SEQUEL: using the GUI

Mahesh B. Patil

Department of Electrical Engineering

Indian Institute of Technology, Bombay

Mumbai-400076, India

e-mail: mbpatil@ee.iitb.ac.in

Руководство пользователя

Оглавление

1 Введение

Целью SEQUEL GUI является предоставление пользователям удобных средств создания схемы, задания параметров анализа и обзора результатов симуляции. Использование GUI подразумевает следующие шаги:

a) ввод схемы;
b) подготовка раздела решения;
c) создание файла схемы (circuit file);
d) запуск SEQUEL с файлом схемы;
e) обзор результатов.

Когда запускается GUI, можно увидеть рабочее окно, схематически показанное на рис.1.

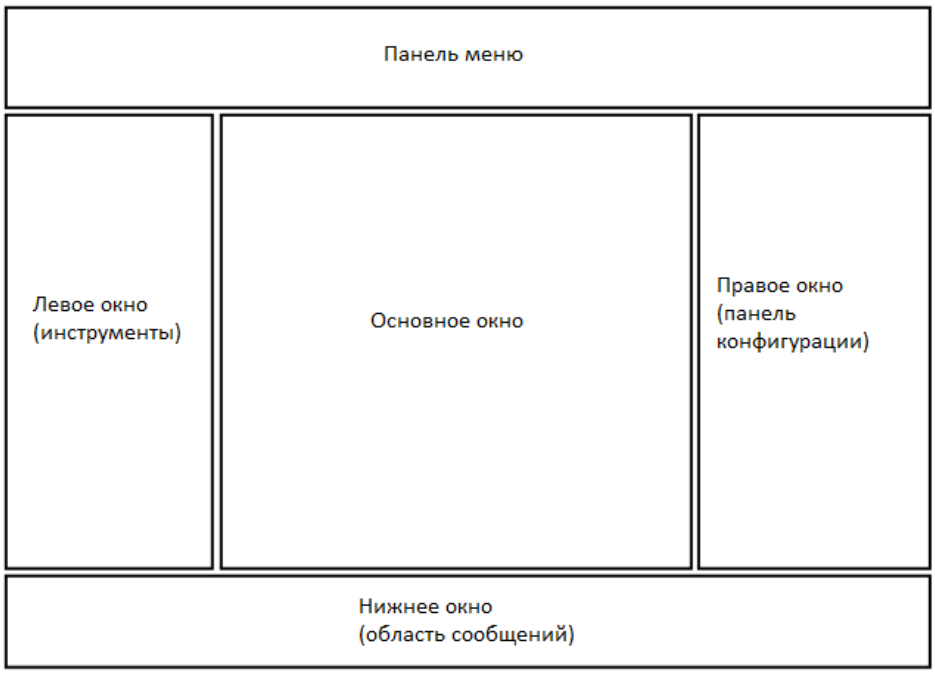

Рис. 1. Схематическое изображение рабочего окна программы

Из всех элементов рабочего окна панель меню остаётся неизменной, а содержание других окон может меняться, в зависимости от конкретной ситуации. Например, когда выбрана закладка «C*ircuit Editor*», в центральном окне появляется рабочее поле чертежа для ввода схемы. А когда выбрана закладка «*Graph*», в основном окне появляются средства выбора для отображения диаграмм, и т.д. Самый лучший способ начать работать с GUI – это запустить существующий пример и посмотреть на результаты, следуя шагам, описанным ниже:

1. Щёлкните (будем подразумевать, если не сказано иное, щелчок левой клавишей мышки) по иконке «*Open Project*» в меню, выберите директорию (в месте установки программы) electronics_gnrl. Проекты SEQUEL, доступные в этой директории (каждый с расширением .sqproj), появятся в отдельном окне. Щёлкните по проекту rc1.sqproj.
2. Щёлкните по закладке «C*ircuit Editor*», чтобы увидеть схему. Вы можете пощёлкать по всем компонентам, чтобы увидеть их параметры в правом окне, если предварительно щёлкните там по закладке «P*roperty Editor*». Вы можете также увидеть список выходных переменных, которые были выбраны для этого проекта, щёлкнув по закладке «*Output Variables*» в правом окне.

3. Щёлкните по закладке «*Solve Blocks*», чтобы увидеть детали анализа, который будет запрошен для этой схемы. В секции «output blocks» (последний раздел) вы найдёте список переменных, набор которых упомянут выше. При работе SEQUEL сделает эти переменные доступными в файле, заданном выходным блоком. Выходной файл записывается в директории /SequelGUI2_Release/SGUI/Output.
4. Щёлкните по закладке «*Circuit File*» и выберите «**Generate CF**». В основном окне появится файл схемы, соответствующий введённой схеме и созданному разделу решения. В файле блок, относящийся к схеме, заключён между begin_circuit и end_circuit, а блок решения между begin_solve и end_solve. Файл схемы можно хранить на жёстком диске, если щёлкнуть по кнопке «**Save CF**», но это не обязательно для выполнения моделирования.
5. Запустите SEQUEL после генерации файла схемы, щёлкнув по кнопке «**Run Solver**» на инструментальной панели меню (этим генерируется файл схемы и выполняется моделирование; в этом смысле запуск генерации файла схемы не обязателен). Сообщение, что решение может быть сгенерировано при выполнении файла схемы, появится в нижнем окне, если щёлкнуть там по закладке «*Solver Output*». Сообщение включает размер матрицы, число итераций (для анализа переходного процесса) и т.д., а также сообщения об ошибках, если они появляются при решении. Когда выполнение завершится, вы увидите сообщение «*SEQUEL: program completed*». В этом простом примере (rc1.sqproj), содержащим маленькую схему, выполнение очень быстрое, вы почти сразу увидите сообщение, что программа выполнила работу.

   Когда программа закончит работу, выходные данные, запрошенные вами, будут записаны в выходные файлы, как описано выше.

6. Теперь щёлкните по кнопке «Graphs». Основное окно образует три колонки. В левой колонке вы увидите имена выходных файлов, которые вы задали в блоке решения (см. выше). Когда вы выделите нужный файл, имена переменных, содержащиеся в этом файле, появятся во второй и третьей колонках. Одна из переменных во второй колонке может быть выбрана в качестве аргумента (ось x), а другая (или другие) в качестве функции (ось y). Щёлкните по кнопке «**Graph It**», чтобы увидеть нужный график. Если нужен другой график, щёлкните по кнопке «Back» и повторите вышеописанную процедуру.

2 Обучающий пример

Чтобы посмотреть на основные функции GUI, мы сейчас создадим SEQUEL проект с начала. Собственно, мы соберём и смоделируем схему, показанную на рис.2, а затем получим графики зависимости V1, V2 от времени.

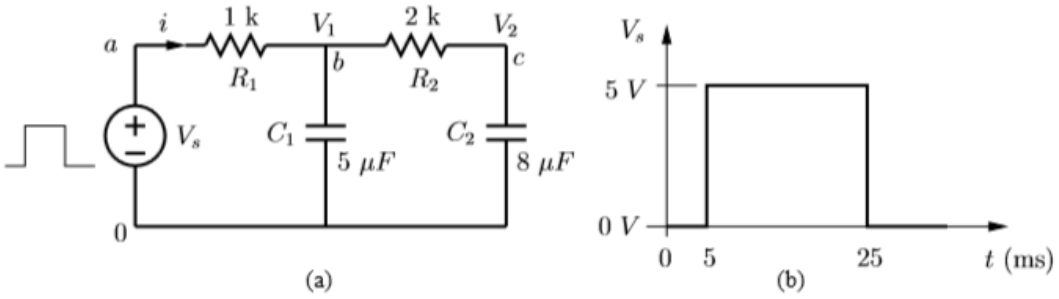

Рис. 2. RC цепь

2.1 Создание схемы

В SEQUEL есть следующие типы элементов:

- Подборка электрических элементов (ece).
- Подборка цифровых элементов (dce).
- Подборка основных элементов (gce).
- Подборка температурных элементов (tce).
- Основные смешанные элементы (gme).

В схеме на рисунке выше включены только элементы типа ece, а именно: резисторы (r.ece), конденсаторы (c.ece) и импульсный генератор (vpulse.ece). Щёлкните по закладке редактора схемы (circuit editor). В основном окне будет рабочее поле для черчения (сборки, создания) схемы, а в левом окне список всех доступных типов элементов. Разверните список (ece) и щёлкните по r.ece. Переместите курсор в свободное место рабочего поля чертежа и дважды щёлкните (напомню, левой клавишей мышки). Появится резистор. Выберите редактор свойств (property editor) в правом окне. Теперь, вернитесь к чертежу и щёлкните по резистору. Он будет подсвечен, а список свойств r.ece отобразится в правом окне. Измените его имя (GUI автоматически даёт имена, но для ясности можно ввести свой идентификатор) на r1, а его сопротивление (появляющееся как реальный параметр) на 1k. Следуя процедуре, описанной выше, создайте остальные элементы, измените их свойства, как этого требует схема. Для генератора импульсов мы можем задать время начала t1 = 5 ms, а время окончания t2 = 25 ms. Однако соответствующие параметры будут t_1=5m, t_2=25m, v_1=0 и v_2=5. Параметры delt_1 и delt_2 – это время нарастания и спада импульса – могут быть заданы подходящим значением, скажем, 10 u (микросекунд). Поверните конденсаторы на 90 градусов по часовой стрелке. Это можно сделать выделением элемента и вводом «r» или заданием «rotation» на 90^0 в окне свойств. Переместите элементы в нужные места либо щелчком по ним с последующим перетаскиванием мышкой, либо щёлкнув по ним, а затем переместив с помощью курсорных клавиш. Заметьте, если элементы выделены (подсвечены), выделение можно снять простым щелчком левой клавиши мышки по свободному месту рабочего поля чертежа. Теперь мы готовы выполнить необходимые соединения. Чтобы соединить узел 1 с узлом 2, щёлкните по узлу 1, отпустите клавишу мышки, переместите курсор к узлу 2 (если необходимо, то к точке желательного перегиба соединения). Продолжайте этот процесс, пока не достигнете узла 2. GUI автоматически выйдет из режима соединения, когда вы щёлкните по узлу 2. Если в середине процесса вам понадобится удалить соединение, нажмите F4, и GUI выйдет из режима соединения. После завершения соединений нам потребуется сообщить SEQUEL об узле, который принимается за общий провод. Для этой цели перенесите элемент ground.ece на чертёж и соедините его с нужным проводом или узлом элемента. Как и с именами элементов, GUI генерирует имена для всех узлов. Программа будет работать с этими именами. Однако для удобства пользователь может придать узлу имя, которое ему больше нравится. Можно это сделать двумя путями:

- Щёлкните по проводу (узлу) и отредактируйте его имя в правом окне (убедитесь, что закладка «*Property Editor*» в правом окне выбрана).
- Щёлкните по элементу, к которому присоединён узел, и отредактируйте имя узла для этого элемента в правом окне.

Если имя изменялось описанным выше способом, узел, связанный со всеми общими элементами, будет обновляться одновременно. Для нашей схемы мы можем задать имена узлов a, b, c, 0, как

показано на рисунке. Удобный способ увидеть имя (заданное или по умолчанию) узла или элемента – это удержать курсор (не нажимая на клавиши мышки) на нужном узле (или элементе) секунды две или около того. Имя появится в окошке около курсора (если это элемент, появится и его тип). Вы можете проверить это на вашей схеме на чертеже. Заключительный шаг в создании описания схемы – это определение выходных переменных. В нашем примере мы хотели бы определить ток через резистор R1 и напряжение узлов b и c в качестве выходных переменных. Это можно сделать следующим образом (Эксперты могут добавлять переменные, удерживая клавишу crtl при щелчке по «*Output Variables*». В этом случае можно добавлять переменные одна за другой и выйти из режима, что очень важно, повторным щелчком по «**Add variable**»).

1. Щёлкните по закладке «*Output Variables*» в правом окне и выберите «**Add Variable**». GUI теперь в режиме добавления выходных переменных. Щёлкните по резистору r1. Появится выпадающее меню, показывая i1, v1 и т.д. Выберите i1. GUI автоматически выйдет из режима добавления переменных.
2. Войдите в режим добавление выходных переменных и щёлкните по узлу (проводу) b.
3. Войдите в режим добавление выходных переменных и щёлкните по узлу (проводу) c.

После выполнения вышеописанных шагов вы заметите, что GUI сгенерированы следующие строки в правом окне var1=i1_of_r1 var2=nodev_of_b var3=nodev_of_c Имена var1, var2, var3 были сгенерированы GUI по умолчанию. Вы можете изменить их на более удобные, дважды щёлкнув по конкретной строке и введя новое имя. Например, можно использовать i1, v_b, v_c вместо var1, var2, var3.

2.2. Создание блоков решения

Мы подготовили схему к моделированию. Следующее, что мы должны сделать, это проинформировать симулятор об анализах, которые предстоит провести. Это то, что содержится в разделе решения. Для схемы выше есть два анализа, которые нам хотелось бы применить:

a) start-up анализ для получения состояния при t=0−;
b) анализ переходного процесса для t > 0.

