)
MAX	максимум
MIN	минимум
STDDEV	стандартное отклонение

2.2.16 Функции, зависящие от времени

Это функции, в которых либо значение функции, либо аргументы имеют размерность единицы времени моделирования.

Функции, возвращающие значения времени

STARTTIME	время начала моделирования
STOPTIME	время остановки моделирования
TIME	текущее время моделирования
TIMESTEP	шаг моделирования

Функции, используемые для проверки данного момента времени

ATSTART	проверка на начало моделирования
TIMECYCLE	проверка цикличности времени или временного интервала
TIMEIS	проверка попадания в данный момент времени или временной интервал

Функции, использующие время как неявный параметр

COSWAVE	косинусоидальная волна
PULSE	периодический импульс
RAMP	линейная функция
SAMPLE	периодическая выборка
SINWAVE	синусоидальная волна
STEP	ступенчатая функция

Оперативные функции

SECONDSTHISRUN	количество секунд, прошедших с начала текущей имитации
SECONDSTHISSTEP	количество секунд, прошедших на текущем шаге моделирования

2.2.17 Тригонометрические функции

Это функции для выполнения тригонометрических действий.

ARCCOS	арккосинус
ARCSIN	арксинус
ARCTAN	арктангенс
COS	косинус
COSH	гиперболический косинус
COSWAVE	косинусоидальная волна
DEGTOGRAD	преобразование градусов в грады
DEGTORAD	преобразование градусов в радианы
GRADTODEG	преобразование градов в градусы
GRADTORAD	преобразование градов в радианы
PI	тригонометрическая константа π
SIN	синус
SINH	гиперболический синус
SINWAVE	синусоидальная волна
TAN	тангенс
TANH	гиперболический тангенс

2.3 Сопоставление функций и выражений в Powersim и других языках

2.3.1 Функции и выражения в Powersim и математике

Все функции и выражения в Powersim так или иначе могут быть переписаны на обычном математическом языке, который обычно используется в математическом анализе, линейной алгебре, теории обыкновенных дифференциальных уравнений и др. (см. табл. 2.8).

Таблица 2.8 Сравнение математических обозначений с обозначениями Powersim

Название операции / объекта	**Математическое обозначение**	**Обозначение в Powersim**
Операция умножения	ab	a * b
Индекс	A_i	A(i), когда *i* – выражение, зависящее от индекса, или LOOKUP (A, I), когда *i* – обычное выражение.
Возведение в степень	x^2	x^2
Извлечение корня	$\sqrt{x}$	SQRT (x)
Суммирование по индексу	$\sum_{i=1}^{n} X_i$	SUM (i=1..n; X(i)) Если X – вектор или массив, то можно использовать функцию ARRSUM(X).
Интегрирование	$A = B + \int C \cdot dt$	level A = B + C*dt или aux A = B + INTEGRATE (C)
Сложная функция	$f(x) = \begin{cases} g(x); x < 0 \\ h(x); x = 0 \\ i(x); x > 0 \end{cases}$	aux f(x) = g(x) | x < 0; h(x) | x = 0; i(x) | x > 0
Вектор	$V=(V_1, V_2, V_3)$	aux V(1..3) = [v1, v2, v3] или aux V(1..3) = VECTOR(v1, v2, v3)
Матрица	$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$	aux A(1..3,1..3) = [[a11,a12,a13],[a21,a22,a23],[a31,a32,a33]]
Скалярное (внутреннее) произведение векторов	$a \cdot b$	SPROD (a, b)
Длина вектора	$\lvert v \rvert$	VECTLEN (v)
Первая производная по времени	$f'(t)$	DERIVN (F)
Вторая производная по времени	$f''(t)$	DERIVN (F, 2)

2.3.2 Функции и выражения в Powersim и Dynamo

Функции в Powersim и Dynamo довольно похожи, хотя имеется ряд отличий приведенных в табл. 2.9.

Таблица 2.9 Сравнение обозначений в Dynamo и Powersim

Dynamo	**Powersim**
(A)(B)	(A)*(B)
A**B	A^B
CLIP(P, Q, R, S)	IF (R >= S, P, Q)
DELAY1(In, Del)	DELAYMTR(In, Del, 1, 0)
DELAY3(In, Del)	DELAYMTR(In, Del, 3, 0)
DELAYP(In, Del, PPL)	DELAYMTR(In, Del, 3, 0)
DLINF3(In, Del)	DELAYINF(In, Del, 3, 0)
DT	TIMESTEP
FIFGE(P, Q, R, S)	IF (R >= S, P, Q)
LOGN(A)	LN (A)
NOISE()	RANDOM(-0.5, +0.5)
NORMRN(Среднее, Ст.откл.)	NORMAL(Среднее, Ст.откл.)
PULSE(H,W,F,I)	IF (TIMECYCLE(F, I, W),H,0)
SAMPLE(X, Интервал, X0)	SAMPLE(X, Интервал, 0, X0)
SMOOTH(In, Del)	DELAYINF(In, Del, 1, In)
SWITCH(P, Q, R)	IF (R = 0, P, Q)
TABHL(Y, X, XL, XH, Dx)	GRAPH(X, XL, Dx, Y) Powersim использует число элементов *Y* для вычисления значения *XH*, явно задаваемого в DYNAMO.
TABLE(Y, X, XL, XH, Dx)	GRAPH(X, XL, Dx, Y)
TABPL(Y, X, XL, XH, Dx)	GRAPHCURVE(X, XL, Dx, Y) В Powersim массив *Y* функции GRAPHCURVE не должен включать конечные нули, как в DYNAMO.
TABXT(Y, X, XL, XH, Dx)	GRAPHLINAS(X, XL, Dx, Y)

Хотя выражения в Powersim во многом схожи с выражениями в Dynamo (Professional Plus), есть определенные отличия, заключающиеся в основном в большем количестве типов в Dynamo и специфике работы с массивам, показанные в табл. 2.10 и проиллюстрированные на уравнениях одной из разобранных ранее моделей.

Таблица 2.10 Сравнение типов переменных в Dynamo и Powersim

Тип	**Dynamo**	**Powersim**
Вспомогательная переменная	A	AUX
Булева переменная	B	В Powersim булевого типа могут быть константы, вспомогательные переменные и переменные уровня.
Константа	C	CONST
Экзогенный параметр	E	В Powersim используются константы или параметры, определяемые динамическими или передаточными объектами.
Инициализируемое значение	I	INIT
Реинициализи-руемое значение	K	В Powersim используется функция INIT

Уровень	L	LEVEL
Вычисленное инициализируемое значение	N	INIT (Используется вместе с переменными уровня)
Параметр	P	—
Переменная темпа	R	В Powersim в качестве переменных темпа могут использоваться константы, вспомогательные переменные и переменные уровня.
Дополнительная переменная	S	В Powersim используются вспомогательные переменные.
Спецификация	SPEC	Только для обозначения установок моделирования (см. п. 1.5.1 и приложения)
Таблица	T	В Powersim в качестве таблиц могут использоваться константы, вспомогательные переменные и переменные уровня.

В Powersim в отличие от Dynamo не используются временные индексы .J, .K, .JK, и .KL, а также по другому оформляются массивы. Рассмотрим это на примере демографической модели "передвижки возрастов", усложненная модификация которой была изложена в п. 1.4.6. Для этой модели имеется следующая потоковая диаграмма (рис. 2.26):

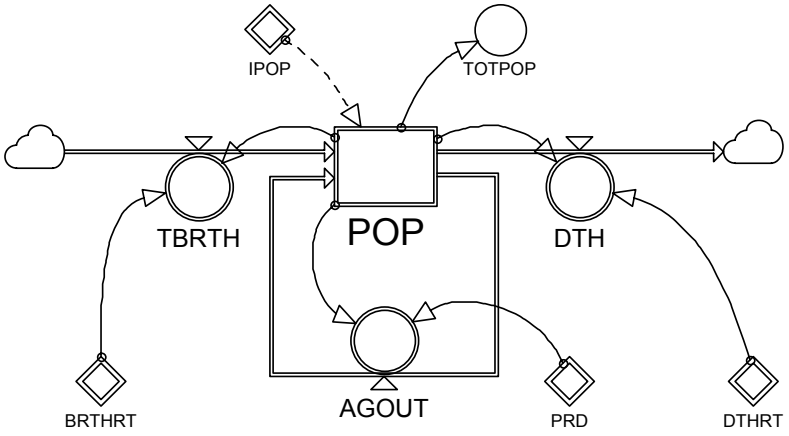


Рисунок 2.26 Потоковая диаграмма для модели "передвижки возрастов"

где *Pop* – Численность населения (чел.), *TBrth* – Рождения (количество родившихся людей, чел./год), *Dth* – Смерти (количество умерших людей, чел./год), *AgOut* – Старение (количество людей, перешедших из одной возрастной группы в другую, чел./год).

Уравнения модели в Powersim

```
range  Age=Youth,Yadlt,Mdlag,Oldst
range  Region =North,South
level  Pop(A=Age,R=Region)=IPop
       +dt*(TBrth(R)|A=Youth;0)+dt*(AgOut(A-1,R)|A>Youth;0)
       -dt*(AgOut)-dt*(Dth)
aux    AgOut(A=Age,Region)=Pop/Prd(A)
aux    Dth(A=Age,Region)=Pop*DthRt(A)
aux    TBrth(R=Region)=Pop(Yadlt,R)*BrthRt(Yadlt)+
       Pop(Mdlag,R)*BrthRt(Mdlag)
aux    TotPop=ARRSUM(Pop)
const  BrthRt(Yadlt..Mdlag)=[12E-3,5E-3]
const  DthRt(Age)=[1.9E-3,0.8E-3,6.6E-3,65E-3]
const  IPop(Age,Region)=[[1000,800],[950,750],[900,700],[400,1000]]
const  Prd(Age)=[12,25,20,1E30]
```

Уравнения модели в Dynamo[2]

```
FOR     AGE=YOUTH,YADLT,MDLAG,OLDST
FOR     AGE2=YADLT-OLDST
FOR     REGION=NORTH,SOUTH
N       POP(AGE,REGION)=IPOP(AGE,REGION)
I       IPOP(*,NORTH)=1000,950,900,400
I       IPOP(*,SOUTH)=800,750,700,1000
L       POP.K(YOUTH,REGION)=POP.J(YOUTH,REGION)+DT*(TBRTH.JK(REGION)
        -AGOUT.JK(YOUTH,REGION)-DTH.JK(YOUTH,REGION))
L       POP.K(AGE2,REGION)=POP.J(AGE2,REGION)+DT*(AGOUT.JK(AGE2-
        1,REGION)-AGOUT.JK(AGE2,REGION)-DTH.JK(AGE2,REGION))
R       TBRTH.KL(REGION)=BRTHRT(1)*POP.K(YADLT,REGION)
        +BRTHRT(2)*POP.K(MDLAG,REGION)
P       BRTHRT=12E-3,5E-3
R       AGOUT.KL(AGE,REGION)=POP.K(AGE,REGION)*DTHRT(AGE)
P       DTHRT=1.9E-3,0.8E-3,6.6E-3,65E-3
A       TOTPOP.K=SUM(POP.K)
```

2.3.3 Функции Powersim и Vensim

В Vensim и Powersim много общего как в отношении диаграммной техники (в Vensim те же типы потоков и уровней, но более простой интерфейс), так и в описании функций [см. табл. 2.11 для различающихся общедоступных (common) функций].

Таблица 2.11 Сравнение обозначений в VensimPLE32 3.0 и Powersim

VensimPLE32 3.0	**Powersim**
ACTIVE INITIAL(X,B)	IF(TIME<>STARTTIME, X, B)
DELAY1(In, Del)	DELAYMTR(In, Del, 1, 0)
DELAY1I(In, Del, Initial)	DELAYMTR(In, Del, 1, Initial)
DELAY3(In, Del)	DELAYMTR(In, Del, 3, 0)
DELAY3I(In, Del, Initial)	DELAYMTR(In, Del, 3, Initial)
DELAYP(In, Del: PPL)	DELAYMTR(In, Del, 3, 0)
DLINF3(In, Del)	DELAYINF(In, Del, 3, 0)
IF THEN ELSE(R, P, Q)	IF (R >< 0, P, Q)
INTEG(R,N)	INEGRATE(R)
INTEGER(X)	INT(X)
LOOKUP([(X_1,Y_1)-(X_N,Y_N)], (X_2,Y_2),...,(X_{N-1}, Y_{N-1}))	GRAPH(X, X_1, Dx, [Y_1,...,Y_N]). Powersim использует число элементов Y для вычисления значения Y_N, количества точек N и X_i.
PULSE(ST,W)	IF (TIME<ST+W,STEP(1,ST),0)
RUMP(S,ST,ET)	IF (TIME<ET,RUMP(S,ST),0)
RANDOM UNIFORM(Min,Max,0)	RANDOM(Min,Max)
SMOOTH(In, Del)	DELAYINF(In, Del, 1, In)
SMOOTHI(In, Del, In0)	DELAYINF(In, Del, 1, In0)
SMOOTH3(In, Del)	DELAYINF(In, Del, 3, In)
SMOOTH3I(In, Del, In0)	DELAYINF(In, Del, 1, In0)
TIME STEP	TIMESTEP
XIDZ(A,B,X)	DIVZX(A,B,X)
ZIDZ(A,B)	A DIV0 B

[2] Professional DYNAMO Plus: Reference Manual, 1986, P. 75.

2.3.4 Функции Powersim и Stella (iThink)

В Stella/iThink и Powersim много общего как в отношении диаграммной техники (в Stella/iThink более простой интерфейс), так и в описании функций (см. табл. 2.12 для различающихся функций).

Таблица 2.12 Сравнение обозначений в Stella/iThink 5.0 и Powersim

Stella/iThink 5.0	**Powersim 3.5c**
A % B	A MOD B
ARRAYMEAN(A[*])	ARRAVG(A) или ARRAVG([A1,A2,...,AN])
ARRAYSTDDEV(A[*])	ARRSTDDEV(A) или ARRSTDDEV([A1,A2,...,AN])
ARRAYSUM(A[*])	ARRSUM(A) или ARRSUM([A1,A2,...,AN])
CAP(Печь), CAP(Конвейер)	Печи и конвейеры не встроены в Powersim, но могут быть смоделированы явно.
CGROWTH(A*100%)	RAMP(A,0)
COOKTIME(Печь)	Печи не встроены в Powersim, но могут быть смоделированы явно.
COUNTER(A,B)	A+(TIME MOD (TIMESTEP*B))
DELAY(Вход, Длительность, Начальное значение)	DELAYPPL(Вход, Длительность, Начальное значение)
DT	TIMESTEP
FORCST(In, LT, FT)	FORCAST(In, LT, FT)
INT(X)	FLOOR(X)
LOG10(X)	LOG(X) или LOG(X, 10)
LOGN(X)	LN – натуральный логарифм X
MEAN(X, ...)	AVG(X, ...)
MOD(A, B)	A MOD B
MONTECARLO(P)	IF(RANDOM(0,100)>P,1,0)
OSTATE(Печь)	Печи не встроены в Powersim, но могут быть смоделированы явно.
PAUSE	PAUSE(TRUE) или PAUSE(1)
POISSON(Среднее)	POISSON(Среднее * TIMESTEP) / TIMESTEP
QELEM(Очередь, Элемент)	Можно смоделировать, используя функцию QUEUE и массивы.
QLEN(Очередь)	Длина очереди является выходным параметром функции QUEUE.
REWORK(A*100%)	Доля возврата (в %) продукции из текущего резервуара (уровня) в предыдущий может быть смоделирована явно.
SMTH1(Вход, Длительность, Начальное значение)	DELAYINF(Вход, Длительность, 1, Начальное значение)
SMTH3(Вход, Длительность, Начальное значение)	DELAYINF(Вход, Длительность, 3, Начальное значение)
SMTHN(Вход, Длительность, N, Начальное значение)	DELAYINF(Вход, Длительность, N, Начальное значение)
SUM(X, Y, Z...)	X+Y+Z+...
SWITCH(A, B)	IF(A > B, 1, 0) или (A > B)

В отличие от Powersim в Stella/iThink предусмотрено несколько типов переменных уровней (запасов): Reservoir (Резервуар), Conveyors (Конвейер), Queues (Очередь) и Ovens (Печь). Переменная уровня из Powersim соответствует Резервуару в Stella/iThink. Другие типы переменных уровней можно явно смоделировать при помощи комбинаций функций

Powersim. Например, конвейер можно смоделировать при помощи диаграммы, представленной на рис. 2.27:

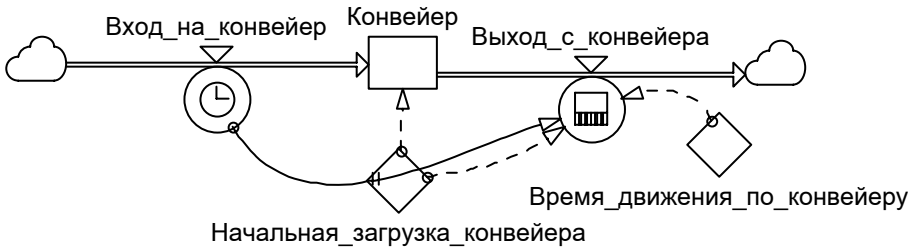


Рисунок 2.27 Потоковая диаграмма для конвейера

Данная диаграмма описывается следующей системой уравнений:

Уравнения уровней

```
level  Конвейер = Начальная_загрузка_конвейера -dt*Выход_с_конвейера
       +dt*Вход_на_конвейер
```

Уравнения темпов

```
aux    Вход_на_конвейер = ...
aux    Выход_с_конвейера = DELAYPPL(Вход_на_конвейер,
       Время_движения_по_конвейеру, Начальная_загрузка_конвейера)
```

Константы

```
const  Время_движения_по_конвейеру = ...
const  Начальная_загрузка_конвейера = ...
```

Очередь в Powersim можно смоделировать при помощи функции QUEUE (см. пример для функции QUEUE в п. 2.4.7). Печь можно смоделировать следующим образом:

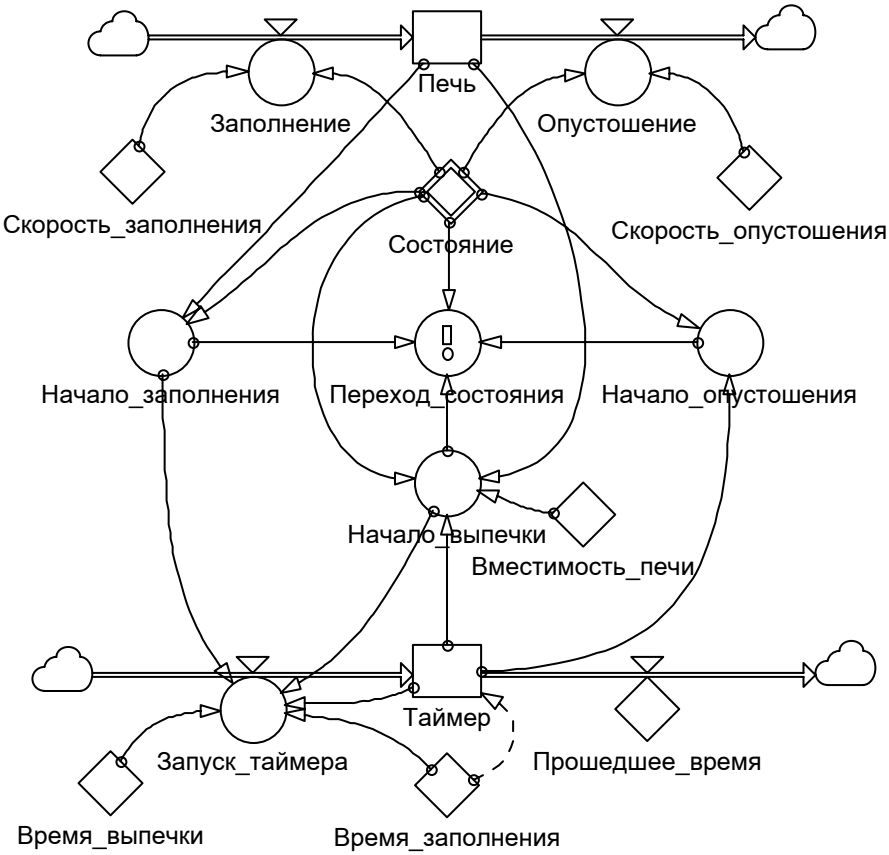


Рисунок 2.28 Потоковая диаграмма для печи

Данная диаграмма описывается следующей системой уравнений:

Уравнения уровней

```
init   Печь = 0
flow   Печь = -dt*Опустошение
       +dt*Заполнение
init   Таймер = Время_заполнения
flow   Таймер = -dt*Прошедшее_время
       +dt*Запуск_таймера
```

Уравнения темпов

```
aux    Заполнение = IF(Состояние(1),Скорость_заполнения,0)
aux    Опустошение = IF(Состояние(3),Скорость_опустошения,0)
aux    Начало_выпечки = Состояние(1) AND ((Таймер<=0) OR
       (Вместимость_печи<=Печь))
aux    Запуск_таймера = PULSEIF(Начало_заполнения,Время_заполнения-
       Таймер)+PULSEIF(Начало_выпечки,Время_выпечки-Таймер)
aux    Начало_заполнения = Состояние(3) AND (Печь<=0)
aux    Начало_опустошения = Состояние(2) AND (Таймер<=0)
aux    Переход_состояния = SHIFTCIF(Начало_заполнения OR
       Начало_опустошения OR Начало_выпечки, Состояние)
```

Константы

```
dim    Состояние = (1..3)
const  Состояние = [1,0,0]
const  Время_заполнения = ...
const  Скорость_заполнения = ...
const  Скорость_опустошения = ...
const  Время_выпечки = ...
const  Прошедшее_время = ...
const  Вместимость_печи = ...
```

2.4 Алфавитный перечень функций Powersim

2.4.1 Унарные и бинарные операторы

- Вычитание

Синтаксис	A - B
Вход	A, B – любые числа.
Результат	Разность A и B.
Группа	Математические функции

- Унарный минус

Синтаксис	- A
Вход	A – любое число.
Результат	Инвертированный, т.е. (0 - A).

! Факториал

Синтаксис	A!
Вход	A – любое число.
Результат	"Не число (=)?", если A < 0; 1, если A = 0; 1*2*3*4*...*A, если A – целое число. В остальных случаях рассчитывается через интеграл по времени INTEGRATE(F(t)) с подынтегральной функцией F(t) = EXP(- t)*t^A и пределами интегрирования от 0 до ∞.
Пример	3! = 6
Группа	Математические функции

% Процент

Синтаксис	A %
Вход	A – любое число.
Результат	Деление на 100.
Пример	A/100 = %
Замечание	% является правым оператором, преобразующим значение из процентов в десятичную дробь, например: 7 % = 7/100 = 0.07.
Группы	Функции преобразования типа, Математические функции
Родственные функции	PCT

*** Умножение**

Синтаксис	A * B
Вход	A, B – любые числа.
Результат	Умножение на B.
Группа	Математические функции
Родственные функции	ARRPROD, MATRIXPROD, PRODC, PRODCR, PRODRC, SPROD, VECTPROD3D

+ Сложение

Синтаксис	A + B
Вход	A, B – любые числа.
Результат	Сложение с B.
Группа	Математические функции
Родственные функции	ADD, ADDCR, ADDRC, ARRSUM, SUM

+ Унарный плюс

Синтаксис	+ A
Вход	A – любое число.
Результат	Неизменный.

/ Деление

Синтаксис A / B
Вход A – любое число, B – любое число, кроме 0.
Результат Деление на B.
Группа Математические функции
Родственные функции INVERT, INVERTC, MOD

; Оператор ограничения

Синтаксис A; B
Вход A, B – любые выражения ограничения.
Результат A если ограничение истинно, иначе – B.
Замечание Оператор ограничения (;) является встроенным оператором и работает вместе с оператором защиты (|) при выборе между альтернативными определениями выражений с массивами. Часто это бывает необходимо при математических операциях над первым или последним индексом по отдельности. Оператор BUT является альтернативой оператору ограничения (;).
Пример См. пример использования (;) в описании оператора защиты (|).
Группа Встроенные функции
Родственные функции |, WHEN, BUT

< Меньше

Синтаксис A < B
Вход A, B – любые числа.
Результат Истина при A < B, Ложь – в противном случае.
Группа Логические функции

< = Меньше или равно

Синтаксис A < = B
Вход A, B – любые числа.
Результат Истинна при A< = B, Ложь – в противном случае.
Группа Логические функции

< > Не равно

Синтаксис A < > B
Вход A, B – любые числа.
Результат Истина при A < > B, Ложь – в противном случае.
Замечание См. замечание для оператора равенства (=).
Группа Логические функции

= Равно

Синтаксис A = B
Вход A, B – любые числа.
Результат Истина при A = B, Ложь – в противном случае.
Замечание Проверки равенства по возможности следует избегать, так как ни один компьютер не способен представить все возможные числа с плавающей запятой точно. Это означает, что даже если результаты двух выражений теоретически должны быть одинаковы, то этого может не быть, когда вычисления выполняются на компьютере, как, например, в следующем "опасном" выражении: IF (TIME = 5, B, C).
Группа Логические функции
Родственные функции TIMEIS

> Больше

Синтаксис A > B
Вход A, B – любые числа.
Результат Истина при A > B, Ложь – в противном случае.
Группа Логические функции

> = Больше или равно

Синтаксис A > = B
Вход A, B – любые числа.
Результат Истина при A > = B, Ложь – в противном случае.
Группа Логические функции

^ Возведение в степень

Синтаксис A ^ B
Вход A, B – любые неотрицательные числа.
Результат Возвращает значение A, возведенное в степень B (например A^B).
Замечание A^0 = 1, 0^B = 0 для любых значений A и B.
Группа Математические функции

| Оператор защиты

Синтаксис A | B
Вход A – любое выражение, B – любое ограничение, например, выражение, включающее в себя переменные индекса, ссылки на диапазоны и целые числа.
Результат A при выполнении B, неопределенность – в противном случае
Замечание Оператор защиты (|) – встроенный оператор, работающий вместе с оператором ограничения (;) при выборе между альтернативными определениями выражений с массивами. Часто это бывает необходимо при математических операциях над первым или последним индексом по отдельности. Оператор WHEN является альтернативой оператору защиты (|).
Пример Копирование элементов из B в A:

```
aux  B(1..3) = ...
aux  A(i=1..3) = B(i+1)|i< 3; B(1)|i = 3
```

Результат выполнения:
A (1) = B (2)
A (2) = B (3)
A (3) = B (1)
Вышеупомянутый алгоритм математически формулируется так:

$$A_i = \begin{cases} B_{i+1}, & i < 3 \\ B_1, & i = 3 \end{cases}$$

Группа Встроенные функции
Родственные функции ;, BUT, WHEN

2.4.2 Функции от A до C

ABS – Абсолютное значение

Синтаксис ABS (X)
Вход X – любое число.
Результат Абсолютное значение X.
Пример ABS (-3) = 3
Группа Функции преобразования типа, Математические функции
Родственные функции ABSC

ABSC – Абсолютное значение комплексного числа или массива

Синтаксис ABSC (A([...,] 2))

Вход A – комплексное число или массив комплексных чисел в декартовой форме.

Результат Абсолютное значение A.

Пример ABSC ([3, 4]) = SQRT (3^2 + 4^2) = 5
ABSC ([[1, 2], [3, 4]]) = [2.2361, 5]

Группа Комплексные функции

Родственные функции ANGLEC, REALC, IMAGC

AND – Логическое И

Синтаксис A AND B

Вход A, B – любые числа.

Результат Истинна, если A и B истинны, и Ложь – в противном случае.

Замечание Оператор AND может также использоваться в определении ограничений, как в следующем примере: A(i, j) | i < 2 AND j > 5. Сходный (но не эквивалентный) способ выражения A AND B: A * B.

Группы Встроенные функции, Логические функции

Родственные функции BOOL, NOT, OR, XOR, TRUE, FALSE

ADD – Сумма массивов

Синтаксис ADD (A(...), B(...))

Вход A, B – массивы равной размерности.

