Введение

Создание сложных и дорогих устройств и систем часто бывает сопряжено не только с материальным риском, но и с опасностью для самой человеческой жизни. Моделирование, т. е. возможность увидеть протекающие процессы в моделируемом объекте без особых затрат и рисков, в наше время стало основой проектирования.

Фактически моделирование бывает двух типов: *физическое и математическое*. При *физическом* моделировании моделируемый объект воплощается в подходящем материале обычно уменьшенного размера при соответствующих условиях (продувка моделей самолетов в аэродинамической трубе, испытание уменьшенных копий гидростанций и т. д.). При *математическом* моделировании основными являются уравнения, описывающие протекающие процессы. Поэтому если для моделирования колебаний металлических ферм моста используется электрическая цепь с протекающими в ней физическими процессами, то моделирование все равно будет математическим, т. к. основано не на идентичности объектов, а на совпадении уравнений, описывающих совершенно разные физические процессы.

Еще Галилей говорил о том, что «природа говорит языком математики». Долгое время человечество математически описывает Природу, строит математические модели. Это фундамент науки. Однако исследование с помощью математики – математическое моделирование – ограничивалось сравнительно простыми системами, и то лишь при их упрощении и линеаризации. С развитием вычислительной техники (причиной развития которой и была все возрастающая сложность задач) стало возможным математически исследовать объекты любой сложности. Появился даже термин *вычислительный эксперимент*. Компьютеры столь существенно изменили математическое моделирование, что следует говорить уже о новом направлении в инженерном деле. Если до середины XX века инженер ассоциировался с логарифмической линейкой, затем с калькулятором и перфокартами, причем сам писал и отлаживал программы, то моделирование в настоящее время напоминает испытание сконструированного устройства и измерение его характеристик.

Существуют разные уровни математического моделирования. Формально математическое описание любого устройства или системы можно усложнять до бесконечности, но нужно ли это делать? Ведь усложнение задачи ограничено возможностями программы, но еще раньше временем вычислений, хотя возможности компьютеров растут год от года.

Особенность моделирования электропривода связана с наличием одновременно функционирующих устройств с разными постоянными времени. Переходные процессы в электромеханической части системы могут протекать в течение единиц и даже десятков секунд, а переходные процессы при переключении силовых транзисторов длятся микросекунды. Разница в длительности процессов доходит до девяти порядков. Пока нет таких программ, которые позволили бы исследовать систему, разделяя в пространстве и во времени такие переходные процессы, – они оказывают существенное влияние на характеристики системы и должны быть учтены. Но эта задача с достаточной точностью решается в MatLab, Simulink.

Отметим также, что большинство программ моделирования занимаются в основном анализом, хоть и не в явной форме, записанных уравнений. Чистые задачи синтеза довольно специфичны, и число их невелико (например, синтез электрических фильтров, логических устройств, корректирующих звеньев, структуры электрических распределительных устройств и некоторые другие задачи).

О каких программах пойдет речь: в первую очередь о программе моделирования систем автоматического регулирования Simulink и ее расширениях, приложения программы MatLab, и программе для моделирования сложных электромеханических устройств ANSYS Maxwell.

Книги, хоть как-то связанные с компьютерной техникой и программным обеспечением, устаревают раньше, чем их возьмет в руки читатель. В данном случае, как показал опыт, удалось, в основном, избежать этого. Первое издание вышло ограниченным тиражом десять лет назад и за это время почти не устарело, т. к. был сделан упор на математические модели, а не их программную реализацию. Неожиданная особенность предмета «моделирование электропривода» оказалась в том, что из вспомогательного курса компьютерных методов расчета и проектирования он превратился в очень эффективный курс обучения современному электроприводу.

Второе издание увеличилось по объему в два раза: добавлены новые главы, посвященные датчикам, регуляторам, фильтрам, в том числе фильтрам Калмана, бездатчиковым системам регулирования, фазовым системам регулирования скорости, прямому управлению моментом, векторной ШИМ, корректорам коэффициента мощности; расширены главы о двигателях.

Правила компьютерного моделирования (серьезные и не совсем)

1. Всякая программа, даже самая хорошая и надежная, содержит ошибки.
2. Вероятность появления ошибки тем выше, чем сложнее или необычнее модель.
3. Компьютерное моделирование всегда имеет смысл — если ничего нового не удалось получить, то удастся понять, почему это произошло.
4. В подавляющем большинстве случаев компьютерная модель дешевле материальной, а компьютерный эксперимент безопаснее натурных испытаний.
5. Никогда не начинай моделирование со сложной модели — ибо если не работает простая модель, то что говорить о сложной.
6. Не делай сложно то, что можно сделать просто; Григорий Сковорода сказал: " Слава тебе, Господи, что ты все нужное сделал простым, а все сложное — ненужным!".
7. Если модель заработала с первого раза — не верь.
8. Если модель не работает — не пытайся обвинить программу, не работает именно модель.
9. В сложной модели главное не программа и не уравнения, описывающие модель. Главное — параметры модели. Они-то, как правило, и не соответствуют действительности.
10. Результаты моделирования никогда не бывают такими, какими их ожидаешь увидеть. Если они все-таки такие, то, значит, что-то не так.
11. Если модель совсем не верна, то или вы загрузили не тот файл, или пора сделать перерыв.