Щёлкните по закладке «*Solve Blocks*». Основное окно теперь можно использовать для создания и редактирования блоков решения следующим образом:

1. Удостоверьтесь, что закладка «*Property Editor*» выбрана в правом окне. Щёлкните по кнопке «**Add Solve Block**». Выделите появившийся блок, щёлкнув по его заголовку. В правом окне выберите тип блока (Start Up) в окошке типа, щёлкнув по кнопке со стрелкой вниз справа или по окну, из выпадающего списка. Далее нам нужно выбрать начальное решение в левом окне для использования в этом блоке. Сделайте это, перетащив элемент initial sol из левого окна в блок решения. По умолчанию значение для initial_sol – это initialize, и оно нам подходит.
2. Добавьте второй блок и измените его тип на transient. Перетащите initial_sol из левого окна в текущий блок основного окна, щёлкните по нему и выберите previous в правом окне (многие компоненты блока решения задаются таким же образом: перетаскиваем ключевого слова из левого окна, выделением его и выбором подходящих свойства в правом окне). Этим создаётся initial_sol=previous (начальное решение=предыдущее) в блоке решения. Перетащите back_euler и задайте для него yes, щёлкнув в правом окне и поставив галочку в появившемся окошке. Перетащите t_start и задайте 0 для него в правом окне. Используйте такие же процедуры чтобы назначить: t_end=100m

delt_const=10u Это всё проинструктирует SEQUEL применить анализ переходного процесса от нуля до 100 миллисекунд с константой Δt=10 μs, используя обратный метод Эйлера. Теперь пора задать имена выходным файлам и то, какие выходные переменные будут записаны в каждый из файлов. Скажем, мы решили записать узловые напряжения v_b и v_c в файл с именем rc_1.dat, а ток i1 в другой файл rc_2.dat. Это можно сделать так. Перетащим output block из левого окна в центральное, щёлкнем по нему. Правое окно покажет разные опции, которые можно задать для выходного блока. Зададим имя файла rc_1.dat. Щёлкнем по полю output variables. Этим отобразится список выходных переменных (которые были определены при создании схемы). Выделим v_b и v_c. Аналогично создадим второй выходной блок с именем rc_2.dat и выберем выходную переменную i1 для него.

2.3 Выполнение программы и обзор графиков

Файл схемы может быть создан щелчком по закладке «*Circuit File*» и затем по кнопке «**Generate CF»** в основном окне. Файл схемы соотносится со схемой, и созданные вами блоки решения появятся в центральном окне. Хотя полезно взглянуть на схемный файл, это не необходимо для пользователя генерировать его, поскольку GUI автоматически это сделает при использовании «**Run Solver**». Запустите SEQUEL этой кнопкой. Щёлкните по закладке «*Solver Output*» в нижнем окне сообщений и убедитесь, что появилось сообщение «SEQUEL: program completed». Щёлкните по закладке «*Graphs*» и посмотрите на графики v_b и v_c от времени, следуя процедуре описанной ранее. Интерфейс Graph позволяет рисовать графики с разными осями y. Это полезно, когда два графика должны изображаться вместе с разными параметрами, то есть, напряжение от 0 до 5 В и ток, меняющийся от $-1\times10-3$ А до $2\times10-3$ А. Пользователь может выбрать левую ось y для напряжения, а правую для тока, просто поставив соответствующую галочку для оси y. Выполните: добавьте узловое напряжение v_a для выходного файла rc_1.dat (первый выходной блок), сгенерируйте файл схемы, запустите SEQUEL и постройте зависимости v_a, v_b, v_c от времени на одном графике.

3 Создание схемы, основные приёмы

Предыдущее упражнение по созданию проекта с самого начала познакомило читателя с некоторыми операциями и возможностями, даваемыми GUI. Теперь мы опишем это более детально. Мы начнём с блока схемы и предполагаем далее, что GUI уже в режиме редактора схемы (circuit editor).

3.1 Глобальные параметры

Положим, что кто-то хочет задать одинаковое значение параметрам разных элементов и, возможно, даже одинаковые параметры в разделе решения. В этом случае этому некто следует определить «global parameter» (целый или действительный) и связать его с интересующим параметром вместо того, чтобы привязывать его к числам. В качестве примера рассмотрим сумматор на ОУ, показанный на рисунке ниже, который разрабатывается для получения одинакового усиления для всех трёх входов. Следовательно, если мы изменим, скажем, Ra с 1 k на 2 k, мы увидим, что Rb и Rc также изменились похожим образом. Это можно применить, используя глобальный параметр, как изложено ниже.

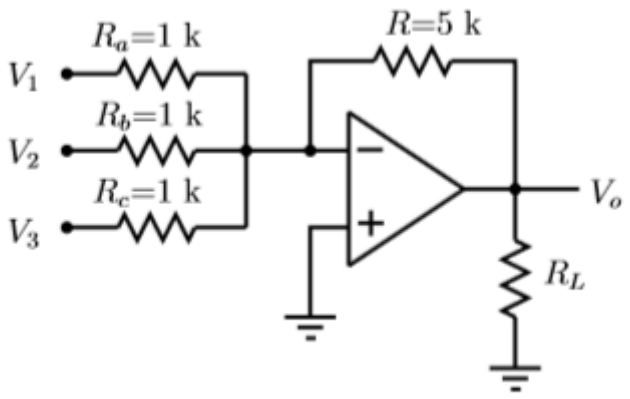

Рис. 3. Сумматор на ОУ

Щёлкните правой клавишей мышки по пустому месту на чертеже, выберите «*Add Global Parameters*» и введите r1 в качестве имени параметра. Щёлкните по пустому месту на чертеже (где нет ни элементов, ни проводов). Выберите закладку «*Property Editor*» в правом окне. Правое окно отобразит список глобальных параметров (который в настоящий момент имеет только r1). Назначьте 1k для r1 в правом окне. Щёлкните по резистору Ra на вашей схеме и отредактируйте реальный параметр r (его сопротивление). Щёлкните в правой половине этого поля, и появится выпадающее меню. Выберите r1. Это сделает задание r=r1. Сделайте такое же задание для Rb и Rc. Если глобальный параметр r1 теперь изменится на 2k, это отразится на каждом из резисторов Ra, Rb, Rc, которым мы назначили глобальный параметр.

3.2 Вычисление выражения

Часто приходится определять параметр элемента в терминах параметра(ов) другого элемента. Например, рассмотрим R-2R лестничную схему, показанную на рис.4. В этой цепи мы хотели бы определить R1a = R1b = R1c = R, скажем, 5 Ом, а R2a = R2b = R2c = R2 = 2R. Это может быть выполнено использованием двух выражений:

```
greal r=5

greal 2r=$((r)*(2))
```

а затем присвоим r сопротивлениям R1a, R1b, R1c и 2r сопротивлениям R2a, R2b, R2c, R2. Разрешены две формы выражений:

a) `greal_name=$(string1(string2))`, где greal_name – это имя глобального реального параметра. string1 может быть выбрано из exp, sin или cos. string2 может быть реальным числом или именем ранее определённого глобального действительного параметра. Например, greal evt=$(exp(vt));
b) `greal_name=$((string1)string2(string3))`, где greal_name – это имя глобального действительного параметра. string1, string3 могут быть действительными числами или именем ранее определённого глобального действительного параметра. string2 используется для указания на операцию и может быть одной из +, -, *, /, **. Например, greal 2r=$((2)*(r)) greal y1=$((x1)**(1.5)).

В обеих формах выражения $ sign и скобки требуются правилами синтаксиса. Операции умножения не разрешаются в единичном greal и должны быть выполнены в серии выражений. Например:

```
greal k=1.3806226e-23
greal q=1.6021918e-19
greal tmpr=300
greal kT=$((k)*(tmpr))
greal vt=$((kT)/(q))
```

3.3 Выбор элементов

Если нам предстоит симулировать схему, то как мы решим, какой компонент выбрать из списка элементов? Есть разные пути для этого.

1. Выбором элементов из списка, если имя элемента известно: например, представьте, мы хотим добавить на чертёж vsrcac.ece. Есть два варианта для выполнения:
    - Раскройте список электрических элементов. Щёлкните по vsrcac.ece. Этот элемент будет подсвечен. Отпустите клавишу мышки. Переместите курсор в нужное место чертежа и дважды щёлкните (левой клавишей мышки).
    - Введите vsrcac в окошко поиска в левом окне. GUI покажет все элементы, чьи имена совпадают со строкой, введённой вами. Щёлкните по vsrcac.ece, переместите курсор в нужное место на чертеже и дважды щёлкните левой клавишей мышки.
2. Если имя элемента не известно, воспользуйтесь существующими SEQUEL проектами. Проверьте, нет ли похожей схемы на ту, что намерены собрать вы. Если есть, найдите имя элемента, который отвечает вашим нуждам.
3. Обратитесь к «on-line» документации, щёлкнув правой клавишей мышки по элементу в списке и выбрав «Help». Выясните, какой элемент наилучшим образом подходит для ваших целей. Во многих случаях сами имена подсказывают назначение элементов, то есть, элементы r.ece, c.ece, l.ece могут использоваться в качестве резистора, конденсатора и индуктивности соответственно.
4. Скопируйте и вставьте из другого проекта: если есть элемент, который уже имеет подходящие для вашего проекта значения, вы можете его скопировать и вставить (Copy/paste механизм работает как внутри проекта, так и между двумя приложениями) одним из следующих способов:
    - Выберите элемент и нажмите **ctrl c**. Этим вы скопируете объект в буфер обмена. Вы можете видеть содержимое буфера обмена (clipboard), выбрав закладку «*Clipboard*» в правом окне. Переместите курсор в место на чертеже, где хотите вставить скопированный элемент и нажмите **ctrl v**.
    - Удерживайте клавишу **ctrl** и, используя мышку, обрисуйте прямоугольник вокруг элементов, которые хотите скопировать. Нажмите **ctrl c**. Этим вы скопируете элементы в буфер обмена. Переместите курсор в место, куда вы хотите вставить скопированные элементы, и нажмите **ctrl v**. Элементы появятся там, куда вы указали. Буфер обмена также может использоваться для операции cut/paste элементов, просто используйте **ctrl x** вместо **ctrl c**. Эта возможность обычно полезна, если вам нужно переместить элемент или группу элементов в другое место на чертеже. Буфер обмена запоминает скопированные/вырезанные объекты в прошлом (даже, если GUI закрыть и запустить вновь), и любые объекты в буфере обмена могут быть выбраны для вставки простым щелчком по ним. Объекты из буфера обмена можно удалить, если щёлкнуть правой клавишей мышки и выбрать «*Remove*». И не забывайте очистить буфера обмена, если скопированные элементы больше вам не нужны.

Нужно быть внимательным, если элемент, который был скопирован из другого проекта, имеет глобальные параметры, связанные с ним. Те же глобальные параметры будут определены для текущего проекта, следуя процедуре, описанной в разделе 3.1.

3.4. Навигация по чертежу

Следующие средства могут использоваться для навигации по чертежу в режиме редактирования схемы.

1. Увеличение и уменьшение вида можно выполнить щелчком по кнопкам «**Zoom In**» или «**Zoom Out**». Аналогичные изменение можно внести, используя колёсико мышки. Есть две другие полезные возможности масштабирования:
    - Кнопка «**Zoom Fit**» с двумя стрелками используется для подстройки сетки так, чтобы схема вписывалась точно в чертёж. Она так же полезна, когда вы «теряете» схему (то есть, она где-то на чертеже, но часть схемы вне поля зрения). Нажмите кнопку «**Zoom Fit**» и вы увидите свою схему.
    - Раздел «Zoom One» пункта «View» основного меню используется для приведения сетки к заданной единице по умолчанию.
2. Весь чертёж может быть перемещён влево/вправо или вверх/вниз с помощью направляющих справа и внизу чертежа. Или можно щёлкнуть левой клавиши мышки по пустому месту на чертеже, удержать кнопку мышки и переместить чертёж мышкой.

3.5. Перемещение элементов

Элементы чертежа могут перемещаться с места на место следующими способами.

1. Щёлкните по элементу левой клавишей мышки и, не отпуская клавишу, переместите элемент в новое место.
2. Щёлкните по элементу правой клавишей мышки и выберите «Move Eelement» из выпадающего меню. Появится диалоговое окно, в котором вы можете задать число точек сетки, на которое хотите переместить элемент по осям x и y. Перемещение выполнится, когда вы нажмёте кнопку **ОК**.
3. Выделите элемент и используйте курсорные клавиши для перемещения.

    Чтобы вышеописанное работало, элемент должен быть «свободен», то есть, он не должен иметь соединений с другими элементами.

4. Используйте операции cut/paste, описанные выше, чтобы удалить элемент с предыдущего места и вставить его в новое. В этом случае, если элемент соединён с другими, все провода, соединяющие его, будут удалены автоматически.

3.6 Поворот и отражение элементов

Щёлкните по нужному элементу. Он будет подсвечен. В правом окне щёлкните по закладке «*Property Editor*». Для поворота, вам предлагается выпадающее меню в окошке «Rotation», где элемент можно повернуть на 0, 90, 180 и 270 градусов (по часовой стрелке). В качестве альтернативы можно использовать выделение элемента и клавишу r, каждое нажатие будет поворачивать элемент на 90 градусов по часовой стрелке. Для отображения элемента вы можете в окошке «Flip» в выпадающем меню использовать «Horizontal» и «Vertical» пункты для соответствующих отражений.

3.7 Маркировка элементов

Выделите элемент, щёлкнув по нему. Введите нужную маркировку в поле «Caption» редактора свойств. Можно заметить, что имена элементов не обязательны; GUI автоматически генерирует имена по умолчанию для каждого элемента, добавленного в схему.

3.8 Добавление текста в чертёж

Для целей документирования или представления полезно добавлять некоторый текст (например, маркировку и значения компонентов) к схеме на чертеже. Это можно сделать щелчком по кнопке «**Add Text Box**», которая переводит GUI в режим ввода текста. Щелчком по подходящему месту на чертеже вы вызываете окно для ввода текста, где можно выбрать шрифт, размер и ввести текст. Этот текст появится на экране. Поместите курсор на текст, появится полупрозрачный для остальной части чертежа прямоугольник. Вы может щёлкнуть по нему, чтобы перенести текст. Когда вы поместите его в нужное место, неплохо бы привести его размер к разумным пределам, чтобы он не мешал в дальнейшей работе (правый угол для этого отмечен, а курсор меняет вид). Также возможно сделать текст невидимым, если щёлкнуть по кнопке «**Toggle Text**». Добавленный текст подразумевает только удобство чтения чертежа пользователем; он игнорируется SEQUEL (поэтому текст можно ввести кириллицей, но проверить, не будет ли он мешать).

3.9 Экспорт схемы

Схема вместе с текстом, добавленным пользователем, если нужно может экспортироваться в таких форматах, как jpg или pdf. Выберите из пункта «View» основного меню в выпадающем списке «Snapshot», выберите подходящий формат, выберите директорию, куда следует записать файл, и введите имя файла. Для завершения операции щёлкните по «**Save**».