Результат Массив суммы A и B.

Пример Определение переменной C:

```
aux C = ADD(A, B)
```

Тот же самый результат можно получить и в случае

```
aux C = A + B,
```

где A, B и C – массивы.

Замечание Комплексные числа в декартовой форме можно складывать, используя функцию ADD.

Группы Комплексные функции, Математические функции

Родственные функции ARRSUM

ADDCR – Сумма комплексных и вещественных чисел или массивов

Синтаксис ADDCR (A([D1, D2, ... Dn,] 2), B([D1, D2, ... Dn]))

Вход A – комплексное число или массив комплексных чисел в декартовой форме, B – вещественное число или массив вещественных чисел.

Результат Сумма A и B.

Пример ADDCR ([3, 4], 5) = [8, 4]

Группа Комплексные функции

Родственные функции ADD, ADDRC

ADDRC – Сумма вещественных и комплексных чисел или массивов

Синтаксис ADDRC (A([D1, D2, ... Dn]), B([D1, D2, ... Dn,] 2))
Вход A – вещественное число или массив вещественных чисел, B – комплексное число или массив комплексных чисел в декартовой форме.
Результат Сумма A и B.
Пример ADDRC (2, [3, 4]) = [5, 4]
Группа Комплексные функции
Родственные функции ADD, ADDCR

ANGLEC – Угол комплексного числа или массива

Синтаксис ANGLEC (A([...,] 2))
Вход A – комплексное число или массив комплексных чисел в декартовой форме.
Результат Угол A, измеряемый в радианах [-π .. π].
Примеры ANGLEC ([0, 1]) = 0
ANGLEC ([[1, 1], [1, -1]]) = [0.7854, 2.3562]
Группа Комплексные функции
Родственные функции ABSC, REALC, IMAGC

ARCCOS – Арккосинус

Синтаксис ARCCOS (X)
Вход Значение угла X в радианах из интервала [-1 .. 1].
Результат Угол, определенный арксинусом входного значения. Угол лежит в интервале [-π /2 .. π /2].
Группа Тригонометрические функции
Родственные функции ARCSIN, ARCTAN, COS, DEGTORAD, GRADTORAD

ARCSIN – Арксинус

Синтаксис ARCSIN (X)
Вход Значение угла X в радианах из интервала [-1 .. 1].
Результат Угол, определенный арккосинусом входного значения. Угол лежит в интервале [0 .. π].
Группа Тригонометрические функции
Родственные функции ARCCOS, ARCTAN, SIN, DEGTORAD, GRADTORAD

ARCTAN – Арктангенс

Синтаксис ARCTAN (X)
Вход Любое значение угла X в радианах.
Результат Угол, определенный арктангенсом входного значения. Угол лежит в интервале (- π /2 .. π /2).
Группа Тригонометрические функции
Родственные функции ARCCOS, ARCSIN, TAN, DEGTORAD, GRADTORAD

ARRAVG – Среднее значение элементов массива

Синтаксис ARRAVG (A(...))
Вход A – любой массив.
Результат Среднее значение элементов A.
Пример ARRAVG ([3, 5, 1, 7]) = 4
Группы Функции для работы с массивами, Статистические функции
Родственные функции AVG, ARRMIN, ARRMAX, ARRPROD, ARRSUM

ARRMAX – Максимальный элемент массива

Синтаксис	ARRMAX (A(...))
Вход	A – любой массив.
Результат	Максимальное значение элементов A.
Пример	ARRMAX ([3, 5, 1, 7]) = 7
Группы	Функции для работы с массивами, Статистические функции
Родственные функции	MAX, ARRMIN, ARRAVG, ARRPROD, ARRSUM

ARRMIN – Минимальный элемент массива

Синтаксис	ARRMIN (A(...))
Вход	A – любой массив.
Результат	Минимальное значение элементов A.
Пример	ARRMIN ([3, 5, 1, 7]) = 1
Группы	Функции для работы с массивами, Статистические функции
Родственные функции	MIN, ARRMAX, ARRAVG, ARRPROD, ARRSUM

ARRPROD – Произведение элементов массива

Синтаксис	ARRPROD (A(...))
Вход	A – любой массив.
Результат	Произведение элементов A.
Пример	ARRPROD ([3, 5, 1, 7]) = 105
Группы	Функции для работы с массивами, Математические функции
Родственные функции	*, ARRAVG, ARRMAX, ARRMIN, ARRSUM, MATRIXPROD, PRODC, PRODCR, PRODRC, SPROD, VECTPROD3D

ARRSTDDEV – Стандартное отклонение элементов массива

Синтаксис	ARRSTDDEV (X(N))
Вход	X – любой массив.
Результат	Стандартное отклонение элементов массива определяется выражением: SQRT (SUM(i = 1..N; (X - ARRAVG (X))^2)) / N), которое можно упростить так: 1/N * SQRT(N* SUM(1..N; X^2) - ARRSUM (X)^2).
Группы	Функции для работы с массивами, Статистические функции
Родственные функции	STDDEV

ARRSUM – Сумма элементов массива

Синтаксис	ARRSUM (A(...))
Вход	A – любой массив.
Результат	Сумма элементов A.
Пример	ARRSUM ([3, 5, 1, 7]) = 16
Группы	Функции для работы с массивами, Математические функции
Родственные функции	+, ADD, ADDCR, ADDRC, ARRAVG, ARRMAX, ARRMIN, ARRPROD, SUM

ASSIGN – Присвоение значения переменной

Синтаксис	ASSIGN (Переменная, X)
Вход	X – любое число.
Выход	Переменная (переменная уровня или константа).
Результат	Значение X.

Замечание ASSIGN – одна из немногих функций, которые имеют побочные эффекты. В данном случае значение первого параметра, который должен быть переменной уровня или константой, заменяется значением второго параметра. Присвоение переменной значения в конце текущего шага моделирования означает, что новое значение переменной будет воздействовать на следующем шаге моделирования. Следовательно, можно утверждать, что использование функции присвоения подразумевает задержку на один шаг моделирования.

Пример Обнуление A при выполнении условия B: IF (B, ASSIGN (A, 0), A). Отметим, что функция IF не вычисляет функцию ASSIGN (A, 0), если B ложно. Если это было бы не так, то функция ASSIGN (A, 0) всегда вычислялась бы и, следовательно, значение A всегда оставалось бы нулевым.

Демо-версия ASSIGN.SIM

Группа Смешанные функции

ATSTART – Проверка на начало моделирования

Синтаксис ATSTART

Результат Истина в самом начале моделирования, Ложь – в противном случае.

Группа Функции, зависящие от времени

Родственные функции STARTTIME, STOPTIME, TIIME, TRUE, FALSE

AVG – Среднее значение (среднее)

Синтаксис AVG (X1, X2, ... , XN)

Вход X1, X2, ..., XN – любые числа (N > = 1).

Результат Среднее значение аргументов: (X1, X2, ..., XN) /N.

Замечание По крайней мере один из аргументов должен быть определен.

Группа Статистические функции

Родственные функции ARRAVG

BOOL – Преобразование числа к булевой форме

Синтаксис BOOL (X)

Вход X – любое число.

Результат 1 при ROUND (X) ≠ 0, 0 – в противном случае.

Пример BOOL (X) = (ROUND (X) <> 0).

Группы Функции преобразования типа, Логические функции

Родственные функции TRUE, FALSE

BUT – Оператор ограничения

Синтаксис A BUT B

Вход A, B – любые выражения ограничений.

Результат A, если ограничение A истинно, B – в противном случае.

Замечание Альтернативное название для оператора ограничения (;).

Пример См. описание оператора защиты (|).

Группа Встроенные функции

Родственные функции WHEN, |, ;

CARTOPOL – Преобразование комплексного числа или массива из декартовой формы в полярную

Синтаксис CARTOPOL (A([...,] 2))
Вход A – комплексное число или массив комплексных чисел в декартовой форме.
Результат Переменная или массив A в полярной форме.
Примеры CARTOPOL ([0, 1]) = [1, 1.5798]
CARTOPOL (POLTOCAR (A)) = A
Группы Комплексные функции, Функции преобразования типа
Родственные функции POLTOCAR, ANGLEC, ABSC

CEIL – Округление вверх до ближайшего целого

Синтаксис CEIL (X)
Вход X – любое число.
Результат Возвращает наименьшее целое число, которое $\geq$ X.
Группа Функции преобразования типа
Родственные функции FLOOR, ROUND, INT, FRAC

COS – Косинус

Синтаксис COS (X)
Вход X – произвольный угол в радианах.
Результат Косинус угла X.
Группа Тригонометрические функции
Родственные функции SIN, TAN, ARCCOS, COSH, DEGTORAD, GRADTORAD, PI

COSH – Гиперболический косинус

Синтаксис COSH (X)
Вход X – любое число.
Результат Гиперболический косинус X.
Группа Тригонометрические функции
Родственные функции SINH, TANH, COS, DEGTORAD, GRADTORAD, PI

COSWAVE – Косинусоидальная волна

Синтаксис COSWAVE (A, P)
Вход A – амплитуда волны, P – период волны.
Результат Зависящая от времени косинусоидальная волна, определяемая уравнением: COSWAVE (A, P) = A * COS (TIME*2*PI/P).
Замечание Функция зависит от времени.
Группа Функции, зависящие от времени, Тригонометрические функции
Родственные функции COS, SINWAVE, TIME

COUNT – Получение числа элементов диапазона или главного индекса диапазона

Синтаксис COUNT (R)
Вход R – название диапазона или переменной индекса.
Результат Число элементов R, если R – диапазон. Число элементов в основном диапазоне R, если R – переменная индекса.

Пример
```
range  Возраст = (Детский,Юношеский,Средний,Пожилой)
const  Возрасты = COUNT(Возраст)
```

Замечание Связь между функциями COUNT, FIRST и LAST следующая:
COUNT (R) = LAST (R) – FIRST (R) + 1.
Группа Встроенные функции
Родственные функции FIRST, LAST

2.4.3 Функции от D до F

DEFAULT – Выбор выражения ограничения по умолчанию

Синтаксис DEFAULT

Результат Истина для всех индексов, которые не являются частью (предыдущих) операторов защиты для данного выражения.

Пример Эквивалентные выражения:
(A(i + 1) | i < 4; A(1) | DEFAULT)
(A(i + 1) | i < 4; A(1))

Замечание Функция DEFAULT работает аналогично пустой защите. Преимуществом является явное объявление "иных" случаев, включающих все случаи, не описанные другими операторами защиты.

Группа Встроенные функции

DEGTOGRAD – Преобразование градусов в грады

Синтаксис DEGTOGRAD (X)

Вход X – угол в градусах.

Результат Эквивалент угла X, измеренного в градах:
DEGTOGRAD (A) = A*400/360.

Группы Функции преобразования типа, Тригонометрические функции

Родственные функции DEGTORAD, GRADTODEG

DEGTORAD – Преобразование градусов в радианы

Синтаксис DEGTORAD (X)

Вход X – угол в градусах.

Результат Эквивалент угла X, измеренного в радианах:
DEGTORAD (A) = A*PI/180.

Группа Функции преобразования типа, Тригонометрические функции

Родственные функции DEGTORAD, RADTODEG, PI

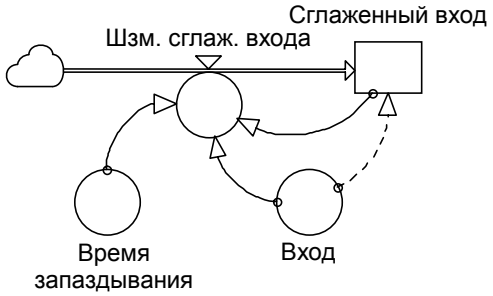
DELAYINF – Информационная задержка N-го порядка

Синтаксис DELAYINF (Вход, Время запаздывания [, Порядок=1 [, Начальное значение (Порядок) = Вход]])

Вход *Вход* – переменная, на которую действует задержка (параметр с запаздыванием). *Время запаздывания* – время, измеряемое в единицах времени моделирования. *Порядок* – положительное целое число, определяющее порядок задержки. Если порядок не задан пользователем, то по умолчанию он равен 1 (см. произвольные начальные параметры в п. 2.1.4). *Начальное значение* – значение задержки, определенное как векторное выражение с числом элементов, равным *Порядку*. По умолчанию, элементы вектора *Начального значения* равны значению *Входа*, если они не заданы явно. Если этот вектор имеет меньшее количество элементов, чем *Порядок*, остальные элементы (см. произвольные начальные параметры) вектора приравниваются последнему элементу вектора.

Результат Экспоненциальная информационная задержка N-го порядка для переменной Входа, использующая время экспоненциального усреднения, равное *Времени запаздывания*, данный *Порядок* и *Начальное значение* задержки.

График Информационная задержка 1-го порядка может быть смоделирована следующим образом:



Значение уровня *Сглаженный_вход* (Выход) соответствует экспоненциальной информационной задержке DELAYINF 1-го порядка (см. п. 2.2.7).

Уравнения

```
init    Сглаженный_вход = ...
flow    Сглаженный_вход = +dt*(Изм._сглаж._входа)
aux     Изм._сглаж._входа = (Вход-Сглаженный_вход)/
        Время_запаздывания
aux     Вход = ...
aux     Время_запаздывания = ...
```

Замечание — *Вход* должен быть переменной [выражения и литералы (именованные числа или массивы) не допускаются], а на диаграмме он должен быть соединен с текущей переменной (*Сглажееный_вход*) при помощи связи с запаздыванием. Если *Начальное значение* не определено, или если *Вход* используется для определения *Начального значения*, то следует использовать на диаграмме обычную связь для соединения *Входа* с текущей переменной. Это связано с тем, что *Вход* будет использоваться и как обычный параметр и как параметр с запаздыванием. *Порядок* должен быть меньше величины *Время_запаздывания / Шаг моделирования*. Иначе функция не будет вести себя правильно (*Время_запаздывания* будет равно значению *Порядок* * *Шаг моделирования*).

Группы — Функции задержки, Функции с памятью

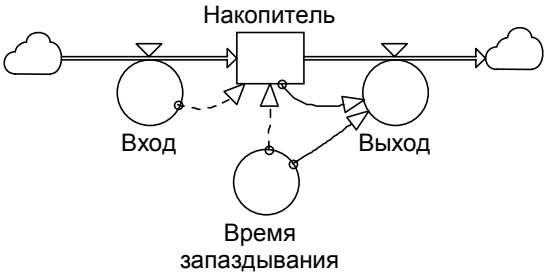
DELAYMTR – Материальная задержка N-го порядка

Синтаксис — DELAYMTR(Вход, Время запаздывания [, Порядок=1 [, Начальное значение (Порядок) = Вход]])

Вход — *Вход* – переменная, на которую действует задержка (параметр с запаздыванием). *Время запаздывания* – время, измеряемое в единицах времени моделирования. *Порядок* – положительное целое число, определяющее порядок задержки. Если порядок не задан пользователем, то по умолчанию он равен 1 (см. произвольные начальные параметры в п. 2.1.4). *Начальное значение* – значение задержки, определенное как векторное выражение с числом элементов, равным *Порядку*. По умолчанию, элементы вектора *Начального значения* равны значению *Входа*, если они не заданы явно. Если этот вектор имеет меньшее количество элементов, чем *Порядок*, остальные элементы (см. произвольные начальные параметры) вектора приравниваются последнему элементу вектора.

Результат — Экспоненциальная материальная задержка N-го порядка для переменной *Входа*, использующая время экспоненциального усреднения, равное *Времени запаздывания*, *Порядок* и *Начальное значение* задержки.

График — Материальная задержка 1-го порядка может быть смоделирована следующим образом:



Значение переменной *Выхода* соответствует экспоненциальной материальной задержке DELAYMTR 1-го порядка (см. п. 2.2.7).

Уравнения

```
init    Накопитель = ...
flow    Накопитель = - dt*(Выход)+ dt*(Вход)
aux     Вход = ...
aux     Выход = Накопитель/Время_запаздывания
aux     Время_запаздывания = ...
```

Замечание См. замечание для функции DELAYINF.

Группы Функции задержки, Функции с памятью

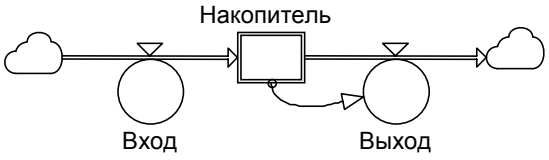
DELAYPPL – Конвейерная (канальная, дискретная) задержка

Синтаксис DELAYPPL (Вход, Время запаздывания [, Начальное значение = Вход])

Вход *Вход* – переменная, на которую действует задержка (параметр с запаздыванием). *Время запаздывания* – время, измеряемое в единицах времени моделирования (см. произвольные начальные параметры в п. 2.1.4). *Начальное значение* – значение задержки (см. произвольные начальные параметры), равное по умолчанию значению *Входа*.

Результат Значение *Входа* в момент времени, равный *t* - *Время запаздывания*, где *t* – текущий момент времени. До тех пор пока *t* < *Время запаздывания* возвращается *Начальное значение*, представляющее собой вектор с одним ненулевым элементом на каждом шаге моделирования и числом элементов, равным *Времени запаздывания / Шаг моделирования*.

График Конвейерную задержку можно смоделировать, используя вектор с количеством элементов, равным *Времени запаздывания* / *Шаг моделирования*, который линейно сдвигается слева направо на каждом шаге моделирования:



Уравнения

```
aux     Вход = ...
init    Накопитель = ...
dim     Накопитель = 1 .. 10
flow    Накопитель (i) = dt*(Вход|i=1;0)-
        dt*(Вывод|i=FIRST(i);0)
aux     Выход =  SHIFTLIF(TRUE, Накопитель)
```

Число элементов вектора *Накопителя* должно равняться числу шагов в интервале, равном *Времени запаздывания*, т.е. величине *Время запаздывания /Шаг моделирования*. См. также в п. 2.2.7.

Замечание См. замечание для функции DELAYINF

Группы Функции задержки, Функции с памятью

DELAYPPLINF – Конвейерная информационная задержка с переменным временем запаздывания

Синтаксис DELAYPPLINF (Вход, Время запаздывания, Максимальное время запаздывания [, Начальное значение = Вход])

Вход *Вход* – переменная, на которую действует задержка (параметр с запаздыванием). *Время запаздывания* – время, измеряемое в единицах времени моделирования. *Максимальное время запаздывания* – время, измеряемое в единицах времени моделирования и используемое в комбинации с *Шагом моделирования* для определения максимального размера оперативной памяти, используемое функцией. *Начальное значение* – значение задержки (см. произвольные начальные параметры в п. 2.1.4), равное по умолчанию значению *Входа*.

Результат Информационная задержка "бесконечного" порядка для *Входа* с заданным *Временем запаздывания*.

Замечание Разница между функциями DELAYPPLINF и DELAYPPL состоит в том, что DELAYPPLINF подстраивается под изменения *Времени запаздывания* в течение всего периода моделирования, в то время как DELAYPPL использует начальное значение *Времени запаздывания* на протяжении всего периода моделирования. Разница между функциями DELAYPPLMTR и DELAYPPLINF заключается в различном отклике на изменение *Времени запаздывания* (см. Функции задержки). Используйте функцию DELAYPPL при постоянном *Времени запаздывания*.

Группы Функции задержки, Функции с памятью

DELAYPPLINFV – Векторная конвейерная информационная задержка с переменным временем запаздывания

Синтаксис DELAYPPLINFV (Вход, Время запаздывания, V (N) [, Порядок])

Вход *Вход* – переменная, на которую действует задержка (параметр с запаздыванием). *Время запаздывания* – время, измеряемое в единицах времени моделирования. *Максимальное время запаздывания*, используемое данной функцией равно *N* * *Шаг моделирования*, где N – число элементов вектора V. V – вектор из N элементов, где число элементов N следует определять по формуле: N = ROUND (*Максимальное время запаздывания* / *Шаг моделирования*), где *Максимальное время запаздывания* – максимальное значение, которое можно выбрать для *Времени запаздывания* в течение периода моделирования.

Выход V – вектор, подбираемый для отражения состояния задержки. *Порядок* – число используемых элементов вектора V (произвольный выходной параметр). *Порядок* определяется по текущему *Времени запаздывания* и *Шага моделирования*: *Порядок* = ROUND (*Время запаздывания* / *Шаг моделирования*)

Результат Информационная задержка "бесконечного" порядка для *Входа* с заданным *Временем запаздывания*.

Замечание Начальное состояние задержки определяется начальным значением вектора V. Разница между функциями DELAYPPLMTRV и DELAYPPLINFV заключается в способе обработки изменений *Времени запаздывания*. Допустим, что изменение *Времени запаздывания* вызывает изменение *Порядка* с 2 до 4 и что вектор V первоначально имеет значения [4, 2, ...].

В случае информационной задержки новые значения вектора V будут равны [4, 4, 2, 2, ...]. Видно, что среднее значение (используемой части) вектора V остается неизменной, в то время как в случае материальной задержки результирующий вектор V был бы равен [2, 2, 1, 1, ...].

Группы Функции для работы с массивами, Функции задержки, Функции с памятью

DELAYPPLMTR – Конвейерная материальная задержка с переменным временем запаздывания

Синтаксис DELAYPPLMTR (Вход, Время задержки, Максимальное время задержки [, Начальное значение = Вход])

Вход *Вход* – переменная, на которую действует задержка (параметр с запаздыванием). *Время запаздывания* – время, измеряемое в единицах времени моделирования. *Максимальное время запаздывания* – время, измеряемое в единице времени моделирования и используемое в комбинации с *Шагом моделирования* для определения максимального размера оперативной памяти, используемое функцией. *Начальное значение* – значение задержки (см. произвольные начальные параметры в п. 2.1.4), равное по умолчанию значению *Входа*.

Результат Материальная задержка "бесконечного" порядка для *Входа* с заданным *Временем запаздывания*.

Замечание Разница между функциями DELAYPPLMTR и DELAYPPL состоит в том, что DELAYPPLMTR подстраивается под изменения *Времени запаздывания* в течение всего периода моделирования, в то время как DELAYPPL использует начальное значение *Времени запаздывания* на протяжении всего периода моделирования. Разница между функциями DELAYPPLMTR и DELAYPPLINF заключается в различном отклике на изменение *Времени запаздывания* (см. Функции задержки). Используйте DELAYPPLMTRV, если необходимо обратиться к внутреннему состоянию задержки. Используйте функцию DELAYPPL при постоянном *Времени запаздывания*.

Группы Функции задержки, Функции с памятью

DELAYPPLMTRV – Векторная конвейерная материальная задержка с переменным временем запаздывания

Синтаксис DELAYPPLMTRV (Вход, Время задержки, V (N) [, Порядок])

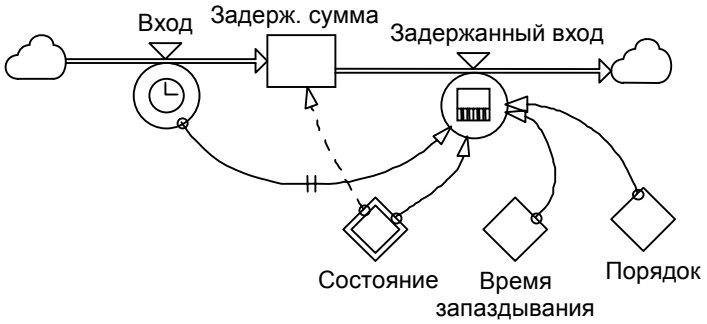
Вход *Вход* – переменная, на которую действует задержка (параметр с запаздыванием). *Время запаздывания* – время, измеряемое в единицах времени моделирования. *Максимальное время запаздывания*, используемое данной функцией равно *N* * *Шаг моделирования*, где N – число элементов вектора V. V – вектор из N элементов, где число элементов N следует определять по формуле: N = ROUND (*Максимальное время запаздывания* / *Шаг моделирования*), где *Максимальное время запаздывания* – максимальное значение, которое можно выбрать для *Времени запаздывания* в течение периода моделирования.

Выход V – вектор, подбираемый для отражения состояния задержки. *Порядок* – число используемых элементов вектора V (произвольный выходной параметр). *Порядок* определяется по текущему *Времени запаздывания* и *Шага моделирования*: *Порядок* = ROUND (*Время запаздывания* / *Шаг моделирования*)

Результат Материальная задержка "бесконечного" порядка для *Входа* с заданным *Временем запаздывания*.

Замечание Начальное состояние задержки определяется начальным значением вектора V. Разница между функциями DELAYPPLMTRV и DELAYPPLINFV заключается в способе обработки изменений *Времени запаздывания*. Допустим, что изменение *Времени запаздывания* вызывает изменение *Порядка* с 2 до 4 и что вектор V первоначально имеет значения [4, 2, ...]. В случае материальной задержки, новые значения вектора V будут равны [2, 2, 1, 1, ...]. Видим, что сумма значений (используемой части) вектора V остается неизменяемой, в то время как в случае информационной задержки результирующий вектор V был бы равен [4, 4, 2, 2, ...].

Пример Рисунок ниже отражает контекст использования DELAYPPLMTRV.



Уравнения:

```
init    Задерж._сумма = ARRSUM (Состояние)
flow    Задерж._сумма = -dt*Задержанный_вход +dt*Вход
aux     Вход = IF(TIME < 50, SIN(TIME/20), 0)
aux     Задержанный_вход = DELAYPPLINFV(Вход,
        Время_запаздывания, Состояние, Порядок)
const   Состояние (1..12) = 0
const   Время_запаздывания = 3
const   Порядок = 0
```

Группы Функции для работы с массивами, Функции задержки, Функции с памятью

DERIVN – Производная N-го порядка по времени

Синтаксис DERIVN (Вход [, Порядок = 1])

Вход *Вход* – выражение, от которого ищется производная. *Порядок* – порядок производной, по умолчанию равный 1 (см. произвольные начальные параметры).

Результат Возвращает производную N-го порядка по времени для *Входа*, определяемую по формуле:

$$\frac{\Delta x}{\Delta t} = \frac{x(t) - x(t - \Delta t)}{\Delta t}.$$

Замечание Результат функции зависит от предыдущих значений *Входа*.

Группы Функции с памятью, Математические функции

Родственные функции INTEGRATE

DIVZ0 – Деление с нулевым результатом при обращении знаменателя в нуль

Синтаксис A DIVZ0 B
Вход A, B – любые числа.
Результат IF (B > < 0, A/B, 0)
Группа Математические функции
Родственные функции DIVZ1, DIVZX

DIVZ1 – Деление с единичным результатом при обращении знаменателя в нуль

Синтаксис A DIVZ1 B
Вход A, B – любые числа.
Результат IF (B > < 0, A/B, 1)
Группа Математические функции
Родственные функции DIVZ0, DIVZX

DIVZX – Деление с заданным результатом при обращении знаменателя в нуль

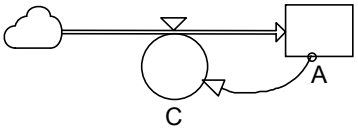
Синтаксис DIVZX (A, B, X)
Вход A, B, X – любые числа.
Результат IF (B > < 0, A/B, X)
Группа Математические функции
Родственные функции DIVZ0, DIVZ1

ELEMCOUNT – Число элементов массива

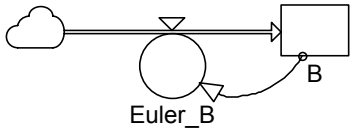
Синтаксис ELEMCOUNT (A(...))
Вход A – любой массив.
Результат Число элементов в массиве A.
Пример ELEMCOUNT ([3, 5, 1, 7]) = 4
Группа Функции для работы с массивами
Родственные функции VECTLEN

EULER – Выборка в начале шага моделирования

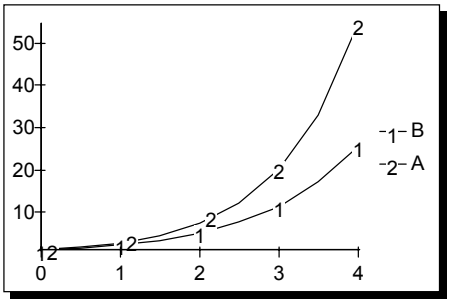
Синтаксис EULER (X)
Вход X – любая переменная (вычисляемый параметр)
Результат Значение X в начале текущего временного шага.
Замечание X должен быть переменной (выражения и литералы не допускаются), и на диаграмме X следует соединить с текущей переменной, используя связь с запаздыванием. Эта функция иногда оказывается полезной наряду с методами интегрирования более высокого порядка, определенными в диалоговом окне Simulation Setup (Установки моделирования), в которых между соседними шагами моделирования производятся дополнительные промежуточные вычисления. Функцию EULER можно использовать, например, если мы хотим, чтобы какой-либо из коэффициентов оставался постоянным в течение промежуточных вычислений при использовании методов интегрирования более высокого порядка.
Пример Две описываемые ниже модели приводят к одинаковому результату при интегрировании по методу Эйлера и к разным результатам при выборе метода интегрирования более высокого порядка (например, Рунге-Кутта 4):




```
init      A = 1
flow      A = + dt*C
aux       C = A
```




```
init      B = 1
flow      B = + dt*Euler_B
aux       Euler_B = EULER(B)
```



Итоги имитации при использовании метода Рунге-Кутта4 (RK4) с фиксированным шагом, равным 0.5.