1. Программное обеспечение моделирования

Математическое моделирование можно проводить с помощью многих программ (MathCad, Ansoft Maxwell, MatLab и др.), но для систем автоматического управления и электропривода вне конкуренции остается приложение MatLab Simulink. Первая версия пакета MatLab была разработана в начале 80-х годов XX века. Развитие и совершенствование этой программы происходило одновременно с развитием средств вычислительной техники. Система MatLab (сокращение от MATrix LABoratory – матричная лаборатория) разработана фирмой The MathWorks, Inc. (США, г. Нейтик, шт. Массачусетс). Ориентирована в первую очередь на обработку массивов данных (матриц и векторов). В настоящее время MatLab представляет собой большую библиотеку функций и приложений. Часть функций носят общий характер и используются во многих приложениях. Они составляют ядро MatLab. Другие используются только в конкретном приложении.

Для моделирования электропривода более всего подходят приложение Simulink и его расширение SimPowerSystem. Simulink – дословно «моделирование связей» – моделирование, в первую очередь, систем автоматического управления путем сборки структурных схем с передаточными функциями в виде отдельных блоков. Но на самом деле можно вводить еще множество других блоков и создавать новые, что позволяет моделировать практически все.

Расширение приложения Simulink SimPowerSystem (*моделирование энергетических систем*) специально создавалось для моделирования электрических цепей, силовой электроники и электропривода.

Simulink – пакет программ для моделирования и анализа динамических систем. Поддерживает линейные и нелинейные системы, смоделированные в непрерывном времени, дискретном времени или одновременно и то и другое. Системы могут иметь сложную структуру.

Моделирование в Simulink обеспечивает *графический интерфейс пользователя* (GUI) для того, чтобы строить модели как блок-схемы, используя мышь. С этим интерфейсом можно рисовать модели так же, как карандашом на бумаге (как в большинстве учебников). Simulink сильно отличается от других программ моделирования, которые требуют записи уравнений на соответствующем языке. Simulink включает всестороннюю библиотеку блоков приемников, источников, линейных и нелинейных компонентов и соединителей. Можно также создать и настроить собственные блоки.

Помимо моделирования MatLab Simulink позволяет использовать компьютер как управляющую систему, заменив отдельные математические блоки реальными устройствами, подключив их через соответствующий интерфейс. Например, вместо моделей инвертора и электрического двигателя можно подключить реальный инвертор и двигатель и управлять ими. Приложение, которое позволяет это делать, называется *мастерской реального времени* Real-Time Workshop. К этому приложению есть еще масса приложений, компиляторов и всевозможных ограничений, плюс дискретное описание во времени. Но самое сложное – аппаратура сопряжения: обычно ее стоимость во много раз превышает стоимость компьютера вместе с программами. Зато и возможности открываются неограниченные.

Еще одно направление, которое развивается многими фирмами и которое есть в MatLab: автоматическое написание программ для микроконтроллеров. Можно для выделенной части Simulink-схемы или субсистемы создать программу на языке СИ или даже выполняемый код для конкретного микроконтроллера, правда, с некоторыми доработками, связанными, например, с расположением реальных подключений микросхемы.

Создание модели в Simulink из встроенных блоков – достаточно простая операция для грамотного в компьютерном отношении человека. Начала работы в Simulink описаны во многих книгах и в Help самого MatLab. Далее приводятся только особенности и тонкости программ, обычно не рассматриваемые в литературе. В прилагаемом списке можно найти

литературу с подробным описанием библиотек Simulink, SimPowerSystem и других приложений.

Для практического применения методов и программ необходима самостоятельная работа за компьютером. На это нацелены лабораторные работы по моделированию электропривода, содержащие всевозможные задания и схемы для моделирования. Третьей частью освоения курса моделирования следует считать эксперимент, когда данные по моделированию сравниваются с данными практической реализации моделей.

1.1. Решатели Simulink

При графическом вводе информации система регулирования собирается из отдельных блоков на экране и соединяется линиями связи, по которым «передается» сигнал. Если это электрическая схема, то элементы схемы соединяются «проводниками», по которым «течет» электрический ток. В действительности такой *графический интерфейс пользователя* (GUI) просто облегчает ввод данных. В конечном итоге получается система обыкновенных дифференциальных уравнений или дифференциально-алгебраических уравнений.

Все существующие системы регулирования делятся на *непрерывные, дискретные и смешанные*. В непрерывных (аналоговых) системах состояние системы изменяется непрерывно, в дискретных (цифровых) – скачком, в смешанных (гибридных) присутствует и то и другое. Типичный пример дискретности – цифровые схемы. Для дискретных систем дифференциальные уравнения заменяются дифференциально-разностными.

Simulink обеспечивает набор программ, известных как *решатели*, для интегрирования обыкновенных дифференциальных или дифференциально-разностных уравнений (ОДУ).

Все решатели *Simulink* образуют две основных категории: *фиксированного шага и переменного шага.*

Решатели фиксированного шага интегрируют ОДУ с постоянным шагом с начала и до конца моделирования. Можно установить размер шага или позволить решателю выбрать размер шага. Вообще при уменьшении размера шага увеличивается точность вычислений при увеличении времени, требуемого для моделирования.

Решатели переменного шага изменяют шаг в течение моделирования, уменьшая его для увеличения точности, когда состояние модели изменяется быстро, и увеличивают шаг при медленном изменении.

В зависимости от типов систем в Simulink представлены непрерывные и дискретные решатели.