3.10 Соединение

Для соединения (заметьте, что соединяются только узлы одного типа, то есть, electrical, digital, general, temperature или thermal power) узла 1 с узлом 2 щёлкните по узлу 1. GUI перейдёт в режим соединения, что отмечается появлением направляющих. Переместите курсор при отпущенной клавише мышки в другое место на чертеже. Щелчок здесь создаёт «якорь» в этом месте. Продолжая таким же образом, завершите соединение щелчком по узлу 2. Режим соединения автоматически выключается. Соединение может проводиться между узлом элемента и некоторой точкой уже существующего соединения. Этим создаётся новое соединение такое же, как и старое (то есть, оба будут иметь одинаковое имя узла). Если в процессе соединения вам понадобится прервать его, нажмите **F4** и GUI выйдет из режима проведения соединений. Важно: размещение узла одного элемента поверх узла другого элемента НЕ создаёт соединения. Лучше разместить оба элемента рядом и провести соединение. Если уж нужно иметь два элемента с перекрывающимися узлами, удостоверьтесь (вручную редактируя раздел «Nodes» каждого элемента), что перекрывающиеся узлы принадлежат к одному и тому же имени узла.

3.11 Удаление соединения

Выделите соединение, щёлкнув по нему. Оно будет подсвечено. Теперь нажмите клавишу **Delete**. Заметьте, что удаление соединения удалит все связанные с узлом элемента другие соединения, а GUI автоматически сгенерирует новые имена узлов для соединённых узлов элемента.

3.12 Маркировка проводов (узлов)

Есть два пути дать имя проводу (соединению):

1. Выберите провод щелчком по нему. Щёлкните по закладке «*Property Editor*» в правом окне и отредактируйте имя узла здесь.
2. Выделите элемент (щелчком по нему), который соединён с нужным проводом. Щёлкните по закладке «*Property Editor*» в правом окне. Появятся свойства элемента с именем связанного с элементом узла, как одно из свойств. Отредактируйте имя здесь.

Следует заметить, что любые изменения в имени узла отразятся на всех элементах, соединённых с ним и также выходными переменными (если они есть), которые содержат имя этого узла.

3.13 Использование «Connectors» для соединения

В списке элементов вы найдёте элементы «connector»: connector_e.ece, connector_d.ece, connector_g.ece, connector_t.ece и connector_p.ece. Это «фальшивые» элементы, и SEQUEL их игнорирует. Но они полезны в следующих ситуациях:

1. Длинные соединения: проводя длинные соединения, вы можете уменьшить их до коротких сегментов, используя коннекторы. Для этого вместо соединения узла 1 непосредственно с узлом 2, присоедините узел 1 к коннектору, а другой коннектор к узлу 2.
2. Проход между пересекающимися и соединёнными проводами: если два провода пересекают один другой, они могут принадлежать или могут не принадлежать к одному узлу, в зависимости от того, где они соединяются. В этом случае коннектор может использоваться для обозначения физического соединения между двумя пересекающимися проводами. Это сделает схему более понятной.

3.14 Использование всплывающей подсказки

Нет необходимости щёлкать по элементу или узлу, чтобы выяснить его имя. Вы можете поместить курсор, не щёлкая клавишей мышки, поверх интересующего вас узла или элемента на некоторое время (секунду или около того), и его имя отобразится в окошке справа от курсора.

3.15 Общий узел

В схеме, содержащей электрические элементы, SEQUEL требуется, чтобы один из узлов был объявлен как общий узел. Это можно сделать, добавив ground.ece на чертёж и соединив эту «землю» с выбранным узлом, как показано на рисунке ниже.

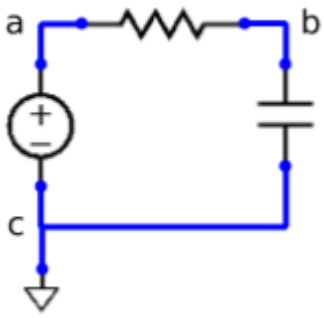

Рис. 5. Объявление, что узел «c» является общим с помощью ground.ece

3.16 Использование «фиктивной» земли

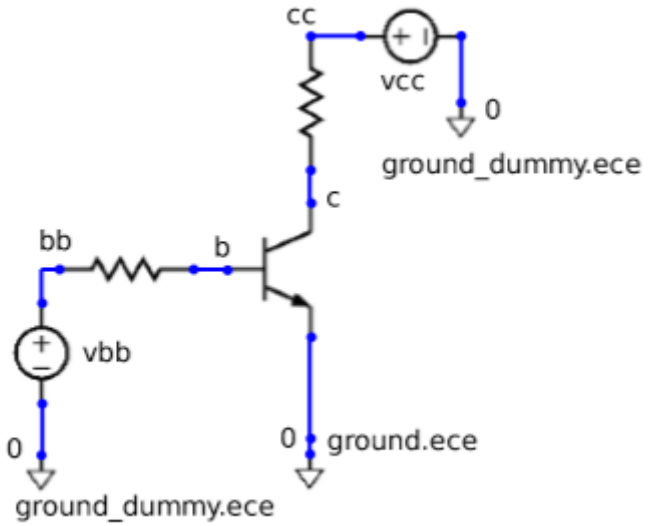

Рис. 6. Использование ground_dummy.ece для обозначения соединения с общим проводом

В некоторых схемах узлы многих элементов требуют соединения с общим проводом, что может загромождать схему. В этом случае мы можем просто отредактировать имя узла, как свойство элемента, вместо того, чтобы физически соединять их общим проводом. Пример показан на рисунке выше. Источник, обозначенный как vcc, имеет два узла, p и n. Мы можем присвоить n=0, сделав 0 именем общего провода в редакторе свойств этого элемента. Хотя этого достаточно для работы SEQUEL, нам не хотелось бы оставлять этот узел в подвешенном состоянии, и было бы неплохо обозначить, что он соединён с общим проводом. Это можно сделать с помощью элемента ground_dummy.ece (имеющего тот же символ, что ground.ece) на чертеже, который подключается к этому узлу, как показано на рисунке. Соответственно другой ground_dummy.ece должен быть подключен к источнику, названному vbb.

3.17 Расширение рабочего пространства

Для больших схем требуется больше места для черчения. Это можно сделать, изменив размер левого, правого и нижнего окон: щёлкните по кромке окна (курсор меняет вид) и измените его размер. Окна могут быть и закрыты, если щёлкнуть по иконке «close» (с крестиком) в заголовке окна. Если необходимо, каждое из окон может быть восстановлено выбором из «panels menu» в пункте выпадающего списка «View» основного меню, где устанавливается видимость окон message area, tool box или configuraion panel.

3.18 Выбор выходных переменных

Выходные переменные используются для записи результатов симуляции в выходные файлы. Есть два момента, связанные с этим процессом: определение интересующих вас переменных и передача симулятору этого определения для записи в выходной файл. О первом следует позаботиться в редакторе схемы, а о втором при создании блоков решения. В редакторе схемы мы определяем все выходные переменные, которые нам интересны. Когда выходные переменные определены, GUI создаёт присваивание:

`outvar_name=string1_of_string2`, где outvar_name – это имя выходной переменной, а выражение справа генерируется автоматически в зависимости от пользовательского выбора.

Например, если пользователь выбрал напряжение узла, названного vcc, тогда генерируется выражение

```
var1=nodev_of_vcc
```

Имя var1 (это могут быть var2, var3 и т.д.) – это имя по умолчанию генерируемое GUI, и оно может редактироваться пользователем после двойного щелчка по var1 и впечатывания нужного имени в этом месте. Для выбора выходной переменной щёлкните по закладке «*Output Variables*» в правом окне, а затем по «**Add Variable**». Курсор превратится в перекрестие, показывая, что GUI теперь в режиме добавления выходных переменных. Когда выходная переменная выбрана, режим добавления переменных автоматически выключается, а курсор возвращается к своему нормальному виду (вспомните о примечании относительно клавиши **ctrl** при добавлении переменных). Разные типы выходных переменных могут выбираться следующим образом:

1. Напряжение узла: щёлкните по нужному узлу (проводу) типа electrical.
2. Напряжение узла (переменный ток): щёлкните по нужному узлу (проводу) типа electrical, скажем, xyz. Вы увидите в правом окне появление nodev_of_xyz. Щёлкните правой клавишей мышки по нему и отметьте AC. Теперь строка изменится на nodev_ac_of_xyz, как этого требуют синтаксические правила SEQUEL.
3. Общая переменная: щёлкните по нужному узлу типа general.
4. Общая переменная (переменный ток): щёлкните по нужному узлу (проводу) типа general, скажем, узлу xyz. Вы увидите появление var_of_xyz в правом окне. Щёлкните правой клавишей мышки и установите AC. Строка изменится на var_ac_of_xyz, как того требуют правила синтаксиса SEQUEL.
5. Цифровая переменная: щёлкните по нужному узлу типа digital.
6. Выходная переменная, относящаяся к элементу. Для большинства элементов одно или несколько его внутренних свойств доступно в качестве выходных переменных. Они перечислены в его описании. Например, для r.ece вы найдёте, что в качестве выходных переменных доступны v1 и i1 (напряжение на нём и ток через него). Их можно выбрать в качестве выходных переменных на уровне схемы, щёлкнув по элементу типа r.ece и выбрав v1 или i1.

4 Раздел решения, общие замечания

Когда создание схемы завершено, нам нужно задать, какой тип анализа будет выполняться программой. В зависимости от схемы имеют значение только некоторые виды анализа, и только они будут выбраны. Например, в схеме с цифровыми элементами нет смысла проводить анализ на переменном токе (AC). Аналогично для машинного привода, где используются такие компоненты, как PI-контроллер, будет интересен анализ переходных процессов (transient), но никак не анализ на постоянном (DC) или переменном токе (AC). Ниже будет беглое описание некоторых видов анализа (SEQUEL поддерживает и анализ шумов, и анализ чувствительности, но они выходят за рамки этого описания), поддерживаемых SEQUEL.

4.1 Анализ на постоянном токе (DC analysis)

Анализ на постоянном токе подходит тогда, когда все интересующие нас переменные не зависят от времени. Если есть элементы, имеющие производные по времени (такие как индуктивности и конденсаторы), их производные по времени приравниваются нулю. Заметьте, что анализ на постоянном токе также охватывает ситуации, когда мы интересуемся вариацией некоторых

значений на постоянном токе (таких как напряжение или ток) относительно параметров цепи (как питающее напряжение или сопротивление). Примеры: вычисление рабочей точки усилителя на транзисторе, передаточные (выход зависит от входа) характеристики биполярных и полевых инверторов.

4.2 Начальный анализ (Start-up analysis)

Начальный анализ важен тогда, когда есть элементы, зависящие от производных по времени, и мы хотим знать решение при t=0. Обратите внимание на схему на рисунке ниже. В анализе на постоянном токе мы должны задать производную dVc/dt равной нулю, поэтому ток (= C dVc/dt) тоже эквивалентен нулю.

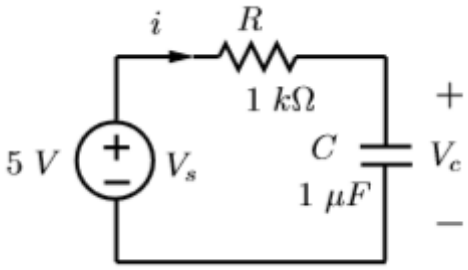

Рис. 7. Пример начального анализа

По этой причине отсутствует падение напряжения на резисторе, а полное напряжение приложено к конденсатору (5 В). Совсем иной предстаёт ситуация при начальном анализе. В этом случае мы имеем начальный параметр (назовём его V0) для конденсатора, которым задаём начальное напряжение на конденсаторе в момент t=0−. Если V0 =2 V, например, тогда условия при t=0− будут Vc =2 V и i = 3 mA, которые совсем иные, чем при анализе на постоянном токе. Начальный анализ обычно используется для задания начальной точки для анализа переходного процесса в соответствующем блоке решения. Для RC цепи на рисунке выше, например, результат симуляции переходного процесса (то есть, решения для t > 0) будет зависеть от начальных условий для цепи (с заданным значением V0), а начальный анализ должен быть выполнен перед анализом переходного процесса. Во множестве других интересующих нас ситуаций, однако, значения разных начальных параметров (как напряжение на конденсаторе и ток через индуктивность) не могут быть заданы заранее, и, следовательно, начальная симуляция частично не обоснована. В этих случаях мы можем начать блок анализа переходного процесса без блока начального анализа. Это выполняется с помощью задания initial_sol при инициализации. В качестве примера рассмотрим реакцию машинного привода при скачке расчётной скорости (см. рис. 8). В этом случае мы применим анализ переходного процесса (без применения начального анализа) для подходящего временного интервала в плане достижения устойчивого состояния, а затем приложим скачок расчётной скорости.

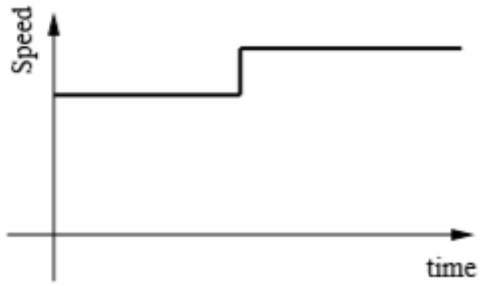

Рис. 8. Пример задания скорости привода

4.3 Анализ переходного процесса

Анализ переходного процесса требуется тогда, когда интересующие нас переменные меняются со временем. Начальная точка для блока анализа переходного процесса будет получена из предыдущего блока решения (установкой initial_sol к предыдущему или файловому значению) или использованием внутренне сгенерированных начальных условий (установкой initial_sol к initialize). В переходном процессе уравнения цепи решаются в нескольких временных точках, а интересующие нас запоминаются в файле для каждой временной точки (или для фиксированных временных интервалов, заданных пользователем). Методы доступные в анализе переходного процесса могут быть в общих чертах разделены на две категории: методы, использующие постоянный шаг времени, и методы, использующие переменный (автоматически рассчитанный) шаг времени. Основой методов автоматических шагов будет использование маленьких временных шагов только тогда, когда это нужно, и больших временных шагов в остальных случаях, чем сокращается время на расчёты.