Группа Функции с памятью
Родственные функции INIT, SAMPLE, SAMPLEIF

EXP – Экспонента числа

Синтаксис EXP (X)
Вход X – любое число.
Результат Возвращается экспонента числа X, т.е. число *e*, возведенное в степень X.
Замечание EXP – это обратная функция для натурального логарифма LN, т.е. LN (EXP (X)) = X.
Группа Математические функции
Родственные функции ^, LN

EXPRND – Экспоненциальное распределение

Синтаксис EXPRND ([Среднее = 1 [, Seed]])
Вход *Среднее* – среднее значение распределения (произвольный параметр, по умолчанию равный 1). *Seed* – параметр, отвечающий инициализацию генератора случайных чисел (произвольный начальный параметр, по умолчанию равный любому значению).
Результат Ряд экспоненциально распределенных случайных чисел с заданным значением *Среднего*.
Примеры EXPRND генерирует экспоненциально распределенные случайные числа со средним значением, равным 1.
EXPRND (5) также как и 5*EXPRND генерирует ряд экспоненциально распределенных чисел со средним значением 5.
Группа Случайные (стохастические) функции
Родственные функции NORMAL, POISSON, RANDOM

FALSE – Логическая ЛОЖЬ

Синтаксис FALSE
Результат Нулевое значение.
Группа Логические функции
Родственные функции THUE

FIRST – Нижний предел диапазона или главного индекса диапазона

Синтаксис FIRST (R)

Вход R – название диапазона или переменная индекса.

Результат Нижний предел R, если R – диапазон или нижний предел основного диапазона R, если R – переменная индекса.

Пример Вычисление численности людей, которые перейдут из одной возрастной группы в другую из-за взросления. Путь переменная ALen обозначает длительность (продолжительность) каждой возрастной группы, а AIn – численность населения, пополняющего данную возрастную группку (см. также пример в п. 1.4.6). Тогда имеем:

```
range    Возраст = Детский,Юношеский,Средний,Пожилой)
aux      AIn(A=Возраст) = Население(A-1)/ALen(A-1)|
         A > FIRST(A); 0
```

Группа Встроенные функции

Родственные функции COUNT, LAST

FLOOR – Округление вниз до ближайшего целого

Синтаксис FLOOR (X)

Вход X – любое число.

Результат Возвращает ближайшее целое число, которое ≤ X.

Примеры FLOOR (3.14) = 3

FLOOR (-5.5) = - 6

Группа Функции преобразования типа

Родственные функции CEIL, INT, FRAC, ROUND

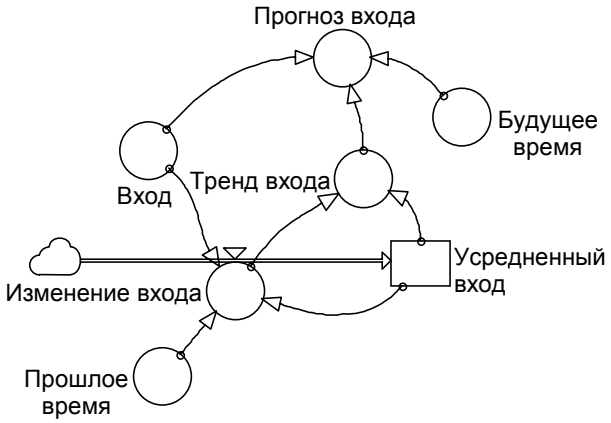
FORCAST – Прогнозирование значения

Синтаксис FORCAST (Вход, Прошлое время, Будущее время [, Начальное значение = 0])

Вход *Вход* – значение, для которого будет выполнен прогноз. *Прошлое время* – положительное число, определяющее время усреднения, используемое при вычислении входного тренда, измеряемое в единицах времени моделирования. *Будущее время* – неотрицательное число, определяющее глубину (горизонт) прогноза, измеряемое в единицах времени моделирования. *Начальное значение* – начальное значение тренда (произвольный начальный параметр, по умолчанию равный нулю).

Результат Прогнозное значение *Входа* в момент времени, равный *Будущему времени*. Данная функция вычисляет экспоненциальное среднее первого порядка для значения Входа, используя время усреднения, равное *Прошлому времени*, а затем экстраполирует тренд на расстояние, равное *Будущему времени*.

График Следующая диаграмма показывает использование функции TREND, и экстраполирует значение *Входа* в будущее.




```
init    Усредненный_вход = ...
flow    Усредненный_вход = + dt*Изменение_входа
aux     ChangeInInput = (Вход - Усредненный_вход)/
        Прошлое_время
aux     Прогноз_входа=Вход + Вход*Тренд_входа*
        Будущее_время
aux     Тренд_входа=Изменение_входа/Усредненный_вход
aux     Вход = ...
aux     Будущее_время = ...
aux     Прошлое_время = ...
```

Замечание Функция зависит от предыдущих значений первого параметра.
Группа Функции с памятью
Родственные функции TREND

FRAC – Дробная часть числа

Синтаксис FRAC (X)
Вход X – любое число.
Результат Дробная (десятичная) часть X.
Замечание Связь между X, INT (X) и FRAC (X) следующая (для всех значений X):
X = INT (X) + FRAC (X)
Примеры FRAC (3.14) = 0.14
FRAC (-5.5) = - 0.5
Группа Функции преобразования типа
Родственные функции CEIL, INT, FLOOR, ROUND

FV – Будущая ценность

Синтаксис FV (Ставка, Периоды, Платеж, Текущая ценность)
Вход *Ставка* – ставка процента за период. *Периоды* – число периодов. *Платеж* – сумма денег, вносимая периодически.
Текущая ценность – современная ценность денег.
Результат Будущая ценность денег (см. п. 2.2.8).
Демо-версия FINANCE.SIM
Группа Финансовые функции
Родственные функции NPV, PMT, PV

2.4.4 Функции от G до I

GETPLAYERS – Количество игроков

Синтаксис GETPLAYERS

Результат Количество игроков, персонально участвующих в игре, определяемое из файла установок игры.

Группа Смешанные функции

Родственные функции GETSIMPLAYERS, GETTOTPLAYERS, PLAYERNUMBER

GETSIMPLAYERS – Количество моделируемых игроков

Синтаксис GETSIMPLAYERS

Результат Количество моделируемых игроков, участвующих в игре, определяемое из файла установок игры.

Группа Смешанные функции

Родственные функции GETPLAYERS, GETTOTPLAYERS, PLAYERNUMBER

GETTOTPLAYERS – Общее количество игроков

Синтаксис GETTOTPLAYERS

Результат Количество персональных и моделируемых игроков, участвующих в игре, определяемое из файла установок игры.

Группа Смешанные функции

Родственные функции GETPLAYERS, GETSIMPLAYERS, PLAYERNUMBER

GRADTODEG – Преобразование градов в градусы

Синтаксис GRADTODEG (X)

Вход Угол X – угол в градах.

Результат Эквивалент угла X, измеренного в радианах:
GRADTODEG (A) = A*360/400.

Группа Функции преобразования типа, Тригонометрические функции

Родственные функции GRADTORAD, DEGTORAD

GRADTORAD – Преобразование градов в радианы

Синтаксис GRADTORAD (X)

Вход Угол X – угол в градах.

Результат Эквивалент угла X, измеренного в радианах:
GRADTORAD (A) = A*PI/200.

Группа Функции преобразования типа, Тригонометрические функции

Родственные функции GRADTODEG, RADTOGRAD, PI

GRAPH – Кусочно-линейный график с горизонтальными асимптотами

Синтаксис GRAPH (X, X1, Dx, Y(N))

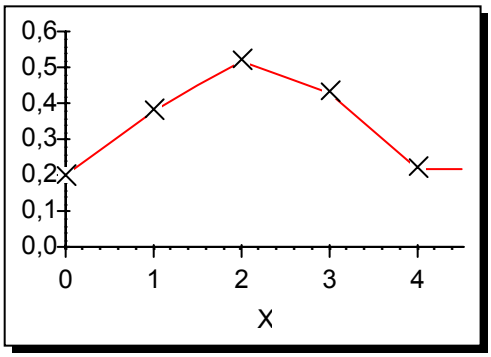
Вход X – любое число (независимая переменная). X1 – Значение X, соответствующее первой выборке, т.е. Y(1). Dx – положительное расстояние между выбираемыми значениями X. Y – вектор.

Результат GRAPH используется для изображения функции по заданному множеству значений функции или ряду равноотстоящих значений *Входа* (см. таблицу ниже):

Вход	**Выход**
X1	Y (1)
X1 + Dx	Y (2)
X1 + 2*Dx	Y (3)
...	...
X1 + (N-1) *Dx	Y (N)

Если X < X1, то возвращается Y (1). Если X > X1 + (N-1)*Dx, то возвращается Y(N), где N – число элементов в Y. Другими словами, функция имеет горизонтальные асимптоты (см. п. 2.2.9). Если значение X лежит между двумя точками ряда X1 + k*Dx, то значение функции вычисляется при помощи линейной интерполяции (см. рисунок ниже).

Пример Выражение GRAPH (X, 0, 1, [0.2, 0.38, 0.52, 0.43, 0.22]) дает следующий график.



Замечание Графические функции можно редактировать как текст или визуально, используя кнопку Graph... (График) диалогового окна Define Variable (Определение переменной), как описано в п. 1.4.6.

Группа Графические функции

Родственные функции GRAPHCURVE, GRAPHLINAS, GRAPHSTEP

GRAPHCURVE – Полиномиальный график с линейными асимптотами

Синтаксис GRAPHCURVE (X, X1, Dx, Y(N))

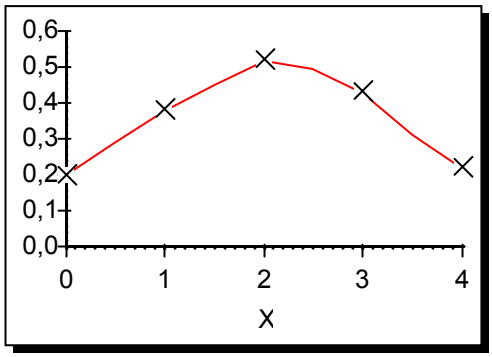
Вход X – любое число (независимая переменная). X1 – Значение X, соответствующее первой выборке, т.е. Y(1). Dx – положительное расстояние между выбираемыми значениями X. Y – вектор. Начальное значение Y используется для остальной части моделирования.

Результат GRAPHCURVE используется для изображения функции по заданному множеству значений функции или ряду равноотстоящих значений *Входа*, как показано в таблице ниже:

Вход	**Выход**
X1	Y (1)
X1 + Dx	Y (2)
X1 + 2*Dx	Y (3)
...	...
X1 + (N-1) *dx	Y (N)

Если значение X лежит между двумя точками ряда X1 + k*Dx, то значение функции вычисляется при помощи интерполяции многочленом 3-го порядка (см. рисунок ниже). Это дает более гладкую функцию, чем GRAPH и GRAPHLINAS. Функция имеет линейные асимптоты (см. п. 2.2.9) для точек, находящихся вне интервала [X1, X1 + (N-1) *Dx].

Пример Выражение GRAPHCURVE (X, 0, 1, [0.2, 0.38, 0.52, 0.43, 0.22]) дает следующий график.



Замечание Графические функции можно редактировать как текст или визуально, используя кнопку Graph... (График) диалогового окна Define Variable (Определение переменной), как описано в п. 1.4.6.

Группа Графические функции

Родственные функции GRAPH, GRAPHLINAS, GRAPHSTEP

GRAPHLINAS – Кусочно-линейный график с линейными асимптотами

Синтаксис GRAPHLINAS (X, X1, Dx, Y(N))

Вход X – любое число (независимая переменная). X1 – Значение X, соответствующее первой выборке, т.е. Y(1). Dx – положительное расстояние между выбираемыми значениями X. Y – вектор, состоящий по крайней мере из одного элемента.

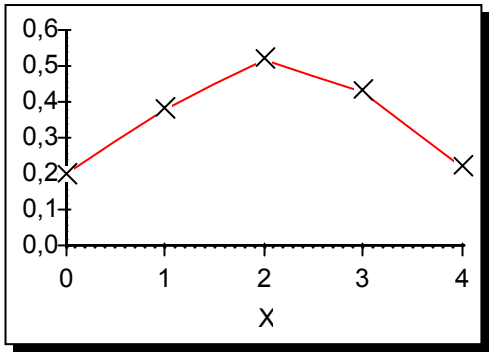
Результат GRAPHLINAS используется для изображения функции по заданному множеству значений функции или ряду равноотстоящих значений *Входа*, как показано в таблице ниже:

Вход	**Выход**
X1	Y (1)
X1 + Dx	Y (2)
X1 + 2*Dx	Y (3)
...	...
X1 + (N-1) *Dx	Y (N)

Если X < X1, то возвращается значение Y(X), вычисленное при помощи линейной экстраполяции (см. п. 2.2.9) по точкам Y(1) и Y(2). Если X > X1 + (N-1)*Dx, то возвращается значение Y(X), вычисленное при помощи линейной экстраполяции по точкам Y(N-1) и Y(N), где N – число элементов в Y. Это означает, что функция имеет линейные асимптоты (см. п. 2.2.9), получаемые на основе двух крайних точек с обоих концов выборки.

Существует следующая вязь между GRAPH и GRAPHLINAS: GRAPH (X, X1, Dx, Y(...)) = GRAPHLINAS (X, X1-Dx, Dx, Z(...)), где Z = [Y(1), Y(1), Y(2), ..., Y(N-1), Y(N), Y(N)].

Пример Выражение GRAPHLINAS (X, 0, 1, [0.2, 0.38, 0.52, 0.43, 0.22]) дает следующий график.



Замечание Графические функции можно редактировать как текст или визуально, используя кнопку Graph... (График) диалогового окна Define Variable (Определение переменной), как описано в п. 1.4.6.

Группа Графические функции

Родственные функции GRAPH, GRAPHCURVE, GRAPHSTEP

GRAPHSTEP – Ступенчатый график с горизонтальными асимптотами

Синтаксис GRAPHSTEP (X, X1, Dx, Y (N))

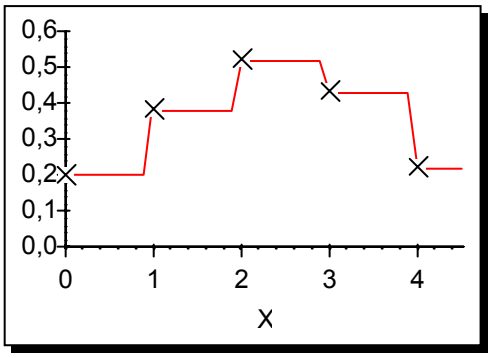
Вход X – любое число (независимая переменная). X1 – Значение X, соответствующее первой выборке, т.е. Y (1). Dx – положительное расстояние между выбираемыми значениями X. Y – вектор, состоящий по крайней мере из одного элемента.

Результат GRAPHSTEP используется для изображения функции по заданному множеству значений функции или ряду равноотстоящих значений *Входа*, как показано в таблице ниже:

Вход	**Выход**
X1	Y (1)
X1 + Dx	Y (2)
X1 + 2*Dx	Y (3)
...	...
X1 + (N-1) *Dx	Y (N)

Если X < X1, то возвращается Y (1). Если X > X1 + (N-1)*Dx, то возвращается Y (N), где N – число элементов в Y. Другими словами, функция имеет горизонтальные асимптоты (см. п. 2.2.9). Если значение X лежит между двумя точками ряда X1 + k*Dx, то значение функции приравнивается значению крайней левой точки интервала (см. рисунок ниже).

Пример Выражение GRAPHSTEP (X, 0, 1, [0.2, 0.38, 0.52, 0.43, 0.22]) дает следующий график.



Замечание Графические функции можно редактировать как текст или визуально, используя кнопку Graph... (График) диалогового окна Define Variable (Определение переменной), как описано в п. 1.4.6.
Следует выбрать *Шаг моделирования* < 1 (в данном примере он равен 0.1), чтобы увидеть разницу между графическими функциями GRAPHSTEP и, например, GRAPH (GRAPHLINAS) на графике, зависящем от времени, если переменная TIME используется как независимая.

Группа Графические функции

Родственные функции GRAPH, GRAPHCURVE, GRAPHLINAS

HIVAL – Самое большое моделируемое значение

Синтаксис HIVAL (X)

Вход X – любое число.

Результат Самое большое значение выражения X за весь период моделирования.

Замечание Функция зависит от предыдущих значений параметра.

Группа Функции с памятью

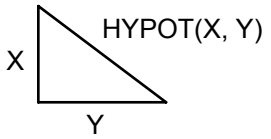
Родственные функции LOVAL, MAX

HYPOT – Гипотенуза

Синтаксис HYPOT (X, Y)

Вход X, Y – любые числа.

Результат Квадратный корень из $X^2 + Y^2$, т.е. длина гипотенузы прямоугольного треугольника. См. рисунок ниже:



Это можно также выразить так: SQRT (X^2 + Y^2).

Группа Математические функции

Родственные функции ABSC, VECTLEN

IF – Арифметическое ЕСЛИ

Синтаксис IF (Условие, Значение1, Значение2)

Вход *Условие* – логическое значение (Истина или Ложь). *Значение1*, *Значение2* – любые числовые выражения (вычисляемые параметры).

Результат *Значение1* вычисляется и возвращается, если Условие истинно. В противном случае вычисляется и возвращается *Значение2*.

Замечание Только одно из выражений, *Значение1* или *Значение2*, вычисляется функцией в зависимости от значения *Условия*. Это означает, что функции с побочными эффектами, например, такие как SOUND и ASSING не будут выполняться, если будут использованы в качестве параметра *Значение2*. Функцию IF можно вкладывать друг в друга до 8 раз включительно:
IF(A<B,C,(IF(C>A,E,(IF(D=B,C,(IF(E<B,A,(IF(B>D,E,(IF(E=A,C,
(IF(A>=D,C,(IF(A=B,D,0))))))))))))))))

Группа Условные функции

IMAGC – Мнимая часть комплексного числа или массива

Синтаксис IMAGC (A ([...,] 2))

Вход A – комплексное число или массив комплексных чисел в декартовой форме.

Результат Мнимая часть A.

Примеры IMAGC ([1, 2]) = 2
IMAGC ([[1, 2], [3, 4]]) = [2, 4]
IMAGC (A) = A (2), где A – комплексное число.

Группа Комплексные функции

Родственные функции ABSC, ANGLEC, REALC

INDEX – Преобразование переменной индекса в скаляр

Синтаксис	INDEX (Idx)
Вход	Idx – переменная индекса.
Результат	Числовое значение Idx.
Замечание	Функцию INDEX также можно использовать для массивов нижних индексов.
Пример	Вычисление суммы квадратов чисел от 1 до 9: SUMM (i = 1..9; INDEX (i)^2).
Группы	Встроенные функции, Функции преобразования типа
Родственные функции	COUNT, LAST, FIRST

INFINITY – Бесконечно большое положительное число

Синтаксис	INFINITY
Результат	Значение, используемое для обозначения числа, которое является слишком большим для представления компьютером.
Замечание	Используйте функцию - INFINITY для обозначения бесконечно большого отрицательного числа. Самое большое число, которое можно сохранить в Powersim составляет 1.0E+300.
Группа	Математические функции
Родственные функции	NAN, TRUE, FALSE

INIT – Начальное значение

Синтаксис	INIT (X)
Вход	X – любое числовое выражение (вычисляемый начальный параметр).
Результат	Начальное значение X.
Замечание	Функция зависит от предыдущего значения параметра, X будет вычислено только на стадии инициализации моделирования, и полученное значение будет возвращаться на оставшейся части интервала моделирования.
Примеры	См. примеры распределений инициализирующих функций (п. 1.5.2).
Группа	Функции с памятью
Родственные функции	SAMPLE, SAMPLEIF, EULER

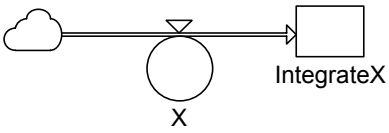
INT – Целая часть числа

Синтаксис	INT (X)
Вход	X – любое число.
Результат	Возвращает целую часть числа X.
Замечание	Существует следующая связь между X, INT (X) и FRAC (X) (для всех значений X): X = INT (X) + FRAC (X).
Примеры	INT (3.14) = 3 INT (-5.5) = -5
Группа	Функции преобразования типа
Родственные функции	CEIL, FLOOR, FRAC, ROUND

INTEGRATE – Интегрирование

Синтаксис	INTEGRATE (X)
Вход	X – переменная (функция), которая интегрируется по времени (параметр с запаздыванием).
Результат	Проинтегрированное значение X, т.е. сумма X_t по всем *Шагам моделирования* от начального момента времени до текущего.

График Функция INTEGRATE (X) дает тот же самый результат, что и следующая модель:




```
init   IntegrateX = ...
flow   IntegrateX = + dt*X
aux    X = ...
```

Замечание Функция зависит от предыдущих значений параметра, X должен быть переменной (выражения и литералы не допускаются), а на диаграмме X необходимо соединить с текущей переменной, используя связь с запаздыванием.

Группы Функции с памятью, Математические функции

Родственные функции DERIVN

INVERT – Обращение квадратной матрицы

Синтаксис INVERT (A(N, N))

Вход A – квадратная матрица.

Результат Обратная матрица A^{-1}.

Группы Функции для работы с массивами, Математические функции

Родственные функции /, ADD, INVERTC, MATRIXPROD

INVERTC – Обращение комплексного числа или квадратной матрицы

Синтаксис INVERTC (A([N, N,] 2))

Вход A – комплексное число или квадратная матрица комплексных чисел в декартовой форме.

Результат Обратная матрица A^{-1}.

Примеры INVERTC ([1, 2]) = [0.2, -0.4]
INVERTC (INVERTC (A)) = A
PRODC (A, INVERTC (A)) = [1, 0]

Группа Комплексные функции

Родственные функции INVERT, ADD, PRODC

2.4.5 Функции от J до M

LAST – Верхний предел диапазона или главного индекса диапазона

Синтаксис LAST (R)

Вход R – название диапазона или переменная индекса.

Результат Верхний предел диапазона R, если R – диапазон. Верхний предел основного диапазона R, если R – переменная индекса.

Пример Вычисление численности людей, которые перейдут из одной возрастной группы в другую из-за взросления. Путь переменная ALen обозначает длительность (продолжительность) каждой возрастной группы, а AOut – численность населения, покидающего данную возрастную группку (см. также пример в п. 1.4.6). Тогда имеем:

```
range   Возраст =(Детский,Юношеский,Средний,Пожилой)
aux     AOut(A=Возраст) = Население(A)/ALen(A)|
        A < LAST(Возраст); 0
```

Группа Встроенные функции

Родственные функции COUNT, FIRST

LIMIT – Ограничение переменной заданным диапазоном

Синтаксис	LIMIT (Переменная, Минимум, Максимум)
Вход	*Переменная* – переменная уровня или константа. *Минимум, Максимум* – любые числа, такие что *Минимум* < = *Максимум*.
Выход	*Переменная* – первоначальное значение, ограниченное интервалом *Минимум .. Максимум*.
Результат	Разность между ограниченным и неограниченным значениями *Переменной*: MAX(Минимум, MIN(Переменная, Максимум)) - Переменная.
Замечание	*Переменная* величина заменяется новой, т.е. граничным значением в конце шага моделирования. Функция имеет побочные эффекты: она изменяет свой первый параметр.
Родственные функции	ASSIGN, MIN, MAX, HIVAL, LOVAL
Группа	Смешанные функции

LN – Натуральный логарифм числа

Синтаксис	LN (X)
Вход	X – любое положительное число.
Результат	Натуральный логарифм X.
Группа	Математические функции
Родственные функции	EXP, LOG

LOG – Логарифм числа по основанию N

Синтаксис	ЛОГАРИФМ (X [, Основание = 10])
Вход	X – любое положительное число. Основание – основание логарифма (произвольный параметр со значением, по умолчанию равным 10).
Результат	Логарифм X по основанию N.
Группа	Математические функции
Родственные функции	EXP, LN

LOOKUP – Поиск по индексу элемента массива

Синтаксис	LOOKUP (A(...), Idx1, Idx2, ..., IdxN)
Вход	A – массив, по крайней мере, N-мерный. Idx1, Idx2, ..., IdxN – индексы в массиве A. Индексы находятся в диапазоне от 1 и выше.
Результат	Значение A(Idx1, Idx2, ..., IdxN). Если один или большее количество индексов находятся вне диапазона, то возвращается NAN. Если N – меньше числа измерений, то возвращается массив.
Замечание	LOOKUP обычно используется, когда необходимо индексировать массив при помощи обычной переменной или выражения вместо выражения для нижнего индекса. Если A – вектор, а Idx – переменная индекса, то выражения A (Idx) и LOOKUP(A, INDEX(Idx)) дают одинаковый результат. Первое выражение предпочтительно из-за удобочитаемости и эффективности с точки зрения быстродействия при моделировании. Если Y – вектор, а X – скаляр, то выражение LOOKUP (Y, X) эквивалентно GRAPHSTEP (ROUND (X), 1, 1, Y).