Непрерывные решатели используют численное интегрирование для вычисления непрерывных переменных модели в любой момент времени. Если в блоке модели указано, что переменная дискретная, то непрерывные решатели учитывают ее дискретность.

Дискретные решатели существуют, прежде всего, для решения просто дискретных моделей (это модели, в которых дифференциальные уравнения заменены на разностные). Они не вычисляют непрерывные переменные модели. Дискретный решатель также учитывает эталонное время (фиксированный шаг) блока.

Дискретный решатель фиксированного шага имеет принципиальное ограничение: он выдает решение для отдельных моментов времени и не может использоваться для моделирования непрерывного состояния системы.

Примечание: Если модель содержит как непрерывные, так и дискретные блоки, нужно использовать непрерывный решатель. Дискретный решатель не может обрабатывать непрерывные состояния.

Simulink обеспечивает два *дискретных решателя*: дискретный решатель фиксированного шага и дискретный решатель переменного шага. Решатель фиксированного шага по

умолчанию выбирает размер шага и, следовательно, ошибку моделирования по изменению состояния в самом быстром блоке модели. Дискретный решатель переменного шага корректирует размер шага моделирования так, чтобы не превысить величину фактической ошибки дискретных изменений состояния модели. Это помогает избежать ненужных шагов и, следовательно, сократить время моделирования.

1.2. Методы решений дифференциальных уравнений

Самый простой метод численного решения дифференциальных уравнения — это *метод Эйлера.* Рассмотрим уравнение

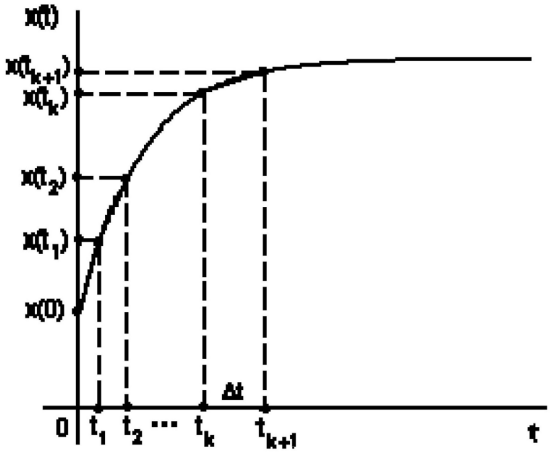

Рис. 1.1. Пример графика функции f(x,t)

$$\frac{dx}{dt} = f(x,t). \qquad (1.2.1)$$

Решением является функция времени x(t), график которой выходит из точки x(0) (начальное условие) (рис. 1.1). Численное решение основывается на последовательном вычислении значений функции x(t), начиная с x(0), с шагом по времени Δt. Как известно, производную приближенно можно представить как отношение приращения Δx к Δt, т. е.

$$\frac{dx}{dt} \approx \frac{\Delta x}{\Delta t} = \frac{x_{k+1} - x_k}{\Delta t} \approx f(x_k, t_k),$$

где x_k – значение функции x(t) в точке t_k; x_{k+1} – значение функции x(t) в точке t_{k+1}, причем $t_{k+1} = t_k + \Delta t$. Из этой формулы получаем

$$x_{k+1} = x_k + f(x_k, t_k) \cdot \Delta t. \qquad (1.2.2)$$

Это и есть *явная формула Эйлера*. Если известно начальное значение x(0), то с шагом Δt получаем остальные точки $x(t_k)$. Ее легко распространить на системы уравнений. Формула Эйлера имеет существенные недостатки: низкая точность и неустойчивость самого вычислительного процесса. Чтобы использовать метод Эйлера, нужно уменьшать шаг интегрирования Δt, что приводит к увеличению времени вычислений.

Повышенной устойчивостью, но самое главное, его можно применять к решению *жестких*[*)] систем уравнений, обладает *неявный метод Эйлера.* Формула (1.2.2) записывается в виде

$$x_{k+1} = x_k + f(x_{k+1}, t_{k+1}) \cdot \Delta t, \qquad (1.2.3)$$

т. е. x_{k+1} вычисляется уже из алгебраического уравнения, в общем случае нелинейного. Для системы дифференциальных уравнений – из системы нелинейных алгебраических уравнений. Шаг интегрирования для ускорения решения может быть переменным: если x(t) изменяется медленно – шаг увеличивается, если быстро – уменьшается.

Из-за низкой точности метод *Эйлера* применяется редко, а из многочисленных одношаговых методов наибольшее распространение получил *метод Рунге – Кутта* четвертого порядка с автоматическим выбором шага в зависимости от заданной точности решения (программы MathCad и MatLab).

[*)] Жесткие – это дифференциальные уравнения, которые описывают переходный процесс, например в линейной системе, как сумму экспонент с различными постоянными времени, отличающимися по величине на много порядков.

Считается, что для расчета сложных электрических цепей самыми экономичными в вычислительном отношении должны быть *многошаговые методы*, в частности *метод прогноза и коррекции*.

В качестве примера приведем формулу для метода *Рунге – Кутта* четвертого порядка.

$$x_{k+1} = x_k + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$

$$k_1 = f\left(x_k, t_k\right)\Delta t,$$

$$k_2 = f\left(x_k + \frac{k_1}{2}, t_k + \frac{\Delta t}{2}\right)\Delta t,$$

$$k_3 = f\left(x_k + \frac{k_2}{2}, t_k + \frac{\Delta t}{2}\right)\Delta t,$$

$$k_4 = f\left(x_k + k_3, t_k + \Delta t\right)\Delta t.$$

Шаг решения может быть постоянным – фиксированным, а может быть переменным. В программе MatLab предусмотрены несколько методов решения дифференциальных уравнений (табл. 1 и 2).