4.4 Анализ установившегося состояния (SSW)

Анализ установившегося состояния может рассматриваться в качестве особого случая анализа переходного процесса. SSW анализ полезен, когда мы интересуемся только периодической установившейся реакцией цепи, то есть, только одним периодом в установившемся состоянии. Рассмотрим цепь на рис. 9a, где f = 50 Гц, иными словами T = 20 мс. Для достижения условий установившегося состояния для этой цепи мы должны начать с самого начала (скажем, с начального условия Vc =0 V) и решить уравнения цепи для нескольких циклов, пока все решения не установятся в периодически устойчивое состояние. Этот подход оказывается неэффективным с точки зрения времени вычислений, поскольку мы можем симулировать сотни, если не тысячи циклов, пока будет достигнуто устойчивое состояние. В действительности мы можем не заботится о результатах этих сотен циклов, но только о том, что случится тогда, когда будет достигнуто устойчивое состояние. SSW делает возможным достичь информации об устойчивом состоянии без выполнения долгого начального переходного процесса. Суть метода показана на рис.9b, где показан результат состояния переменной Vc от t=0 до t=T. Если мы начнём с Vc(0)=0 V, численное решение уравнений цепи даст результат Vc(T) = 0.12 V (зелёная кривая). Vc(0) = −0.4 V результат при Vc(T) = 0 V (синяя кривая) и т.д.

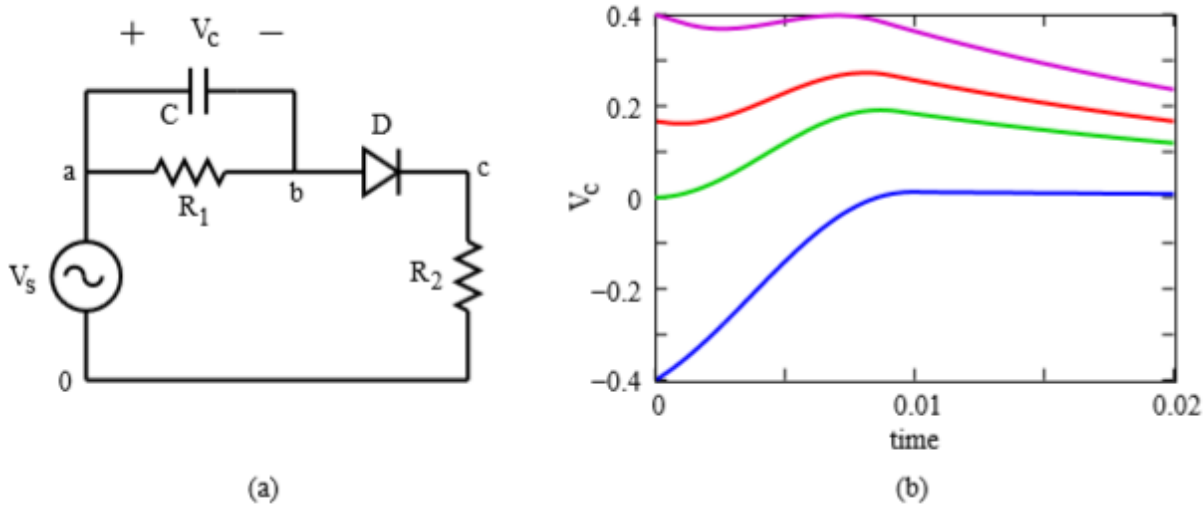

Рис. 9. Иллюстрация SSW анализа: (a) схема, (b) результат для Vc

Если предположить, что Vc(T) заканчивается, будучи эквивалентно Vc(0), мы получим нужное решение, поскольку это в полной мере отвечает условию для периодического установившегося состояния. Если есть другие установившиеся переменные в цепи (как токи через индуктивности и напряжения на конденсаторах), все они должны удовлетворять условию периодичности, то есть, Xi(0)=Xi(T), где Xi – это переменная состояния. Смысл SSW анализа в получении множества X1(0), X2(0)… такого, что после решения уравнений цепи для одного периода, мы завершаем процесс с X1(T)=X1(0), X2(T)=X2(0) и т.д. Это достигается применением метода Ньютона-Рафсона [1]-[5], как показано в «SSW N-R loop» на рисунке.

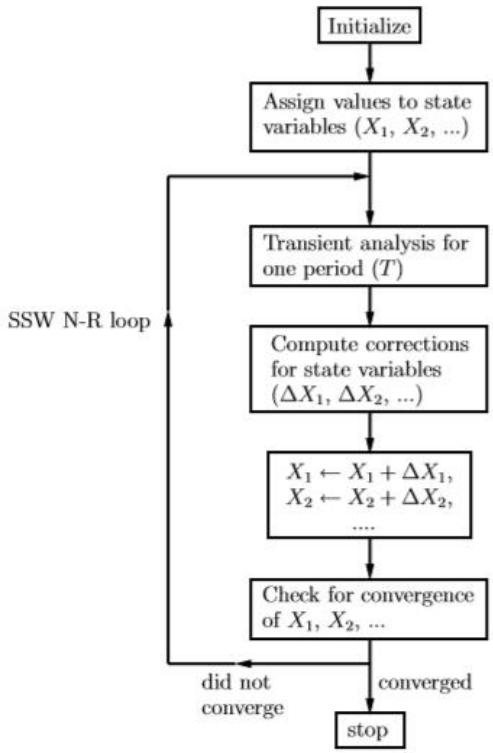

Рис. 10. Алгоритм SSW анализа

Заметьте, что если цепь нелинейная, блок анализа переходного процесса будет влечь за собой отдельный N-R цикл (для каждой временной точки). «Внутренние» N-R итерации (в блоке анализа переходного процесса) и «внешние» N-R итерации (в SSW loop) управляются разными параметрами метода, как мы увидим позже.

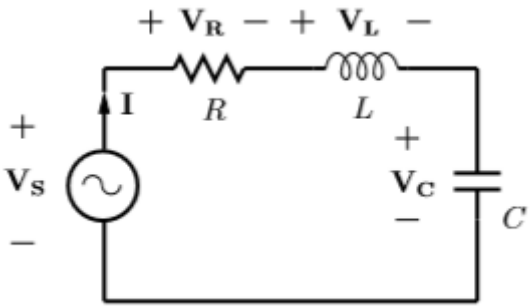

Рис. 11. Последовательная RLC цепь

4.5 Анализ на переменном токе (AC)

Анализ на переменном токе должен выполняться тогда, когда нас интересуют частотные характеристики цепи или системы. Анализ на переменном токе подразумевает вычисления с комплексными числами (phasors, векторы) и может быть классифицирован следующим образом:

1. Компоненты, не зависящие от смещения: рассмотрим цепь на рис. 11. Здесь для заданной частоты ω импеданс для R, L и C может быть вычислен как R, jωL и −j/ωC, и они не зависят ни от каких-либо смещений на постоянном токе, которые может иметь источник. В этом случае мы можем выполнить анализ на переменном токе непосредственно, то есть, без предварительного анализа на постоянном токе.
2. Зависящие от смещения компоненты: рассмотрим усилитель с общим эмиттером, показанный на рис. 12a. В схеме есть нелинейный элемент, собственно биполярный транзистор, чьё мало-сигнальное поведение (AC) зависит от рабочей точки, то есть, состояния на постоянном токе. В частности, если используется π эквивалентная схема для модели BJT в мало-сигнальной ситуации (см. рис. 12 b), тогда gm = IC/VT и rπ = β/gm зависят от постоянного тока коллектора Ic. Ясно, что не было бы смысла выполнять анализ на переменном токе напрямую; AC solve блок должен быть предварён блоком DC solve так, чтобы зависящие от смещения величины в мало-сигнальной цепи получили расчёты от симулятора.

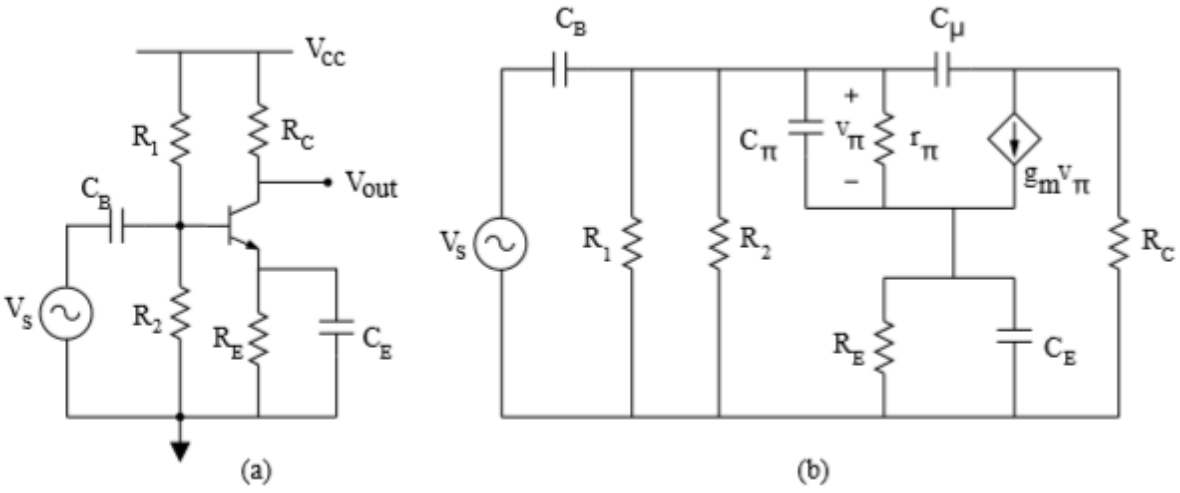

Рис. 12. Усилитель с общим эмиттером: (a) схема и (b) мало-сигнальная эквивалентная цепь

5 Разделы решения, N-R параметры

Нелинейные уравнения могут возникнуть в разного типа анализах: DC, start-up, transient и SSW. SEQUEL подобно большинству других симуляторов электрических цепей использует метод Ньютона-Рафсона (N-R) для решения нелинейных уравнений. Метод N-R управляется следующими параметрами:

5.1 Максимальное число итераций

`* itmax_newton (optional, default=150)` используется для задания максимального числа N-R итераций. Если N-R процесс не сходится в itmax_newton итерациях, тогда, во-первых, в DC или start-up анализе программа останавливается и отображается сообщение об ошибке. Во-вторых, в анализе переходных процессов или SSW анализе программа может остановиться или продолжиться повторением N-R процесса с меньшим временным шагом, в зависимости от метода обозначенного пользователем (будет обсуждено позже).

5.2 Критерий сходимости

N-R процесс называется сходящимся, когда удовлетворяются некоторые критерии сходимости. SEQUEL допускает, чтобы удовлетворялись следующие нормы (критерии).

`* norm_2 (optional, default=1e-10)`, относящаяся к

$$\epsilon_{\mathrm{RHS(2)}} = \frac{1}{N}\sqrt{\sum_{i=1}^{N} f_i^2}\,. \qquad (1)$$

Система уравнений будет решена при fi = 0; так модуль fi представляет отклонение от нуля и является мерой ошибки.

`* delxmax_kcl (optional, default=1e-9)`, относящаяся к

$$\epsilon_{\mathrm{KCL}} = \max \text{ over all nodes} \left(\sum_k i_k \right). \qquad (2)$$

Другими словами, в *каждом* узле мы вычисляем сумму всех токов, втекающих в узел, а затем берём максимум по всем узлам. €KCL должно быть в идеале нулём. `delxmax_kcl` – это отклонение (в амперах) от нуля, которое мы готовы терпеть. Заметьте, что `delxmax_kcl` существенно только тогда, когда есть электрические элементы цепи.

`* delxmax_volt (optional, default=1e-4)` относится к

$$\epsilon_{\Delta V} \;=\; \text{max over all nodes}\,(\Delta V_i)\;, \tag{3}$$

где ΔVi (в узле i) – это изменение напряжения узла от одной N-R итерации к другой. Как только N-R процесс сходится, все значения ΔV будут приближены к нулю. €ΔV – это отклонение (в вольтах) от нуля, которое мы готовы терпеть. Заметьте, что `delxmax_volt` существенно только тогда, когда в цепи есть электрические элементы.

`* delxmax_tmpr (optional, default=0.1)` относится к

$$\epsilon_{\Delta T} \;=\; \text{max over all temperature nodes}\,(\Delta T_i)\;, \tag{4}$$

где ΔTi (в температурном узле i) – это изменение температуры (в градусах Цельсия или Кельвина) от одной N-R итерации к другой. Как только N-R процесс сходится, все значения ΔV будут приближены к нулю. €ΔT – это отклонение от нуля, которое мы готовы терпеть. Заметьте, что `delxmax_tmpr` существенно только тогда, когда в цепи есть термоэлементы.

`* delxmax_pwr (optional, default=1e-7)` относится к

$$\epsilon_{\Delta P} \;=\; \text{max over all power nodes}\,(\Delta P_i)\;, \tag{5}$$

где ΔPi (в узле мощности i) – это изменение температурной мощности (в ваттах) от одной N-R итерации к другой. Когда N-R процесс сходится, все значения ΔP будут приближены к нулю. €ΔP – это отклонение от нуля, которое мы готовы терпеть. Заметьте, что `delxmax_pwr` существенно только тогда, когда в цепи есть термоэлементы.