Демо-версия	LOOKUP.SIM
Группа	Функции для работы с массивами
Родственные функции	VECTOR, INDEX, GRAPH, GRAPHCURVE, GRAPHLINAS, GRAPHSTEP

LOVAL – Самое маленькое моделируемое значение

Синтаксис	LOVAL (X)
Вход	X – любое число.
Результат	Самое маленькое значение выражения X за весь период моделирования.
Замечание	Функция зависит от предыдущих значений параметра.
Группа	Функции с памятью
Родственные функции	HIVAL, МИНИМУМ

MATRIXPROD – Произведение матриц

Синтаксис	MATRIXPROD (A(m, n), B(n, p))
Вход	A, B – двумерные массивы (матрицы), в которых число столбцов первой матрицы должно равняться числу строк второй.
Результат	Матрица C размерности m x p, содержащая произведение матриц A и B. Элементы C (r, s) результирующей матрицы C определяется по правилу:
Уравнения	`aux  C(r=1..m,s=1..p) = SUM(i=1..n;A(r,i)*B(i,s))`
Группы	Функции для работы с массивами, Математические функции
Родственные функции	ARRPROD, PRODC, PRODCR, PRODRC, SPROD, TRANSPOSE, VECTPROD3D

MAX – Максимум

Синтаксис	MAX (X1, X2, ... XN)
Вход	X1, X2, ... XN – любые числа.
Результат	Максимальный элемент.
Группа	Статистические функции
Родственные функции	ARRMAX, HIVAL, MIN, AVG, STDDEV

MIN – Минимум

Синтаксис	MIN (X1, X2, ... XN)
Вход	X1, X2, ... XN – любые числа.
Результат	Минимальный элемент.
Группа	Статистические функции
Родственные функции	ARRMIN, MAX, AVG, STDDEV

MOD – Остаточный член от деления

Синтаксис	A MOD B
Вход	A – любое число, B – любое число кроме нуля.
Результат	Остаточный член для A / B, определяемый как такое значение R, что A = B*k + R, где k – целое число и ABS (R) < ABS (B).
Группа	Математические функции
Родственные функции	Оператор деления (/)

2.4.6 Функции от N до P

NAN – Не число

Синтаксис NAN
Результат Значение, используемое для представления недопустимого числа.
Группа Математические функции
Родственные функции INFINITY, TRUE, FALSE

NORMAL – Нормальное (гауссово) распределение

Синтаксис NORMAL ([Среднее=0 [, Стандартное отклонение=1 [, Seed]]])
Вход *Среднее* – среднее значение распределения, по умолчанию равное 0 (произвольный параметр). *Стандартное отклонение* – отклонение от среднего значения распределения, по умолчанию равное 1 (произвольный параметр). *Seed* – параметр, отвечающий за инициализацию генератора случайного числа (произвольный начальный параметр, по умолчанию равный произвольному значению).
Результат Генерирует ряд нормально распределенных случайных чисел заданными значениями *Среднего* и *Стандартного отклонения*.
Примеры Распределения результатов испытаний
5 + NORMAL генерирует нормально распределенные случайные числа со *Средним* = 5 и *Стандартным отклонением* = 1.
20 + 3*NORMAL генерирует нормально распределенные случайные числа со *Средним* = 20 и *Стандартным отклонением* = 3.
NORMAL (20, 3) генерирует такое же распределение, что и выше.
NORMAL (5) генерирует нормально распределенные случайные числа со *Средним* = 5 и *Стандартным отклонением* = 1.
Группа Случайные (стохастические) функции
Родственные функции EXPRND, POISSON, RANDOM

NOT – Логическое НЕ

Синтаксис NOT A
Вход A – логическое значение (Истина или Ложь).
Результат Истинна, если A Ложь, Ложь – в противном случае.
Пример NOT 1 = 0
NOT 0 = 1
Замечание NOT выражается так же следующим способом: 1 - A.
Родственные функции FALSE, TRUE, AND, OR, XOR
Группа Логические функции

NPV – Чистая приведенная ценность

Синтаксис NPV (Платеж, Ставка процента)
Вход *Платеж* – размер састичного платежа. *Ставка процента* – ставка процента в единицу времени.
Результат Чистая приведенная ценность.
Демо-версия FINANCE.SIM
Замечание См. п. 2.2.8.
Группы Финансовые функции, Функции с памятью
Родственные функции FV, PMT, PV

OR – Логическое ИЛИ

Синтаксис	A OR B
Вход	A, B – логические значения (Истина или Ложь).
Результат	Истина, если по крайней мере один из A или B истинен, Ложь – в противном случае.
Замечание	Оператор OR может также использоваться в определении ограничений: A (i) | i < 2 OR i> 5 OR выражается так же следующим способом: A + B - A* B.
Группа	Встроенные функции, Логические функции
Родственные функции	XOR, AND, NOT, BOOL, TRUE, FALSE

OTHERWISE – Оператор защиты по умолчанию

Синтаксис	OTHERWISE
Вход	A – любое выражение.
Результат	A для всех индексов, которые не являются частью (предыдущих) операторов защиты для данного выражения.
Замечание	OTHERWISE эквивалентен оператору DEFAULT (См. замечание для DEFAULT).
Группа	Встроенные функции
Родственные функции	| , WHEN, DEFAULT

PAUSEIF – Условная пауза

Синтаксис	PAUSEIF (Условие)
Вход	*Условие* – произвольное условие. Когда данный аргумент становится равным Истине, моделирование приостанавливается и может быть возобновлено (например, используя кнопку Пауза), даже если *Условие* все еще истинно.
Результат	Значение *Условия*.
Группа	Функции управления
Родственные функции	PAUSEWHILE, STOPIF, STOPRUNIF

PAUSEWHILE – Условная длящаяся пауза

Синтаксис	PAUSEWHILE (Условие)
Вход	*Условие* – произвольное условие. Когда данный аргумент становится равным Истине, моделирование приостанавливается и может быть возобновлено (например, используя кнопку *Пауза*), если *Условие* не ложно.
Результат	Значение *Условия*.
Группа	Функции управления
Родственные функции	PAUSEIF, STOPIF, STOPRUNIF

PCT – Преобразование числа в процент

Синтаксис	PCT (X)
Вход	X – любое число.
Результат	X * 100.
Группы	Функции преобразования типа, Математические функции
Родственные функции	%

Pi – Тригонометрическая константа π

Синтаксис PI
Результат Значение π, т.е. 3.141592654...
Группа Тригонометрические функции
Родственные функции DEGTORAD, GRADTORAD, RADTODEG, RADTOGRAD

PLAYERNUMBER – Номер текущего игрока

Синтаксис PLAYERNUMBER
Результат Номер текущего игрока.
Группа Смешанные функции
Родственные функции GETPLAYERS, GETSIMPLAYERS, GETTOTPLAYERS

PMT – Частичный платеж (взнос или поток платежей)

Синтаксис PMT (Ставка процента, Периоды, Текущая ценность, Будущая ценность)
Вход *Ставка процента* – учетная ставка процента за период. *Периоды* – количество периодов. *Текущая ценность, Будущая ценность* – настоящая и будущая ценность денег соответственно.
Демо-версия FINANCE.SIM
Группа Финансовые функции
Результат Периодические платежи (см. п. 2.2.8).

POISSON – Распространение Пуассона

Синтаксис POISSON ([Среднее=1 [, Seed]])
Вход *Среднее* – среднее значение распределения (произвольный параметр, по умолчанию равный 1). *Seed* – параметр, отвечающий за инициализацию генератора случайных чисел (произвольный начальный параметр, по умолчанию равный произвольному значению).
Результат Генерирует ряд случайных чисел по распределению Пуассона с заданным значением *Среднего.*
Группа Случайные (стохастические) функции
Родственные функции EXPRND, NORMAL, RANDOM

POLTOCAR – Преобразование комплексного числа или массива из полярной формы в декартову

Синтаксис POLTOCAR (A([...,] 2))
Вход A – комплексное число или массив комплексных чисел в полярной форме.
Результат A в декартовой форме.
Примеры POLTOCAR ([3, 3.141592]) = [-3, 0]
CARTOPOL (POLTOCAR (A)) = A
Группы Комплексные функции, Функции преобразования типа
Родственные функции CARTOPOL, ANGLEC, ABSC

POLY – Полином (многочлен)

Синтаксис POLY (X, A0 [, A1, ...,An])
Вход X – любое число, A0...An – любые числа (коэффициенты полинома).
Результат Полиномиальная функция X, определяемая как:
$a_0 + a_1 x + a_2 x^2 + \quad + a_n x^n$.
Группа Математические функции

PROD – Произведение выражений по индексам

Синтаксис PROD (Dim1, Dim2, ..., DimN; Выражение)

Вход Dim1, Dim2, ..., DimN – определения индексов (измерений), по которым ведется "суммирование". Каждое измерение определяется как произвольная переменная индекса, сопровождаемая знаком (=) и диапазоном. Несколько измерений можно определить сразу, перечислив их через запятую. *Выражение* – любое выражение, возможно, включающее индексы *Dim1*, *Dim2*, ..., *DimN*.

Результат Произведение значений *Выражения* по индексам *Dim1*, *Dim2*, ..., *DimN*.

Замечание Функция PROD не имеет обычного синтаксиса, как у других функций, так как можно определять измерения (как переменные типа массива). Отметим, что для разделения измерений и выражения используется точка с запятой (;). Связь между стандартной математической системой обозначений и функцией PROD иллюстрируется ниже:

Примеры PROD (i = 1..3; A(i)) = A (1) * A (2) * A (3)
PROD (1..3; 2) = 8
PROD (i = 1..5; INDEX (i)) = 5! = 120

Группа Функции для работы с массивами

PRODC – Произведение комплексных чисел, векторов или массивов

Синтаксис PRODC (A([[M,] N,] 2), B([[M,] N,] 2))

Вход A, B – комплексные числа, вектора или массивы в декартовой форме.

Результат Произведение A и B.

Группа Комплексные функции

Родственные функции MATRIXPROD, PRODRC, PRODCR

PRODCR – Произведение комплексных и вещественных чисел, векторов или массивов

Синтаксис PRODCR (A([[M,] N,] 2), B([[M,] N]))

Вход A – комплексное число, вектор или массив в декартовой форме. B – вещественное число, вектор или массив.

Результат Произведение A и B.

Пример PRODCR ([3, 4], 5) = [15, 20]

Группа Комплексные функции

Родственные функции PRODC, PRODRC

PRODRC – Произведение вещественных и комплексных чисел, векторов или массивов

Синтаксис PRODRC (A([[M,] N]), B([[M,] N,] 2))

Вход A – вещественное число, вектор или массив. B – комплексное число, вектор или массив в декартовой форме.

Результат Произведение A и B.

Пример PRODRC (2, [3, 4]) = [6, 8]

Группа Комплексные функции

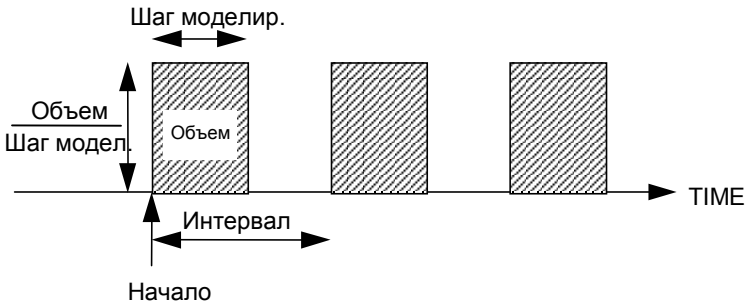
Родственные функции PRODC, PRODCR

PULSE – Периодический импульс

Синтаксис PULSE (Объем, Начало, Интервал)

Вход *Объем* – объем импульса (вычисляемый параметр). *Начало* – время появления первого импульса, измеренное в единицах времени моделирования. *Интервал* – интервал времени между начальными точками соседних импульсов, измеренный в единицах времени моделирования.

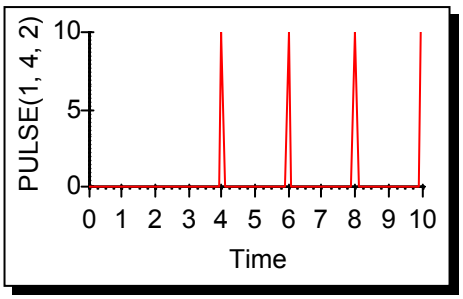
Результат *Объем / Шаг моделирования* или 0 в зависимости от текущего времени TIME и значений *Начала* и *Интервала*. Импульс появляется в *Начальный* момент времени и повторяется через промежуток времени, равный *Интервалу*, как показано ниже.



Замечание Функция зависит от времени TIME и *Шага моделирования*. Существует следующая связь между функциями PULSE и PULSEIF:

PULSE (V, F, I) = PULSEIF (TIMECYCLE (F, I), V)

Пример Выражение PULSE (10, 4, 2) дает следующий график при моделировании с шагом 0.1:



Демо-версия INVENTOR.SIM

Группа Функции, зависящие от времени

Родственные функции PULSEIF, TIME

PULSEIF – Условный импульс

Синтаксис PULSEIF (Условие, Объем)

Вход *Условие* – условие, определяющее появление (Истина) или не появление (Ложь) импульса. *Объем* – объем импульса (вычисляемый параметр).

Замечание Функция зависит от *Шага моделирования.*

Результат Если *Условие* Ложно, то возвращается 0, иначе – *Объем / Шаг моделирования*.

Группа Условные функции

Родственные функции PULSE, TRUE, FALSE, TIMEIS, TIMECYCLE

PV – Текущая (настоящая) ценность

Синтаксис PV (Ставка процента, Периоды, Платеж, Будущая ценность)

Вход *Ставка процента* – ставка процента за период. *Периоды* – количество периодов. *Платеж* – размер периодического платежа. *Будущая ценность* – будущее значение ценности денег (произвольный параметр, по умолчанию равный 0).

Результат Текущая (настоящая) ценность (см. п. 2.2.8).

Демо-версия FINANCE.SIM

Группа Финансовые функции

Родственные функции FV, NPV, PMT

2.4.7 Функции от Q до S

QUEUE – Очередь со многими клиентами и серверами

Синтаксис QUEUE (Клиенты [,QMAX=100 [,QInit[(K[,2])]=0 [, Готовность серверов = TRUE [, QLen [, QWait]]]]])

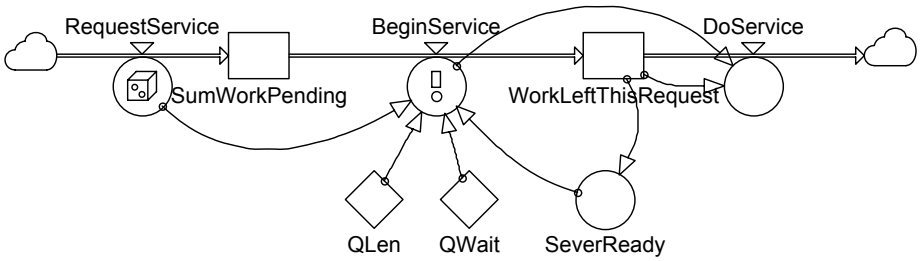
Вход *Клиенты* – скаляр или вектор с просьбами клиентов об услугах. Каждый элемент *Клиентов* содержит количество работы, запрашиваемое соответствующим клиентом у сервера. *QMax* – максимальная емкость очереди в терминах количества клиентских запросов (произвольный начальный параметр, по умолчанию равный 100). *Qinit*: когда *QInit* является скаляром, то очередь первоначально пуста (произвольный начальный параметр, по умолчанию равный 0). *Qinit* (K): когда *QInit* является вектором, то очередь инициализируется с K элементами со временами вступления в очередь, равными 0, и временами обслуживания *i*-го элемента, равными *QInit* (i). *QInit* (K, 2): когда *QInit* является матрицей K x 2, то очередь инициализируется с K элементами со временем вступления *i*-го элемента в очередь, равным *QInit* (i, 1), и временами обслуживания *i*-го элемента, равными *QInit* (i, 2). *Готовность серверов* – скаляр или вектор с логическими значениями, определяющими, какие серверы являются доступными для обслуживания клиентов (произвольный параметр, по умолчанию, равный TRUE, т.е. Истине).

Выход *QLen* – фактическая длина очереди (произвольный выходной параметр). *QWait* – время ожидания самого последнего выхода из очереди (произвольный выходной параметр).

Результат Скаляр или вектор с одним элементом на каждый сервер, определяющий поток работ для каждого сервера. Если *Готовность серверов*(i) = FALSE (Ложь), то *Результат*(i) будет нулевой. Отметим, что запросы не складываются вместе при возвращении функцией QUEUE. Например, три запроса с размерами 0.1, 0.4 и 0.3 не будут возвращены как один запрос с размером 0.8.

Замечание Ненулевые элементы *Входа* добавляются (конкатенируются) к внутреннему списку запросов в очереди, ожидающих обслуживания. Для каждого готового к работе сервера, выбирается какой-либо элемент с переднего фланга очереди и приписывается к соответствующему элементу вектора *Результата*. Если очередь становится пустой, то остающимся элементам вектора *Результата* присваиваются нулевые значения.

Пример График ниже поясняет работу системы с одним клиентом и одним сервером:



где RequestService – запрос на обслуживание, – сумма накопленных работ, BeginService – начало обслуживания, WorkLeftThis-Request – оставшийся объем работы с данным запросом, DoService – обслуженные запросы, SeverReady – готовность сервера к новой работе. Данная система описывается следующими уравнениями:

```
aux     BeginService = QUEUE (RequestService,100,0,
        SeverReady, QLEN, QWait)
aux     DoService = MIN(WorkLeftThisRequest/TIMESTEP+
        BeginService,1)
const   QLen = 0
const   QWait = 0
aux     RequestService=POISSON(1/5*TIMESTEP)/TIMESTEP
        *RANDOM (1, 10)
aux     SeverReady = WorkLeftThisRequest < = TIMESTEP
init    SumWorkPending = 0
flow    SumWorkPending = -dt*BeginService + dt*
        RequestService
init    WorkLeftThisRequest = 0
flow    WorkLeftThisRequest = -dt*DoService + dt*
        BeginService
```

- Здесь клиентский запрос достигает среднего значения каждую пятую единицу времени с запросом на время обслуживания, лежащее в диапазоне от 1 до 10. (См. RequestService).
- Когда сервер готов, то функция QUEUE удаляет клиентский запрос из очереди и возвращает время обслуживания, запрошенное клиентом. (См. BeginService).
- Сервер выполняет максимум одну единицу обслуживания в каждый единичный момент времени. (См. DoService).
- Сервер готов для нового запроса, если у него не остается никакой работы по обработке текущего запроса или если он завершил обслуживание текущего запроса внутри текущего *Шага моделирования*. (См. SeverReady).
- SumWorkPending не является необходимым для работы модели.
- QLEN и QWait – выходные параметры, модифицируемые функцией QUEUE с задержкой на один *Шаг моделирования*.

Другой пример Очередь со многими клиентами и серверами (см. п.2.2.10).
Демо-версия QUEUE1. SIM
Группа Функции с памятью
Родственные функции Все функции задержки, SHIFTLCNT, SHIFTCIF, SHIFTLIF

RADTODEG – Преобразование радиан в градусы

Синтаксис RADTODEG (X)
Вход Угол X – угол в радианах.
Результат Эквивалент угла X, измеренного в градусах:
RADTODEG (A) = A*180/PI.
Группа Функции преобразования типа
Родственные функции DEGTORAD, PI, RADTOGRAD

RADTOGRAD – Преобразование радиан в грады

Синтаксис RADTOGRAD (X)

Вход Угол X – угол в радианах.

Результат Эквивалент угла X, измеренного в градах:
RADTOGRAD (A) = A*200/PI.

Группа Функции преобразования типа

Родственные функции GRADTORAD, PI, RADTODEG

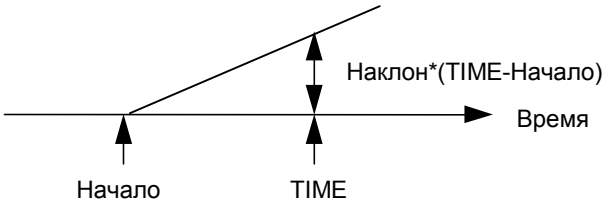
RAMP – Линейная функция

Синтаксис RAMP (Наклон, Начало)

Вход *Наклон* – наклон графика линейной функции (см. рис. Ниже). *Начало* – время появления пилообразного сигнала.

Результат 0, если время TIME < *Начало*, и (*Наклон* * TIME – *Начало*) – в противном случае. Функция RAMP определяется следующим уравнением:

RAMP (S, F) = IF (TIME < F, 0, (TIME – F) *S)



Демо-версия INVENTOR.SIM

Группа Функции, зависящие от времени

RANDOM – Равномерное распределение

Синтаксис RANDOM ([Минимум=0 [, Максимум = Минимум + 1 [, Seed]]])

Вход *Минимум* – оценка минимального возвращаемого значения (произвольный параметр, по умолчанию равный 0). *Максимум* – оценка максимального возвращаемого значения (произвольный параметр, по умолчанию равный *Минимуму* + 1). *Seed* – параметр, отвечающий за инициализацию генератора случайных чисел (произвольный начальный параметр, по умолчанию равный произвольному значению).

Результат Случайное число, равномерно распределенное между *Минимумом* и *Максимумом*.

Примеры 5 + RANDOM или RANDOM (5) возвращает равномерно распределенные случайные числа между 5 и 6.
20 + 3* RANDOM или RANDOM (20, 23) возвращает равномерно распределенные случайные числа между 20 и 23.
Существует большее количество примеров применения функции RANDOM, например, распределение результатов испытаний.

Группа Случайные (стохастические) функции

Родственные функции EXPRND, NORMAL, POISSON

REALC – Вещественная часть комплексного числа

Синтаксис REALC (A ([...,] 2))

Вход A – комплексное число или массив комплексных чисел в декартовой форме.

Результат Вещественная часть A.

Примеры REALC ([1, 2]) = 1

REALC ([[1, 2], [3, 4]]) = [1, 3]
REALC (A) = A (1) , где A – комплексное число.

Группа Комплексные функции

Родственные функции ABSC, ANGLEC, IMAGC

ROUND – Округление до ближайшего целого

Синтаксис ROUND (X)

Вход X – любое число.

Результат X, округленное до ближайшего целого числа, согласно формуле: ROUND (X) = IF (X> = 0, FLOOR (X + 0.5), CEIL (X –0. 5)).

Группа Функции преобразования типа

Родственные функции CEIL, FLOOR, INT, FRAC

RUN – Номер текущей имитации

Синтаксис RUN

Результат Порядковый номер имитации (прогона): 1 – в течение первой имитации, 2 – в течение второй и т.д.

Замечание Функция зависит от номера имитации. Моделирование обычно содержит только одну имитацию. Однако можно задать несколько имитаций, используя диалоговое окно Run Setup (Настройки имитации). Затем Powersim автоматически начнет новую имитацию после завершения предыдущей. Это будет продолжаться до тех пор, пока не будет выполнено желаемое количество имитаций. Данная особенность может использоваться, например, при проверке чувствительности модели. Функция RUN может использоваться для перебора различных вариантов тестовых испытаний за время выполнения серии последовательных имитаций.

Примеры Распределение результатов испытаний.

Группа Смешанные функции

Родственные функции RUNCOUNT, TIME, STARTTIME, STOPTIME, TIMESTEP, ATSTART

RUNCOUNT – Количество имитаций

Синтаксис RUNCOUNT

Результат Количество имитаций, заданное в диалоговом окне Run Setup (Настройки имитации).

Примеры Распределение результатов испытаний.

Группа Смешанные функции

Родственные функции RUN

SAMPLE – Периодическая выборка

Синтаксис SAMPLE (Вход, Начало, Интервал [, Начальное значение = 0])

Вход *Вход* – любое числовое выражение, для которого производится выборка (вычисляемый параметр). *Начало* – время первой выборки, измеряемое в единицах времени моделирования. *Интервал* – интервал проведения выборки, измеряемый в единицах времени моделирования. *Начальное значение* – значение, возвращаемое функцией SAMPLE до наступления времени первой выборки (произвольный начальный параметр, по умолчанию равный 0).

Результат SAMPLE – периодически устанавливается равным *Входу* и сохраняет это значение до наступления следующего времени выборки. Первая выборка берется в *Начальный* момент времени, а новые выборки берутся через каждый *Интервал* времени. SAMPLE возвращает *Начальное значение* до наступления времени первой выборки (*Начала*).

Замечание Функция зависит от времени и от предыдущих значений *Входа.*

Группы Функции, зависящие от времени; Функции с памятью

Родственные функции SAMPLEIF, INIT, EULER, TIME

SAMPLEIF – Условная выборка

Синтаксис SAMPLEIF (Условие, Вход [, Начальное значение = 0])

Вход *Условие* – условное выражение (Истинна или Ложь), определяющее запуск или не запуск процедуры выборки. *Вход* – любое числовое выражение, для которого нужно произвести выборку (вычисляемый параметр). *Начальное значение* – значение, возвращаемое SAMPLEIF до тех пор, пока условие не будет впервые выполнено (произвольный начальный параметр, по умолчанию равный 0).

Результат Последнее выборочное значение *Входа* или *Начальное значение* (до наступления времени первой выборки, т.е. до выполнения условия).

Замечание Функция зависит от предыдущих значений *Входа*. Существует следующая вязь между функциями SAMPLE и SAMPLEIF:
SAMPLE (X, I, F, X0) = SAMPLEIF (TIMECYCLE (F, I), X, X0)

Группа Функции с памятью, Условные функции

Родственные функции SAMPLE, INIT, EULER, TIMEIS, TIMECYCLE

SCANEQ – Поиск равного компонента вектора

Синтаксис SCANEQ (Вектор (...), X [, Направление поиска = FALSE])

Вход *Вектор* – проверяемый вектор. X – любое число. *Направление поиска* – если Истина, то *Вектор* просматривается с конца, иначе – с начала (произвольный параметр, по умолчанию равный FALSE (Ложь)).

Результат Положение (начинающееся с 1) первого (или последнего) элемента *Вектора*, равного X. Если X не является элементом *Вектора*, то возвращается 0.0.

Группа Функции для работы с массивами

Родственные функции SCANNEQ, SCANLT, SCANLTEQ, SCANGT, SCANGTEQ

SCANGT – Поиск большего компонента вектора

Синтаксис SCANGT (Вектор (...), X [, Направление поиска = FALSE])

Вход *Вектор* – проверяемый вектор. X – любое число. *Направление поиска* – если Истина, то *Вектор* просматривается с конца, иначе – с начала (произвольный параметр, по умолчанию равный FALSE (Ложь)).

Результат Положение (начинающееся с 1) первого (или последнего) элемента *Вектора*, большего X. Если все элементы *Вектора* меньше или равны X, то возвращается 0.0.

Группа Функции для работы с массивами

Родственные функции SCANEQ, SCANNEQ, SCANLT, SCANLTEQ, SCANGTEQ

SCANGTEQ – Поиск большего или равного компонента вектора

Синтаксис	SCANGTEQ (Вектор (...), X [, Направление поиска = FALSE])
Вход	*Вектор* – проверяемый вектор. X – любое число. *Направление поиска* – если Истина, то *Вектор* просматривается с конца, иначе – с начала (произвольный параметр, по умолчанию равный FALSE (Ложь)).
Результат	Положение (начинающееся с 1) первого (или последнего) элемента *Вектора*, большего или равного X. Если все элементы *Вектора* меньше X, то возвращается 0.0.
Группа	Функции для работы с массивами
Родственные функции	SCANEQ, SCANNEQ, SCANLT, SCANLTEQ, SCANGT

SCANLT – Поиск меньшего компонента вектора

Синтаксис	SCANLT (Вектор (...), X [, Направление поиска = FALSE])
Вход	*Вектор* – проверяемый вектор. X – любое число. *Направление поиска* – если Истина, то *Вектор* просматривается с конца, иначе – с начала (произвольный параметр, по умолчанию равный FALSE (Ложь)).
Результат	Положение (начинающееся с 1) первого (или последнего) элемента *Вектора*, меньшего X. Если все элементы *Вектора* больше или равны X, то возвращается 0.0.
Группа	Функции для работы с массивами
Родственные функции	SCANEQ, SCANNEQ, SCANLTEQ, SCANGT, SCANGTEQ

SCANLTEQ – Поиск меньшего или равного компонента вектора

Синтаксис	SCANLTEQ (Вектор (...), X [, Направление поиска = FALSE])
Вход	*Вектор* – проверяемый вектор. X – любое число. *Направление поиска* – если Истина, то *Вектор* просматривается с конца, иначе – с начала (произвольный параметр, по умолчанию равный FALSE (Ложь)).
Результат	Положение (начинающееся с 1) первого (или последнего) элемента *Вектора*, меньшего или равного X. Если все элементы *Вектора* больше X, то возвращается 0.0.
Группа	Функции для работы с массивами
Родственные функции	SCANEQ, SCANNEQ, SCANLT, SCANGT, SCANGTEQ

SCANNEQ – Поиск неравного компонента вектора

Синтаксис	SCANNEQ (Вектор (...), X [, Направление поиска = FALSE])
Вход	*Вектор* – проверяемый вектор. X – любое число. *Направление поиска* – если Истина, то *Вектор* просматривается с конца, иначе – с начала (произвольный параметр, по умолчанию равный FALSE (Ложь)).
Результат	Положение (начинающееся с 1) первого (или последнего) элемента *Вектора*, неравного X. Если все элементы *Вектора* равны X, то возвращается 0.0.
Группа	Функции для работы с массивами
Родственные функции	SCANEQ, SCANLT, SCANLTEQ, SCANGT, SCANGTEQ

SECONDSTHISRUN – Количество секунд, прошедших с начала текущей имитации

Синтаксис	SECONDSTHISRUN
Результат	Число секунд, прошедшее с момента начала текущей имитации.
Группа	Функции, зависящие от времени
Родственные функции	SECONDSTHISSTEP

SECONDSTHISSTEP – Количество секунд, прошедших на текущем шаге моделирования

Синтаксис SECONDSTHISSTEP

Результат Число секунд, прошедшее на текущем шаге моделирования.