Если *шаг переменный*:

Таблица 1

Решатель	Тип уравнения	Порядок точности	Когда использовать
ode45	нежесткий	средний	В большинстве случаев. Должен быть первым пробным решателем
ode23	нежесткий	низкий	Для задач с низкой точностью или для того, чтобы решать умеренно жесткие уравнения
ode113	нежесткий	ниже высшего	Для задач с высокой точностью или для решения в вычислительном отношении быстро изменяющейся задачи
ode15s	жесткий	ниже среднего	Если `ode45` решает медленно, т. к. жесткая задача
ode23s	жесткий	низкий	При низкой точности, для решения жестких систем уравнений и при постоянной массовой матрице
ode23t	умеренно жесткий	низкий	Для умеренно жестких задач, без числового демпфирования
ode23tb	жесткий	низкий	Для решения жестких уравнений с низкой точностью

Алгоритмы

ode45 – базируется на явной формуле Рунге – Кутта (4, 5). Это одношаговый метод – для вычисления $y(t_n)$ требуется решение только в предыдущей точке времени $y(t_{n-1})$. Вообще, `ode45` – лучшая функция для «начальной попытки» для большинства задач.

ode23 – реализация явной формулы Рунге – Кутта (2, 3). Может быть более эффективной, чем `ode45`, при больших допусках и в присутствии умеренной жесткости. Подобно `ode45`, `ode23` – одношаговый метод.

ode113 – переменного порядка Адамса – Башфорта – Мултона (Adams – Bashforth – Moulton). Может быть более эффективен, чем `ode45`, при малых допусках. `ode113` – многошаговый метод – обычно требуется решение в нескольких предыдущих точках времени, чтобы вычислить текущее решение.

Вышеупомянутые алгоритмы предназначены для решения нежестких уравнений. Если они слишком медленные, используется один из методов для жестких систем уравнений.

ode15s – метод переменного порядка, основанный на формулах дифференцирования назад (NDF). Использует обратные формулы дифференцирования (BDF, известный как метод ускорения), которые являются обычно менее эффективными. Подобно `ode113`, `ode15s` – многошаговый метод. Если задача жесткая или `ode45` расходится или неэффективен, используется `ode15s`. Аналогично при решении дифференциально-алгебраической задачи.

ode23s базируется на модифицированной формуле Розенброка (Rosenbrock) второго порядка. Поскольку это одношаговый метод, то может быть более эффективен, чем `ode15s`, при больших допусках. Может решать некоторые виды жестких задач, для которых `ode15s` неэффективен.

ode23t – реализация формулы трапеций со «свободной» интерполяцией. Используется только для умеренно жестких уравнений при решении без числового демпфирования.

ode23tb – реализация TR-BDF2, первая ступень – неявная формула Рунге – Кутта трапециевидной формы и второй ступени – с формулой обратного дифференцирования второго порядка. Используется та же самая итеративная матрица в оценке обеих ступеней. Подобно `ode23s`, этот метод может быть более эффективным, чем `ode15s` при больших допусках.

Примечание: Для *жестких* задач решение может измениться за очень короткое время по сравнению с интервалом интегрирования. Методы, не предназначенные для жестких задач, неэффективны на интервалах, где решение изменяется медленно из-за достаточно маленького шага.

Если *шаг фиксированный:*

Таблица 2

Решатель	Метод	Точность	Метод интегрирования
`ode1`	явный	низкая	Метод Эйлера
`ode2`	явный	средняя	Метод Heun's
`ode3`	явный	выше средней	Формула Bogacki – Shampine
`ode4`	явный	высокая	Формула Рунге – Кутта четвертого порядка (RK4)
`ode5`	явный	высокая	Формула Dormand – Prince
`ode14x`	неявный	от низкого до высокого	Метод Ньютона с экстраполяцией

Точность и время моделирования зависят от размера шага: чем меньше шаг, тем более точный результат, но дольше моделирование. Однако это имеет свои пределы и при слишком малом шаге увеличивается число итераций, а следовательно, инструментальная ошибка. Шаг интегрирования может выбрать Simulink (значение по умолчанию) или установить пользователь.

Simulink устанавливает размер шага на *фундаментальное эталонное время* модели, если оно указано, или как результат деления *времени моделирования*[*)] на 50, если модель не имеет никакого дискретного состояния.

Еще раз отметим, что с дискретным решателем с фиксированным шагом нельзя получить непрерывные значения.

Если тип решателя устанавливается как решатель фиксированного шага, Simulink устанавливает решатель `ode3`, т. е. выбирает решатель, пригодный к обработке и непрерывного, и дискретного состояния с умеренными вычислительными затратами. Однако это не гарантирует, что заданный по умолчанию решатель точно вычислит непрерывную переменную модели или что решение не может идти быстрее с менее сложным решателем. В зависимости от динамики модели, нужно выбрать другой решатель или *эталонное время*, чтобы достичь приемлемой точности или сократить время моделирования.

- **Выбор фиксированного шага непрерывного решателя**

Вообще, практически невозможно заранее определить, какой решатель и размер шага дадут приемлемые результаты для непрерывной модели за минимальное время вычисления. Определение лучшего решателя для каждой модели требует эксперимента.