Какие критерии сходимости будут проверяться симулятором? Это зависит от дополнительных утверждений, как следующие.

```
* chk_rhs2=yes (no) for checking (not checking) norm_2 (default=yes)

* chk_delx_kcl=yes (no) for checking (not checking) delxmax_kcl (default=no)

* chk_delx_volt=yes (no) for checking (not checking) delxmax_volt
(default=no)

* chk_delx_tmpr=yes (no) for checking (not checking) delxmax_tmpr
(default=no)

* chk_delx_pwr=yes (no) for checking (not checking) delxmax_pwr (default=no)
```

В качестве примера представьте, что пользователь задал, что критерий norm_2 должен быть проверен SEQUEL (установкой флага chk_rhs2=yes), и norm_2 было задано как 1e-9. В этом случае программа вычислит значение €RHS(2) (см. уравнение 1) в каждой N-R итерации. Если оно меньше, чем 10^{-9}, N-R процесс называется сходящимся; если нет, программа попытается провести другую

N-R итерацию, вплоть до максимума `itmax_newton` итераций. Иногда полезно записать одну или более норм, описанных выше, в консоль (она появится в нижнем окне GUI, когда выбрана закладка «Solver Output»). В частности, если N-R процесс перестал сходиться, информация о норме может помочь в решении проблемы сходимости. Следующие флаги могут использоваться для выбора норм для записи в консоль.

```
* write_rhs2=yes (no) for writing (not writing) RHS(2) as defined in Eq. 1
(default=no)

* write_delx_kcl=yes (no) for writing (not writing) KCL as defined in Eq. 2
(default=no)

* write_delx_volt=yes (no) for writing (not writing) ΔV as defined in Eq. 3
(default=no)

* write_delx_tmpr=yes (no) for writing (not writing) ΔT as defined in Eq. 4
(default=no)

* write_delx_pwr=yes (no) for writing (not writing) ΔP as defined in Eq. 5
(default=no)
```

Поскольку SEQUEL – это программа для общих целей, допускающая разные типы переменных, таких как electrical, general и thermal, norm_2 задаётся общим ограничением $\frac{1}{N}\sqrt{\sum_{i=1}^{N} f_i^2}$,

то есть, вычисление касается функции fi всех типов (electrical, general и thermal). Следует запомнить, что минимум norm_2 достижим в зависимости от «шкалы» проблемы. Рассмотрим цепь, показанную на рис. 13, где R – это обычный резистор, а R1 – нелинейный элемент, удовлетворяющий:

$$i = k_1 i + k_2 i^2 + k_3 i^3 . \tag{6}$$

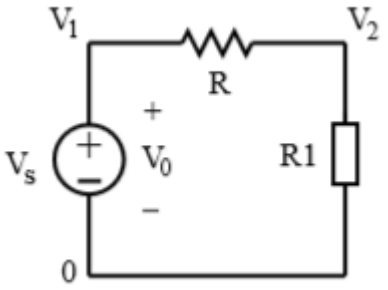

Рис. 13. Пример нелинейной цепи

Давайте рассмотрим два набора параметров (V0 = 4 остаётся постоянным).

(a) R=1 Ω, k1 =k2 =k3 =1 и

(b) R=1 kΩ, k1 =k2 =k3 =10−3,

где k1, k2, k3 в A/V , A/V 2, A/V 3 соответственно. Второй случай – это просто масштабированная версия первого, а решение (в терминах узлового напряжения) одинаково в обоих случаях: V2 = 1V. Метод N-R даёт следующие результаты в случае (a):

```
1 norm_rhs_2= 0.444444444          v2=0.000000000E+00
2 norm_rhs_2= 1.51111111           v2=0.200000000E+01
3 norm_rhs_2= 0.370467194          v2=0.135849057E+01
4 norm_rhs_2= 0.0562189282         v2=0.106553526E+01
5 norm_rhs_2= 0.00219068161        v2=0.100265975E+01
6 norm_rhs_2= 3.76580074E-06       v2=0.100000458E+01
7 norm_rhs_2= 1.1187525E-11        v2=0.100000000E+01
8 norm_rhs_2= 0.                   v2=0.100000000E+01
```

Для случая (b) метод даёт:

```
 1 norm_rhs_2= 0.444444444         v2=0.000000000E+00
 2 norm_rhs_2= 0.00151111111       v2=0.200000000E+01
 3 norm_rhs_2= 0.000370467194      v2=0.135849057E+01
 4 norm_rhs_2= 5.62189282E-05      v2=0.106553526E+01
 5 norm_rhs_2= 2.19068161E-06      v2=0.100265975E+01
 6 norm_rhs_2= 3.76580074E-09      v2=0.100000458E+01
 7 norm_rhs_2= 1.11874622E-14      v2=0.100000000E+01
 8 norm_rhs_2= 1.41534746E-17      v2=0.100000000E+01
 9 norm_rhs_2= 1.41534746E-17      v2=0.100000000E+01
10 norm_rhs_2= 1.41534746E-17      v2=0.100000000E+01
```

После пяти итераций решение совпадает до третьего знака после запятой в обоих случаях (см. колонку V2). Однако в случае (a) norm_2 фактически достигает нуля в 8й итерации, пока она остаётся конечной и имеет тенденцию к переполнению (norm_2 часто переполняется в некоторой точке из-за конечной точности компьютера; иначе можно произвольно уменьшать норму и увеличивать количество итераций; это же и о других нормах) около 1.4×10−17. Если мы задаём критерий norm_2 и задаём значение norm_2 как 1×10−18, сходимость достижима для случая (a), но не для случая (b). Однако эта точность много больше, чем требуется нам, поскольку решение (v2) уже достаточно точно при пятой итерации. Если при симуляции появляется сообщение «convergence not reached» не стоит паниковать, нужно уменьшить допустимое отклонение и попробовать ещё раз. Мы видели, что разность между следующими значениями v2 становится меньше, так как N-R процесс сходится. Мы могли бы использовать эту разность в качестве критерия сходимости. Это можно сделать установкой chk_delx_volt=yes. Максимально приемлемое значение (delxmax_volt) может быть задано подходящим небольшим числом, скажем, 0.1m. Каков наилучший выбор для критерия сходимости? В чисто электрической цепи delxmax_kcl и delxmax_volt могут устанавливаться обоснованно (заметим, что можно задавать множественные критерии сходимости; например, при задании chk delx kcl=yes и chk delx volt=yes программа использует выражения (2) и (3) для решения о достижении сходимости). Например, если мы знаем, что токи в цепи в диапазоне миллиампер, мы можем задать delxmax_kcl=1n (1 nA). Однако если цепь имеет разные типы элементов (как электрический привод для мотора, PI контроллер и т.д.), наиболее подходящим выбором была бы norm_2, которая учитывается всеми типами функций. Хотя значение по умолчанию (1e-10) работает хорошо во многих случаях, выбор значения для norm_2 иногда может быть затруднён. Если N-R процесс прерывается из-за расходимости с каким-то значением norm_2, полезно записать norm_2 в консоль (установкой write_rhs2=yes) и решиться на подходящее значение. Часто помогает, если взглянуть на существующие примеры. Есть шансы, что вы найдёте похожий пример (в смысле элементов и связанных значений) на тот, что вы пытались симулировать, а критерии сходимости, заданные в этом примере, могут работать и с вашей схемой. И, наконец, заметьте, что с линейными цепями

нет вопросов по сходимости, поскольку решение получается из единственного шага обратного преобразования матрицы без использования N-R процесса.

5.3 Демпфирование N-R итераций

Наиболее полезный и известный аспект N-R метода – это его квадратичная сходимость («квадратичная» сходимость означает, что ошибка спадает квадратично; например, если ошибка имеет порядок 10−3 в текущей итерации, то она станет порядка 10−6 в следующей). Однако это не всегда устойчиво: если начальное приближение не «закрыто» для решения, N-R процесс может не сходиться. Демпфирование N-R процесса можно использовать в подобных случаях для поддержания сходимости. Основная идея демпфирования проста. Вместо обновления вектора решения, как

$$\mathbf{x}^{(n+1)} \leftarrow \mathbf{x}^{(n)} + \Delta\mathbf{x}^{(n)}, \qquad (7)$$

мы обновим его как

$$\mathbf{x}^{(n+1)} \leftarrow \mathbf{x}^{(n)} + k\,\Delta\mathbf{x}^{(n)}, \qquad (8)$$

где k – это «демпфирующий фактор» между 0 и 1. Другими словами, вместо использования всей коррекции Δx(n) мы используем только её часть, то есть, k Δx(n). Рисунки 14 и 15 иллюстрируют, как демпфирование может использоваться для успешной сходимости, когда стандартный метод отказывает. Рисунок 14 показывает, что стандартный N-R метод не сходится с начальным приближением, x(0) =1.5. С другой стороны, с демпфированием N-R метод сходится с тем же приближением (рис. 15).

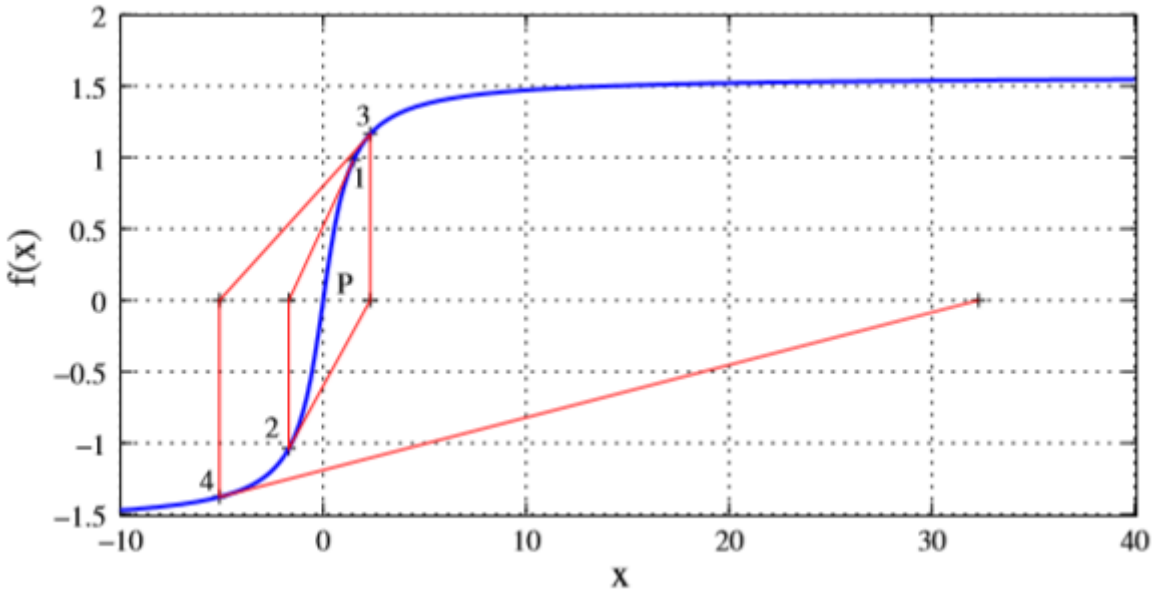

Рис. 14. Применение N-R метода к f(x) = tan−1 x = 0, где x = 1.5, как начальное приближение.

Заметьте, что N-R метод не сходится в этом случае.

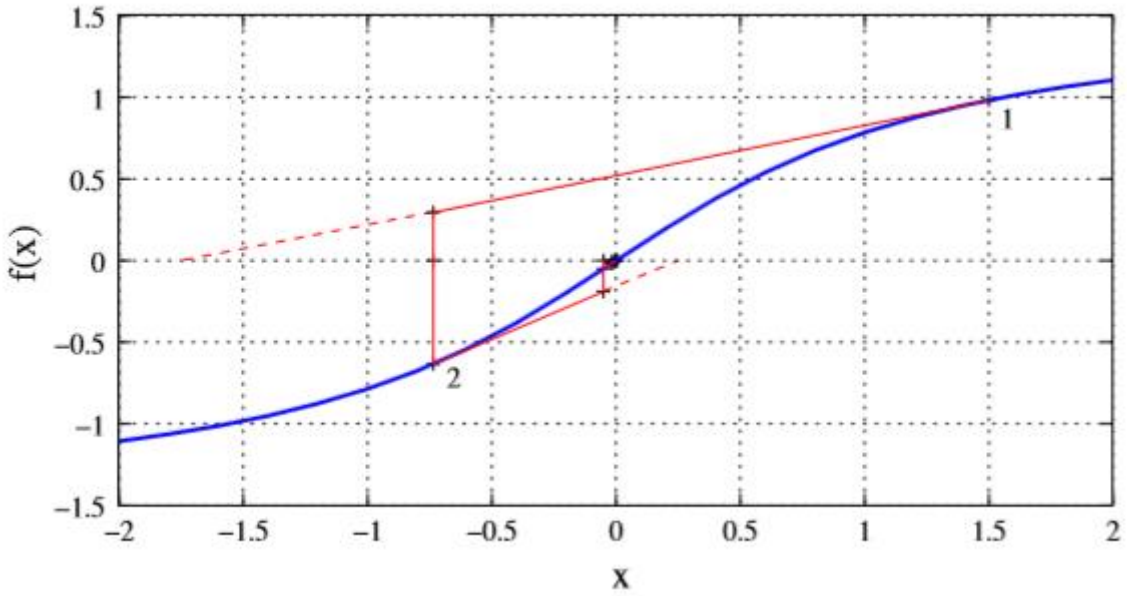

Рис. 15. Применение N-R метода к f(x) = tan−1 x = 0, где x = 1.5, как начальное приближение, а демпфирующий фактор k = 0.7.

Хотя демпфирование вносит улучшение в сходимость, оно понижает уровень сходимости. Демпфирование, таким образом, следует использовать тогда, когда отказывает стандартный N-R метод. Хотя даже, если демпфирование полезно, оно может применяться только для нескольких первых итераций. Это поможет закрыть решение, и, когда это происходит, можно успешно применить стандартный N-R метод. Из соображений такого сценария следующие параметры могут задаваться в SEQUEL блоках решения для использования демпфированного N-R метода.

* `dmp=yes (no)` для N-R метода с (без) демпфированием (default=no)

* `dmp_k` (optional, default=0.8): значение k из (8)

* `dmp_newt_max` (optional, default=10): количество N-R итераций, для которых применяется демпфирование. После dmp_newt_max итераций используется стандартный N-R метод, общее количество итераций (демпфированных+стандартные) будет задано через itmax_newton.

Параметры dmp_k и dmp_newt_max действительны только тогда, когда dmp установлено в yes.