Группа Функции, зависящие от времени

Родственные функции SECONDSTHISRUN

SELECTDECISION – Выбор решения

Синтаксис SELECTDECISION (Игрок, Решение, Предполагаемое решение, Моделируемое решение, Пустое решение)

Вход *Игрок* – индекс в измерении игрока. *Решение* – значение, принятое данным игроком. *Предполагаемое решение* – значение, предлагаемое другими игроками по отношению к данному игроку. *Моделируемое решение* – моделируемая переменная решения данного игрока. *Пустое решение* – значение решения, используемое для неучаствующего игрока.

Результат Одно из значений (Решено, Предлагается, Моделируется или Пустое решение) в зависимости от значения *Игрока*, текущего режима игры (стратегический или операционный), числа игроков и числа моделируемых игроков.

Пример

```
Aux  Цена (p = Диапазон игроков) = SELECTDECISION(
     INDEX(p),Решение(p),Предполагаемое решение(p),
     Моделируемое решение(p), 0)
```

Группа Смешанные функции

SHIFTCIF – Условный циклический сдвиг компонент вектора

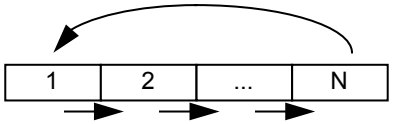
Синтаксис SHIFTCIF (Условие, Вектор(N))

Вход *Условие* – условное выражение (Истина или Ложь), определяющее запуск или не запуск сдвига. *Вектор* – любая векторная переменная уровня или константа.

Выход *Вектор* – неизменный или сдвигаемый в зависимости от значения *Условия*. Если сдвиг осуществляется, то компоненты *Вектора* сдвигаются следующим образом:

Вектор [N] = Вектор [N-1]
...
Вектор [3] = Вектор [2]
Вектор [2] = Вектор [1]
Вектор [1] = Вектор [N]

Это означает, что все компоненты кроме последнего копируются в следующую позицию. Последний же копируется в первую позицию, как показано ниже:



Результат Последний компонент *Вектора*, т.е. *Вектор* (N).

Замечание Функция имеет побочные эффекты по второму параметру.

Группы Функции для работы с массивами, Условные функции

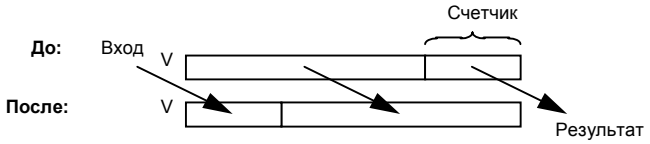
Родственные функции SHIFTLIF, SHIFTLCNT

SHIFTLCNT – Линейный сдвиг компонент вектора

Синтаксис SHIFTLCNT (Счетчик, Вектор(N) [, Вход = 0])

Вход *Счетчик* – счетчик смещения компонент *Вектора* (положительное значение – смещение вправо, отрицательное – влево), *Вектор* – вектор с N элементами, *Вход* – значение, которое будет присвоено освобожденным компонентам (произвольный входной параметр)

Выход *Вектор* – неизменный или сдвигаемый в зависимости от значения *Счетчика*.



Результат Сумма элементов, перемещенных за пределы данного *Вектора*.

Замечание Функция имеет побочные эффекты по второму параметру. Данная функция обрабатывает сдвиги в обоих направлениях, а также дробный сдвиг, т.е. дробные значения *Счетчика* смещений.
SHIFTLCNT (1, V, 0) = SHIFTLIF (TRUE, V).
Для константы *Время запаздывания*
SHIFTLCNT (Счетчик, Время запаздывания, Вход) = DELAYPPL (Вход, Время запаздывания, 0),
где *Счетчик* = ELEMCOUNT(*Состояние задержки*) / *Время запаздывания* * *Шаг моделирования*, и *Состояние задержки* – вектор со всеми компонентами, первоначально равными нулю.

Группа Функции для работы с массивами

Родственные функции SHIFTLIF, SHIFTCIF

SHIFTLIF – Условный линейный сдвиг компонент вектора

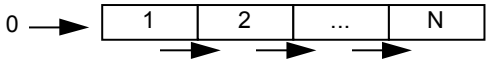
Синтаксис SHIFTLIF (Условие, Вектор(N))

Вход *Условие* – условное выражение (Истина или Ложь), определяющее запуск или незапуск сдвига. *Вектор* – любая векторная переменная уровня или константа.

Выход *Вектор* – неизменный или сдвигаемый в зависимости от значения *Условия*. Если сдвиг осуществляется, то компоненты *Вектора* сдвигаются следующим образом:

Вектор [N] = Вектор [N-1]
...
Вектор [3] = Вектор [2]
Вектор [2] = Вектор [1]
Вектор [1] = Вектор [N]

Это означает, что все компоненты копируются в следующую позицию, кроме последнего, который отбрасывается. Первый компонент полагается равным нулю.



Результат Последний компонент *Вектора*, т.е. *Вектор* (N).

Замечание Функция имеет побочные эффекты по второму параметру.

Пример Периодического сдвига можно достигнуть, используя функцию TIMECYCLE при определении *Условия*, например, следующим образом: SHIFTCIF (TIMECYCLE (T, I), V)

Группы Функции для работы с массивами, Условные функции

Родственные функции SHIFTCIF, SHIFTLCNT

SIGN – Знак числа

Синтаксис SIGN (X)
Вход X – любое число.
Результат + 1 при положительном X, -1 при отрицательном X, 0 – в противном случае.
Пример SIGN (X) = IF (X < 0, -1, IF (X> 0, 1, 0))
Группы Функции преобразования типа, Математические функции

SIN – Синус

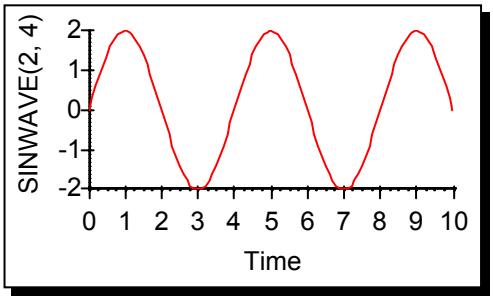
Синтаксис SIN (X)
Вход Угол X – угол в радианах.
Результат Синус угла X.
Группа Тригонометрические функции
Родственные функции COS, TAN, ARCSIN, SINH, DEGTORAD, GRADTORAD, PI

SINH – Гиперболический синус

Синтаксис SINH (X)
Вход Значение X.
Результат Гиперболический синус входного значения X.
Группа Тригонометрические функции
Родственные функции COSH, TANH, SIN, DEGTORAD, GRADTORAD, PI

SINWAVE – Периодическая синусоида

Синтаксис SINWAVE (Амплитуда, Период)
Вход *Амплитуда* – амплитуда волны. *Период* – период волны.
Результат Зависящая от времени синусоида, определяемая уравнением: SINWAVE (A, P) = A* SIN (TIME*2*PI / P).
Пример Выражение SINWAVE (2, 4) дает следующий график.



Замечание Функция зависит от времени.
Демо-версия INVENTOR.SIM
Группы Функции, зависящие от времени, Тригонометрические функции
Родственные функции SIN, TIME, COSWAVE

SOUND – Звук

Синтаксис SOUND (Вход)
Вход *Вход* – звук, который будет проигран. Значение округляется до ближайшего целого числа и используется для выбора звука из следующей таблицы:

Вход	Звук
-1	Звук, воспроизводящийся через компьютерный динамик
0	Нет звука
1	Системный звук "Звездочка"
2	Системный звук "Восклицание"
3	Системный звук "Рука"
4	Системный звук "Вопрос"
5	Системный звук "Стандартный (по умолчанию)"

Числа вне вышеупомянутого диапазона трактуются как -1.

Результат Значение *Входа*.

Замечание *Входные* значения от 1 до 5 выбирают звук из раздела [Звуки] файла WIN.INI.

Группа Смешанные функции

SPROD – Скалярное произведение векторов

Синтаксис SPROD (A(N), B(N))

Вход A, B – векторные выражения с одинаковым числом элементов.

Результат Скалярное произведение A и B, определяемые следующим выражением: SUM (i = 1..N; A (i) * B (i)).

Группы Функции для работы с массивами, Математические функции

Родственные функции ARRPROD, MATRIXPROD, PRODC, PRODCR, PRODRC, VECTPROD3D

SQRT – Квадратный корень

Синтаксис SQRT (X)

Вход X – любое неотрицательное число.

Результат Квадратный корень числа X.

Пример Связь между SQRT и оператором ^ следующая:
SQRT (X) = X ^ 0.5.

Группа Математические функции

STARTTIME – Время начала моделирования

Синтаксис STARTTIME

Результат Время начала моделирования, определяемое в диалоговом окне Simulation Setup (Установки моделирования).

Группа Функции, зависящие от времени

Родственные функции STOPTIME, TIME, TIMESTEP, ATSTART

STDDEV – Стандартное отклонение

Синтаксис STDDEV (X1, X2, ..., XN)

Вход X1, X2, ... XN – любые числа.

Результат Стандартное отклонение аргументов, определяемое как:

$$S = \sqrt{\frac{\sum_{i=1}^{N}(X_i - \overline{X})^2}{N}},$$

где $\overline{X}$ – среднее значение, равное AVG (X1, X2, ..., XN).

Замечание Данное стандартное отклонение является смещенной (на малых выборках) оценкой теоретического стандартного отклонения. Для получения несмещенной оценки надо умножить функцию STDDEV на функцию SQRT(N/(N – 1)).

Группа Статистические функции

Родственные функции ARRSTDDEV

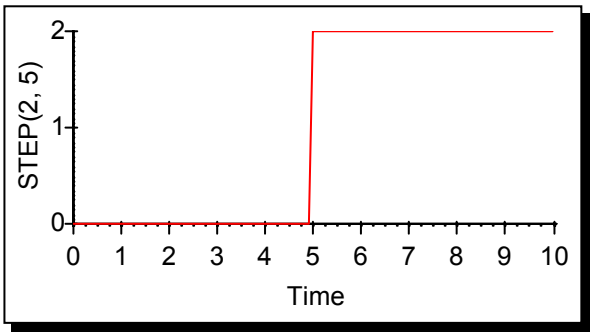
STEP – Ступенчатая функция

Синтаксис STEP (Высота, Начало)

Вход *Высота* – числовое выражение, определяющее высоту ступеньки. *Начало* – выражение, определяющее время появления ступенчатого сигнала.

Результат 0, если время TIME < *Начала*, *Высота* – в противном случае.

Пример Функция STEP (2, 5) дает следующий график:



Замечание Функция зависит от времени.
Демо-версия INVENTOR.SIM
Группа Функции, зависящие от времени
Родственные функции RAMP, PULSE

STOPIF – Условная остановка всех имитаций

Синтаксис STOPIF (Условие)
Вход *Условие* – условие остановки моделирования. Когда *Условие* выполняется, то текущая и все последующие имитации прекращаются. Количество имитаций определяется в диалоговом окне Run Setup (Настройки имитации).
Результат Значение *Условия*
Группа Функции управления
Родственные функции PAUSEIF, PAUSEWHILE, STOPRUNIF

STOPRUNIF – Условная остановка текущей имитации

Синтаксис STOPRUNIF (Условие)
Вход *Условие* – условие остановки моделирования. Когда *Условие* выполняется, то текущая управляемая имитация прерывается и происходит переход к следующей имитации. Количество имитаций определяется в диалоговом окне Run Setup (Настройки имитации).
Результат Значение *Условия*
Группа Функции управления
Родственные функции PAUSEIF, PAUSEWHILE, STOPIF

STOPTIME – Время остановки моделирования

Синтаксис STOPTIME
Результат Время остановки моделирования, определяемое в диалоговом окне Run Setup (Настройки имитации).
Группа Функции, зависящие от времени
Родственные функции STARTTIME, TIME, TIMESTEP, ATSTART

STRATEGICMODE – Проверка нахождения игры в стратегическом режиме

Синтаксис STRATEGICMODE
Результат Истина, если модель выполняется в стратегическом режиме. Ложь, если модель выполняется в операционном режиме. Стратегический режим запускается, когда пользователь нажимает кнопку Try Decision (Испытать решение) на панели принятия решений (см. п. 1.6.2), в то время как при нажатии кнопки Accept Decision (Принять решение) запускается операционный режим.
Группа Смешанные функции
Родственные функции SELECTDECISION

SUM – Суммирование выражений по индексами

Синтаксис	SUM (Dim1, Dim2, ..., DimN; Выражение)
Вход	Dim1, Dim2, ..., DimN – определения индексов (измерений), по которым идет "суммирование". Каждое измерение определяется как произвольная переменная индекса, сопровождаемая знаком (=) и диапазоном. Можно задать несколько одинаковых измерений сразу, перечислив их через запятую. *Выражение* – любое выражение, возможно, включающее индексы Dim1, Dim2, ..., DimN.
Результат	Сумма значений *Выражения* по измерениям Dim1, Dim2, ..., DimN.
Замечание	Функция SUM не имеет обычного синтаксиса, как у других функций, так как можно задавать измерения (как переменные типа массива). Отметим, что для разделения измерений и выражения используется точка с запятой (;). Связь между стандартными математическими обозначениями и функцией SUM иллюстрируется ниже:
Примеры	SUM (i = 1..3; A(i)) = A(1) + A(2) + A (3) SUM (1..3; 2) = 6 SUM (i = 3..9; INDEX (i)) = 42
Группа	Встроенные функции
Родственные функции	ADD, ADDCR, ADDRC, ARRSUM

2.4.8 Функции от T до V

TAN – Тангенс

Синтаксис	TAN (X)
Вход	Угол X – угол в радианах.
Результат	Тангенс угла X.
Группа	Тригонометрические функции
Родственные функции	COS, SIN, ARCTAN, TANH, DEGTORAD, GRADTORAD, PI

TANH – Гиперболический тангенс

Синтаксис	TANH (X)
Вход	Значение X.
Результат	Гиперболический тангенс входного значения X.
Группа	Тригонометрические функции
Родственные функции	COSH, SINH, TAN, DEGTORAD, GRADTORAD, PI

TIME – Текущее время моделирования

Синтаксис	TIME
Результат	Текущее время моделирования, отсчитываемое от момента времени начала моделирования STARTTIME и увеличиваемое на величину TIMESTEP за каждый *Шаг моделирования*.
Замечание	Функция зависит от времени.
Группа	Функции, зависящие от времени
Родственные функции	STARTTIME, STOPTIME, TIMESTEP, ATSTART

TIMECYCLE – Проверка цикличности времени или временного интервала

Синтаксис	TIMECYCLE (Начало, Интервал [, Длительность = 0])
Вход	*Начало* – начальное проверяемое время. *Интервал* – время между проверяемыми моментами времени. *Длительность* – длина интервала (произвольный параметр, по умолчанию равный 0).

Результат Истина, если текущее время моделирования лежит внутри интервала (*Начало* + *k***Интервал*, *Начало* + *k***Интервал* + *Продолжительность*), где *k* – неотрицательное целое число.

Замечание Функция зависит от времени и *Шага моделирования* TIMESTEP, величина которого принимается во внимание при проверке времени TIME.

Группа Функции, зависящие от времени

Родственные функции TIMEIS, TIME, TRUE, FALSE

TIMEIS – Проверка попадания в данный момент времени или временной интервал

Синтаксис TIMEIS (Момент времени [, Продолжительность = 0])

Вход *Момент времени* – числовое выражение, определяющее проверяемое время. *Продолжительность* – продолжительность проверяемого интервала времени (произвольный параметр, по умолчанию равный 0).

Результат Истина, если текущее время находится в интервале (*Момент времени*, *Момент времени* + *Продолжительность*).

Замечание Функция зависит от времени и *Шага моделирования* TIMESTEP. Выражение TIME = T не всегда дает тот же самый результат, что и TIMEIS (T), поскольку функция TIMEIS учитывает размер *Шага моделирования*.

Группа Функции, зависящие от времени

Родственные функции TIMECYCLE, TIME, TRUE, FALSE, ATSTART

TIMESTEP – Шаг моделирования

Синтаксис TIMESTEP

Результат *Шаг моделирования*, определяемый в диалоговом окне Simulation Setup (Установки моделирования).

Группа Функции, зависящие от времени

Родственные функции TIME, STARTTIME, STOPTIME, PULSE, PULSEIF

TRANSPOSE – Транспонирование матрицы

Синтаксис TRANSPOSE (A(M, N))

Вход A – выражение с двумерным массивом.

Результат Двумерный N x M массив (матрица), в котором строки и столбцы поменяны местами.

Уравнения Транспонированная матрица TA для матрицы A может быть найдена следующим образом:

```
aux     A(M,N) = ...
aux     TA(i=1..N, j=1..M) = A(j,i)
```

Группы Функции для работы с массивами, Математические функции

Родственные функции MATRIXPROD, TRANSPOSEC

TRANSPOSEC – Транспонирование матрицы комплексных чисел

Синтаксис TRANSPOSEC (A ([[M,] N,] 2))

Вход A – комплексное число, вектор или матрица в декартовой форме.

Результат Транспонированная матрица R ([N [, M,]] 2), где
R (i, j, 1..2) = A(j, i, 1..2)

Замечание Если A имеет менее трех измерений, то возвращается неизменной.

Группа Комплексные функции

Родственные функции TRANSPOSE

TREND – Тренд

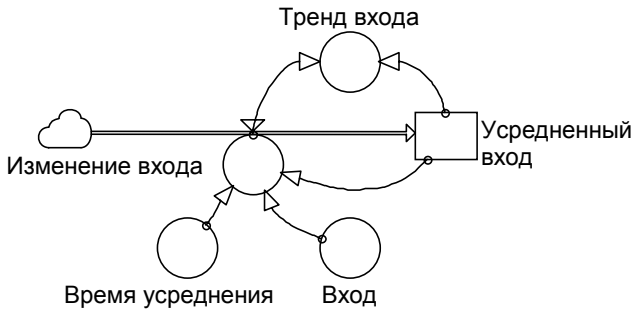
Синтаксис TREND (Вход, Время усреднения [, Начальное значение = 0])

Вход *Вход* – числовое выражение, которое будет исследовано через некоторое время. *Время усреднения* – промежуток времени, за который проводится усреднение для получения экстраполяции тренда. *Начальное значение* – начальное значение функции TREND (произвольный начальный параметр, по умолчанию равный 0).

Результат Экспоненциальное среднее 1-го порядка для темпа изменения *Входа*, использующее данное *Время усреднения*. Значение выражается как относительное изменение *Входа* в единицу времени.

График Функцию TREND можно смоделировать следующим образом:

```
init   Усредненный_вход = ...
flow   Усредненный_вход = + dt*Изменение_входа
aux    Изменение_входа = (Вход - Среднее Входа)/
       Время_усреднения
aux    Тренд_входа = Изменение_входа/Усредненный_вход
aux    Время_усреднения = ...
aux    Вход = ...
```



Замечание Функция пользует структуру функции информационной задержки (см. DELAYINF) и зависит от предыдущих значений первого параметра.

Группа Функции с памятью

TRUE – Логическая ИСТИНА

Синтаксис TRUE

Результат Значение, равное 1.

Замечание При проверке условий любое значение, получаемое при округлении до отличного от нуля числа, оценивается как Истина.

Группа Логические функции

Родственные функции FALSE, BOOL

VECTLEN – Длина вектора

Синтаксис VECTLEN (A(N))

Вход A – любое векторное выражение.

Результат Длина вектора, т.е. квадратный корень из суммы квадратов абсолютных значений элементов: SQRT (SUM (i = 1..N; A(i) ^ 2)).

Группы Функции для работы с массивами, Математические функции

Родственные функции ABS, ABSC, ELEMCOUNT, HYPOT

VECTOR – Создание вектора

Синтаксис VECTOR (Elem1, Elem2, ..., ElemN)

Вход Elem1, Elem2, ..., ElemN – элементы вектора. Элементы могут или быть скалярами или массивами с равными размерностями.

Результат Массив с элементами Elem1, Elem2, ..., ElemN. Результирующий массив имеет еще одну размерность по сравнению с каждым из элементов ElemI.

Группа Функции для работы с массивами

Родственные функции LOOKUP

VECTPROD3D – Векторное произведение трехмерных векторов

Синтаксис VECTPROD3D (A(3), B(3))

Вход A, B – выражения с компонентами трехмерных векторов.

Результат Вектор с тремя компонентами, представляющий векторное произведение A и B.

Группы Функции для работы с массивами, Математические функции

Родственные функции ARRPROD, MATRIXPROD, PRODC, PRODCR, PRODRC, SPROD

2.4.9 Функции от W до Z

WHEN – Оператор защиты

Синтаксис A WHEN B

Вход A – любое выражение. B – любое ограничение, например, выражение, возводящее в степень переменные индекса, ссылки на диапазон и целые числа.

Результат A, если B истинен, неопределенность – в противном случае.

Замечание Альтернативным названием оператора защиты является оператор (|).

Группа Встроенные функции

Родственные функции ; , | , BUT

XOR – Логическое исключающее ИЛИ

Синтаксис A XOR B

Вход A, B – логические значения (Истина или Ложь)

Результат Истина, если либо A, либо B равно Истине, Ложь – в противном случае, т.е. (A AND NOT B) OR (NOT A AND B) или (проще): BOOL (A) < > BOOL (B).

Замечание При использовании < > и = с логическими значениями, хорошая практикой является первоначальное использование оператора BOOL. Это гарантирует, что истинные значения будут равны 1. Пример ниже поясняет это, поскольку два выражения не дают одинаковых результатов: равенство 3 = 4 является ложным, тогда как равенство BOOL (3) = BOOL (4) истинно.

Сходный способ выражения A XOR B следующий: A + B - 2*A* B.

Группа Логические функции

Родственные функции XOR, AND, NOT, BOOL, TRUE, FALSE

2.5 Сообщения об ошибках

В данном разделе рассматриваются сообщения об ошибках, возникающие при попытках утвердить путем нажатия кнопки Set (Установить) недопустимые определения переменных или объявления размерностей переменных в диалоговом окне Define Variable (Определение переменной), см. п. 1.4.6.

Ошибка C0001 – '...' ожидается

Когда определение переменной не полно, согласно синтаксису Powersim, то данное сообщение об ошибке указывает на то, что могло бы отсутствовать. Автоматически Powersim выбирает ту часть определения, куда нужно вставиться отсутствующий элемент.

Например, если Вы введете "+" в поле Definition (Определение), диалога Define Variable (Определение переменной) и нажмете кнопку Set (Установить), то сообщение об ошибке укажет, что: "Ожидается операнд". Это связано с тем, что оператор "+" требует последующего операнда. Допустимое определение выглядело бы так: "A + B", подразумевая, что операнды A и B связаны с определяемой переменной.

Примеры элементов, которые могут быть указаны как ожидаемые в данном сообщении об ошибке:

Элемент	Пример	Объяснение
, (Запятая)	IF (A > 0, B C)	Запятая между аргументами B и C при вызове функции
Операнд	+	Оператор "+" должен сопровождаться, например, названием переменной, вызовом функции, или числом
)	IF (A > 0, B, C	Правая круглая скобка после последнего аргумента функции
(	IF A> 0, B, C)	Левая круглая скобка перед первым аргументом функции
Конец определения	A B	
Целое число	i = 1 ..	Определение размерности массива, в котором отсутствует верхний предел диапазона

Ошибка C0002 – Ошибка в числе

Выбранное число не допустимо с точки зрения синтаксиса Powersim. Пример допустимого числа: "1.23".

Ошибка C0003 – Выражение с индексом выходит за рамки диапазона

Определение содержит ссылку на переменную типа массив, использующую нижний индекс, в которой выбранный индекс превышает фактический диапазон для индекса вызванной переменной. В следующем примере определение B неверно, так как содержит только три элемента.

```
dim     A = 1..3
aux     B = A(4)
```

Ошибка C0004 – Несоответствие размерностей

Размерность выбранного выражения не соответствует объявленной размерности определяемой переменной. См. примеры ниже:

Определение	Комментарии
`aux B(i=1..4)=[1,2]`	**Неверно:** есть только два элемента в определении B, в то время как переменная B объявлена с 4 элементами.
`aux B(i=1..4)=[1,2,5,7]`	Верно: определение (i = 1..4) и определение ([1,2,5,7]) содержит 4 элемента.
`aux A(1..2)= TIME` `aux B(i=1..4)= A`	**Неверно:** так как A объявлен с 2 элементами, а B объявлен с 4 элементами, то переменной A нужно присвоить индекс для использования при определении B.

`aux A(1..2) = TIME`
`aux B(i=1..4)= A(i)|`
`i=1..2;0`
	Верно: несмотря на различные размерности в определении A и B, переменную B еще можно определить в терминах A, так как A был снабжен индексом и ограничен для соответствия размерности переменной B.

Ошибка C0005 – Неизвестное название '...'

Набранная текстовая строка не является известной переменной или функцией. Возможно, это обусловлено наличием орфографической ошибки.

Ошибка C0006 – Запрещенная ссылка на переменную

Выбранная переменная не связана с текущей переменной, или она ошибочно соединена с использованием связи с запаздыванием вместо обычной связи. Закройте диалоговое окно Define Variable (Определение переменной), и нарисуйте связь от выбранной переменной к определяемой переменной, используя соответствующий тип связи.

Ошибка C0007 – Соединенная переменная '...' не используется

Все соединенные с текущей переменной переменные должны использоваться для ее определения. Список Linked Variables (Связанные переменные) в диалоговом окне Define Variable (Определение переменной) содержит все связанные переменные. Если некоторые из связанных переменных не нужны для определения определяемой переменной, то следует закрыть диалоговое окно и удалить указанные неиспользуемые связи. Это можно сделать, выделив сначала неиспользуемые связи, а затем нажав кнопку Clear (Очистка) в меню Edit (Правка).

Ошибка C0008 – Переменная индекса ограничена дважды

Переменная индекса используется дважды в списке защиты, как в примере ниже:

```
dim     A = (i = 1..10)
aux     A = TIME, WHEN i> 2 AND i < 7 BUT TIME/2 OTHERWISE
```

Переменная индекса i используется в двух различных защитах в одном и том же списке защиты. Допустимое определение может быть следующим:

```
dim     A = (i = 1..10)
aux     A = TIME, WHEN i = 3..6 BUT TIME/2 OTHERWISE
```

Ошибка C0009 – Внутренняя ошибка во время второго прохода!

Эта ошибка никогда не должна произойти. Однако если она произошла, то, пожалуйста, сохраните документ и свяжитесь с компанией Powersim.

Ошибка C0010 – Ошибка проверки размерности! ('...')

Функция не принимает размерность выбранного параметра. Если воспользоваться следующим определением переменной:

```
aux     B = ARRMAX (3),
```

то появится сообщение об ошибке: Ошибка проверки размерности! ("Ожидается массив!") Замечание: функции, в которых в качестве параметров используются массивы, используют переменное число параметров. Это означает, что если используемая переменная не снабжена индексом, то функция использует все измерения.

Ошибка C0011 – Неоднозначное выражение ограничения

Некоторые из элементов текущей переменной типа массив определены дважды. В примере ниже переменная определена дважды при i = 3:

```
dim     A = (i = 1 .. 4)
aux     A = 5 WHEN i < = 3 BUT 3 WHEN i > 2
```

См. также выражения ограничений в п. 2.4.

Ошибка C0012 – Незаконченное выражение ограничения

Некоторые из элементов текущей переменной типа массив не определены. В примере ниже переменная не определена при i = 3.

```
dim     A = (i = 1 .. 4)
aux     A = 5 WHEN i < 3 BUT 3 WHEN i > 3
```

См. также выражения ограничений в п. 2.4.