Вот предложенный MatLab способ выбора лучшего решателя фиксированного шага экспериментально. Сначала используется один из решателей переменного шага для моделирования до уровня желаемой точности. Анализируются результаты моделирования. Затем используется `ode1` для моделирования с заданным по умолчанию размером шага. Сравнивается результат моделирования с `ode1` и с решателем переменного шага. Если результаты те же самые в пределах указанного уровня точности, то лучший решатель фиксированного шага для данной модели – `ode1`. Дело в том, что `ode1` – наиболее простой из решателей фиксированного шага Simulink и, следовательно, имеет наименьшее время моделирования для текущего размера шага.

Если `ode1` не дает точных результатов, повторяются предыдущие шаги с другими решателями фиксированного шага, пока не найдется тот, который дает точные результаты с наименьшими вычислительными затратами. Самый эффективный способ сделать это состоит в том, чтобы использовать двойную поисковую методику. Сначала – `ode3`. Если дает точные результаты – `ode2`. Если `ode2` дает точные результаты, это лучший решатель для данной модели; иначе `ode3` – лучше. Если `ode3` не дает точные результаты – `ode5`. Если `ode5` дает точные результаты – `ode4`. Если `ode4` дает точные результаты, выбирается как решатель для данной модели; иначе `ode5`.

Нужно отметить, что с увеличением возможностей вычислительной техники многие задачи решаются сразу с завышенной точностью за приемлемое время, хотя сложные системы требуют настройки в вычислительном отношении.

- **Определение ошибки решателя переменного шага**

Решатели используют стандартные локальные методы защиты от ошибок. В течение каждого шага решатель вычисляет значения переменной в конце шага и определяет *локальную ошибку*. Сравнивается локальная ошибка с *допустимой ошибкой*, которая является функцией относительной ошибки (*rtol*) и абсолютной ошибки (*atol*). Если ошибка больше допустимой ошибки для *любой* переменной, решатель изменяет размер шага и пробует снова:

- *Относительная ошибка* каждой переменной. Значение по умолчанию 1e-3 означает, что вычисленное значение имеет точность 0.1%.
- *Абсолютная ошибка* – пороговое значение допустимой ошибки.

Если определено `auto` (значение по умолчанию), Simulink устанавливает абсолютный допуск для каждой величины первоначально 1e-6. По мере моделирования Simulink сбра-

[*)] Время моделирования – это время моделируемого процесса и не равно времени вычисления.

сывает абсолютную ошибку для каждой переменной к максимальному значению. Таким образом, если вычисляемая величина изменяется от 0 до 1 и `reltol` – 1e-3, то к концу моделирования `abstol` установлен на 1e-3. Если величина изменяется от 0 до 1000, то `abstol` установлен 1.

Если установка не удовлетворительная, пользователь может ее изменить. Часто для получения соответствующих значений абсолютной ошибки приходится выполнять повторное моделирование.

- **Когда какой решатель выбирать?**

Для непрерывных систем с непрерывным или фиксированным шагом, если состояние модели изменяется быстро или содержит разрывы, решатель переменного шага может значительно сократить время моделирования, потому что для такой модели при переменном шаге может потребоваться меньше шагов для достижения сопоставимой точности.

Если планируется компилировать модель с целью получения выполняемого файла для работы в *реальном масштабе времени* (мастерская реального времени Real-Time Workshop может работать только с дискретными блоками), нужен дискретный *решатель фиксированного шага,* т. к. компьютерные системы в реальном масштабе времени работают с фиксированным размером шага. Решатель переменного шага может заставить моделирование пропускать аварийные ситуации, которые могут произойти в компьютерной системе в реальном масштабе времени.

Если компиляция модели не планируется, выбор между переменным шагом и фиксированным шагом зависит от размеров модели и ее динамики.

Для дискретной системы точность моделирования определяется шагом времени. Если используется слишком большой шаг по времени, точность может быть недостаточной. Единственный способ узнать это состоит в том, чтобы повторить моделирование с различными шагами по времени и найти компромисс между приемлемым шагом времени и временем моделирования. Обычно шаг в 20–50 мкс дает хорошие результаты моделирования переходных процессов при коммутациях в энергосистемах на частоте в 50 Гц или в системах, использующих силовые электронные устройства типа диодов и тиристоров.

Шаг уменьшается для биполярных транзисторов с изолированным затвором (IGBT), полевых транзисторов (FET) и запираемых тиристоров (GTO), работающих на высоких частотах переключений.

Например, при моделировании инвертора с широтно-импульсной модуляцией (PWM), работающего на 8 кГц, требуется шаг времени, по крайней мере, 1 мкс.

Для малых систем алгоритмы с переменным шагом по времени обычно более быстры, чем с постоянным шагом, потому что меньше шагов интегрирования. Однако для больших систем, которые содержат много частей или много нелинейных блоков, типа электронных реле, выгоднее *дискретизация модели* с заменых непрерывных блоков на дискретные, но это не всегда возможно.