5.4 gmin продвижение

Эффективный путь получить сходимость N-R итераций в так называемом «gmin stepping». Что основано на том факте, что N-R метод, скорее всего, сходится, когда начальное приближение близко к решению. Рассмотрим усилитель на биполярном транзисторе, показанный на рис. 16 (для упрощения не показаны разделительные конденсаторы; в любом случае они представляют обрыв на постоянном токе). Уравнения модели для BJT – exp(VBE/VT) and exp(VBC/VT), когда используется плохое начальное приближение. Это та ситуация, в которой использование gmin stepping может оказаться полезным

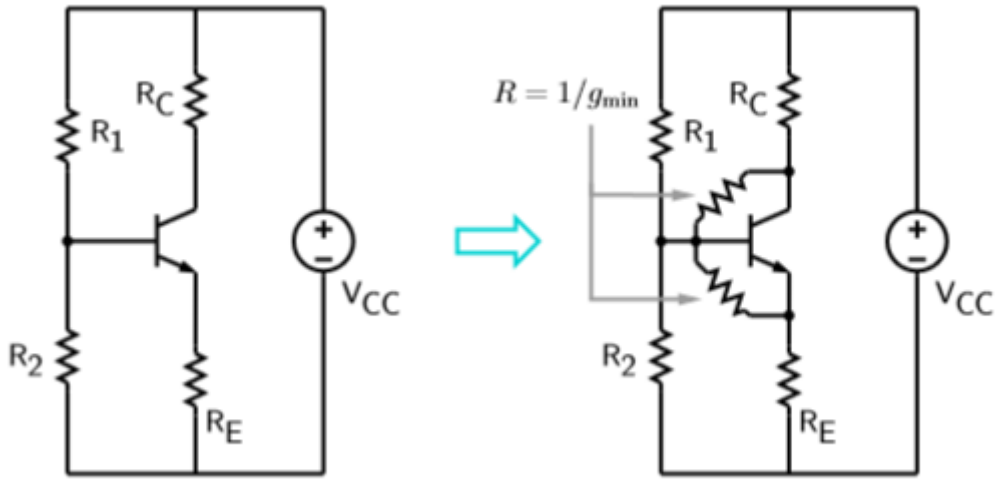

Рис. 16. Пример, показывающий использование gmin stepping

В gmin stepping резисторы с R = 1/gmin (отсюда название gmin stepping) добавлены, как показано на рисунке. Если сопротивление мало, BJT эффективно обходится, а цепь становится линейной, приводя к быстрой сходимости N-R итераций. Решение, полученное таким образом, используется для начального приближения к новой проблеме, в которой значения сопротивлений (для вновь добавленных резисторов) увеличиваются от прежних значений. Этот процесс повторяется, пока проходные сопротивления не становятся очень большими (то есть, 1012 Ом), и становятся эффективны открытые цепи, и так получено решение для оригинальной проблемы. Следующие параметры определяют gmin продвижения в SEQUEL:

* `gmin_step=yes (no)`, если gmin stepping будет (нет) использовано (default=no)

* `gmin_start` задаёт начальное значение gmin (default=1Ω−1)

* `gmin_end` задаёт конечное значение gmin (default=1×10−12 Ω−1)

* `gmin_npoints` задаёт количество gmin шагов, которые будут использованы при прохождении (логарифмически) от gmin_start до gmin_end. (default=50)

Параметры gmin_dmp, gmin_dmp_k, gmin_dmp_newt_max, gmin_itmax_newton имеют то же значение, что и dmp, dmp_k, dmp_newt_max, itmax_newton соответственно, о которых мы говорили раньше, исключая то, что они используются только во время gmin stepping процедуры.

6 Схемы, включающие цифровые элементы

Цифровые (логические) элементы, такие как вентили и триггеры, поддерживаются SEQUEL совсем не так, как аналоговые элементы. Для цепей с аналоговыми элементами задача в том, чтобы составить уравнения цепи и решить их в одном действии обратного преобразования (если проблема линейна) или в нескольких N-R итерациях (если проблема не линейна). Итог этого выполнения, то есть, решение – это вектор действительных чисел, представляющий узловое напряжение, ток в ветви и т.д. Для цифровых цепей, с другой стороны, мы не интересуемся актуальными (аналоговыми) значениями напряжения; нас интересует только, будет ли это высокий («1») или низкий («0») уровень. Это требует совсем другого – и вычислительно более простого – подхода.

6.1 Анализ переходного процесса

Наиболее эффективный путь (и один из применяемых в SEQUEL) для анализа переходных процессов схем, включающих цифровые элементы, это так называемая «event-driven, управляемая событиями» симуляция, в которой мы наблюдаем за входом каждого цифрового элемента и отражаем «события» (0 → 1 или 1 → 0) на его выходе только тогда, когда изменения ожидаются, как результат изменения его входа(ов). Интервал времени между изменениями входа(ов) элемента и событиях в его выходном узле(ах) определяется значением задержки для конкретного элемента. Анализ переходных процессов цифровой схемы выполняется планированием событий и обработкой их в плановые времена. В качестве примера рассмотрим схему на рис. 17 [6]. Изначально A=0, B=0 и C=1. Для t > 0 входы изменяются так, как показано на рис. 18. При t = 0, D=E=1 и F=0. При t = t1 один из входов (A) изменился.

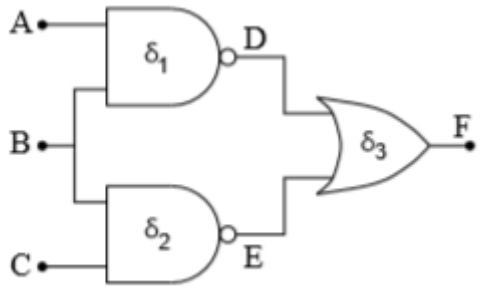

Рис. 17. Простая цифровая цепь, задержка каждого вентиля отмечена

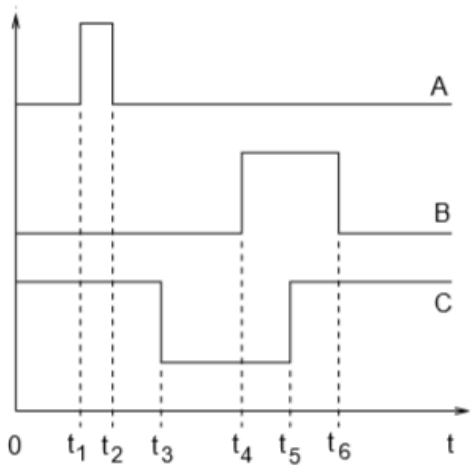

Рис. 18. Воздействия, приложенные к входам цифровой цепи на предыдущем рисунке

Теперь следует найти ответы на следующие вопросы: (a) какие вентили будут затронуты этими «событиями»? (b) они действительно станут причиной изменения каких-либо переменных? Если так, то на какой момент времени эти изменения («события») будут запланированы? Получается, что события в t1 не сказываются на выходе вентиля (узел D), поскольку его второй вход всё ещё 0. Фактически, в этом примере нет событий, которые должны планироваться до t5. Поскольку для перехода в t5 узел E должен измениться с 1 на 0. Это изменение или «событие» должно быть спланировано на t = t5 + δ2, где δ2 – это задержка для вентиля. Когда событие узла E выполняется или «обработано», вентиль OR должен быть проверен на возможное изменение его выхода, поскольку узел E обслуживает входной узел вентиля OR. Ясно, что обработка цифровых элементов

полностью отлична от обработки аналоговых элементов, и в отношении вычислений всё упрощается, становится почти тривиально.

Два основных отличия между симуляцией аналоговых и цифровых элементов выявляются в обсуждении выше: (a) когда привлекается аналоговый элемент, возникает необходимость в решении системы уравнений. С цифровыми элементами достаточно присвоения логических значений. Таким образом, при одинаковой «сложности» (при одинаковом количестве узлов) требуемое время CPU на один временной шаг будет получаться много больше для аналоговых элементов. (b) симуляция цифровых элементов – это «event-driven» [7], [8], то есть, данный цифровой элемент нуждается только в обработке, если один или более его входов изменились.

6.2 DC и Start-Up анализ

Симуляция DC (или start-up) другая. Здесь, поскольку мы ищем решение только в одной временной точке, нет необходимости в планировании или обработке событий. Вместо этого всё, что требуется – это получить значения всех цифровых переменных так, чтобы они согласовывались друг с другом. Рассмотрим пример на рис. 19.

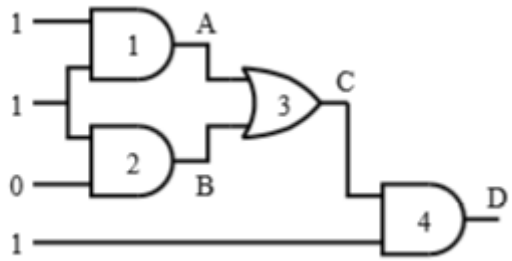

Рис. 19. Цифровая схема с постоянными (DC) входами

В качестве отправной точки, скажем, мы назначим для выходов каждого вентиля (то есть, узлы A, B, C, D) будут 0. При этих условиях мы выполним следующие «проходы».

Проход 1: мы полагаем, что узлы A и B должны быть 1 и 0, соответственно. Заметьте, что эта изменённая информация не доступна для вентиля OR при этом проходе.

Проход 2: мы полагаем, что узел C должен быть 1.

Проход 3: мы полагаем, что узел D должен быть 1.

Таким образом, в конце этих трёх проходов мы получили решение, которое совместимо с логическим поведением всех элементов. Этот процесс управляется следующими параметрами.

* `n_pass_dgtl` (optional, default=20): максимальное число проходов, которые нужно сделать, чтобы получить DC или start-up решение для схемы, включающей цифровые элементы.

В смешанных схемах с аналоговыми и цифровыми элементами значения цифровых переменных должны быть согласованы в отношении цифровых и аналоговых элементов. Рассмотрим схему на рис. 20 (A-to-D и D-to-A элементы, требующие преобразования сигналов в нужный тип, не показаны на рисунке). Перед тем, как начать анализ переходного процесса этой цепи, мы постараемся получить начальное решение.

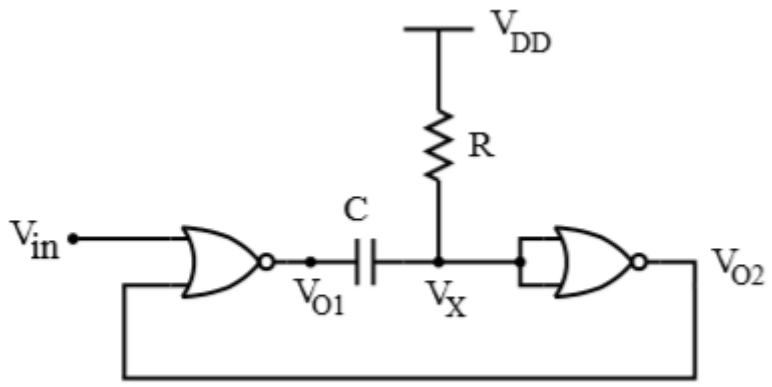

Рис. 20. Схема со смешанными сигналами

Значение V_{01}, например, должно быть таким, чтобы оно удовлетворяло логическому поведению вентиля NOR и аналоговому решению для RC цепи. То же относится и к другим переменным. Для анализа на постоянном токе или начального анализа на рис. 21 показан алгоритм, реализованный в SEQUEL. Теперь мы имеем внутренние проходы для получения решения для цифровых переменных (X) и внешние проходы, чтобы убедиться, что аналоговые переменные (x) также согласованы с цифровыми переменными (X). Параметр n_pass_mixed управляет внешними проходами:

* `n_pass_mixed` (optional, default=20): максимальное число внешних проходов (см. рис. 21), которое нужно сделать для получения DC или start-up решения для схемы, включающей цифровые и аналоговые элементы. Заметьте, что внутренние проходы в этом случае всё ещё управляются параметром n_pass_dgtl, который был рассмотрен ранее.

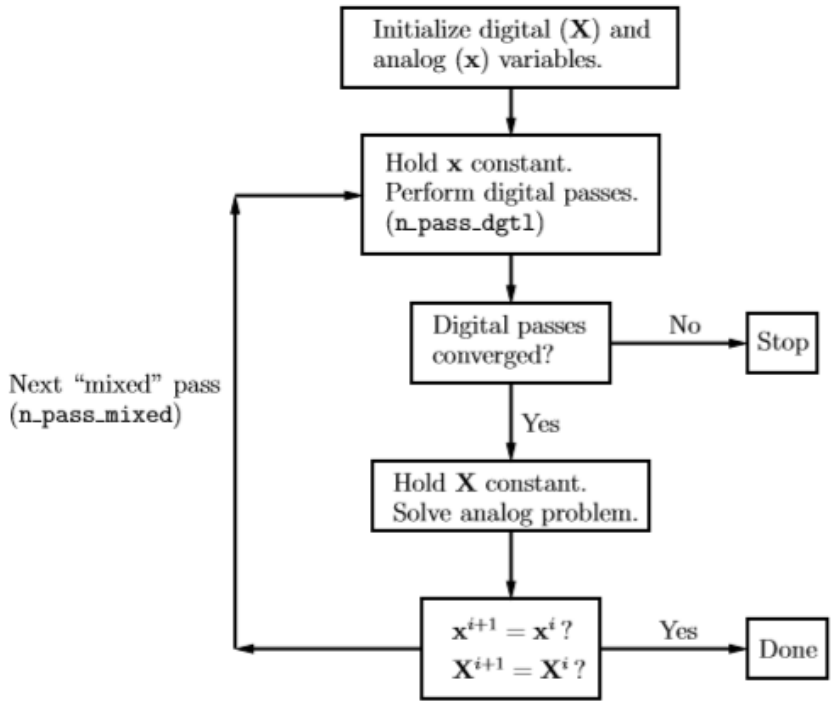

Рис. 21. Алгоритм для смешанного анализа в условиях DC или start-up

7 Синтаксис блока решения

Назначение блоков решения в передаче некоторых уведомлений симулятору, таких как: (a) какой анализ следует применить к схеме; (b) какой метод следует использовать; (c) значения параметров метода; (d) какие переменные следует сохранить и т.д. В этом разделе мы обсудим подобные детали, начиная с карты метода, а затем поговорим об остальном.