Ошибка C0013 – Единственным верным числом в определениях потоков является 0

Выбранное число не допустимо в определении потока. Его можно заменить нулем (0) для получения допустимого определения потока. Эта ошибка возникает, когда редактируется заданное по умолчанию определение уровня через потоки (сгенерированное из потоков, соединенных с данным уровнем на потоковой диаграмме). Можно подставить поток (или части потока, если уровень представлен массивом) с нулевым, но не с другим числовым значением. В примере ниже использование числа 10 в определении *Уровня* через потоки недопустимо.

```
dim     Уровень = (i = 1..2)
init    Уровень = 0
flow    Уровень = + dt*(Темп | i = 1; 10)
const   Темп = 10
```

Ошибка C0014 – Единственной допустимой ссылкой на темп является '...'

Выделенный текст не распознан как название соединенного темпа, характеризующего данный поток. Замените выделенный текст правильным названием темпа или другим допустимым определением потока.

Ошибка C0015 – Не достаточно памяти для добавления переменной

Powersim не может выделить достаточно памяти для добавления определения новой переменной в текущую модель. Попробуйте освободить память, например, закрыв другие выполняющиеся приложения.

Ошибка C0016 – Функция '...' не может использоваться как внутренний "n"-ый параметр функции '...'

Функции, которые распределяют память или генерируют внутренние уровни, не могут использоваться как вычисляемые параметры. В следующем примере второй параметр функции IF не допустим, поскольку функция, генерирующая внутренние уровни, используется в качестве вычисляемого параметра.

```
aux     A = TIME
aux     B = IF(TIMEIS(5),DELAYMTR(A,3),0)
```

Ошибка C0017 – '...' – переменная, соединенная связью без запаздывания

Выбранную переменную необходимо соединить с текущей переменной, используя связь с запаздыванием. Это требуется для всех входных аргументов функции задержки. В примере ниже A необходимо соединить с B, используя связь с запаздыванием при допустимом определении B.

```
aux     A = TIME
aux     B = DELAYINF(A,1)
```

Ошибка C0018 – Переменная '...' со связью с запаздыванием не используется

Фактическая переменная, соединенная с текущей переменной при помощи связи с запаздыванием, должна использоваться как параметр с запаздыванием для корректности определения. В первом примере ниже A – параметр с запаздыванием, в то время как во втором примере A – параметр без запаздывания:

```
aux     B = DELAYINF(A,1)     "параметр с запаздыванием"
aux     B = SQRT(A)           "параметр с запаздыванием"
```

Ошибка C0019 – Выражения ограничений не допустимы внутри функций, возвращающих массивы

Отбор указывает, что Вы пробовали использовать выражение ограничения как часть аргумента функции, которая возвращает массив. В примере ниже выражение ограничения "1 WHEN i = 1; 2" не допустимо внутри функции LOOKUP, так как в данном случае функция возвращает массив.

```
dim     A = (1..3,1..2)
const   A = [[2,4],[6,1],[3,5]]
dim     B = (i = 1..2)
aux     B = LOOKUP(A,1 WHEN i = 1; 2)
```

Ошибка C0020 – Переменные индекса не допустимы в функциях, возвращающих массивы

Функция, в которой используется выбранная переменная индекса, возвращает массив. Следовательно, она не принимает параметры, выраженные в терминах переменных индекса. В примере ниже использование переменной индекса i внутри функции VECTOR не допускается.

```
dim     A = (i = 1..2)
aux     A = VECTOR(1,B(i))
dim     B = (1..2)
const   B = 1
```

Ошибка C0021 – Не возможно преобразовать '...' в вещественное число

Выбранное выражение содержит недопустимую попытку преобразования диапазона в вещественное число. В примере ниже определение A не допустимо, поскольку первая функция пытается преобразовывать *Область* в вещественное число. Выражение для B допустимо, так как диапазон *Областей* преобразуется в значение индекса, которое является целым числом.

```
range   Область = СЕВЕР,ВОСТОК,ЮГ,ЗАПАД
aux     A = FIRST(Область)
dim     B = (R = Область)
const   B = 10, WHEN R = FIRST(Область) BUT 0
```

Ошибка C0022 – Нельзя смешивать элементы из различных числовых диапазонов

Выбранная спецификация диапазона не допустима, так как элементы происходят из различных переменных типа диапазон. В примере ниже предпринимается попытка смешения элементов из различных диапазонов *Возраст* и *Область* при определении размерности диапазона A.

```
range   Возраст = МОЛОДОЙ,ВЗРОСЛЫЙ,СТАРЫЙ
range   Область = СЕВЕР,ВОСТОК,ЮГ,ЗАПАД
dim     A = (i = МОЛОДОЙ..ЮГ)
```

Ошибка C0023 – Минимум больше максимума в диапазоне '...'

Выбранная спецификация размерности не допустима. В примере ниже размерность B не допустима, так как минимум всегда должен быть меньше максимума:

```
dim     B = (i = 4..3)
aux     B = A(i)
```

Ошибка C0024 – Не возможно выполнить операцию '...' + '...'

Операция сложения (+), предпринятая для выбранной части объявления размерности, не допустима. В примерах ниже определение A не допустимо, в то время как B и C допустимы. Общее правило состоит в том, что можно складывать элемент диапазона и число, но не два элемента диапазона.

```
range   Область = СЕВЕР,ВОСТОК,ЮГ,ЗАПАД
dim     A = (i = СЕВЕР..СЕВЕР + ЮГ)      "не допустимо"
dim     B = (i = СЕВЕР..2 + СЕВЕР)       "допустимо"
dim     C = (i = СЕВЕР..СЕВЕР + 2)       "допустимо"
```

Ошибка C0025 – Не возможно выполнить операцию '...' - '...'

Операция вычитания (–), предпринятая для выбранной части объявления размерности, не допустима. В примерах ниже определение A и B не допустимы, в то время как B и C допустимы. Общее правило состоит в том, что можно вычитать число из элемента диапазона, но не наоборот.

```
range    Область = СЕВЕР,ВОСТОК,ЮГ,ЗАПАД
dim      A = (i = СЕВЕР..4 - ЮГ)            "не допустимо"
dim      B = (i = СЕВЕР..ЗАПАД - МОЛОДОЙ)   "не допустимо"
dim      C = (i = СЕВЕР..ЗАПАД - СЕВЕРНЫЙ)  "допустимо"
dim      D = (i = СЕВЕР..ЗАПАД - 2)         "допустимо"
```

Ошибка C0026 – Переменные индекса нельзя преобразовать, используя поддиапазон

Предпринимается недопустимая попытка преобразования переменной индекса с использованием поддиапазона. В примере ниже приводится недопустимое определение, поскольку оно содержит преобразование переменной индекса с использованием поддиапазона *Подобласть*.

```
range   Возраст = МОЛОДОЙ,ВЗРОСЛЫЙ,СТАРЫЙ
range   Область = СЕВЕР,ВОСТОК,ЮГ,ЗАПАД
range   Подобласть = СЕВЕР..ВОСТОК
dim     A = (i = 1..3)
Aux     A = B(Подобласть(i))
dim     B = (Возраст)
const   B = 1
```

Ошибка C0027 – Индекс '...' используется дважды

Это сообщение об ошибке больше не используется.

Ошибка C0028 – Диапазон '...' плохо определен

Выбранный диапазон плохо определен. Если диапазон определен как глобальный, необходимо открыть диалоговое окно Define Range (Определение диапазона) и переопределить диапазон. Например:

```
range   ErrRange = 1..0
```

Ошибка C0029 – Необходима связь от переменной '...'

Фактическую переменную необходимо соединить с текущей переменной, используя связь. В примере ниже требуется инициализирующая связь от A к B, так как функция DELAYINF использует свой первый аргумент, т.е. A, в качестве заданной по умолчанию инициализации задержки.

```
aux     A = TIME
aux     B = DELAYINF(A,1,1)
```

Ошибка C0030 – Размерность переменной '...' плохо определена

Размерность выбранной переменной не верна. Переопределите правильно размерность выбранной переменной. В примере ниже размерности переменных A и B не допустимы.

```
range   ErrRange = 1..0
dim     A = (ErrRange)
aux     A = 1
aux     B = A(1)
```

Ошибка C0031 – Вложенные ограничения не допустимы

Выражения ограничений нельзя вкладывать друг в друга. В примере ниже A не допустимо, в то время как B допустимо, хотя выражает то же самое, что и A, используя оператор AND.

```
dim     A = (i = 1..3, j = 1..2)
const   A = (1 WHEN i = 1 BUT 0 ИНАЧЕ) WHEN j = 1 BUT 2
        OTHERWISE
dim     B = (i = 1..3, j = 1..2)
const   B = 1 WHEN i = 1 AND j = 1 BUT 2 WHEN j > 1 BUT 0
        OTHERWISE
```

Ошибка C0032 – Элемент '...' не определен

Выбранный элемент диапазона лежит вне заданного интервала диапазона. В примере ниже как A, так и B не допустимы.

```
range   Область = СЕВЕР,ВОСТОК,ЮГ,ЗАПАД
dim     A = (i = Область(1)..Область(7))
dim     B = (i = СЕВЕР..СЕВЕР + 4)
```

Ошибка C0033 – Слишком сложное выражение

Определение превышает предел сложности для выражений в Powersim. Появление ошибки указывает, когда был достигнут предельный уровень сложности. Пример ниже становился допустимым при удалении одной из функций SIN.

```
aux     A = SIN(SIN(SIN(SIN(SIN(SIN(SIN(SIN(SIN(SIN(SIN
        (SIN(SIN(TIME))))))))))))
```

Ошибка C0034 – Арифметические выражения, включающие кратные индексы, не допустимы

Выбранный индекс массива не допустим, поскольку он состоит из кратных переменных индекса. В примере ниже определение A не допустимо, так как предпринимается попытка использовать $i + j$ как индекс для B.

```
dim     A = (i = 1..3, j = 1..3)
aux     A = B(i + j)
dim     B = (1..5)
```

Ошибка C0035 – Не возможно составить функцию '...' из-за отсутствия прав доступа к библиотеке функций '...'

Обозначенная функция принадлежит библиотеке функций, которую данная система Powersim не имеет права использовать.

Ошибка C0036 – Целое число выходит за рамки допустимого диапазона

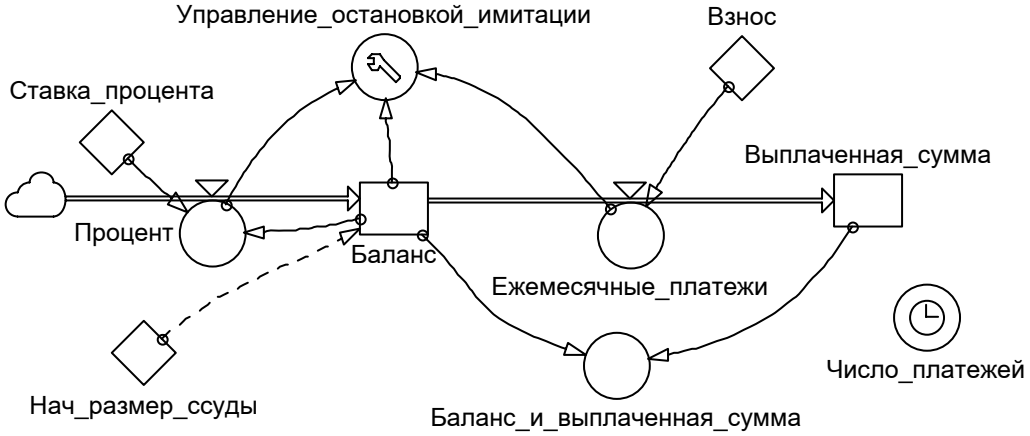
Целые числа должны лежать в диапазоне [-32750 .. 32750]. Нижеуказанная размерность не допустима.

```
dim     A = (i = 32750..32750 + 1)
```

Приложения*

1. Модель погашения ссуды (FINANCE.SIM)

Потоковая диаграмма:



Уравнения модели:

Уравнения уровней

```
init   Баланс = MAX(0,Нач_размер_ссуды)
flow   Баланс = +dt*Процент-dt*Ежемесячные_платежи
doc    Баланс = Текущая сумма долга
unit   Баланс = руб.
init   Выплаченная_сумма = 0
flow   Выплаченная_сумма = +dt*Ежемесячные_платежи
doc    Выплаченная_сумма = Общая сумма, выплаченная кредитору
unit   Выплаченная_сумма = руб.
```

Уравнения темпов

```
aux    Ежемесячные_платежи = Взнос
doc    Ежемесячные_платежи = Размер ежемесячного платежа по ссуде
unit   Ежемесячные_платежи = руб./месяц
aux    Процент = Баланс*(Ставка_процента/12)
doc    Процент = Накопленный процент
unit   Процент = руб./месяц
```

Вспомогательные уравнения

```
aux    Баланс_и_выплаченная_сумма = Выплаченная_сумма+Баланс
doc    Баланс_и_выплаченная_сумма = Общая сумма денег, которая будет
       выплачена кредитору – в отличие от выплаченной суммы, она включает
       все взносы плюс остаток (Баланс)
unit   Баланс_и_выплаченная_сумма = руб.
aux    Управление_остановкой_имитации=STOPIF(Баланс+Процент<Ежемесячные_
       платежи)
doc    Управление_остановкой_имитации = Функция управления, которая
       останавливает моделирование, если баланс плюс текущий процент
       меньше, чем ежемесячный взнос
unit   Управление_остановкой_имитации = безразмерная
```

* В приложениях дан русский перевод моделей, разработанных сотрудниками компании Powersim (Umar Imrana Ahmed, Daniel E. Angelakis, Arne-Helge Byrknes, Raymond P. Capece) и включенных в демонстрационную версию пакета Powersim.

Дополнительные уравнения

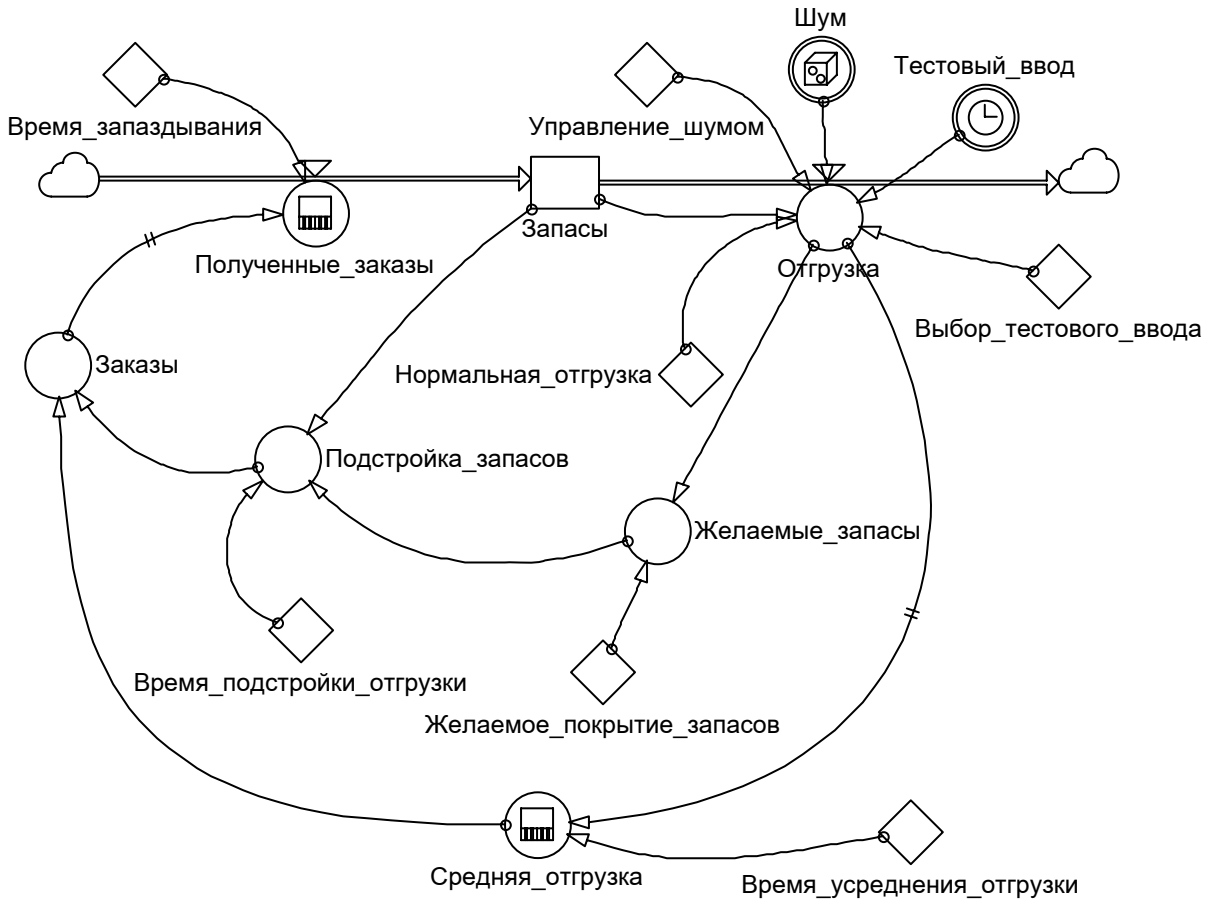
```
aux    Число_платежей = TIME
doc    Число_платежей = Число произведенных платежей
unit   Число_платежей = ед.
```

Константы

```
const  Взнос = 0
doc    Взнос = Ежемесячный взнос, установленный кредитором
unit   Взнос = руб./месяц
const  Нач_размер_ссуды = 0
doc    Нач_размер_ссуды = Размер ссуды
unit   Нач_размер_ссуды = руб.
const  Ставка_процента = 0%
doc    Ставка_процента = Годовая процентная ставка
unit   Ставка_процента = безразмерная

spec   начало = 0
spec   конец = 300
spec   dt = 1
spec   метод = РК4 (перемен. шаг)
spec   абс.ошибка = 0.01
spec   отн.ошибка = 0.01
```

2. Модель управления запасами товаров (INVENTOR.SIM)

Потоковая диаграмма:



Уравнения модели:

Уравнения уровней

```
init   Запасы = 300
flow   Запасы = -dt*Отгрузка+dt*Полученные_заказы
doc    Запасы = Число изделий в наличии
unit   Запасы = ед./неделя
```

Уравнения темпов

```
aux    Отгрузка = MIN(Нормальная_отгрузка,Запасы) + LOOKUP(Тестовый_ввод,
       Выбор_тестового_ввода)+ LOOKUP(Шум,Управление_шумом)
doc    Отгрузка = Количество отгруженных за неделю запасенных материалов.
       Функция LOOKUP используется для того, чтобы выбрать элемент век-
       тора Тестового_ввода по значению переменной Выбор_тестового_ввода
aux    Полученные_заказы = ROUND(DELAYMTR(Заказы,Время_запаздывания,3))
doc    Полученные_заказы = Материальная задержка 3-го порядка по
       отношению к заказам
unit   Полученные_заказы = ед./неделя
```

Вспомогательные уравнения

```
aux    Желаемые_запасы = Отгрузка*Желаемое_покрытие_запасов
doc    Желаемые_запасы = Желаемый для поддержания уровень запасов
unit   Желаемые_запасы = ед.
aux    Заказы = ROUND(Средняя_отгрузка+Подстройка_запасов)
unit   Заказы = ед.
aux    Подстройка_запасов = (Желаемые_запасы-
       Запасы)/Время_подстройки_отгрузки
doc    Подстройка_запасов = Разница между желаемыми и реальными запасами
unit   Подстройка_запасов = ед./неделя
aux    Средняя_отгрузка = DELAYINF(Отгрузка,Время_усреднения_отгрузки)
doc    Средняя_отгрузка = Сглаженная по времени средняя отгрузка
unit   Средняя_отгрузка = ед./неделя
```

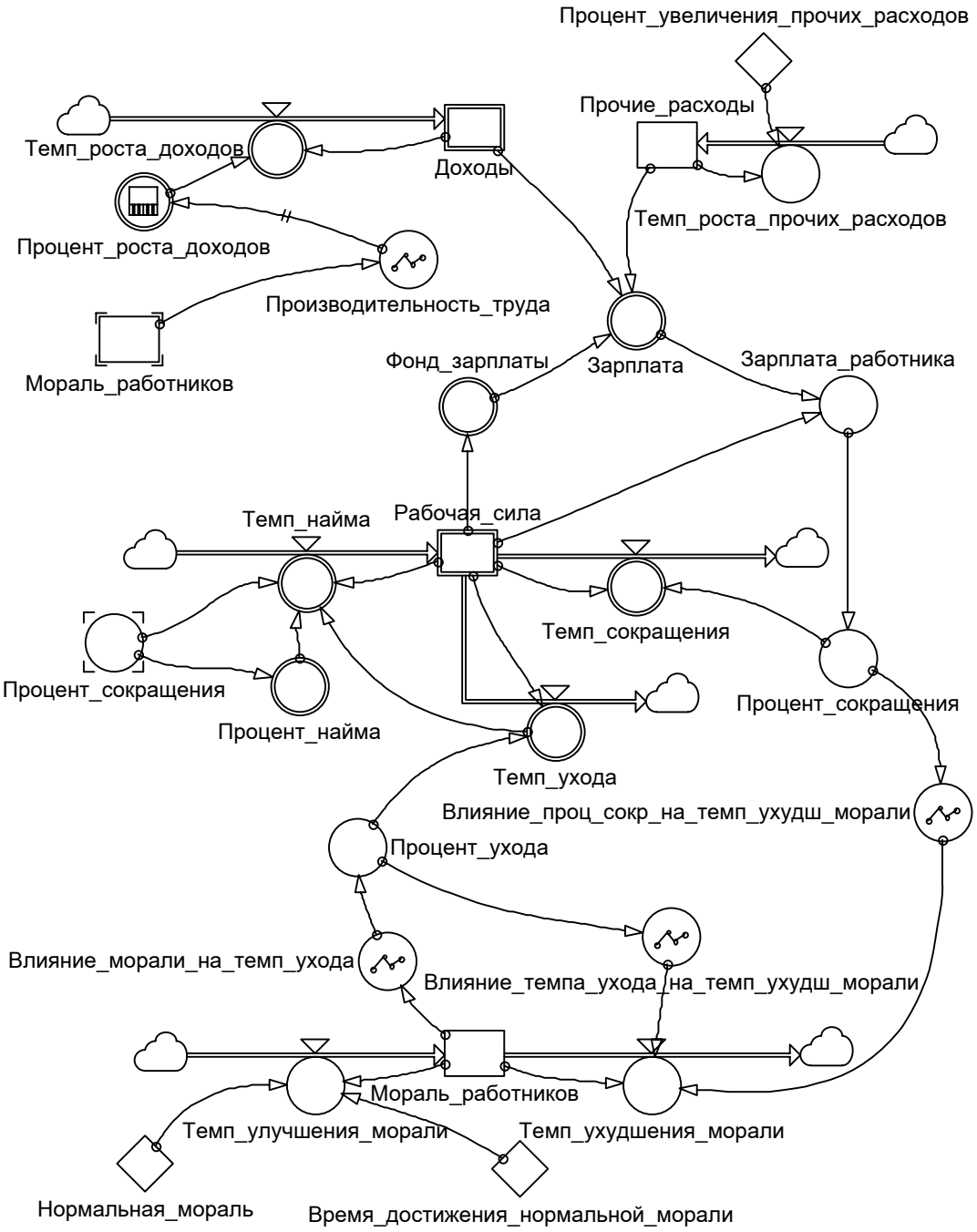
Константы

```
dim    Тестовый_ввод = (i=1..4)
aux    Тестовый_ввод = STEP(10,10) WHEN i=1 BUT RAMP(20,10) WHEN i=2 BUT
       PULSE(10,10,200) WHEN i=3 BUT STEP(SINWAVE(10,25),15)
unit   Тестовый_ввод = ед./неделя
dim    Шум = (i=1..2)
aux    Шум = RANDOM(10,20) WHEN i=1 BUT 0 WHEN i=2
doc    Шум = Случайная составляющая отгрузки
unit   Шум = ед./неделя
const  Время_запаздывания = 3
doc    Время_запаздывания = Количество недель, необходимое для получения
       заказанных материалов
unit   Время_запаздывания = ед./неделя
const  Время_подстройки_отгрузки = 2
doc    Время_подстройки_отгрузки = Количество недель, необходимое для
       устранения диспропорции между желаемыми запасами и реальными
unit   Время_подстройки_отгрузки = неделя
const  Время_усреднения_отгрузки = 2
doc    Время_усреднения_отгрузки = Количество недель, необходимое для
       усреднения отгрузки
unit   Время_усреднения_отгрузки = неделя
const  Выбор_тестового_ввода = 1
doc    Выбор_тестового_ввода = Выбор тестового ввода определяется
       переменной Тестовый_ввод: 1: STEP(10,10); 2: RAMP(20,10);
       3: PULSE(10,10,200); 4: STEP(SINWAVE(10,25),15)
const  Желаемое_покрытие_запасов = 3
doc    Желаемое_покрытие_запасов = Запасов должно хватать для покрытия
       как минимум 3-х недельного предложения
```

```
unit   Желаемое_покрытие_запасов = неделя
const  Нормальная_отгрузка = 100
doc    Нормальная_отгрузка = Количество товара, отгружаемое еженедельно в
       обычных условиях
unit   Нормальная_отгрузка = ед./неделя
const  Управление_шумом = 1
doc    Управление_шумом = 1: Включить шум. 2: Выключить шум.

spec   начало = 0
spec   конец = 100
spec   dt = 1
spec   метод = РК2 (фикс. шаг)
```

3. Модель учета морали персонала фирмы (MORALE.SIM)[*]

Потоковая диаграмма:



[*] Под моралью в данном случае понимается, прежде всего, психологический микроклимат внутри фирмы.