1.3. Эталонное время

Каждый блок Simulink имеет *эталонное время*, даже непрерывные блоки (например, Integrator) и блоки, которые не имеют состояния, типа Gain. Дискретные блоки позволяют определять их эталонное время через параметр Sample Time. Непрерывные блоки имеют бесконечно малое эталонное время, названное *непрерывным эталонным временем*. Блок, который ни дискретен, ни непрерывен, имеет *неявное эталонное время*, наследуемое его входом. Неявное эталонное время непрерывно, если любой из входов блока непрерывен. Иначе неявное эталонное время дискретно. Неявное эталонное время дискретной выборки равно самому короткому входному эталонному времени, если все входные эталонные времена кратны значениям целого числа самого короткого времени. В противном случае неявное эталонное время равно входному *фундаментальному эталонному времени.*

Фундаментальное эталонное время дискретной системы – самый большой делитель целого числа фактических эталонных времен системы. Например, предположим, что система имеет эталонные времена 0.25 и 0.5 секунд. Фундаментальное эталонное время в этом случае – 0.25 секунды. Предположим вместо него эталонные времена – 0.5 и 0.75 секунды. В этом случае фундаментальное эталонное время снова 0.25 секунды.

В Simulink можно установить фиксированный или переменный шаг дискретного решателя, чтобы рассчитать дискретную систему. Фиксированный шаг решателя равняется фундаментальному *эталонному* времени дискретной системы. Решатель изменяет переменный шаг так, чтобы равняться расстоянию между фактическими тактами эталонного времени. Следующая диаграмма (рис. 1.2) иллюстрирует различие между фиксированным шагом и переменным.

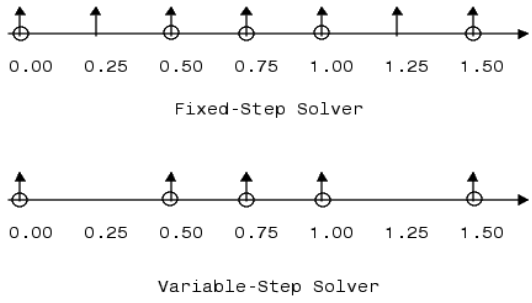

Рис. 1.2. Диаграмма эталонного времени для фиксированного шага и переменного шага.

В диаграмме стрелки указывают шаги моделирования, а круги представляют такты эталонного времени. Диаграмма иллюстрирует решающее устройство переменного шага, требующее, чтобы при моделировании было как можно меньше шагов, если фундаментальное типовое время меньше, чем любое из фактических эталонных времен моделируемой системы. С другой стороны, решающее устройство фиксированного шага требует, чтобы возможно меньше памяти было занято и быстрее шло решение, если эталонное время системы фундаментально. Это может иметь преимущество в случае компиляции модели Simulink (использование мастерской реального времени Real-Time Workshop).

1.4. Дискретизация моделей

При дискретизации модели выборочно заменяют непрерывные блоки Simulink эквивалентными дискретными элементами. *Дискретизация – определяющий шаг в разработке цифрового регулятора и для аппаратного обеспечения моделирования*. Можно использовать его для подготовки непрерывных моделей к компиляции в Real-Time Workshop, которое поддерживает только дискретные блоки. В основе дискретного описания линейных систем лежит дискретное Z-преобразование.

1.5. Типы данных, поддержанные Simulink

Таблица 3

Имя	Описание
`double`	Двойная точность с плавающей точкой
`single`	Обычная точность с плавающей точкой
`int8`	Обычная 8 бит целая
`uint8`	Без знака 8 бит целая
`int16`	Обычная 16 бит целая
`uint16`	Без знака 16 бит целая
`int32`	Обычная 32 бита целая
`uint32`	Без знака 32 бита целая

Simulink поддерживает все встроенные типы данных MatLab, кроме *int64* и *uint64*. Термин *встроенный тип данных* относится к типам данных, определенным непосредственно MatLab в противоположность типам данных, *определенным пользователями* MatLab. Таблица 3 – список встроенных типов данных MatLab, поддерживаемых Simulink; таблица 4 – диапазоны чисел и точность, с которой эти числа представлены.

Таблица 4

Тип данных	Нижний предел	Верхний предел	Точность
`single`	$2^{-126} \approx 10^{-38}$	$2^{128} \approx 3 \cdot 10^{38}$	$2^{-23} \approx 10^{-7}$
`double`	$2^{-1022} \approx 2 \cdot 10^{-308}$	$2^{1024} \approx 2 \cdot 10^{308}$	$2^{-52} \approx 10^{-16}$
`int(n)`	-2^{n-1}	$2^{n-1} - 1$	1
`uint(n)`	0	2^{n-1}	1

Помимо встроенных типов, Simulink определяет *булевой (1 или 0) тип*, представленный внутренне значениями uint8. Также много блоков Simulink поддерживают тип данных с *фиксированной точкой*. См. блоки Simulink в документации Simulink относительно типов данных, поддержанных определенными блоками. Если документация для блока не определяет тип данных, входы или выходы блока имеют тип данных только двойной точности.

1.6. Сигналы в Simulink

Сигналы – потоки данных, которые появляются на выводах блоков Simulink при моделировании.

Сигнал может быть одномерным – *вектор* – и двумерным – *матрица*. Матрица с одним элементом – *скаляр*. *Вектор-строка* – двумерная матрица, которая имеет одну строку. *Вектор-столбец* – двумерная матрица, которая имеет один столбец. Некоторые блоки могут работать с сигналами любых размеров. Некоторые могут принять или вывести только скалярные или векторные сигналы (см. описание блока).

По умолчанию тип данных сигналов Simulink – двойная точность с плавающей точкой (double). Сигналы Simulink могут быть комплексными числами. Некоторые блоки работают только с сигналами булевого типа (0 и 1). Переход от одного вида сигналов к другому осуществляется блоком преобразования сигналов Data Type Conversion.