7.1 Карта метода

В карте метода можно отобразить большое количество параметров, и она выглядит слегка пугающе для начинающего. Однако только часть из них будет применяться в заданном анализе. Кроме того, их значения по умолчанию зачастую достаточно хороши, чтобы не требовать внимания пользователя. В дальнейшем мы перечислим параметры метода для каждого типа анализа и опишем их назначение.

7.1.1 DC/start-up анализ

* `N-R parameters`: как описано в разделе 5. Заметьте, что это относится только к нелинейным проблемам.

* `n_pass_dgtl и n_pass_mixed`: как описано в разделе 6.2. Заметьте, что это относится только к случаю наличия цифровых элементов в схеме.

7.1.2 AC анализ

Для анализа на переменном токе нет параметров метода, поскольку проблемы AC всегда линейны (как мы видели в разделе 4, даже нелинейные цепи линеаризуются для мало-сигнального AC анализа). Таким образом, есть только один метод получить решение, то есть, решение матрицы уравнений с комплексными переменными. Нам не нужно задавать частоту (частоты), требуемую для анализа. Это делается установками set_freq или vary_freq, которые мы обсудим позже.

7.1.3 Transient анализ

В анализе переходного процесса нам следует побеспокоиться о двух (более или менее независимых) проблемах:

1. Решение уравнений в заданной точке времени: это похоже на DC случай, а N-R параметры из раздела 5 прекрасно здесь подходят, если проблема нелинейная.
2. Дискретизация времени: это связано с тем, что интересующий нас временной интервал делится на более мелкие интервалы, а какой метод используется для дискретизации по времени, определяется задачей. Следующие опции метода могут использоваться в этом случае.

* `back_euler=yes` (no) для использования (не использования) обратного метода Эйлера с постоянным шагом по времени (default=no)

* `back_euler_auto=yes` (no) для использования (не использования) обратного метода Эйлера с автоматическим шагом по времени (default=no)

* `trapezoidal=yes` (no) для использования (не использования) метода трапеций с фиксированным шагом по времени (default=no)

* `trapezoidal_auto=yes` (no) для использования (не использования) метода трапеций с автоматическим шагом по времени (default=no)

* `gear2=yes` (no) для использования (не использования) метода Гира второго порядка с фиксированным шагом по времени (default=no)

* `trbdf2=yes` (no) для использования (или нет) метода Trapezoidal/BDF2 (default=no)

Если пользователь не задаёт какой-то метод для анализа переходного процесса, применяется метод трапеций.

Параметры, не зависящие от метода: эти параметры применимы ко всем методам.

* `t_start` (required, no default): время начала анализа переходного процесса

* `t_end` (required, no default): время окончания анализа переходного процесса

* `delt_const` (required, no default): в методе с постоянным шагом по времени (back_euler, trapezoidal и gear2) delt_const задаёт постоянную шага по времени Δt. В методах с автоматическим шагом по времени (back_euler_auto, trapezoidal_auto и trbdf2) delt_const определяет начальное значение Δt, последующие шаги по времени автоматически вычисляются симулятором.

* `delt_min` (optional, default=0.0002×delt_const) задаёт минимальный шаг по времени, который применит симулятор. Заметьте, что этот параметр относится даже к методам с постоянным шагом по времени, поскольку некоторые элементы могут заставить применить временные точки в интервале меньше, чем delt_const и delt_min, а затем определяется минимальный временной шаг, принимаемый симулятором. Пример показан на рис. 22.

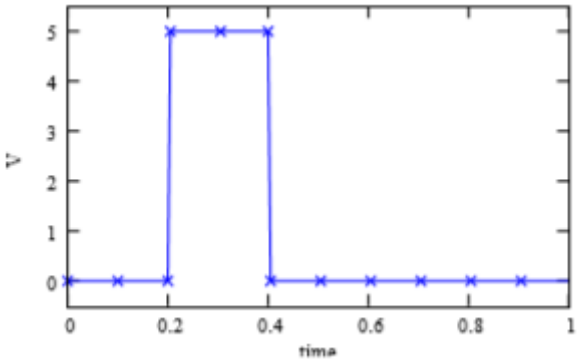

Рис. 22. Осциллограмма получена с clock 1.gce. Были использованы следующие параметры: delt const=0.1, delt min=0.0001. Отметьте дополнительные временные точки (которые были форсированы clock 1.gce) около t=0.2 и t=0.4.

* `delt_max` (optional, default=10×delt_const) задаёт максимальный временной шаг, принимаемый симулятором.

* `itmax_trns` (optional, default=50000) задаёт максимальное количество временных шагов, которое предстоит выполнить симулятору. Это, в основном, «предохранитель», чтобы быть уверенным, программа не будет безостановочно работать, если, например, было задано слишком большое значение для t_end или слишком маленькое для delt_const.

Параметры, зависящие от метода: в методах авто-шагов есть дополнительные параметры, которые управляют временем шага, как описано далее.

1. Метод `back_euler_auto`: если проблема нелинейная, временно шаг может быть вычислен на основе сходимости N-R метода, как иллюстрируется диаграммой на рис. 23.

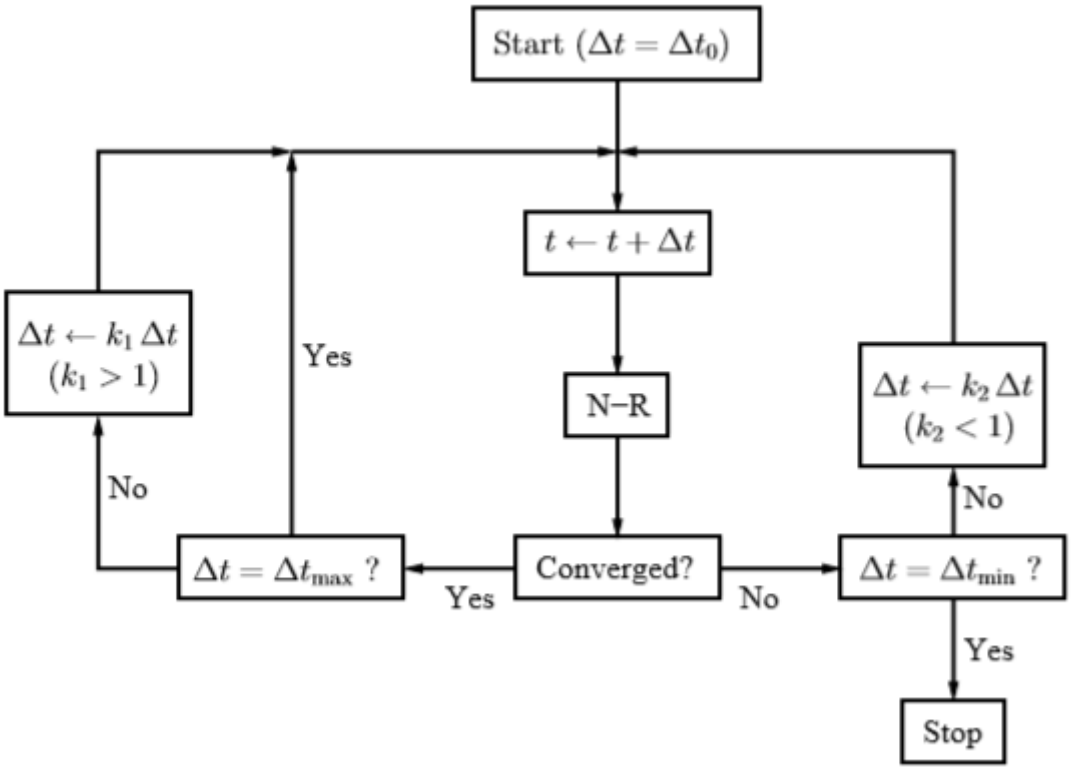

Рис. 23. Диаграмма метода авто-шагов на основе сходимости N-R итераций

Для заданного значения Δt, если метод N-R сходится (в itmax_newton итерациях), мы принимаем решение и делаем следующее значение Δt больше предыдущего вплоть до максимального, заданного delt_max. Если метод N-R не сходится, мы уменьшаем временной шаг и делаем следующую попытку. Если нет сходимости с меньшим допустимым временным шагом (delt_min), программа останавливается. Следующие параметры управляют временным шагом.

* `fctr_stepred` (optional, default=0.05): фактор (k2 на рис. 23), которым уменьшается временной шаг, если процесс N-R расходится.

* `fctr_stepinc` (optional, default=1.5): фактор (k1 на рис. 23), которым временной шаг увеличивается, если процесс N-R сходится.

* `itmax_stepred` (optional, default=20): максимальное количество раз (в заданной точке времени), когда временной шаг может уменьшаться, если процесс N-R не сходится.

2. `trapezoidal_auto` method: этот метод использует тот же алгоритм (рис. 23) и параметры, что и метод back_euler_auto.
3. `trbdf2` method: если проблема линейная, она решается напрямую, и нет необходимости в использовании методов итерации. По этой причине метод, основанный на сходимости, такой, как показано на рис. 23, исключается. В этих случаях мы можем использовать приближение на базе оценки локального уменьшения ошибки (LTE). В TR-BDF2 схеме [9] временной шаг Δti+1 (то есть, интервал между ti и ti+1) делится на две части: (a) Δt1 = γ Δti+1 (γ < 1), где применяется метод трапеций. Мы будем называть это TR шагом. (b) Δt2 =

(1−γ) Δti+1, где применяется схема второго порядка Backward Difference (BDF2). После того, как мы выполним один TR и один BDF2 шаг (то есть, мы теперь в точке времени ti+1), осуществляется уменьшение локальной ошибки, и на основе относительной величины этой ошибки с заданным пользователем значением отклонения, текущий временной шаг либо принимается, либо вырезается.

Если ошибка оказывается слишком большой, вырезается предыдущий временной шаг, TR-BDF2 шаг выполняется с меньшим временным шагом, и ошибка вычисляется вновь. Временной шаг может быть вновь принят или вырезан, в зависимости от ошибки. Основная идея во многом та же, что изображена на рис. 23, за исключением того, что используется оценка LTE (вместо сходимости N-R процесса) для определения, будет ли принят или вырезан заданный временной шаг. Более детально метод описан в [9]. Следующие параметры применимы для метода trbdf2.

* `trbdf2_tolr` (optional, default=1e-5): «допуск» для метода TR-BDF2 (A рис. (43) в [9])

* `trbdf2_gamma` (optional, default=0.586): значение γ в предыдущем обсуждении. Значение по умолчанию было получено при решении уравнения (33) в [9].

Для сходимости разные параметры анализа переходного процесса представлены на рис. 24.

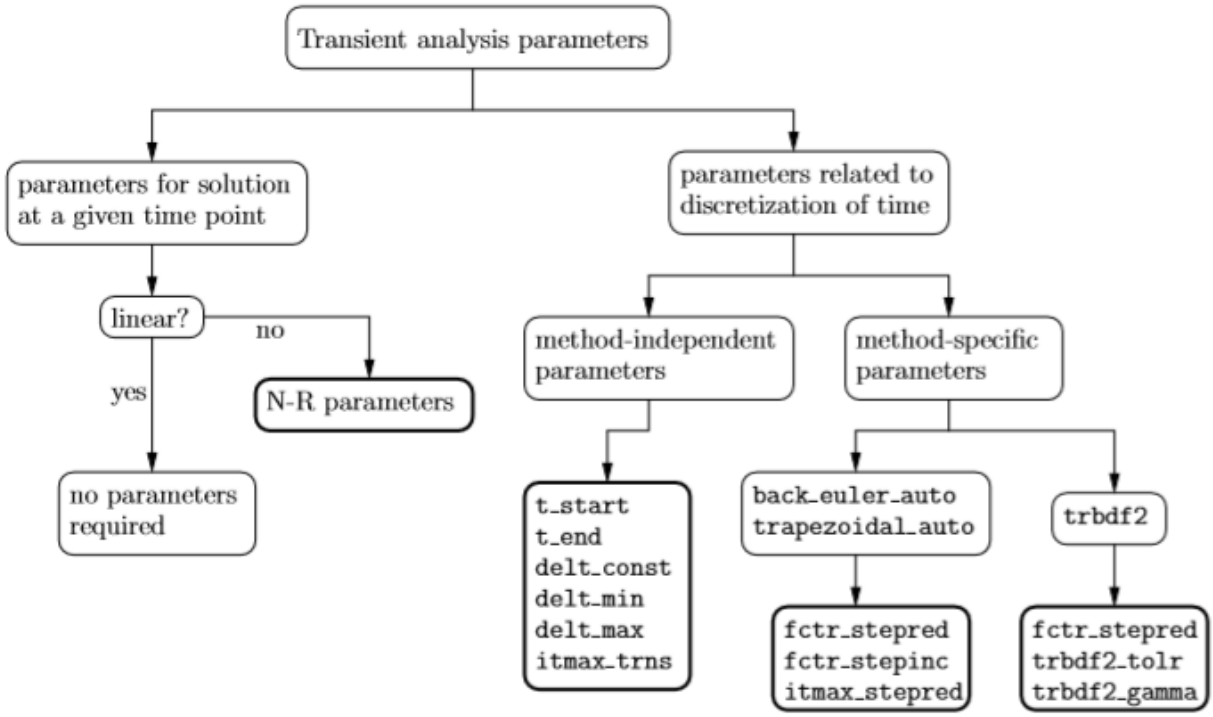

Рис. 24. Параметры анализа переходного процесса

7.1.4 SSW анализ

Как показано в разделе 4.4, анализ касается внешних SSW N-R итераций и анализа переходных процессов в каждой внешней итерации. Вместе с тем, все параметры, описанные в разделе 7.1.3 для анализа переходного процесса, применимы для SSW анализов, исключая, что выбор методов разрешён (поскольку интересующий временной интервал для SSW много меньше, чем типичный для анализа переходного процесса, обычно нет особой необходимости использовать методы автоматического выбора шага, а back euler или trapezoidal хорошо работают) для back_euler, trapezoidal, back_euler_auto и trapezoidal_auto. Хотя параметр t_end, останавливающий время в анализе переходного процесса, не относится к SSW анализам, поскольку мы интересуемся только одним циклом (в течение T). Вдобавок к параметрам анализа переходного процесса используются следующие параметры SSW анализа.

* `ssw_period`: период (T) сигнала

* `ssw_frequency`: это другой вариант задания периода. Если ssw_frequency (f0) задана, ssw_period вычисляется внутренне как T = 1/f0

Заметьте, что только один из параметров ssw_period и ssw_frequency должен быть задан.

* `ssw_period_mult` (optional, default=1): этот параметр (целое) используется для задания количества рассматриваемых циклов. Выбор ssw_period_mult отличным от 1 не имеет смысла с точки зрения вычислений, поскольку решение будет просто повторяться на интервале T, и в последующих интервалах не будет новой информации. Однако ssw_period_mult> 1 полезно, когда пользователь намерен получить график, который явно показывает периодическую природу сигнала.

Параметры ssw_itmax_newton, ssw_norm, ssw_dmp, ssw_dmp_k и ssw_dmp_newt_max похожи на те, что обсуждались в разделе 5.3, исключая то, что они применяются к SSW N-R итерациям (внешний цикл на рис. 10), как описано ниже.

* `ssw_itmax_newton` (optional, default=20) используется для задания максимального числа N-R итераций в цикле SSW N-R. Если N-R процесс не сходится в ssw_itmax_newton итерациях, программа выведет сообщение об ошибке и остановится.

* `ssw_norm`: значение допуска, используемое для сравнения с $\epsilon^{\mathrm{SSW}}_{\mathrm{RHS}(2)}$, чтобы решить, будут ли SSW N-R итерации сходиться. $\epsilon^{\mathrm{SSW}}_{\mathrm{RHS}(2)}$ получается из

$$\epsilon^{\mathrm{SSW}}_{\mathrm{RHS}(2)} = \frac{1}{N'}\sqrt{\sum_{i=1}^{N'} f_i^2}, \tag{9}$$

где N’ – это количество переменных состояния в уравнениях.

* `ssw_dmp=yes` (no) индицирует, что демпфирование будет (не будет) использовано для SSW N-R итераций (optional, default=no)

* `ssw_dmp_k` (optional, default=0.8): значение k (см. уравнение 8) для SSW N-R цикла

* `ssw_dmp_newt_max` (optional, default=10): количество SSW N-R итераций, для которых будет применено демпфирование. После ssw_dmp_newt_max итераций используется стандартный N-R метод, общее количество итераций (damped+standard) задаётся ssw_itmax_newton.

Параметры ssw_dmp_k и ssw_dmp_newt_max существенны только тогда, когда ssw_dmp установлено в yes.

7.2 Другие составляющие блока решения

Помимо установки метода разделы решения могут иметь несколько других составляющих, которые могут использоваться в зависимости от типа блока решения анализа.

7.2.1 initial sol декларация

Целью данного заявления будет задание того, как генерировать начальное решение, то есть, «отправную точку» для анализа. Следующие опции доступны для этой декларации.

1. `initial_sol initialize` показывает, что начальное решение должно быть сгенерировано внутри симулятора. Это опция по умолчанию.
2. `initial_sol file filename=string` показывает, что начальное решение должно быть прочитано из файла, имя которого задано строкой (string).
3. `initial_sol previous` показывает, что предыдущее решение (содержащееся в последнем блоке) должно использоваться как начальное решение.

Начальное решение применимо к DC, start-up, transient и SSW анализу. В частности, если проблема нелинейная, тогда начальное решение поддерживает отправную точку для N-R процесса. Заметьте, что для transient и SSW анализа начальное решение предоставляет отправную точку N-R процесса только для первых временных точек. В дальнейшем решение, полученное в t_i, используется как стартовая точка для $t_i + 1$. Для AC анализа начальное решение не требуется.

7.2.2 Синтаксис выходного блока

Назначение выходного блока (output block) информировать симулятор о переменных, которые должны сохраниться в выходных файлах, об именах выходных файлов и т.д. Допустим один или более выходной блок в каждом блоке решения. Следующие декларации применимы в выходном блоке:

1. `begin_output` – показывает начало выходного блока.
2. filename=string1 limit_lines=string2 append=string3 эта декларация задаёт имя выходного файла, который должен быть создан, и связанные опции.

   string1 задаёт имя файла, например, filename=rc_circuit.dat.

   Верхний предел количество строк в файле может быть задано в string2, то есть, limit_lines=200000. Это «предохранитель» для защиты от непреднамеренного создания огромных выходных файлов; если количество строк, запоминаемых в выходном файле превышает limit_lines, программа напишет сообщение об ошибке и остановится. Значение по умолчанию 100000.

Ключевое слово `append` показывает, будет ли файл открываться в режиме «append». string3 может быть yes или no (default=no). Режим append (прикреплять) полезен, если пользователь желает разделить симуляцию в разные блоки решения, но записать данные в один выходной файл для графического изображения всего вместе. Например, рассмотрим частотную характеристику режекторного фильтра. Мы можем пожелать разделить симуляцию на три части: (a) f < f0, (b) f≈f0 и (c) f > f0. Это позволит нам использовать относительно грубое разрешение по частоте в блоке решения для (a) и (c) и хорошее разрешение для (b). Однако мы предпочли бы

записать выходные данные (в зависимости от частоты) в один и тот же выходной файл для дальнейшего обозрения. Это может быть сделано с append=yes в блоке решения, относящегося к (b) и (c).

3. `variables`: список, показывающий информацию, которая будет записана в выходной файл. Следующие опции могут использоваться в списке:

* имена выходных переменных, определённых в схеме (для AC переменных, скажем, v1 нам необходимо задать, что сохранять – амплитуду или фазу v1; это можно сделать, записывая в список mag of v1 или phase of v1).

* `solution` для обозначения всего решения. Обычно это делается для сохранения решения с целью его использования в качестве начальной точки в другом блоке решения. Если решение задано, не дозволяются другие строки в декларации переменных.

Переменные цифровых и аналоговых типов нельзя записывать в один и тот же выходной файл.

4. `control`: string1=string2 (и т.д.). Эта декларация, не обязательная, используется для двух целей:
    - для управления выходными временными точками, в которых выход записывается в файл при анализе переходного процесса в блоке решения (не надо путать с временными точками, в которых уравнения схемы имеют реальное решение). Следующие комбинации string1 и string2 важны в этом случае.

      * `string1`: фиксированный интервал для задания интервала между последующими выходными точками времени
      `string2`: действительное число

      * `string1`: `out_tstart`, чтобы показать, что выход не будет записываться до заданного времени
      `string2`: действительное число

      * `string1`: `out_tend`, чтобы показать, что выход не будет записываться после заданного времени
      `string2`: действительное число

    - для управления тем, как фазовая AC (комплексная) переменная будет записываться в выходной файл. Это значило бы, в сущности, возможность избежать «скачка» с -180^0 к $+180^0$ в графике фазо-частотной характеристики. Такие прыжки, хотя технически не неправильные, могут быть неприятны, когда вам приходится интерпретировать график. Чтобы получить непрерывный график мы можем использовать тот факт, что любой угол ϑ эквивалентен $\vartheta \pm 360^0$. Следующие комбинации string1 и string2 применимы в этом случае.

      * `string1`: `min_phase`, чтобы показать минимальное значение фазы, которое записывается в файл
      `string2`: действительное число (градусы)

      * `string1`: `max_phase`, чтобы показать максимальное значение фазы, которое записывается в файл
      `string2`: действительное число (градусы)

Либо min_phase, либо max_phase может быть задано, но не оба сразу. По определению задано «no» для обоих.

* `string1: delta_phase`, чтобы показать максимально разрешённую разность между значениями фазы в последующих точках частоты
`string2`: действительное число (градусы, default=200)

В основном, значения по умолчанию работают хорошо; однако если есть нежелательные скачки в графике фазы, пользователь должен задать вышеописанные параметры, чтобы избежать этих скачков.

5. `end_output` показывает конец выходного блока.

7.2.3 Декларация set parm

```
set_parm string1_of_string2=string3
```

Это заявление присваивает заданные значения целым или действительным параметрам элементов, перекрывая определение в схеме. Это дополнительная декларация. Разрешается более одного set_parm заявления в блоке решения. Здесь string1 – это имя параметра, string2 – имя элемента, а string3 – целое или действительное число, зависящее от типа связанного параметра.

7.2.4 Декларация set stparm

```
set_stparm string1_of_string2=real_number
```

Это заявление присваивает заданные значения параметру start-up элемента типа ece, gce или tce, перекрывая определение в схеме. Это дополнительная декларация. Разрешается более одного set_stparm заявления в блоке решения. Здесь string1 – это имя start-up параметра, а string2 – имя элемента.

7.2.5 Декларация vary parm

Цель этого заявления – варьировать действительный параметр элемента. Это дополнительная декларация. Разрешается более одного заявления vary parm, и в этом случае они работают подобно вложенным do или for формулировкам. Параметр, появляющийся в последнем заявлении vary parm, варьируется первым, затем варьируется предпоследний и т.д. Декларация vary parm приложима к DC, start-up и AC блокам решения. Они могут быть одной из трёх форм, показанных ниже.

1. `vary_parm string1_of_string2 from real_number_1 to real_number_2 + type=linear n_points=integer`

   Это показывает, что параметр, названный string1, элемента, названного string2, будет варьироваться линейно от действительного числа 1 до действительного числа 2, общее число точек будет задано через n точек. Заметьте, что + в первой колонке (здесь или где-то ещё) просто указывает на продолжение декларации в синтаксисе SEQUEL.

2. `vary_parm string1_of_string2 from real_number_1 to real_number_2 + type=log n_points=integer`

   Это показывает, что параметр, названный string1, элемента, названного string2, будет варьироваться логарифмически от действительного числа 1 до действительного числа 2,

общее количество точек будет задано n точками. Заметьте, что действительные числа в этом случае должны быть положительными.

3. `vary_parm string1_of_string2 type=table x1 x2 ..`

   Это показывает, что параметр, названный string1, элемента, названного string2, будет варьироваться. Первое действительное число (x1), появляющееся после type=table, будет назначено в качестве первого значения параметра и т.д.

   Есть другой способ использования декларации vary parm, где string2 замещается ключевым словом glbl, показывающим, что все действительные параметры элемента, которые были приписаны к глобальным действительным параметрам string1, будут варьироваться.

7.2.6 Декларация vary parm sync

Декларация vary_parm_sync (справедливая для DC и AC анализа) полезна, когда параметры для разных элементов варьируются «синхронно» один с другим. В качестве примера рассмотрим схему с несколькими резисторами r1, r2 и т.д. Если мы хотим варьировать некоторые из них вместе, тогда следует использовать следующие заявления:

```
vary_parm_sync r_of_r1 from 2 to 10 type=linear n_points=5
vary_parm_sync r_of_r2 from 1 to 5 type=linear n_points=5
```

Декларации выше будут приводить к тому, что анализ повторится пять раз с (R1,R2)=(2,1), (4,2), (6,3), (8,4) и (10,5). Синтаксис для декларации vary_parm_sync тот же, что и для vary_parm.

7.2.7 Декларация set freq

```
set_freq=real_number
```

Это заявление используется для установки частоты в AC анализе, когда анализ должен быть выполнен для единственной частоты.

7.2.8 Декларация vary freq

Цель данной декларации варьировать частоту заданным образом в AC анализе. Декларация vary freq может быть в одной из трёх форм, заданных ниже.

1. `vary_freq from real_number_1 to real_number_2 + type=linear n_points=integer`

   Это показывает, что частота будет варьироваться линейно от действительного числа 1 до действительного числа 2, общее количество точек задаётся как n точек.

2. `vary_freq from real_number_1 to real_number_2 + type=log n_points=integer`

   Это показывает, что частота будет варьироваться логарифмически от действительного числа 1 до действительного числа 2, общее количество точек задаётся как n точек.

3. `vary_freq type=table x1 x2 ..`

Это показывает, что первое действительное число (x1), появляющееся после type=table, это первое значение для частоты и т.д.

В блоке решения AC анализа должны быть либо одно заявление set freq, либо одно vary freq (но не оба сразу).

7.2.9 Декларация set tmpr

```
set_tmpr tmpr_name=real_number
```

Это заявление используется для (не обязательно) задания значений температуры. Допускается более одного заявления set tmpr. Здесь tmpr – это имя температурного узла в схеме. Декларация set tmpr может использоваться в AC анализе или анализе чисто цифровой схемы; в остальных случаях игнорируется.

Ссылки

[1] M. B. Patil, M. C. Chandorkar, B. G. Fernandes, and K. Chatterjee, “Computation of steady-state response in power electronic circuits,” IETE J. Research, vol. 48, no. 6, pp. 471-477, Nov. 2002.

[2] T. J. Aprille and T. N. Trick, “Steady-state analysis of nonlinear circuits with periodic inputs,” Proc. IEEE, vol. 60, pp. 108-114, 1972.

[3] T. J. Aprille and T. N. Trick, “A computer algorithm to determine the steady-state response of nonlinear oscillators,” IEEE Trans. Circuit Theory, vol. 19, pp. 354-360, 1972.

[4] F. R. Colon and T. N. Trick, “Fast periodic steady-state analysis for large-signal electronic circuits,” IEEE J. Solid-State Circuits, vol. 8, pp. 260-269, 1973.

[5] T. N. Trick, F. R. Colon, and S. P. Fan, “Computation of capacitor voltage and inductor current sensitivities with respect to initial conditions for the steady-state analysis of nonlinear periodic circuits,” IEEE Trans. Circuits and Systems, vol. 22, pp. 391-396, 1975.

[6] M. B. Patil, “A public-domain program for mixed-signal simulation,” IEEE Trans. Education, pp. 187-193, May 2002.

[7] R. Raghuram, Computer Simulation of Electronic Circuits, Wiley Eastern, New Delhi, 1989.

[8] P. Antognetti, D. O. Pederson, and H. de Man, Computer Design Aids for VLSI Circuits, NATO ASI Series, Martinus Nijhoff Publishers, The Hague, 1984.

[9] R. E. Bank, W. M. Coughran, W. Fichtner, E. H. Grosse, D. J. Rose, and R. K. Smith, “Transient simulation of silicon devices and circuits,” IEEE Trans. Electron Devices., vol. 32, no. 10, pp. 1992-2006, 1992.