Уравнения модели:

Уравнения уровней

```
dim    Доходы = (1..2)
init   Доходы = 47000000
flow   Доходы = +dt*Темп_роста_доходов
doc    Доходы = Средства, заработанные за предшествующие 12 месяцев
       Первый элемент вектора отражает реальные доходы, а второй –
       потенциальные
unit   Доходы = руб.
init   Мораль_работников = 100
flow   Мораль_работников = -dt*Темп_ухудшения_морали+dt*Темп_улучшения_
       морали
doc    Мораль_работников = Уровень морали компании
unit   Мораль_работников = ед.морали
init   Прочие_расходы = 30000000
flow   Прочие_расходы = +dt*Темп_роста_прочих_расходов
doc    Прочие_расходы = Расходы, не связанные с оплатой труда работников
       фирмы
unit   Прочие_расходы = руб.
dim    Рабочая_сила = (1..2)
init   Рабочая_сила = 400
flow   Рабочая_сила = -dt*Темп_ухода-dt*Темп_сокращения+dt*Темп_найма
doc    Рабочая_сила = Велична рабочей силы компании: первый элемент
       вектора отражает политику сокращения производства, а второй – нет
unit   Рабочая_сила = чел.
```

Уравнения темпов

```
dim    Темп_найма = (i=1..2)
aux    Темп_найма = IF(Процент_сокращения>0,0,Рабочая_сила*Процент_найма
       +Темп_ухода) WHEN i=1 BUT Рабочая_сила*Процент_найма WHEN i=2
doc    Темп_найма = Скорость найма работников: первый элемент вектора
       отражает политику сокращения производства, а второй – нет
unit   Темп_найма = чел./месяц
dim    Темп_роста_доходов = (1..2)
aux    Темп_роста_доходов = Доходы*Процент_роста_доходов
doc    Темп_роста_доходов = Скорость ежемесячного увеличения доходов
       фирмы: первый элемент вектора отражает реальный темп роста
       доходов, а второй – потенциальный
unit   Темп_роста_доходов = руб./месяц
aux    Темп_роста_прочих_расходов = Прочие_расходы*Процент_увеличения_
       прочих_расходов
doc    Темп_роста_прочих_расходов = Скорость увеличения расходов, не
       связанных с оплатой труда работников фирмы
unit   Темп_роста_прочих_расходов = руб./месяц
dim    Темп_сокращения = (i=1..2)
aux    Темп_сокращения = Процент_сокращения*Рабочая_сила(1) WHEN i=1
       BUT 0 WHEN i=2
doc    Темп_сокращения = Скорость увольнения работников вследствие
       сокращения производства: первый элемент вектора отражает политику
       сокращения производства, а второй – нет
unit   Темп_сокращения = чел./месяц
aux    Темп_улучшения_морали = (Нормальная_мораль-Мораль_работников)/
       Время_достижения_нормальной_морали
doc    Темп_улучшения_морали = Скорость обычного улучшения морали
unit   Темп_улучшения_морали = ед.морали/месяц
dim    Темп_ухода = (i=1..2)
aux    Темп_ухода = Рабочая_сила(1)*Процент_ухода WHEN i=1 BUT 0 WHEN i=2
doc    Темп_ухода = Скорость добровольного увольнения работников
unit   Темп_ухода = чел./месяц
```

```
aux    Темп_ухудшения_морали = Мораль_работников*(Влияние_проц_сокр_на_
       темп_ухудш_морали+Влияние_темпа_ухода_на_темп_ухудш_морали)
doc    Темп_ухудшения_морали = Скорость ухудшения морали вследствие
       добровольного увольнения сотрудников и увольнения, связанного с
       сокращением производства
unit   Темп_ухудшения_морали = ед.морали/месяц
```

Вспомогательные уравнения

```
aux    Влияние_морали_на_темп_ухода = GRAPH(Мораль_работников,0,10,
       [0.058,0.051,0.047,0.04,0.034,0.028,0.025,0.019,0.017,0.011,0
       "Min:0;Max:0.4"])
unit   Влияние_морали_на_темп_ухода = безразмерная
aux    Влияние_проц_сокр_на_темп_ухудш_морали = GRAPHCURVE(Процент_
       сокращения,0,0.03,[0,0.1,0.3,0.61,0.73,0.74,0.75,0.75,0.75,0.81,
       0.99"Min:0;Max:1"])
unit   Влияние_проц_сокр_на_темп_ухудш_морали = безразмерная
aux    Влияние_темпа_ухода_на_темп_ухудш_морали = GRAPH(Процент_ухода,0,
       0.02,[0,0.07,0.1,0.13,0.14,0.16,0.17,0.19,0.2,0.21,0.22"Min:0;
       Max:1"])
unit   Влияние_темпа_ухода_на_темп_ухудш_морали = безразмерная
dim    Зарплата = (i=1..2)
aux    Зарплата = Доходы(1)-Прочие_расходы-Фонд_зарплаты(1) WHEN i=1
       BUT Доходы(2)-Прочие_расходы-Фонд_зарплаты(2) WHEN i=2
doc    Зарплата = Зарплата работников за предшествующие 12 месяцев:
       первый элемент вектора отражает реальную зарплату, а второй -
       потенциальную
unit   Зарплата = руб.
aux    Зарплата_работника = Зарплата(1)/Рабочая_сила(1)
doc    Зарплата_работника = Годовая зарплата одного работника фирмы
unit   Зарплата_работника = руб./чел.
aux    Производительность_труда = GRAPH(Мораль_работников,0,10,[0.51,
       0.52,0.52,0.53,0.54,0.56,0.6,0.67,0.76,0.88,1"Min:0;Max:1"])
doc    Производительность_труда = качество/количество выпускаемой
       работником продукции. В обычном режиме они равны 1 и изменяются
       под воздействивем изенений морали работника
unit   Производительность_труда = безразмерная
dim    Процент_найма = (i=1..2)
aux    Процент_найма = IF(Процент_сокращения>0,0,0.015) WHEN i=1 BUT
       0.015 WHEN i=2
doc    Процент_найма = Процент найма новых работников
unit   Процент_найма = %/месяц
dim    Процент_роста_доходов = (i=1..2)
aux    Процент_роста_доходов = 0.0178*DELAYPPL(Производительность_труда,
       12,1) WHEN i=1 BUT 0.0178 WHEN i=2
doc    Процент_роста_доходов = Ежемесячный процент увеличения доходов,
       при котором увеличивается годовой доход: первый элемент вектора
       отражает реальный процент увеличения дохода, а второй -
       потенциальный
unit   Процент_роста_доходов = %/месяц
aux    Процент_сокращения = IF(Зарплата_работника<18000,0.05,0)
doc    Процент_сокращения = Политика фирмы такова, что если зарплата
       работника становится меньше 18000 руб., то тогда происходит
       сокращение производства и увольнение работников
unit   Процент_сокращения = %/месяц
aux    Процент_ухода = Влияние_морали_на_темп_ухода
doc    Процент_ухода = Процент работников, увольняющихся за месяц
unit   Процент_ухода = %/месяц
```

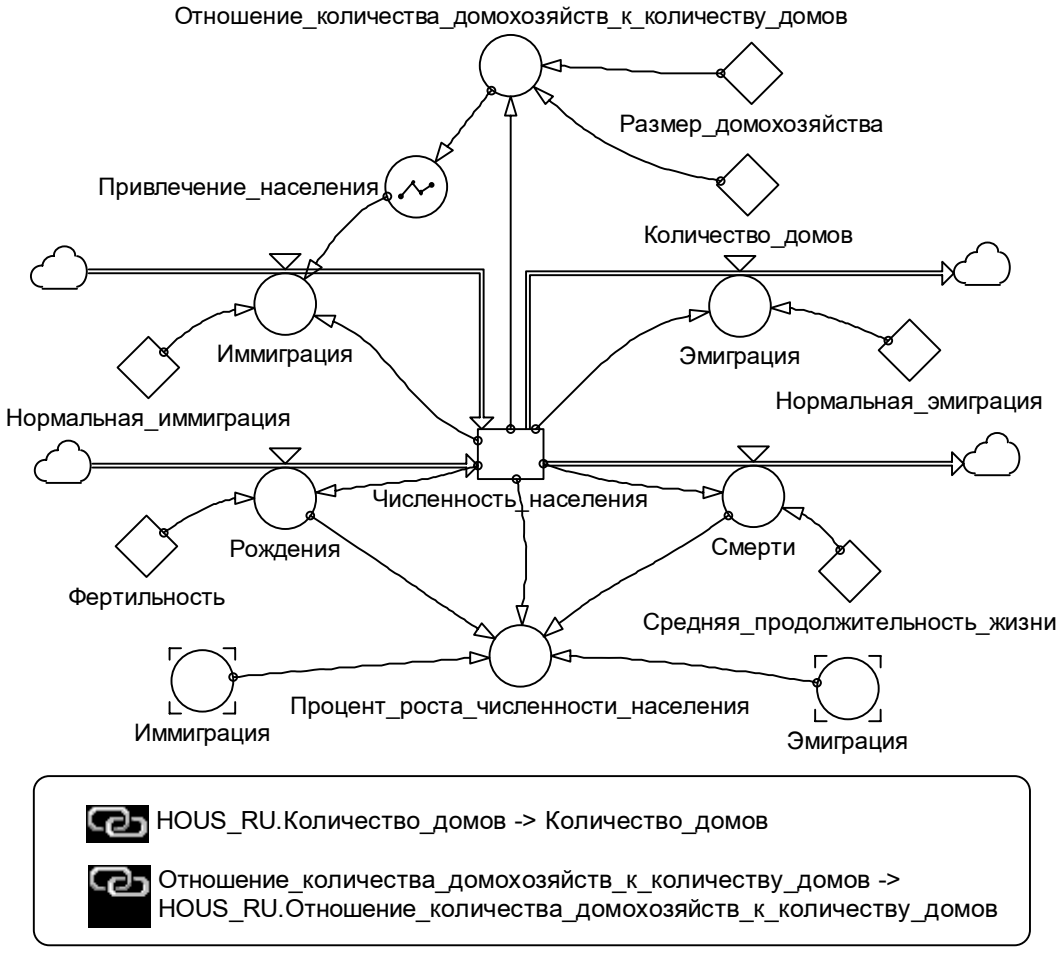
```
dim    Фонд_зарплаты = (1..2)
aux    Фонд_зарплаты = 30000*Рабочая_сила
doc    Фонд_зарплаты = Расходы, связанные с оплатой труда работников
       фирмы
unit   Фонд_зарплаты = руб.
```

Константы

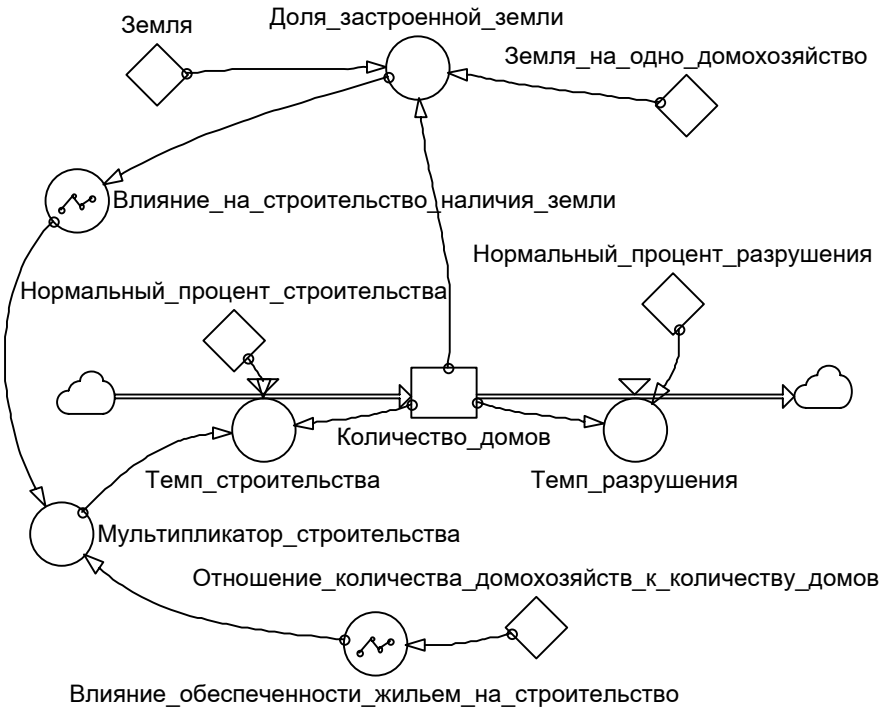
```
const  Время_достижения_нормальной_морали = 6
doc    Время_достижения_нормальной_морали = Время, необходимое для
       приведения морали фирмы в нормальное состояние
unit   Время_достижения_нормальной_морали = месяц
const  Нормальная_мораль = 100
doc    Нормальная_мораль = Нормальный уровень морали фирмы
unit   Нормальная_мораль = ед.морали
const  Процент_увеличения_прочих_расходов = 0.0135
doc    Процент_увеличения_прочих_расходов = Процент увеличение расходов,
       не связанных с оплатой труда работников фирмы
unit   Процент_увеличения_прочих_расходов = %/месяц

spec   начало = 0
spec   конец = 36
spec   dt = 1
spec   метод = Эйлер (фикс. шаг)
```

4. Модель динамики численности населения (POPULATN.SIM)

а) Потоковая диаграмма главной модели:



б) Потоковая диаграмма комодели:



а) Уравнения главной модели:

Уравнения уровней

```
init   Численность_населения = 50000
flow   Численность_населения = -dt*Смерти +dt*Рождения -dt*Эмиграция
       +dt*Иммиграция
doc    Численность_населения = Численность населения города
unit   Численность_населения = чел.
```

Уравнения темпов

```
aux    Иммиграция = Численность_населения*Привлечение_населения*
       Нормальная_иммиграция
doc    Иммиграция = Численность населения, приезжающего в город на ПМЖ за
       год
unit   Иммиграция = чел./год
aux    Рождения = Численность_населения*Фертильность
doc    Рождения = Число родившихся за год
unit   Рождения = чел./год
aux    Смерти = Численность_населения/Средняя_продолжительность_жизни
doc    Смерти = Число умерших за год
unit   Смерти = чел./год
aux    Эмиграция = Численность_населения*Нормальная_эмиграция
doc    Эмиграция = Численность населения, покидающиего город за год
unit   Эмиграция = чел./год
aux    Отношение_количества_домохозяйств_к_количеству_домов =
       Численность_населения/(Количество_домов*Размер_домохозяйства)
doc    Отношение_количества_домохозяйств_к_количеству_домов =
       Обеспеченность жильем населения. Это отношение меньше 1 при
       избытке жилья и больше 1 при дефиците жилья
unit   Отношение_количества_домохозяйств_к_количеству_домов =
       безразмерная
aux    Привлечение_населения = GRAPH(Отношение_количества_домохозяйств_к_
       количеству_домов,0,0.2,[1.4,1.4,1.35,1.3,1.15,1,0.8,0.65,0.5,0.45,
       0.4"Min:0;Max:1.4"])
```

```
doc   Привлечение_населения = Мультипликатор привлекательности домов для
      населения – нелинейная функция от Отношения_количества_домо-
      хозяйств_к_количеству_домов (при увеличении этого соотношения
      начинает сокращаться иммиграция, так как спрос на дома начинает
      превышать предложение домов)
unit  Привлечение_населения = безразмерная
```

Дополнительные уравнения

```
aux   Процент_роста_численности_населения = (Рождения-Смерти+Иммиграция-
      Эмиграция)/Численность_населения
doc   Процент_роста_численности_населения = Чистый темп увеличения
      численности городского населения как процент от его общей
      численности
unit  Процент_роста_численности_населения = 1/год
```

Константы

```
const Количество_домов = 0
unit  Количество_домов = шт.
const Нормальная_иммиграция = 0.1
doc   Нормальная_иммиграция = Нормальный темп иммиграци как процент от
      численности городского населения
unit  Нормальная_иммиграция = 1/год
const Нормальная_эмиграция = 0.07
doc   Нормальная_эмиграция = Нормальный темп эмиграци как процент от
      численности городского населения
unit  Нормальная_эмиграция = 1/год
const Размер_домохозяйства = 4
doc   Размер_домохозяйства = Средний размер домохозяйства (количество
      человек в расчете на 1 дом)
unit  Размер_домохозяйства = чел./шт.
const Средняя_продолжительность_жизни = 67
doc   Средняя_продолжительность_жизни = Средняя продолжительность жизни
      городского населения
unit  Средняя_продолжительность_жизни = год
const Фертильность = 0.03
doc   Фертильность = Нормальная годовая фертильность как процент от
      численности городского населения
unit  Фертильность = 1/год

spec  начало = 1900
spec  конец = 2000
spec  dt = 1
spec  метод = Эйлер (фикс. шаг)
```

б) Уравнения комодели:

Уравнения уровней

```
init  Количество_домов = 14000
flow  Количество_домов = +dt*Темп_строительства - dt*Темп_разрушения
doc   Количество_домов = Общее количество имеющихся домов
unit  Количество_домов = шт.
```

Уравнения темпов

```
aux   Темп_разрушения = Количество_домов*Нормальный_процент_разрушения
doc   Темп_разрушения = Темп ежегодного разрушения домов (в результате
      износа)
unit  Темп_разрушения = шт./год
```

```
aux   Темп_строительства = Количество_домов*Нормальный_процент_
      строительства*Мультипликатор_строительства
doc   Темп_строительства = Темп ежегодного строительсва новых домов
unit  Темп_строительства = шт./год
aux   Влияние_на_строительство_наличия_земли = GRAPH(Доля_застроенной_
      земли, 0,0.1,[1,0.92,0.83,0.75,0.66,0.57,0.47,0.35,0.24,0.11,0
      "Min:0;Max:1.6"])
doc   Влияние_на_строительство_наличия_земли = Нелинейная функция от
      доли застроенной земли (при увеличении этой доли происходит
      сокращение строительства новых домов)
unit  Влияние_на_строительство_наличия_земли = безразмерная
aux   Влияние_обеспеченности_жильем_на_строительство = GRAPH(Отношение_
      количества_домохозяйств_к_количеству_домов,0,0.2,[0.2,0.25,0.35,
      0.5,0.7,1,1.35,1.6,1.8,1.95,2"Min:0;Max:2"])
doc   Влияние_обеспеченности_жильем_на_строительство = Нелинейная
      функция от Отношения_количества_домохозяйств_к_количеству_домов
      (при увеличении этого отношения происходит увеличение
      строительства, так как спрос на дома начинает превышать
      предложение домов)
unit  Влияние_обеспеченности_жильем_на_строительство = безразмерная
aux   Доля_застроенной_земли =
      (Количество_домов*Земля_на_одно_домохозяйство)/Земля
doc   Доля_застроенной_земли = Процент застроенной территории
unit  Доля_застроенной_земли = безразмерная
aux   Мультипликатор_строительства = Влияние_обеспеченности_жильем_
      на_строительство*Влияние_на_строительство_наличия_земли
doc   Мультипликатор_строительства = Мультипликатор, отражающий влияние
      количества домов и земли на темп строительства новых домов
unit  Мультипликатор_строительства = безразмерная
```

Константы

```
const Земля = 8000
doc   Земля = Имеющаяся территория под потенциальную застройку
unit  Земля = акры
const Земля_на_одно_домохозяйство = 0.1
doc   Земля_на_одно_домохозяйство = Земельный участок одного
      домохозяйства
unit  Земля_на_одно_домохозяйство = га/шт.
const Нормальный_процент_разрушения = 0.015
doc   Нормальный_процент_разрушения = Среднегодовой процент разрушения
      старых домов от общего количества домов
unit  Нормальный_процент_разрушения = 1/год
const Нормальный_процент_строительства = 0.07
doc   Нормальный_процент_строительства = Среднегодовой процент
      строительства новых домов от общего количества домов
unit  Нормальный_процент_строительства = 1/год
const Отношение_количества_домохозяйств_к_количеству_домов = 0
unit  Отношение_количества_домохозяйств_к_количеству_домов = безразмер-
      ная

spec  начало = 1900
spec  конец = 2000
spec  dt = 1
spec  метод = Эйлер (фикс. шаг)
```

5. Модель жизненного цикла товара (PRODUCT.SIM)

Потоковая диаграмма:

Уравнения модели:

Уравнения уровней

```
init   Не_покупатели = Потенциальный_рынок
flow   Непокупатели = -dt*Темп_привыкания_к_товару
doc    Не_покупатели = Число людей, еще не купивших данный товар, но
       имеющих склонность стать покупателями товара
unit   Не_покупатели = чел.
init   Общие_маркетинговые_расходы = 0
flow   Общие_маркетинговые_расходы = +dt*Маркетинговые_расходы
unit   Общие_маркетинговые_расходы = руб.
init   Покупатели = 0
flow   Покупатели = +dt*Темп_привыкания_к_товару
doc    Покупатели = Число людей, привыкнувших к товару (принявших его) и
       купивших его
unit   Покупатели = чел.
init   Эффективность_маркетинга = 0
flow   Эффективность_маркетинга = +dt*Изменение_эффективности_маркетинга
doc    Эффективность_маркетинга = Реальный эффект от маркетинговых
       расходов, связанный с запаздыванием распространения и информации о
       товаре
unit   Эффективность_маркетинга = руб./месяц
```

Уравнения темпов

```
aux   Изменение_эффективности_маркетинга = (Маркетинговые_расходы-
      Эффективность_маркетинга)/Время_изменения_эффективности_маркетинга
doc   Изменение_эффективности_маркетинга = Изменение эффекта от
      маркетинговых расходов
unit  Изменение_эффективности_маркетинга = руб./месяц/месяц
aux   Маркетинговые_расходы = LOOKUP(Автоматическая_и_ручная_настройка_
      бюджета_маркетинга, Выбор_бюджета)
doc   Маркетинговые_расходы = Ежемесячные расходы, связанные с
      маркетингом
unit  Маркетинговые_расходы = руб./месяц
aux   Темп_привыкания_к_товару = MAX(0,(Не_покупатели*Влияние_на_
      рекламу) + Распространение_информации_о_товаре)
doc   Темп_привыкания_к_товару = Скорость привыкания к товару (принятия
      товара) и его покупки, когда не покупатели становятся покупателями
unit  Темп_привыкания_к_товару = чел./месяц
```

Вспомогательные уравнения

```
dim   Автоматическая_и_ручная_настройка_бюджета_маркетинга = (i=1..2)
aux   Автоматическая_и_ручная_настройка_бюджета_маркетинга = GRAPH(TIME,
      0,2,[800,3300,6800,10500,15300,20000,21900,22500,20800,17500,11500,
      0,0,0,0,0,0,0,0,0,0"Min:0;Max:30000"]) WHEN i=1 BUT Ручная_
      настойка_бюджета_маркетинга WHEN i=2
doc   Автоматическая_и_ручнная_настройка_бюджета_маркетинга = Первый
      элемент - автоматическое регулирование бюджета маркетинга, второй
      элемент - ручное регулирование бюджета маркетинга
unit  Автоматическая_и_ручная_настройка_бюджета_маркетинга = руб./месяц
aux   Влияние_на_рекламу = GRAPH(Эффективность_маркетинга,0,5000,[0,
      0.007,0.021,0.039,0.066,0.103,0.138,0.165,0.186,0.192,0.195"Min:0;
      Max:0.2"])
doc   Влияние_на_рекламу = Нелинейное влияние рекламы на темп привыкания
      к товару (темп принятия товара)
unit  Влияние_на_рекламу = 1/месяц
aux   Проникновение_на_рынок = Покупатели/Емкость_рынка
doc   Проникновение_на_рынок = Освоенная часть рынка
unit  Проникновение_на_рынок = безразмерная
aux   Распространение_информации_о_товаре = (1-Проникновение_на_рынок)*
      Покупатели*Работа_с_покупателями
doc   Распространение_информации_о_товаре = Ежемесячное число людей,
      привыкших к товару (принявших товар) благодаря устной рекламе
      (от друзей и знакомых)
unit  Распространение_информации_о_товаре = чел./месяц
```

Дополнительные уравнения

```
aux   Объем_продаж = Темп_привыкания_к_товару*Средние_удельные_продажи
doc   Объем_продаж = Количество проданного за месяц товара
unit  Объем_продаж = ед./месяц
```

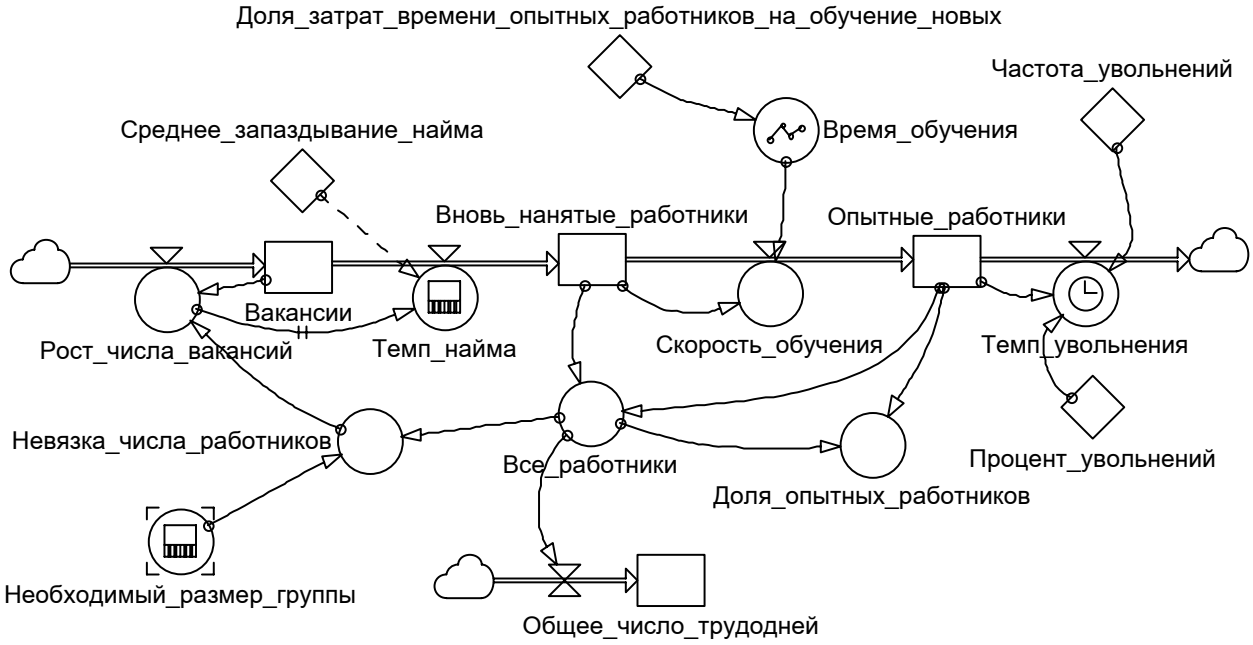
Константы

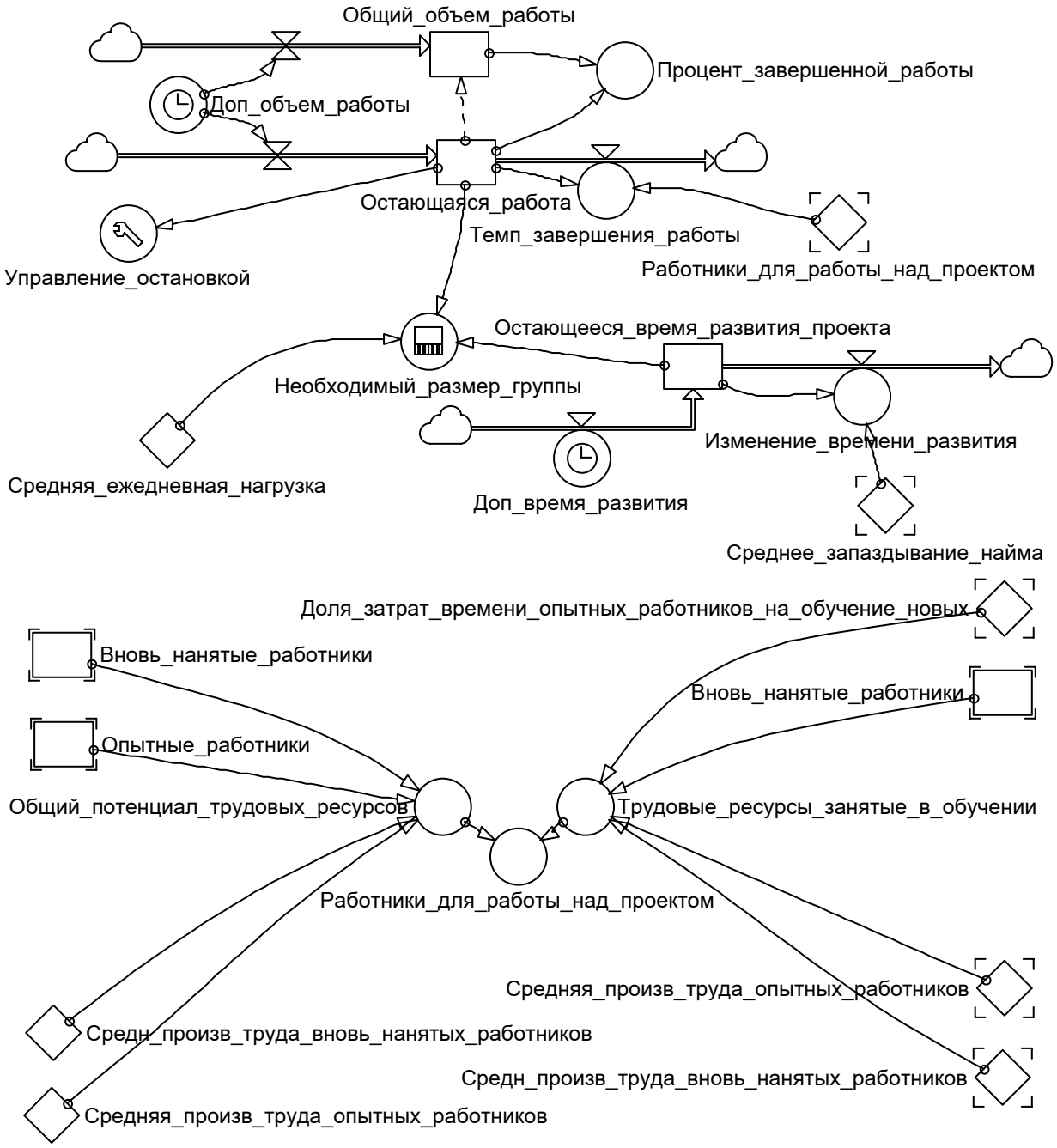
```
const Время_изменения_эффективности_маркетинга = 1
doc   Время_изменения_эффективности_маркетинга = Время, необходимое для
      диффузии маркетинговой информации
unit  Время_изменения_эффективности_маркетинга = месяц
const Выбор_бюджета = 1
doc   Выбор_бюджета = Параметр управления, позволяющий производить
      автоматический или ручной выбор маркетинговых расходов
unit  Выбор_бюджета = безразмерная
```

```
const  Емкость_рынка = 1000000
doc    Емкость_рынка = Потенциальный размер рынка для данного товара
unit   Потенциальный_рынок = чел.
const  Работа_с_покупателями = 0.167
doc    Работа_с_покупателями = Среднне число клиентов, интересовавшихся
       товаром за месяц, в расчете на одного продавца (0.167 интересовав-
       шихся товаром за месяц означает, что за год товаром интересовались
       2 человека)
unit   Работа_с_покупателями = чел./чел./месяц
const  Ручная_настойка_бюджета_маркетинга = 3000
doc    Ручная_настойка_бюджета_маркетинга = Настройка расходов
       маркетинга, заданная по умолчанию
unit   Ручная_настойка_бюджета_маркетинга = руб./месяц
const  Средние_удельные_продажи = 1.2
doc    Средние_удельные_продажи = Среднее количество купленного товара в
       расчете на одного покупателя
unit   Средние_удельные_продажи = ед./чел.

spec   начало = 0
spec   конец = 35
spec   dt = 0.25
spec   метод = РК4 (фикс. шаг)
```

6. Модель управления проектом (PROJMGMT.SIM)

Потоковая диаграмма:





Уравнения модели:

Уравнения уровней

```
init   Вакансии = 0
flow   Вакансии = +dt*Рост_числа_вакансий -dt*Темп_найма
doc    Вакансии = Число свободных рабочих мест в компании
unit   Вакансии = чел.
init   Вновь_нанятые_работники = 5
flow   Вновь_нанятые_работники = -dt*Скорость_обучения +dt*Темп_найма
doc    Вновь_нанятые_работники = Обычно это работники без соответсвующего
       опыта работы
unit   Вновь_нанятые_работники = чел.
init   Общее_число_трудодней = 0
flow   Общее_число_трудодней = +dt*Все_работники
doc    Общее_число_трудодней = Общее число человекодней, потраченное на
       работу по проекту в любой точке жизненного цикла проекта
unit   Общее_число_трудодней = чел.*день
```

```
init   Общий_объем_работы = Остающаяся_работа
flow   Общий_объем_работы = +dt*Доп_объем_работы
doc    Общий_объем_работы = Объем работы, который необходимо выполнить
       для завершения проета
unit   Общий_объем_работы = трудодень
init   Опытные_работники = 5
flow   Опытные_работники = -dt*Темп_увольнений +dt*Скорость_обучения
doc    Опытные_работники = Количство работников, способных к саморазвитию
       (самосовершенствованию)
unit   Опытные_работники = чел.
init   Остающаяся_работа = 1000
flow   Остающаяся_работа = -dt*Темп_завершения_работы+dt*Доп_объем_работы
doc    Остающаяся_работа = Количество трудодней, остающихся по проекту
unit   Остающаяся_работа = трудодень
init   Остающееся_время_развития_проекта = 100
flow   Остающееся_время_развития_проекта = +dt*Доп_время_развития
       -dt*Изменение_времени_развития
doc    Остающееся_время_развития_проекта = Отрезок времени, остающийся
       для развития проекта (этот параметр нужен для расчета необходи-
       мого размера рабочей группы, способной завершить в срок всю работу
       по проекту). Таким образом, значение этого параметра никогда не
       может быть меньше запаздывания найма новых работников, так как
       бессмысленно заполнять вакансии работниками, которых нельзя затем
       использовать в проекте
unit   Остающееся_время_развития_проекта = день
```

Уравнения темпов

```
aux    Все_работники = Опытные_работники+Вновь_нанятые_работники
doc    Все_работники = Общее число работников компании
unit   Все_работники = чел.
aux    Доп_время_развития = IF(TIME=50,80,0)
doc    Доп_время_развития = Темп добавления времени к остающемуся времени
       развития проекта. В данном случае ко времени развития проекта
       добавляются 80 дней, если общий объем работы увеличивается на 1000
       трудодней в момент времени 50.
unit   Доп_время_развития = день/день
aux    Доп_объем_работы = IF(TIME=50,1000,0)
doc    Доп_объем_работы = Темп добавления объема работы по проекту. В
       данном случае, исходя из консеравтивной начальной оценки, большой
       объем работы добавляется на 50-й день
unit   Доп_объем_работы = трудодень/день
aux    Изменение_времени_развития = IF(Остающееся_время_развития_проекта>
       Среднее_запаздывание_найма,1,0)
doc    Изменение_времени_развития = Темп, с которым отсчитывается
       (уменьшается) время развития проекта
unit   Изменение_времени_развития = день/день
aux    Рост_числа_вакансий = MAX(0,Невязка_числа_работников-Вакансии)
doc    Рост_числа_вакансий = Темп регистрирования новых вакансий
unit   Рост_числа_вакансий = чел./день
aux    Скорость_обучения = Вновь_нанятые_работники/Время_обучения
doc    Скорость_обучения = Скорость обучения новых работников
unit   Скорость_обучения = чел./день
aux    Темп_завершения_работы = MIN(Работники_для_работы_над_проектом,
       Остающаяся_работа)
doc    Темп_завершения_работы = Скорость завершения работ по проекту
unit   Темп_завершения_работы = трудодень/день
aux    Темп_найма = DELAYPPL(Рост_числа_вакансий,Среднее_запаздывание_
       найма,0)
```

```
doc    Темп_найма = Скорость найма компанией новых сотрудников
unit   Темп_найма = чел./день
aux    Темп_увольнений = IF(TIME MOD (1/Частота_увольнений) = 0,
       ROUND(Опытные_работники*Процент_увольнений),0)
doc    Темп_увольнений = Скорость увольнения работников. В начальный
       момент времени никто не увольняется
unit   Темп_увольнений = чел./день
```

Вспомогательные уравнения

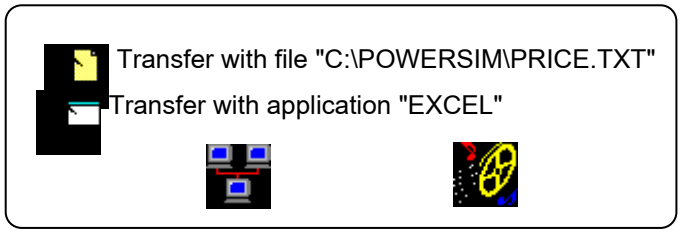
```
aux    Время_обучения = GRAPH(Доля_затрат_времни_опытных_работников_на_
       обучение_новых,0,0.1,[200,108,65,40,36,31,30,28,27,26,25"Min:0;
       Max:200"])
doc    Время_обучения = Среднее время приобретения необходимого опыта
       новыми работниками
unit   Время_обучения = день
aux    Доля_опытных_работников = Опытные_работники/Все_работники
doc    Доля_опытных_работников = Доля общей рабочей силы, способной к
       саморазвитию (самосовершенствованию)
unit   Доля_опытных_работников = безразмерная
aux    Невязка_числа_работников = Необходимый_размер_группы-Все_работники
doc    Невязка_числа_работников = Диспропорция между имеющимся и
       необходимым для работы по проекту количеством работников
unit   Невязка_числа_работников = чел.
aux    Необходимый_размер_группы = HIVAL(ROUND((Остающаяся_работа/
       Остающееся_время_развития_проекта)/Средняя_ежедневная_нагрузка))
doc    Необходимый_размер_группы = Общий размер группы, необходимый для
       завершения работы по проекту
unit   Необходимый_размер_группы = чел.
aux    Общий_потенциал_трудовых_ресурсов = Средн_произв_труда_вновь_
       нанятых_работников*Опытные_работники+Средняя_произв_труда_опытных_
       работников*Вновь_нанятые_работники
doc    Общий_потенциал_трудовых_ресурсов = Общие производительные
       возможности трудовых ресурсов компании
unit   Общий_потенциал_трудовых_ресурсов = трудодень/день
aux    Процент_завершенной_работы = PCT(1-(Остающаяся_работа/
       Общий_объем_работы))
doc    Процент_завершенной_работы = Доля завершенной работы по проекту
unit   Процент_завершенной_работы = безразмерная
aux    Работники_для_работы_над_проектом = Общий_потенциал_трудовых_
       ресурсов-Трудовые_ресурсы_занятые_в_обучении
doc    Работники_для_работы_над_проектом = Доля трудовых ресурсов,
       которую можно использовать для выполнения реальной работы по
       проекту
unit   Работники_для_работы_над_проектом = трудодень/день
aux    Трудовые_ресурсы_занятые_в_обучении = (Средн_произв_труда_вновь_
       нанятых_работников+Средняя_произв_труда_опытных_работников)*Вновь_
       нанятые_работники*Доля_затрат_времени_опытных_работников_на_обуче-
       ние_новых
doc    Трудовые_ресурсы_занятые_в_обучении = Ежедневные затраты
       внутрифирменных трудовых ресурсов для обучения новых работников
unit   Трудовые_ресурсы_занятые_в_обучении = трудодень/день
aux    Управление_остановкой = STOPIF(Остающаяся_работа=0)
doc    Управление_остановкой = Остановка имитации в случае, когда
       остающаяся работа по проекту равна 0, т.е. когда проект завершен
unit   Управление_остановкой = безразмерная
```

Константы

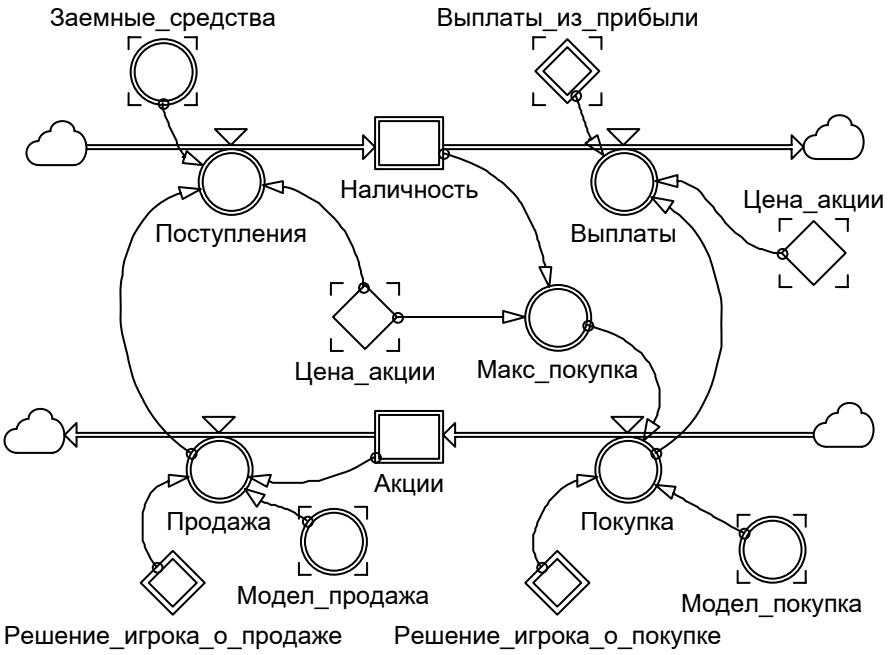
```
const  Доля_затрат_времени_опытных_работников_на_обучение_новых = 0.20
doc    Доля_затрат_времени_опытных_работников_на_обучение_новых = Доля
       времени, потраченного персоналом компании на обучение новых
       работников
unit   Доля_затрат_времени_опытных_работников_на_обучение_новых =
       безразмерная
const  Процент_увольнений = 10%
doc    Процент_увольнений = Средний процент увольнения
unit   Процент_увольнений = безразмерная
const  Средн_произв_труда_вновь_нанятых_работников = 1
doc    Средн_произв_труда_вновь_нанятых_работников = Средняя
       потенциальная производительность труда опытных (обученных)
       работников
unit   Средн_произв_труда_вновь_нанятых_работников = трудодень/день/чел.
const  Среднее_запаздывание_найма = 40
doc    Среднее_запаздывание_найма = Среднее время (в днях), необходимое
       для рекламы вакансий, интервьюирования, отбора и найма новых
       работников
unit   Среднее_запаздывание_найма = день
const  Средняя_ежедневная_нагрузка = 1
doc    Средняя_ежедневная_нагрузка = Средняя ежедневная трудовая нагрузка
       работников (1 при полной загруженности)
unit   Средняя_ежедневная_нагрузка = трудодень
const  Средняя_произв_труда_опытных_работников = 0.33
doc    Средняя_произв_труда_опытных_работников = Средняя потенциальная
       производительность труда новых (не опытных) работников
unit   Средняя_произв_труда_опытных_работников = трудодень/день/чел.
const  Частота_увольнений = 1/100
doc    Частота_увольнений = Средняя частота увольнений
unit   Частота_увольнений = 1/день

spec   начало = 1
spec   конец = 300
spec   dt = 1
spec   метод = Эйлер (фикс. шаг)
```

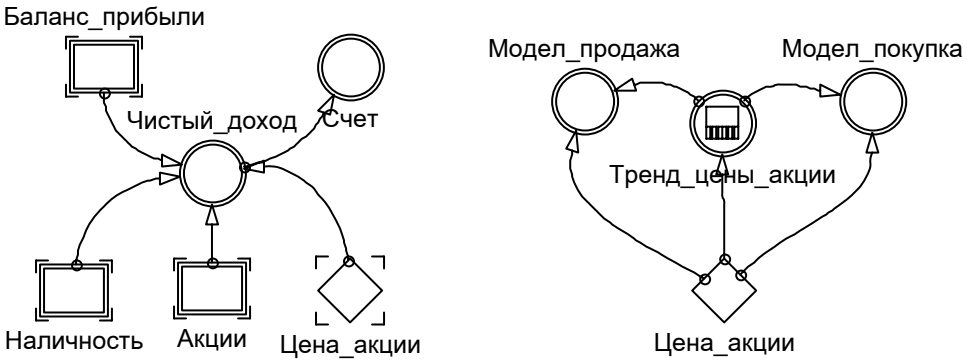
7. Модель торговли акциями на бирже (STOCKMKT.SIM)

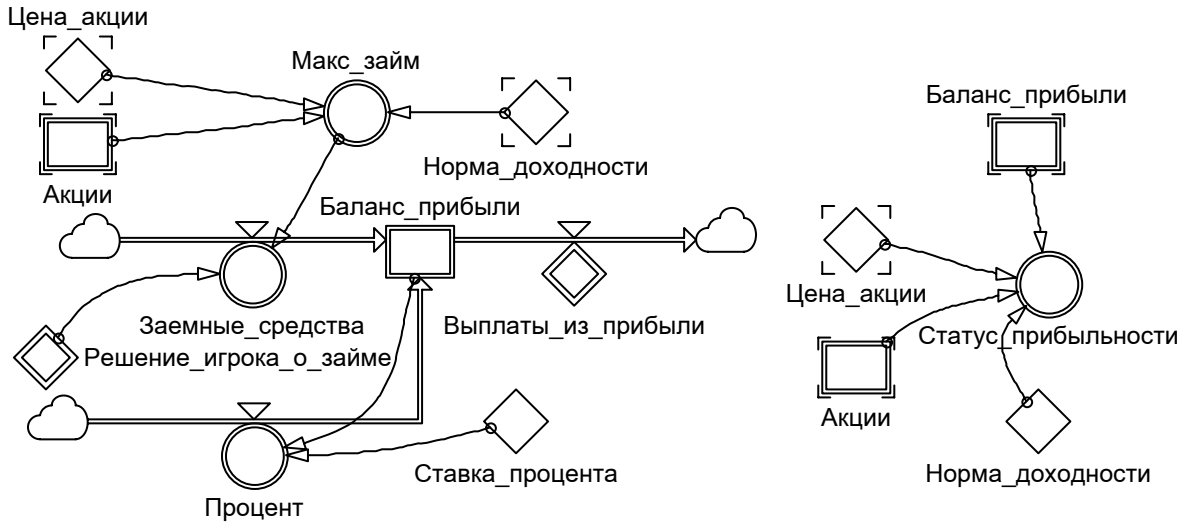
Потоковая диаграмма*:



* Модель считывает текущие значения цен акций из файла C:\Powersim\Price.txt и передает текущие значения чистого дохода в файл Shares.xls приложения MS Excel. Кроме того, если Статус_прибыльности становится равным 1, то средство мультимедиа вызывает диалоговое окно MessageBox с предупреждением о прибыльности. Возможно два варианта проведения игрового моделирвания: 1) игра "человек-машина", 2) игра "человек-человек". Для этого заданы специальным образом свойства объекта "Сетевая игра" (см. п. 1.6.1).







Уравнения модели:

Уравнения уровней

```
dim    Акции = (1..2)
init   Акции = 0
flow   Акции = +dt*Покупка-dt*Продажа
doc    Акции = Общее количество акций на бирже
unit   Акции = шт.
dim    Баланс_прибыли = (1..2)
init   Баланс_прибыли = 0
```

```
flow   Баланс_прибыли = +dt*Процент-dt*Выплаты_их_прибыли+dt*Заемные_
       средства
doc    Баланс_прибыли = Прибыль игроков
unit   Баланс_прибыли = руб.
dim    Наличность = (1..2)
init   Наличность = 10000
flow   Наличность = -dt*Выплаты +dt*поступления
doc    Наличность = Располагаемые денежные средства у игроков
unit   Наличность = руб.
```

Уравнения темпов

```
dim    Выплаты = (1..2)
aux    Выплаты = Цена_акции*Покупка + Выплаты_их_прибыли
doc    Выплаты = Выплаты, связанные с покупкой акций и возвращением
       займов
unit   Выплаты = руб./день
dim    Заемные_средства = (1..2)
aux    Заемные_средства = MIN(Решение_игрока_о_займе, Макс_займ)
doc    Заемные_средства = Заемные средства, ограниченные максимальной
       суммой займа
unit   Заемные_средства = руб./день
dim    Покупка = (i=1..2)
aux    MIN(Макс_покупка, SELECTDECISION(INDEX(i), ROUND(Решение_игрока_
       о_покупке), 0, Модел_покупка, 0))
doc    Покупка = Покупка акций игроками или машиной (моделируемая
       покупка)
unit   Покупка = шт./день
dim    Поступления = (1..2)
aux    Поступления = Цена_акции*Продажа + Заемные_средства
doc    Поступления = Поступления от продажи акций и от займов
unit   Поступления = руб./день
dim    Продажа = (i=1..2)
aux    Продажа = MIN(Акции, SELECTDECISION(INDEX(i), Решение_игрока_
       о_продаже, 0, Модел_продажа, 0))
doc    Продажа = Продажа акций игроками или машиной (моделируемая
       покупка)
unit   Продажа = шт./день
dim    Процент = (1..2)
aux    Процент = Баланс_прибыли*Ставка_процента
doc    Процент = Накопленные проценты на прибыль игроков
unit   Процент = руб./день
```

Вспомогательные уравнения

```
dim    Макс_займ = (1..2)
aux    Макс_займ = (Акции*Цена_акции)*((1/Норма_доходности)-1)
doc    Макс_займ = Размер максимально доступного займа, зависящий прямо
       пропорционально от количества и цены акций и обратно
       пропорционально – от нормы доходности акций
unit   Макс_займ = руб.
dim    Макс_покупка = (i=1..2)
aux    Макс_покупка = FLOOR(Наличность/Цена_акции)
doc    Макс_покупка = Максимальное количество акций, которое может купить
       текущий игрок
unit   Макс_покупка = шт.
dim    Модел_покупка = (1..2)
aux    Модел_покупка = IF((Тренд_цены_акции > 0) AND (Цена_акции <23) OR
       (Тренд_цены_акции > 0.02), 400, 0)
doc    Модел_покупка = Количество покупаемых акций, моделируемое
       компьютером (компьютер покупает 400 акции, если цена на них не
```

```
      падает и составляет менее 23 руб./шт., или если она растет более
      чем на 2% в день)
unit  Модел_покупка = шт./день
dim   Модел_продажа = (1..2)
aux   Модел_продажа = IF((Тренд_цены_акции < -0.01) AND (Цена_акции >
      30) OR (Цена_акции > 40), 400, 0)
doc   Модел_продажа = Количество продаваемых акций, моделируемое
      компьютером (компьютер продает 400 акции, если цена на них падает
      менее чем на 1% в день и составляет более 23 руб./шт., или если
      цена превышает 40 руб./шт.)
unit  Модел_продажа = шт./день
dim   Статус_прибыльности = (1..2)
aux   Статус_прибыльности = IF(Баланс_прибыли > (Акции*Цена_акции*
      (1-Норма_доходности)),1,0)
doc   Статус_прибыльности = Если статус прибыльности становится равным
      1, то средство мультимедиа вызывает диалоговое окно MessageBox с
      предупреждением о прибыльности
unit  Статус_прибыльности = безразмерная
dim   Счет = (i=1..2)
aux   Счет = Чистый_доход(1)/Чистый_доход(2) WHEN i=1 BUT
      Чистый_доход(2)/Чистый_доход(1) WHEN i=2
doc   Счет = Счет определяется как сумма активов текущего игрока,
      деленная на сумму активов его конкурентов
unit  Счет = пункты
dim   Тренд_цены_акции = (1..2)
aux   Тренд_цены_акции = TREND(Цена_акции, 8, Цена_акции)
doc   Тренд_цены_акции = Тренд цены акции, усредненный за 8 дней
unit  Тренд_цены_акции = руб./шт./день
dim   Чистый_доход = (1..2)
aux   Чистый_доход = Наличность + (Акции * Цена_акции) - Баланс_прибыли
doc   Чистый_доход = Чистый доход игроков
unit  Чистый_доход = руб.
```

Константы

```
dim   Выплаты_их_прибыли = (1..2)
const Выплаты_их_прибыли = 0
doc   Выплаты_их_прибыли = Начальные выплаты из прибыли игроков
unit  Выплаты_их_прибыли = руб./день
const Норма_доходности = 0.5
doc   Норма_доходности = Норма доходности по операциям с ценными
      бумагами (50%)
unit  Норма_доходности = безразмерная
dim   Решение_игрока_о_покупке = (1..2)
const Решение_игрока_о_покупке = 0
doc   Решение_игрока_о_покупке = Начальное количество акций, покупаемое
      игроками на текущем шаге моделирования (далее изменяется игроками)
unit  Решение_игрока_о_покупке = шт./день
dim   Решение_игрока_о_продаже = (1..2)
const Решение_игрока_о_продаже = 0
doc   Решение_игрока_о_продаже = Начальное количество акций, продаваемое
      игроками на текущем шаге моделирования (далее изменяется игроками)
unit  Решение_игрока_о_продаже = шт./день
dim   Решение_игрока_о_займе = (1..2)
const Решение_игрока_о_займе = 0
doc   Решение_игрока_о_займе = Начальное количество заемных средств у
      игроков (далее изменяется игроками)
unit  Решение_игрока_о_займе = руб./день
```

```
const  Ставка_процента = 0.04/250
doc    Ставка_процента = Ежедневная ставка процента по счетам игроков
       (4% за 250 дней)
unit   Ставка_процента = безразмерная
const  Цена_акции = 0
doc    Цена акции = Начальная цена акции (далее загружается из файла
       C:\Powersim\Price.txt)
unit   Цена_акции = руб./шт.

spec   начало = 0
spec   конец = 250
spec   dt = 1
spec   метод = Эйлер (фикс. шаг)
```

Содержание файла настроек игры (Stock.psn)

```
[GameFiles]
File1=0,STOCK1.PSS
File2=0,STOCK2.PSS
```

Содержание файла настроек игрока для варианта игры "человек-машина" (Stock1.pss)

```
[Game]
Name=Игра с компьютером ("человек-машина")
Strategic=0
Symmetric=1
Players=1
SimPlayersBits=0
DecisionFiles=stock%d.txt
DecisionPeriod=1
DecisionVariables0=Решение_игрока_покупать(%d),Решение_игрока_о_займе(%d),
Выплаты_из_прибыли(%d),Решение_игрока_продавать(%d)
```

Содержание файла настроек игрока для варианта игры "человек-человек" (Stock2.pss)

```
[Game]
Name=Игра двух игроков ("человек-человек")
Strategic=0
Symmetric=1
Players=2
SimPlayersBits=0
DecisionFiles=stock%d.txt
DecisionPeriod=1
DecisionVariables0=Решение_игрока_покупать(%d),Решение_игрока_о_займе(%d),
Выплаты_из_прибыли(%d),Решение_игрока_продавать(%d)
```

Литература

1. *Гультяев А.* Визуальное моделирование в среде MATLAB: учеб. курс. – СПб.: Питер, 2000.
2. *Дудорин В.И., Алексеев Ю.Н.* Системный анализ экономики на ЭВМ. – М.: Финансы и статистика, 1986.
3. *Емельянов А.А., Власова Е.А.* Имитационное моделирование в экономических информационных системах. – М.: МЭСИ, 1996.
4. *Емельянов В.В., Ясиновский С.И.* Введение в интеллектуальное имитационное моделирование сложных дискретных систем и процессов. Язык РДО. – М.: Изд-во "АНВИК", 1998.
5. *Евсюхина К., Чесалова М.* Работа с пакетом динамического моделирования POWERSIM / Под. ред. А. Масаловича. – М., 1997.
6. *Калянов Г.Н.* CASE-технологии: консалтинг в автоматизации бизнес-процессов. – М.: Горячая линия – Телеком, 2000.
7. *Клейнен Дж.* Статистические методы в имитационном моделировании. – М.: Статистика, 1978.
8. *Нейлор Т.М.* Машинные имитационные эксперименты с моделями экономических систем. – М.: Мир, 1975.
9. *Сидоренко В.Н.* Системная динамика. – М.: ТЕИС, 1998.
10. *Сидоренко В.Н., Матросов И.В., Латышев К.В.* Язык разработки системно-динамических моделей RUSIM / Сб. тезисов международной конф. "Ломоносов 2000". – М.: МАКС Пресс, 2000.
11. *Советов Б.Я., Яковлев С.А.* Моделирование систем. – М.: Высш. Шк., 1998.
12. *Урезченко В.М.* Построение имитационных моделей с использованием принипов системной динамики. – М.: МИФИ, 1989.
13. *Уткин Э.А.* Консалтинг. – М.: Ассоциация авторов и издателей "ТАНДЕМ", Издательство ЭКМОС, 1988.
14. *Форрестер Дж.* Мировая динамика. – М.: Наука, 1978.
15. *Шеннон Р.* Имитационное моделирование – искусство и наука. – М.: Мир, 1978.
16. *Шрайбер Т.Дж.* Моделирование на GPSS. – М.: Машиностроение, 1980.
17. *Hansen G.A.* Tools for Business Process Reengineering / IEEE Software. 1994.
18. *Richardson G.P., Pugh A.L.* Introduction to System Dynamics Modeling with DYNAMO. – Cambridge, MA: MIT Press, 1981.
19. *Forrester Jay W.* System Dynamics in Management Education. – Sloan School of Managament, MIT, 1989.
20. Ithink User's Guide. – High Performance System, 1996.
21. Powersim User's Guide. – Powersim Co., 1996.
22. *Sterman J.D.* Business Dynamics: System Thinking and Modeling for a Complex World. – Irwin/McGraw-Hill: New York, 2000.
23. Vensim: Personal Learning Edition. User's Guide. – Ventana Systems Inc., 1996.

Алфавитный указатель функций Powersim

Сидоренко Владимир Николаевич Системно-динамическое моделирование в среде POWERSIM: Справочник по интерфейсу и функциям. – М.: МАКС-ПРЕСС, 2001. – 159 с.

Издательство ООО "МАКС ПРЕСС"
Лицензия ИД №00510 от 01.12.99 г.
Подписано к печати 20.06.2000 г.
Усл. Печ.л. 10. Тираж 100 экз. Заказ .
Тел. 939-3890, 939-38-91, 928-1042. Тел./Факс 939-3891.
119899, Москва, Воробьевы горы, МГУ.