Виртуальный сигнал – сигнал, который представляет другой сигнал графически. Виртуальные сигналы – просто графические объекты. Они не имеют никакого математического или физического значения. Simulink игнорирует их при моделировании.

Управляющий сигнал – сигнал, используемый одним блоком, чтобы запустить выполнение другого блока.

При создании модели Simulink должны соблюдаться *правила размеров параметра и сигнала*: параметры блока должны иметь те же самые размеры, что и соответствующие входные сигналы. Блок может иметь скалярные параметры, соответствующие нескалярным сигналам, но в этом случае Simulink расширяет скалярный параметр, чтобы иметь те же самые размеры, что и соответствующий сигнал.

1.7. Блоки Simulink

Все блоки Simulink подразделяются по двум основным категориям: *невиртуальные* и *виртуальные блоки*.

Невиртуальные блоки играют активную роль в моделировании системы. Если добавляется или удаляется невиртуальный блок, то изменяется поведение модели.

Виртуальные блоки, напротив, не играют никакой активной роли в моделировании; они помогают организовывать модель графически. Некоторые блоки Simulink виртуальные в некоторых обстоятельствах и невиртуальные в других. Такие блоки называют *условно виртуальными* блоками.

1.8. Субсистема (SubSystem)

Подсистема (субсистема) представляет систему внутри другой системы. Подсистема – удобный способ создания обозримых схем моделей, правда, при потере наглядности.

В зависимости от способа управления субсистемой различают: Enable Subsystem – подсистема, которая работает, пока есть внешний сигнал; Trigger Subsystem – подсистема, выполнение которой запускается изменением полярности управляющего сигнала, подобно триггеру, даже отдельный импульс запускает подсистему. Enable and Trigger Subsystem – подсистема, выполнение которой сочетает Enable- и Trigger-управление внешним входным сигналом.

С появлением новых версий Simulink возможности субсистем расширяются (см. соответствующую литературу).

1.9. Особенности приложения SimPowerSystem

В блок-схемах Simulink информация – сигнал – передается по линям связи от одного блока другому только в одном направлении – указано стрелкой. Это общее свойство блок-схем систем автоматического управления. В SimPowerSystem вводятся электрические цепи, в которых сигнал, например электрический ток, может протекать в обоих направлениях. Это отражено в моделях. Поэтому в SimPowerSystem присутствуют две сети: *силовая*, которая моделирует электрическую цепь, и информационная, моделирующая системы автоматического управления. К сожалению, в программе не предусмотрено выделение силовой цепи, например, синим цветом.

Эти две сети напрямую не подключаются. Чтобы на осциллограф Scope подать напряжение силовой цепи, нужно включить измерительный прибор – вольтметр Voltage Measurement. Сигнал с вольтметра подается на осциллограф. Точно так же для управления силовой цепью служат преобразователи сигналов в напряжения или токи. Силовая электроника хотя и работает в силовой цепи, но управляется информационной цепью. В этом ее кардинальное отличие от других программ по моделированию электрических цепей, например MicroCap.

1.10. Упрощение моделей

При решении сложных задач большой размерности используется метод *диакоптики* – замены одной задачи большой размерности несколькими, меньшей размерности. Для электромеханических систем практически всегда возможно разделение на некоторое число независимых фрагментов, по крайней мере, в вычислительном отношении.

В общем случае используется *макромоделирование* и *структурная декомпозиция*. *Макромодели* – математические модели реальных устройств различного уровня сложности. *Структурная декомпозиция* – расщепление сложного объекта на части, объединенные внешними связями.

Несколько условно можно выделить несколько типов задач: самая простая задача – построение упрощенной модели по ее математической зависимости. Чаще встречаются ситуации, когда нужно построить сложную модель или ее численный вариант для данного набора параметров. Общей аналитической зависимости нет. В математике замена численно не выражаемой зависимости вычисляемой функцией называется *аппроксимацией*. Построение модели системы по экспериментальным данным носит название *структурной идентификацией*. Если математическая зависимость известна, но нет параметров модели – *параметрическая идентификация*. Проведение экспериментов для построения моделей – планирование эксперимента при статистическом характере результатов и ошибках опытов.

Надо отметить, что, хотя современные ЭВМ и позволяют моделировать очень сложные объекты и системы, при большом количестве сложных связанных объектов моделировать их вместе без упрощения практически невозможно.

2. Электропривод

2.1. Основные функции и определения

Электроприводом называется электромеханическая система, преобразующая электрическую энергию в механическую с определенным законом движения. Электропривод бывает регулируемый и нерегулируемый. Далее будем рассматривать только регулируемый электропривод, хотя процесс пуска двигателя при непосредственном подключении к сети, как правило, обязательная часть моделирования регулируемого электропривода, – типичная нерегулируемая система. В свою очередь регулируемый электропривод делится на разомкнутый и замкнутый.

В *разомкнутых системах* в установившемся состоянии двигатель непосредственно подключен к питающей сети, а управление осуществляет аппаратура пуска и торможения (контакторы, реле, тиристорные преобразователи). Схемы управления таких систем электропривода конструктивно оформляют в виде станций управления. В схемах станций должны быть типовые узлы управления, защиты и блокировки.

Замкнутые системы управления электроприводом – это системы автоматического управления с обратной связью, осуществляющие заданный набор функций управления.

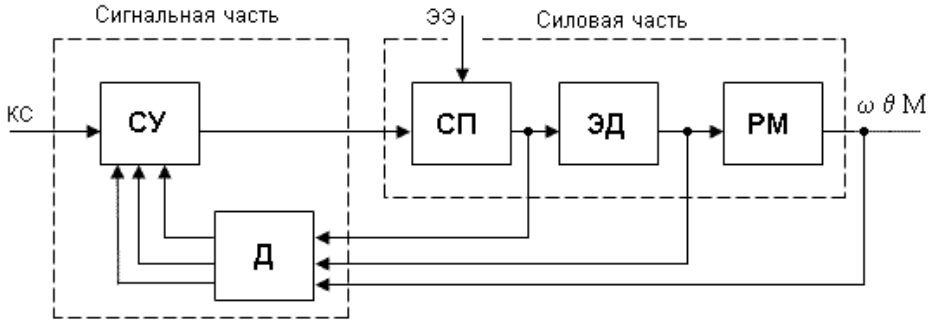

Рис. 2.1. Структурная схема замкнутого электропривода

Структурная схема замкнутого электропривода представлена на рис. 2.1, где:
СУ – система управления, выдает сигналы управления силовому преобразователю;
Д – датчики скорости, угла поворота, тока и т. д.;
СП – силовой преобразователь, в основном, полупроводниковый коммутатор;
ЭД – электродвигатель постоянного или переменного тока;
РМ – рабочий механизм: фрезерный станок, рули управления, лебедка крана и т. д;
КС – командный сигнал, задаваемый или человеком, или другой системой управления;
ЭЭ – электрическая энергия для питания привода;
ω θ М – скорость, угол поворота, момент на выходе системы.

Замкнутые системы управления электроприводами могут иметь различные виды управления: 1) системы стабилизации – поддержание постоянства управляемой координаты; 2) следящие системы – управление координатой по заданному заранее неизвестному закону; 3) программное управление – управление с целью изменения управляемой координаты по закону, определенному заранее и заданному программой.

Системы стабилизации в большинстве случаев являются системами стабилизации скорости. Но встречаются системы стабилизации других параметров технологических процессов, например момента.

Следящие системы представляют собой системы регулирования положения рабочего механизма. Типичным примером следящей системы может служить система управления

радиолокационной антенной, следящей за летящим объектом, характер движения которого заранее неизвестен.

Программное управление положением рабочего механизма, которое должно изменяться по заданной программе, в настоящее время осуществляется цифровыми микропроцессорами.

2.2. Математические модели

Во всех учебниках по электроприводу обязательно рассматривается линеаризация математических моделей элементов систем управления. Причина этого – математический аппарат, разработанный для линейных систем и расчетов вручную. Для простейших систем с двигателями постоянного тока с таким упрощением можно согласиться, но для современных систем управления этого просто нельзя сделать.

В дальнейшем рассмотрим модели элементов силовой части электропривода: 1) полупроводниковых преобразователей; 2) электрических двигателей; 3) датчиков.

2.3. Элементная база силовой электроники

Основными приборами силовой электроники в области коммутируемых токов до 50 А являются: диоды (Diode); тиристоры (Thyristor, SCR); биполярные транзисторы (BPT); биполярные транзисторы с изолированным затвором (IGBT); полевые транзисторы с изолированным затвором (MOSFET); силовые интегральные схемы (Power IC); интеллектуальные силовые интегральные схемы (Smart Power IC).

В области коммутируемых токов более 50 А основными приборами силовой электроники являются: силовые модули на базе биполярных транзисторов; силовые модули на базе IGBT; тиристоры; запираемые тиристоры (GTO); диоды.

Сами GTO в последние годы были модернизированы (ABB Mitsubisi), и появился новый класс приборов — тиристор, коммутируемый по затвору (GCT — Gate Thyristor или IGCT — Integrated Gate Commutated Thyristor).

Биполярные транзисторы (BPT)

Силовые биполярные транзисторы в диапазоне до 50 А находят применение в основном в массовом и дешевом бытовом и промышленном оборудовании.

Из-за сложности и большой стоимости схем управления (драйверов), низкого быстродействия и недостаточной стойкости к перегрузкам на сегодняшний день устарели. Однако быстродействующие BPT пока имеют важное преимущество перед MOSFET по показателю «коммутирующая мощность/цена» для диапазона напряжений более 400 В. Поэтому биполярные силовые транзисторы останутся эффективным компонентом для дешевых массовых применений (например, ключевые источники питания — SMPS).

Тиристоры (SCR)

Несмотря на такие очевидные достоинства, как низкое падение напряжения (1.2–1.5 В для среднего диапазона напряжений и немногим более для высоковольтного диапазона), высокая плотность тока, наивысшее значение показателя «коммутируемая мощность/площадь кремния», высокие коммутируемые напряжения (сегодня 8–10 кВ) и токи (5 кА), простота и низкая стоимость схем управления, стойкость к перегрузкам по току, высокая надежность прижимной таблеточной конструкции, – этот класс приборов силовой электроники сегодня можно отнести к устаревшим из-за одного существенного недостатка – невозможности выключения по управляющему электроду. Этот прибор все больше и больше будет вытесняться полностью управляемыми приборами: IGBT и IGCT. Так как SCR имеет все же наивысшее значение показателя «коммутирующая мощность/цена», то две области останутся предпочтительными для их применения: