

Ep. 6.7
535

ISSN 0136-3549
3134-3823

TALLINNA
POLÜTEHNILISE INSTITUUDI
TOIMETISED

535
ТРУДЫ ТАЛЛИНСКОГО
ПОЛИТЕХНИЧЕСКОГО
ИНСТИТУТА

ТПИ
'82

АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ
В ЭЛЕКТРОНИКЕ
СИСТЕМА SPADE

535

**TPI
'82**

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3

●

**АВТОМАТИЗАЦИЯ
ПРОЕКТИРОВАНИЯ
В
ЭЛЕКТРОНИКЕ -
СИСТЕМА
SPADE**

Электротехника и автоматика XX

Таллин 1982

ПРЕДИСЛОВИЕ

The structure of a system
reflects the structure
of the organization that
built it.

(Conway's Law)

Опубликованные в настоящем сборнике статьи отражают результаты работы над системой SPADE, которая представляет собой, с одной стороны, инструментальную систему для разработки пакетов прикладных программ, а с другой стороны - набор пакетов для проведения различных расчетов при проектировании электронных систем.

Материалы сборника, касающиеся системы SPADE, как инструментального средства, соответствуют конкретному продукту - 3-й версии системы. Статьи, в которых рассматриваются приложения системы, не могут быть отражением конечного продукта, так как прикладные пакеты являются непрерывно развивающимися и в ходе эксплуатации. Поэтому эти статьи следует рассматривать как исследования предметной области и вычислительных процедур на основе идеологии, принятой в системе SPADE. Но авторы убеждены, что эта идеология является достаточно общей для того, чтобы сделать результаты этих исследований применимыми и при использовании других инструментальных систем.



ПРИНЦИПЫ ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ SPADE

I. Предпосылки

Система SPADE, которая реализована на ЕС ЭВМ, имеет предшественников: систему SAATAN, реализованную на ЭВМ Минск-22 (1970-1976 гг.) и систему SPADE на ЭВМ Минск-32 (1976-1979 гг.). Опыт разработки и эксплуатации этих систем, анализ некоторых других инструментальных систем (ПРИЗ, ИСП), существующих пакетов программы (АНРЗ, НАР2) и их описаний (SCPTRE, MAPC, СПАРС) являлись основными источниками при разработке рассматриваемой системы.

На этой основе сделаны следующие выводы:

1. Процесс машинного проектирования является сложным процессом в системе человек-машина, который очень часто сводится к непрерывной отладке, так как возникающие задачи не могут быть решены существующими программными средствами.

2. Обеспечить необходимую скорость решения задач (от постановки задачи до получения результатов) можно только на базе некоторой инструментальной системы, позволяющей максимально автоматизировать процесс разработки программных средств и сосредоточить внимание разработчика на существенных (содержательных) аспектах проблемы.

3. Существенно, чтобы инструментальная система помогала разработке на всех этапах решения задачи - качество ее определяется самым слабым звеном (может быть в большей степени, чем в технических системах).

4. Желательно, чтобы процедуры в системе были однотипными - как разработка и развитие самого пакета, так и решение прикладных задач с помощью него должны подчиняться одной и той же технологии. Это обеспечивает саморазвитие

системы, позволяет с самого начала разработки начинать решение прикладных задач и существенно ускоряет внедрение системы.

5. Крайне необходимо соблюдать строгую дисциплину при разработке математического обеспечения. Практика показывает, что человек склонен не поддерживать такой дисциплины, и стандартные средства предлагают ему массу возможностей для нарушения элементарных правил программирования (ради "эффективности"). Поэтому нужно, чтобы инструментальная система лишала разработчика таких возможностей.

6. К сожалению, очень распространено мнение, что эффективность программы определяется скоростью решения прикладных (часто - тестовых) задач [1]. В действительности, эффективность определяется полными затратами на решение всех задач, которые действительно решаются и затратами на разработку программы. Ввиду того, что последние часто существенно превышают затраты на решение, следует основное внимание уделять уменьшению затрат на разработку пакетов.

7. Всегда, когда возможно, следует предпочитать динамические структуры статическим. Это относится как к структурам данных, так и к процессам.

8. Одним из самых важных показателей качества пакета является обработанность выходной информации. Вывод массы результатов-полуфабрикатов, требующих от пользователей рутинной сортировки, поиска и дополнительных расчетов, может свести на ноль все достоинства пакета. Поэтому базовая система должна обеспечить возможность отображения в выходных документах минимального количества, в удобочитаемой форме и только существенной информации. Для инженера-проектировщика основной формой нужно считать графическое представление, а таблицы играют вспомогательную роль.

2. Основная идеология

Перечисленные выше выводы содержат один очень важный момент - критикуемые явления связаны почти без исключения с процедурной ориентацией прикладных пакетов, т.е. тем, что разработка пакета рассматривается как процесс составления программ. Выход из такого положения заключается в признании

ведущей роли данных - эту тенденцию можно считать основной в самом программировании в последние годы [2, 3, 4].

Основной конструктивный вывод, сделанный из сказанного, может быть выражен следующей формулой:

машинное проектирование=обработка данных,

которая являлась ведущим тезисом при разработке системы SPADE.

Эту формулу следует понимать в самом широком смысле: как относительно решения прикладных задач, так и относительно разработки пакетов:

1) процесс решения прикладной задачи есть обработка данных;

2) эта обработка управляется данными, к созданию и модификации которых сводится процесс разработки пакетов.

Необходимо отметить, что в процессе машинного проектирования сами обрабатываемые данные существенно влияют на вычислительный процесс, т.е. принимают участие в управлении (причем не только их структура, но и содержание).

В качестве задач создания данных следует рассматривать такие традиционные компоненты систем машинного проектирования, как 1) описание исходных данных, 2) трансляция, 3) оформление выходного документа. Непосредственным следствием такого подхода являются: единая структура описания данных, универсальные транслятор и формирователь выходных документов.

Управление вычислительным процессом определяется в основном связями (отношениями) между данными. Возможность автоматизации процесса управления была показана Э. Тьуту построением теории вычислительных моделей [5].

По нашему мнению, существенное отличие машинного проектирования от традиционной обработки информации заключается в преобладающей роли функциональных связей по сравнению со структурными связями. Если структурные связи являются конечными, строгими, то функциональные связи являются нестрогими, неконечными, неоднозначными, неточными. Они, как правило, описывают лишь принципиальную воз-

возможность вычисления, осуществимость которого может существенно зависеть от содержания самих данных. Сложность функциональных связей, по-видимому, является одной из причин, почему созданные пакеты просто реализуют конкретные функциональные отношения. Усилия разработчиков были сосредоточены на разработку и реализацию соответствующих алгоритмов, а информационная сторона проблемы оставалась на заднем плане.

Работа системы SPADE базируется на следующих принципах:

1) существует база данных задачи - набор данных, отображающий конкретную предметную область, в которой решается задача;

2) элементы базы связаны между собой отношениями, которые позволяют вычислять одни значения из других;

3) пользователь может ввести данные в базу посредством (входного) языка описания данных;

4) пользователь может вывести данные из базы указанием данных и формы выходного документа (описание заказа);

5) используя набор отношений, система пытается вычислить необходимые значения для выполнения заказа.

Таким образом, система SPADE в принципе предполагает диалог пользователь - база данных.

Предполагается, что разработчик пакета (или сам пользователь) заранее ввел в систему необходимые описания данных и отношений, хотя не исключено пополнение этих наборов во время решения конкретной проблемы. Ввод дополнительных отношений или объектов дает пользователю возможность управлять решением своей задачи, получить специальные результаты, учитывающие конкретный характер задачи.

Система SPADE требует от разработчика пакета точного и исчерпывающего описания всех данных, в том числе и так называемых "промежуточных". В основу системы положен тезис, что не должно быть неопределенных, лишенных смысла данных. Анализ существующих алгоритмов показывает, что все операции, как и сами данные, имеют ясную физическую интерпретацию, что должно быть отображено в системе данных. Все данные в базе должны быть доступны пользователю.

Три основных вопроса, которые необходимо было решать при создании системы согласно вышеприведенным принципам, были

1) как определять и интерпретировать данные из предметной области, как их описывать?

2) как определять и описывать отношения, как реализовывать соответствующие вычисления?

3) как управлять вычислительным процессом?

Принципиальное решение этих проблем излагается в настоящей статье, более подробное описание приводится в последующих статьях сборника.

3. Мир глазами системы SPADE

В системе SPADE предполагается, что при решении любой задачи имеем дело с некоторой совокупностью объектов ("системой"), которые имеют нечто общее (хотя бы то, что они — суть объекты данной совокупности).

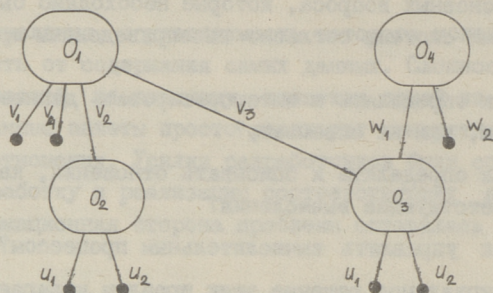
Каждый объект имеет набор атрибутов, которым соответствуют значения. Такая картина является весьма общей и принятой в теории обработки информации [6].

Разумеется, представление реального мира может быть различным и определяется не только существом реальных объектов, но и нашим подходом и целями. Анализ разработанных пакетов программ показывает, что в большинстве случаев различные авторы строят похожие модели.

Подчеркнем, что объекты могут представлять самые различные вещи: резистор или все резисторы данной схемы, передаточную функцию или полином, таблицу как выходной документ или критерий оптимальности и т.д.

Единственные связи между объектами определяются тем, что значением атрибута может быть другой объект. Так как такое соотношение создает подчиненность, то тем самым определяется и некоторая иерархия в данной совокупности объектов.

Рекурсия, определенная указанной подчиненностью, сшивается на значениях, не являющихся объектами в определенном выше смысле — на терминальных значениях (фиг. I).



Фиг. 1. Объекты O_2 и O_3 являются значениями атрибутов v_2 и v_3 объекта O_1 , объект O_3 также значением атрибута w_1 объекта O_4 . Все остальные атрибуты имеют терминальные значения.

В системе SPADE терминальное значение – вектор, который представляет собой упорядоченную последовательность элементов одного из следующих типов:

- 1) целый,
- 2) вещественный,
- 3) комплексный,
- 4) логический,
- 5) текстовой.

На выборе в качестве основного понятия вектора, как и указанного набора типов, отражается специфика той прикладной области, для которой создана система.

Совершенно ясно, что мы можем иметь дело с объектами, которые характеризуются одним и тем же набором атрибутов. Такие объекты считаем принадлежащими одному классу или имеющими определенный тип. Принадлежащие данному классу объекты идентифицируются именем объекта. Система SPADE демократична в том смысле, что все объекты, несмотря на их подчиненность, считаются равноправными членами данной совокупности.

Нетрудно видеть, что тип объекта можно рассматривать как отношение в смысле, принятом для реляционных баз данных [6], причем каждая строка представляет собой конкретный объект (ключом является имя объекта), а каждый столбец – атрибут.

Поэтому можно сказать, что при составлении модели предметной области определение абстрактных объектов означает по-

строение схемы базы данных, а при решении задачи генерируется конкретная база данных задачи. Для реализации связи "пользователь → база данных" используется транслятор, который более подробно рассматривается в [7].

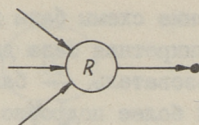
Если структурные модели предметной области могут быть построены без особых сложностей, и различные реализации, по-видимому, дают качественно примерно эквивалентные результаты, то построение функциональной модели является крайне сложной и ответственной задачей, качество решения которой в основном определяет и конечный результат. Проблема усложняется и тем, что функциональные отношения оказывают влияние и на структурные связи. Это часто приводит к необходимости корректировки структурной модели при модификации функциональных связей.

В большинстве работ, посвященных автоматизации управления вычислительным процессом, функциональные отношения рассматриваются в самом общем виде – они могут иметь несколько входов (известных) и несколько выходов (искомых) [5, 8, 9, 10, 11, 12]. Это является естественным обобщением понятия подпрограммы, которая в принципе позволяет изменять значения нескольких параметров.

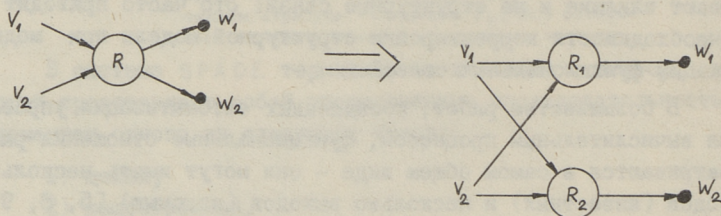
Такая общность может оказаться существенным ограничением, если не пренебречь неопределенностью функциональных отношений, как это часто делают [9]. Во-первых, отметим, что при наличии нескольких выходов трудно определить применимость некоторого отношения, так как часть выходов могут быть определены, а некоторые – нет. Во-вторых, наличие нескольких выходов существенно усложняет планирование ввиду того, что поиск необходимо вести по группам величин, а также из-за потенциальной избыточности. В-третьих, реальные функциональные отношения являются настолько сложными, что реализация любого разбиения на входы-выходы требует отдельной программной реализации, причем некоторые разбиения могут оказаться бессмысленными.

Эту проблему можно сравнивать с созданием сложного технического устройства, которое состоит из большого количества элементарных компонентов, количество типов которых невелико. При этом функциональная простота компонентов имеется как на высоких уровнях иерархии (функциональная схема), так и на низких (принципиальная схема).

Поэтому в системе SPADE в качестве основного типа функционального отношения используется элементарное отношение, имеющее один выход (терминальное значение, т.е. вектор) (фиг. 2). Совершенно ясно, что отношение с n выходами может быть заменено на n элементарных (фиг. 3) и поэтому принципиальных ограничений при построении сложных отношений не существует.



Фиг. 2. Отношение с одним выходом (элементарное отношение).



Фиг. 3. Представление отношения с двумя выходами в виде двух элементарных.

Применение элементарных отношений позволяет ставить каждому отношению однозначно в соответствие значение (но не наоборот). Это позволяет просто следить за версиями (поколениями) значения и породивших их отношений, что очень важно при модификациях (в т.н. многовариантных задачах).

Мощность семантической модели определяется полнотой системы отношений. Если можно было бы ограничиться только структурой отношений, то планирование решения оказалось относительно простым, о чем свидетельствуют соответствующие публикации [9, 10]. Но уже в [5] была отмечена необходимость использования некоторых внешних условий при определении применимости отношения.

Отметим здесь некоторые дополнительные проблемы, возникающие в связи с функциональными отношениями:

- 1) несуществование выхода при данных входных значениях;
- 2) многозначность выхода;

- 3) необратимость (практически всех) отношений;
- 4) вырождения при определенных значениях входов;
- 5) неэквивалентность различных путей вычисления.

Все это может привести к довольно трудным проблемам как при составлении семантической модели предметной области, так и при самом планировании решения. Некоторые конкретные проблемы будут рассмотрены в последующих статьях данного сборника. Там же указываются некоторые пути их решения.

4. Функциональная структура системы

Система SPADE спроектирована для работы под управлением ОС ЕС, преимущественно в режиме MVT. В качестве базового языка используется ФОРТРАН IV, хотя небольшое количество модулей составлено на АССЕМБЛЕРЕ.

Все модули могут быть разделены на следующие группы:

- 1) фундаментальные модули,
- 2) организующие модули,
- 3) отладочные модули,
- 4) операторные модули,
- 5) внешние модули.

Фундаментальные модули образуют некоторое расширение операционной системы и в эту группу входят: модули фортран-ной среды, модули, обеспечивающие работу со структурами данных, модули динамического управления памятью и программами, модули, реализующие отдельные часто употребляемые функции. Эти модули доступны всем остальным программам системы и входят в ядро системы.

Организующие модули выполняют специфические для системы функции: трансляция, организация вычислений, оформление результатов. Эти модули в основном загружаются динамически в зависимости от хода решения.

Отладочные модули обеспечивают развитие самой системы, проверку и тестирование новых модулей и слежение за процессом решения задачи. Часть из этих модулей входит в состав ядра.

Операторные модули обеспечивают систему реализацией вычислений, предусмотренных отношениями. Модули этой группы загружаются динамически в ходе решения. Все они имеют стандартную структуру и стандартный интерфейс с системой. Практически любой модуль, в принципе реализующий функциональную связь и разработанный вне системы SPADE, можно превращать в стандартный путем добавления к нему модуля-шапки.

Внешние модули — это "чужие" программы, разработанные вне системы SPADE. Они вызываются динамически и поэтому могут быть совершенно независимыми, но с целью экономии памяти и для предотвращения неприятностей с повторным открытием наборов данных, желательно модуль обрабатывать МОДИФИКАТОРом системы SPADE. МОДИФИКАТОР есть специальная программа, которая позволяет исключить из загрузочного модуля те подпрограммы, которые входят в состав ядра системы SPADE. В частности, исключаются модули фортранной среды.

Таким образом, ядро системы представляет собой одну программу, включающую все фундаментальные модули, некоторые организующие и отладочные модули. Главный организующий модуль осуществляет загрузку и запуск транслятора, решателя и оформителя. Ядро имеет оверлейную структуру и редактируется отдельно. В зависимости от ресурсов и характера задач могут быть использованы различные оверлейные структуры.

С функциональной точки зрения в работе системы имеется три основных обрабатывающих компонента:

- 1) транслятор,
- 2) решатель,
- 3) оформитель.

Основные группы данных, принимающие участие в решении задачи, следующие:

- 1) рабочие,
- 2) управляющие.

Рабочие данные состоят из:

- 1) базы данных задачи (внутренние),
- 2) исходные данные (внешние),
- 3) результаты (внешние).

Две последние подгруппы являются просто отображением части внутренних данных.

Управляющие данные по сути являются семантическими моделями. Транслятор и решатель используют семантическую модель предметной области решаемой задачи, которая в 3-й версии системы четко делится на две части: описания объектов и описания отношений. Оформитель использует фиксированную модель выходных документов.

5. Управление памятью и программами

Память под данные и загружаемые модули распределяется динамически, через макрокоманды супервизора GETMAIN и FREEMAIN. Так как система распределения памяти должна быть доступной и ФОРТРАН-программам, то используется относительная адресация – с началом отсчета в непоименованной общей области. Доступ к зарезервированным участкам памяти осуществляется через фиктивные массивы, определенные в декларации COMMON. Таким образом достигается возможность использования выделяемого участка в том же модуле, т.е. нет необходимости в резервировании памяти вне модуля, в котором она требуется.

Для того, чтобы исключить из динамически загружаемых программ модули, которые уже входят в состав ядра, разработан управляемый загрузчик, который при загрузке программ разрешает все ссылки к модулям ядра. Такая возможность обеспечивается обработкой объектных или загрузочных модулей перед редактированием МОДИФИКАТОРом – одной из вспомогательных программ системы SPADE. Необходимо отметить, что системные макрокоманды LINK и LOAD позволяли бы загружать только независимые программы.

6. Обработка текстовых данных

Ввиду того, что система SPADE почти полностью независима от решаемых задач, а для идентификации данных используются имена, необходимо иметь специальные средства для обработки текстовых значений. Так как все тексты, которые являются в выходных документах, определены в семантической модели предметной области или в тексте данной задачи, то недо-

пустимо ограничивать длину текстовых значений, как это часто делается в языках программирования. Чтобы предотвратить трудности, возникающие при обработке текстовых данных переменной длины и уменьшить расход памяти на размещение текстов, в системе используется кодирование текстов.

Любое текстовое значение кодируется двухбайтовым числом, что позволяет одновременно хранить до 32000 значений. Для кодирования используется хэширование [13]. Все процессы поиска, сравнения и т.д. выполняются над кодами, т.е. целыми числами фиксированного формата.

Описанный метод обработки оказался более эффективным, чем предполагалось. Несмотря на то, что кодируются и все выходные данные (в том числе и образы чисел), в течение опытной эксплуатации, когда решались небольшие задачи, не было случая переполнения области для 2039 значений, которая была зарезервирована. При этом не применялись какие-либо средства для экономии, т.е. все текстовые значения кодировались и сохранялись до конца задания.

7. Физическая структура данных

Как уже было указано выше, основным информационным элементом в структурах данных системы SPADE является вектор. Структурная связь подчинения отображается с помощью ссылок сверху вниз, причем каждая ссылка поименована. Элемент верхнего уровня может иметь несколько подчиненных, т.е. реализуется отношение типа 1:m. Реализация такого отношения возможна с помощью вектора указателей, объединяющего все ссылки к подчиненным данному элементу. Естественная интерпретация такой структуры - дерево. Именно древовидные структуры приняты в качестве основных в системе SPADE как самые простые. Более подробно связь этих структур с семантическими моделями рассматривается в [7].

Ядро системы содержит несколько модулей для работы с такими векторными структурами. Они выполняют естественные функции: чтение и запись элементов данных, извлечение адреса данных и создание указателя (присоединение структуры), наращивание вектора значений.

Основной вид идентификации данных в таких структурах - через последовательность имен, что эквивалентно использова-

нию составного имени. В действительной работе используются коды имен.

Так как вся связь с базой данных задачи происходит через эти модули, то они используются во всех организующих программах системы. Ввиду необходимости обработки данных с помощью квантора общности (имена неизвестны), указанные модули позволяют обращаться к данным также посредством индексов, т.е. порядковых номеров в векторах указателей.

Из некоторых вариантов модулей в ходе эксплуатации выделялись те, которые допускали смешанную идентификацию, т.е. на некоторых уровнях используются индексы, на других - идентификаторы. Они оказались наиболее удобными в реализации программ ввиду простой реализации выборки при любых сочетаниях кванторов существования (имя) и общности (индекс).

8. Заключение

Опыт разработки и эксплуатации первых версий системы в общем доказал справедливость указанных выше принципов. Существенно улучшились дисциплина работы, согласованность данных, до минимума удалось свести дублирование как в данных, так и в программных модулях. Возможность вывода любых данных повышала удобство применения прикладных программ, которые стали короткими, содержательными и легко понятными. Уже после завершения первой версии транслятора сложилась ситуация, где сами разработчики для отладки всех средств предпочитали использовать саму систему SPADE.

За все это необходимо платить дополнительным расходом машинного времени, так как накладные расходы при решении конкретной задачи безусловно больше, чем при использовании специальной программы. Но на наш взгляд, "... people, not machines, are the primary cost of today's computer systems; we should be prepared to waste hours of computer time, if it will save us months or years of people time" /2, p. 80/.

Л и т е р а т у р а

1. Петренко А.И., Тимченко А.П., Слусар П.Б. Сравнение программ схемотехнического проектирования на базе набора тестовых задач. - ИВУЗ "Радиоэлектроника", т. 23, 1980, № 6, с. 5-12.

2. Yourdon E., Constantine L.L. Structured design: fundamentals of a discipline of computer program and systems design. Englewood Cliffs, New Jersey, Prentice Hall, 1979. 473 p.

3. Tassell van D. Program style, design, efficiency, debugging and testing. Englewood Cliffs, New Jersey, Prentice Hall, 1978. 323 p.

4. Брукс Ф.П. Как проектируются и создаются программные комплексы. М., Наука, 1979, 151 с.

5. Тыгу Э.Х. Решение задач на вычислительных моделях. - ЖВМ и МФ, т. 10, 1970, № 3, с. 716-733.

6. Мартин Дж. Организация баз данных в вычислительных системах. М., Мир, 1980. 662 с.

7. Венделин Е. Структуры данных и входной язык. См. наст. сб. с. 19.

8. Тыгу Э.Х. Решатель вычислительных задач. - ЖВМ и МФ, т. 11, 1971, № 4, с. 992-1004.

9. Карзанов А.В., Фараджев И.А. Планирование вычислений при решении задач на вычислительных моделях. - Программирование, 1975, № 4, с. 19-24.

10. Вальковский В.А. О синтезе оптимальных программ на базе вычислительных моделей. - Программирование, 1980, № 6, с. 27-36.

11. Бухштаб Ю.А., Горлин А.И., Камнин С.С., Корягин Д.А., Любимский Э.З. Об одном методе планирования расчетных цепочек. - Программирование, 1981, № 3, с. 34-38.

12. Тамм Б.Г., Тыгу Э.Х. Пакеты программ. - Изв. АН СССР, Техническая кибернетика, 1977, № 5, с. III-124.

13. Х о л л П. Вычислительные структуры. Введение в
нечисленное программирование. М., Мир, 1978. 214 с.

V. Kukk

System SPADE: Principles of Operation

Summary

The paper discusses some problems of the design of CAD systems. System SPADE has been designed to be a basis for different CAD-systems. The basic assumption is that computation process can be interpreted as a modification of the "task data base", and so, the user has a direct access to any item in it; he can enter, or modify, or display everything. The design of computer models, the computations control process and the structure of the system are described.

СТРУКТУРЫ ДАННЫХ И ВХОДНОЙ ЯЗЫК.

I. Введение

Исследование предметной области является первым этапом разработки пакета прикладных программ, в результате которого составляется ее формальное описание - модель. Как было показано в [1], модель предметной области состоит из двух частей: описания данных и описания связывающих их отношений.

При решении задачи образуется совокупность данных, которая пополняется на каждом шаге решения, пока не будут получены необходимые величины. Эта совокупность не что иное как конкретизация модели соответствующей предметной области для данной задачи. Для того, чтобы создать совокупность данных, необходимо выделить некоторые объекты и присвоить значения их атрибутам. Средством для проведения этой процедуры является входной язык.

Основная функция транслятора заключается таким образом в преобразовании данных, описанных в тексте, во внутреннее представление, т.е. в совокупность данных, используемых при решении. Это преобразование сопровождается сравнением содержания текста с описанием модели.

Принципиальная сторона вопроса о данных, вычислительном процессе и моделях предметных областей обсуждалась в [1], в настоящей же статье подробно рассматривается проблема описания данных в модели предметной области, проблема создания данных при решении задачи и связанные с этим вопросы их представления на входном языке.

2. Структура данных

Данные в системе SPADE делятся на объекты, характеризующиеся некоторым набором атрибутов. При описании модели предметной области задаются типы этих объектов, атрибуты, а также контрольные величины для проверки их значений. Вся информация собрана в иерархическую структуру со следующими уровнями:

- а) все описания данных;
- б) типы объектов;
- в) атрибуты;
- г) контрольные величины для последующей проверки значений.

При описании данных на атрибуты объектов могут быть наложены следующие ограничения.

1) Тип значений.

Задается тип значений, который может быть использован при написании текста на входном языке: словесное значение, целое, действительное или комплексное число и тип значений, в которые должны быть преобразованы значения из текста. Им может быть код текстовой константы или любой формат числа, принятый в ФОРТРАНе.

2) Количество значений.

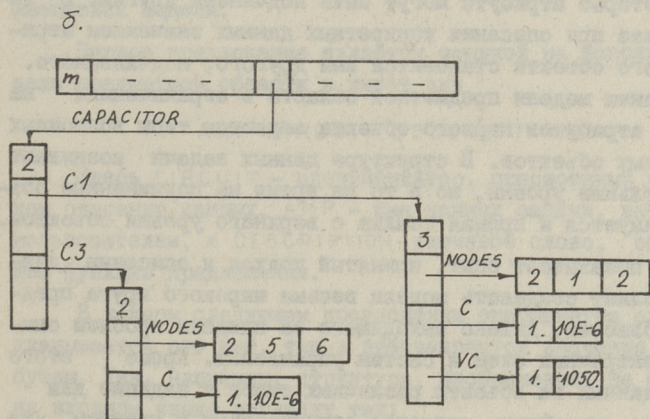
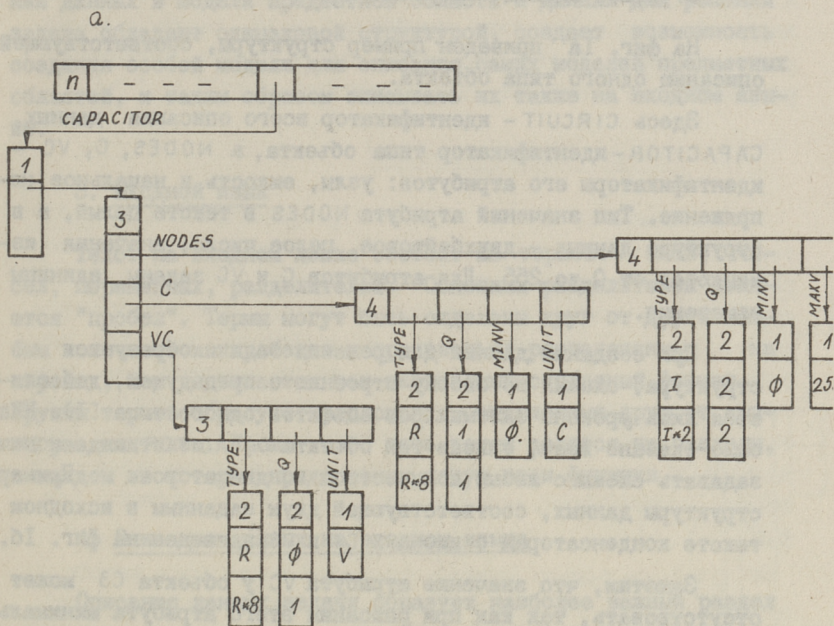
Указывается минимально необходимое для вычислений количество значений и максимальное их количество.

3) Допустимые значения.

Допустимые значения могут быть заданы:

- а) наименьшим и наибольшим значением;
- б) перечнем значений;
- в) перечнем значений, задаваемых пользователем в тексте и перечнем значений, которые возникают при этом в структуре данных;
- г) смешанным способом: интервалом и отдельными значениями.

4) Стандартные значения, которые присваиваются атрибуту, если его описание в тексте отсутствует.



Фиг. 1. Структура описания объекта в модели (а) и структура описания двух экземпляров объектов из конкретной задачи (б).

5) Класс единиц измерения.

На фиг. 1а приведен пример структуры, соответствующей описанию одного типа объекта.

Здесь CIRCUIT - идентификатор всего описания данных, CAPACITOR - идентификатор типа объекта, а NODES, C, VC - идентификаторы его атрибутов: узлы, емкость и начальное напряжение. Тип значений атрибута NODES в тексте целый, а в структуре данных - двухбайтовое целое число, значения изменяются от 0 до 255. Для атрибутов C и VC заданы единицы измерения.

При создании данных для решения задачи образуется структура, схожая по своему строению с предыдущей, добавляется лишь уровень экземпляров объектов одного типа [I]. Как было сказано выше, описав тип объекта CAPACITOR, мы можем задавать схемы с любым количеством конденсаторов. Пример структуры данных, соответствующей двум заданным в исходном тексте конденсаторам с именами C1 и C3, приведен на фиг. 1б.

Заметим, что значение атрибута VC у объекта C3 может отсутствовать, так как при описании этого атрибута минимальное количество значений задано равным 0.

Некоторые атрибуты могут быть подчинены другим. В таком случае при описании конкретных данных значением атрибута одного объекта становится имя другого, подчиненного. При описании модели предметной области в ограничениях на значения атрибутов первого объекта задаются типы возможных подчиненных объектов. В структуре данных задачи возникают дополнительные уровни, но в то же время на подчиненные объекты образуется и прямая ссылка с верхнего уровня объектов.

Как показывает опыт, принятый подход к описанию данных позволяет создавать модели весьма широкого круга предметных областей, далеко выходящего за пределы проблем описания электронных схем и систем управления. Кроме этого деление данных на объекты различных типов и задание для них набора атрибутов является естественным и удобным для разработчика. Следует заметить, что вышеописанный подход является практически единственным способом задания данных в системе. Здесь был приведен пример с описанием схемы, но таким же образом описываются, например, и формы выходных

документов, о чем подробнее говорится в [4]. То, что описания данных в модели предметной области и данных для решения задачи обладают одинаковой структурой, создает возможность создания особой модели для описания самих моделей предметных областей, и таким образом описывать их также на входном языке.

3. Входной язык

Текст на входном языке состоит из термов: слов, чисел, переменных, разделителей. Основным разделителем является "пробел". Термы могут быть отделены друг от друга любым количеством пробелов и произвольно расположены на строках, чем достигается так называемый свободный формат. Группы термов образуют предложения, отделенные друг от друга разделителем. Кроме этого весь текст делится на разделы, в каждом из которых предложения несут свои функции.

4. Описание данных на входном языке

Описание данных задачи образует наиболее важный раздел исходного текста, в нем может быть описано неограниченное количество экземпляров объектов, соответствующих какой-либо имеющейся модели.

Первое предложение является ссылкой на используемую модель предметной области и имеет вид

AMP CIRCUIT DESCRIPTION;

Здесь CIRCUIT - идентификатор, присвоенный разработчиком описанию данных, AMP - имя данных задачи, выбранное пользователем, а DESCRIPTION - ключевое слово, определяющее функцию предложения.

В каждом следующем предложении описывается один объект: указывается его имя, тип и присваиваются значения его атрибутам. Так, например, структура, изображенная на фиг. 1б, на входном языке выглядит так:

C1 CAPACITOR NODES = 1,2 C = 1.MKF VC = -1.KV;

C3 CAPACITOR NODES = 5,6 C = 1.MKF;

CAPACITOR – идентификатор типа объекта (из описания данных в модели), C1 и C3 – имена, которые пользователь дает экземплярам объектов. Значения атрибутов C и VC заданы с единицами измерения, соответствующими классу единиц, описанному разработчиком (вектор с идентификатором UNIT).

5. Описание единиц измерения

Для описания единиц измерения разработчик создает объект типа UNITS, атрибуты которого зависят от единиц измерения, используемых в конкретной предметной области. Каждый атрибут определяет класс единиц, например, единицы измерения напряжения, емкости и т.д. Идентификатор для этих атрибутов выбирает разработчик и они же указываются при описании контрольных величин атрибутов других объектов в векторе UNIT (см. фиг. 1а). Допустимые значения задаются двумя множествами: значениями, задаваемыми пользователем в тексте и значениями, создаваемыми в структуре данных (см. фиг. 2а).

В тексте задачи единицы измерения могут быть заданы объектом типа UNITS, где каждому атрибуту (класс единиц) присваивается значение.

Если объект UNITS в тексте не задан, то в структуре данных он создается с атрибутами, значения которых соответствуют стандартным. Он является исключением из общего правила, так как подструктуры данных для неуказанных в тексте объектов в общем случае не создаются. Исключением является также и то, что тип UNITS зафиксирован в трансляторе и другой тип объекта для задания единиц измерения использовать нельзя.

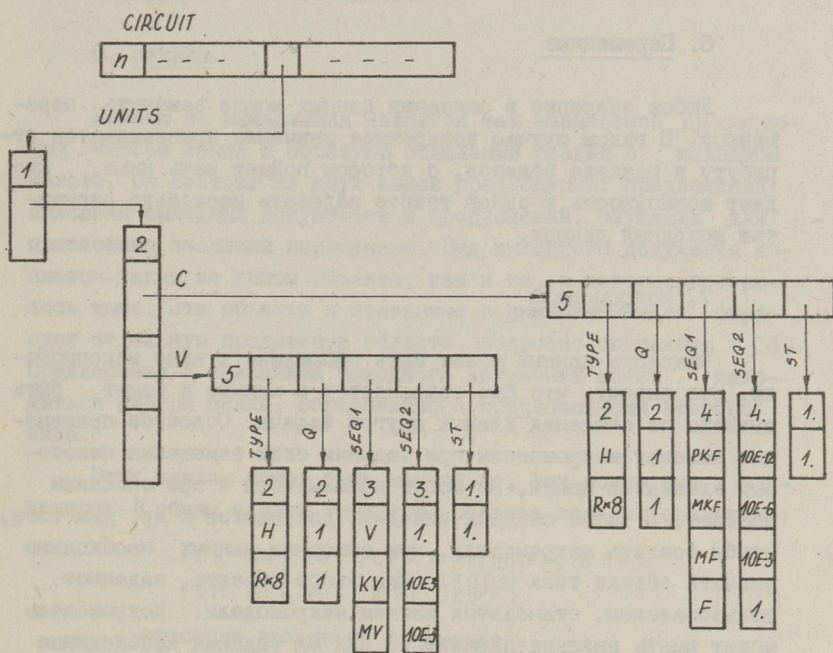
При описании других объектов в тексте можно изменить значение единицы измерения, указав ее непосредственно за значением атрибута. Например,

```
U1 UNITS V=KV C=F;
```

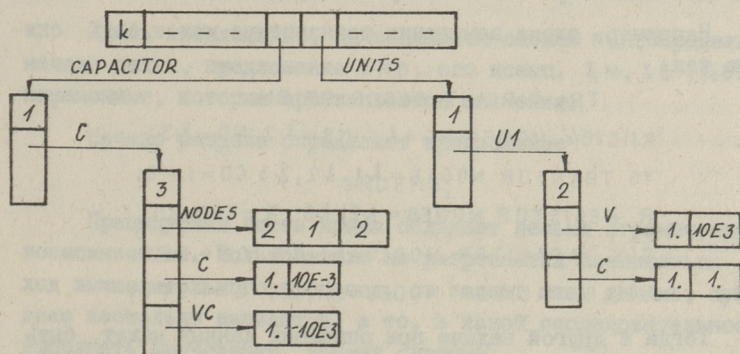
```
C1 CAPACITOR NODES=1.2 C=1.MKF VC=-1;
```

Здесь в объекте C1 единицы измерения напряжения взяты из U1, а единица измерения емкости указана непосредственно за значением. В любом случае значения атрибутов попадают в структуру данных нормированными, т.е. умноженными

а.



б.



Фиг. 2. Структура объекта UNITS в модели (а) и объектов U1 и C в описании данных задачи (б).

на значение элемента вектора SEQ 2 (фиг. 2а) соответствующего заданной единице (см. фиг. 2б).

6. Переменные

Любое значение в описании данных можно заменить переменной. В таком случае конкретное значение присваивается атрибуту в разделе заказов, о котором пойдет речь ниже. Это дает возможность в одном тексте задавать несколько вариантов исходных данных.

7. Макромодели

Описание данных может быть оформлено в виде макромодели. Это значит, что оно сохраняется в архиве и может быть вызвано из описания данных другой задачи. Основное применение находят макромодели при задании схем замещения некоторых элементов цепей, но могут применяться и при описании подсхем, а также сложных выходных документов и др. Для того, чтобы создать макромодель, при описании данных необходимо указать объект типа MODEL. Имя этого объекта, заданное пользователем, становится именем макромодели. Макромодель может иметь внешние параметры. Для их задания необходимые значения в описании данных оставляют переменными, и те же переменные являются значениями атрибутов объекта MODEL, причем идентификаторы этих атрибутов определяет пользователь.

Например, схема замещения транзистора может быть описана так:

TRL CIRCUIT DESCRIPTION;

'KPC104' MODEL NG=&1 NS=&2 ND=&3;
TS TRANSOR NODES=&1, &2, &3 CO=1.MS;
R RESISTOR NODES=&2, &3 R=100.KO;
CGS CAPACITOR NODES=&1, &3 C=4.5 PF;
GGD CAPACITOR NODES=&1, &3 C=1.5 PF;

Тогда в другой задаче при описании данных может быть использован объект с типом 'KPC104' и атрибутами NG, NS, ND:

T1 'KPC104' NG=28 NS=2 ND=1;

К образованной в ходе трансляции структуре данных объекты из макромодели присоединяются без дополнительного ур-

ня, так что для решения задачи их использование дополнительных трудностей не создает.

8. Заказы

Заказы на вычисления являются так называемой процедурной частью языка и образуют отдельный раздел в исходном тексте. Он состоит из двух видов предложений: предложений описания выходных документов и предложений, служащих для присвоения значений переменным. Вид выходного документа является таким же типом объекта, как и те, о которых говорилось выше. Эти объекты и связанные с ними отношения образуют отдельную предметную область, подробно описанную в [4]. Предложение с описанием выходного документа может находиться внутри цикла, образованного предложениями второго вида.

Этих циклов может быть несколько, они могут быть вложенными. В общем случае структура раздела заказов следующая:

```
FOR &N=28 &C1=2.PF, 10.PF, 18.PF;
```

```
FOR &C2.PF, 10.PF, 18.PF;
```

```
< описание выходного документа >
```

```
END;
```

```
END;
```

Здесь предложение, начинающееся словом FOR, определяет начало цикла, предложение END; его конец. &N, &C1, &C2 - переменные, которым присваиваются значения.

Начало раздела определяет предложение

```
ORDERS;
```

Процедурная часть языка обладает весьма ограниченными возможностями. Пользователю не разрешается вмешиваться в ход вычислительного процесса, он задает лишь данные, пусть даже несколько вариантов, а то, в какой последовательности выполнять вычисления, решает система.

Достоинства такого подхода подробнее рассматриваются в [5].

9. Структура исходного текста

Кроме описанных выше разделов в текст включается еще один - шапка. Функции предложений этого раздела заключаются в задании некоторых общих для всей задачи заглавий и режимов работы системы при ее решении. Например, предложение

"TITLE 'PART OF A DIFFERENTIAL PREAMPLIFIER';

описывает заголовок, который будет печататься на всех страницах выходного документа. Здесь символ " является признаком предложения шапки, TITLE - ключевым словом, определяющим функцию предложения. Предложения шапки находятся в начале текста перед описанием данных, и структура всего текста таким образом следующая:

- 1) шапка,
- 2) описание данных,
- 3) заказы.

Текст заканчивает предложение

EOT;

Описание данных может состоять из нескольких разделов, но задача решается с использованием только одного, последнего, в остальных могут быть описаны макромодели.

10. Некоторые дополнительные возможности

Как видно из предыдущих пунктов статьи, описание данных на входном языке является довольно многословным. Подробное описание повышает наглядность текста, но при наличии большого количества данных становится неудобным. Облегчение труда пользователя происходит за счет применения различных сокращений.

Первые предложения разделов описания данных и заказов включают ключевые слова DESCRIPTION и ORDERS. Им соответствуют сокращенные варианты DSCN и ORD.

Описание типа объекта в предметной области содержит кроме вышеуказанной информации некоторую комбинацию символов. Используя эту комбинацию для задания имени объекта в тексте описания задачи, тип объекта может быть опущен. Кроме этого при задании однотипных объектов необходимо задать тип толь-

ко первого из них, остальным объектам тип присваивается автоматически. Имя объекта также не является обязательной частью его описания, оно может быть сгенерировано транслятором.

Описание атрибутов может не содержать их идентификаторов, в таком случае при задании их значений придерживаются очередности созданной при описании модели предметной области. При задании однотипных объектов с одинаковыми значениями атрибутов значения могут быть взяты из предыдущего предложения. Вместо повторяющегося атрибута в тексте указывается *.

Применение всех вышеизложенных возможностей читатель может проследить на примере, приведенном в конце статьи.

Использование сокращений может свести объем исходного текста до минимума, но в то же время понизить его наглядность.

II. Заключение

К настоящему времени создано множество систем машинного проектирования, особенно систем, решающих задачи проектирования электронных схем. Все они имеют свои входные языки. Эти языки служат в основном для описания данных. Можно заметить, что и манера описания данных в них примерно одинакова. В этом смысле язык системы SPADE ничем особенным не отличается. Основным же его достоинством разработчики системы считают гибкость, т.е. то, что для описания новой предметной области, а значит и для создания новой версии языка требуется лишь создание нового комплекта данных, а не изменение программ. Причем создание этих данных происходит также средствами того же языка. Такое рекурсивное использование оказывается весьма удобным.

/ НАЧАЛО ТЕКСТА НА ЯЗЫКЕ /
/ СИСТЕМЫ * SPAD * /

/*****/

"PROBLEM FILTER1

"PROGRAM LFCA /ПРОГРАММА ЧАСТОТНОГО АНАЛИЗА ЦЕПЕЙ/

"OPTION LET ДОПУСКАЕТСЯ К РЕШЕНИЮ
ДАЖЕ ПРИ ОБНАРУЖЕНИИ ОШИБОК В ЗАДАНИИ/

"TITLE 'LOW PASS FILTER, EXAMPLE 1

/*****/

/ НАЧАЛО ОПИСАНИЯ ДАННЫХ /

FILTER CIRCUIT DESCRIPTION:

RA RESISTOR NODES=1,4 R=0.125K01

C1 CAPACITOR NODES=2,5 C=13.725NF1

C3 3,5 16.56NF1

C5 1,5 9.2596NF1

L2 INDUCTOR NODES=3,2 L=0.19465MH1

L4 1,3 0.1789MH1

RK 2,5 RK1 /RK ВАРЬИРУЕТСЯ/

/*****/

GIVEN GENERAL*DATA NFREQ=1

OPTION CONTROL NN=5 ORDER=3:

ORIGINAL TRANSMITTANCE INPUT1=4 INPUT2=5

OUTPUT1=2 OUTPUT2=5 TYPE=VOLT*GAIN1

/*****/

/ОПРЕДЕЛЕНИЕ ЧАСТОТНОЙ ХАРАКТЕРИСТИКИ/

F1 FREQ*RESPONSE INITIAL*FREQ=0.1592E4

FINAL*FREQ=0.1592E6 STEP=0.041

/КОНЕЦ ОПИСАНИЯ/

/*****/

/НАЧАЛО ЗАКАЗОВ/

ORDERS:

/ПЕРВЫЙ ЗАКАЗ: ЭЛЕМЕНТЫ ЦЕПИ/

FOR ARK=1,0K01

TP TABLE /ТАБЛИЦА/

OBJECT=ORIGINAL,RESISTORS /ОБЪЕКТ - РЕЗИСТОРЫ/
LIST=NAME,NODE1,NODE2,RESISTANCE /ПАРАМЕТРЫ/

TC TABLE

OBJECT=ORIGINAL,CAPACITORS
LIST=NAME,NODE1,NODE2,CAPACITANCE

TL TABLE

OBJECT=ORIGINAL,INDUCTORS
LIST=NAME,NODE1,NODE2,INDUCTANCE

END:

/ВТОРОЙ ЗАКАЗ: ГРАФИК/

FOR ARK=0.5K0,1,0K0,1.5K01

P TWO*FUN*PLOT

OBJECT=F1

X*LIST=REB Y*LIST=REAL*PART Z*LIST=IMAG*PART

X*AXIS=LOGARITHMIC

Y*SCALE=GIVEN*LIMITS

Y*LIMITS=1,1

Z*SCALE=GIVEN*LIMITS

Z*LIMITS=1,1

DEVICE=PLOTTER

:

END:

/*****/

EOT:

0

Л и т е р а т у р а

1. К у к к В. Принципы функционирования системы. См. наст. сб. с. 3.

2. Петренко А.И., Власов А.И., Тимченко А.П., Мачуговский В.С., Ладогубец В.В. Общая характеристика пакета прикладных программ для решения задач схемотехнического проектирования. - Электронное моделирование, Киев, 1979, № 2, с. 96-107.

3. Rubner - Petersen Th. NAP2 - a nonlinear analysis program for electronic circuits. Users manual 16/5-73. Technical University of Denmark, 1973. 320 p.

4. Леппиксон В. Формирование выходных документов. См. наст. сб. с. 47.

5. Кукк В., Венделин Е. Управление вычислительным процессом. См. наст. сб. с. 39.

E. Vendelin

Data Structures and Input Language

Summary

The paper discusses the problems of data description for computer-aided design of circuits and control systems. The data structures generated by the translator of CAD system SPADE are described and the main rules of the input language are presented.

В. Куки, Е. Венделин

УПРАВЛЕНИЕ ВЫЧИСЛИТЕЛЬНЫМ ПРОЦЕССОМ

I. Введение

По мнению авторов, самым дорогим компонентом существующих пакетов программ, предназначенных для проектирования электронной аппаратуры, являются собственно вычислительные процедуры. В то же время разработка именно этих процедур имеет наименьшую поддержку со стороны средств автоматизации проектирования — практически лишь языки программирования высокого уровня.

Вычислительные модули являются в действительности очень сложными программами, хотя в принципе алгоритмы расчета имеют относительно простую логическую структуру. Это подтверждается исследованиями, проведенными для сравнения различных методов расчета, например, в виде операторной записи алгоритмов [1]. Даже методы решения жестких систем дифференциальных уравнений высокого порядка могут быть представлены всего лишь некоторыми элементами структурной программы, если только не вникать в подробности на низших уровнях (стандартные циклы, распределение памяти и т.п.), несущественные для самого алгоритма. Как правило, алгоритмы расчета приобретают особую ясность и прозрачность, если их интерпретировать на языке предметной области [2, 3].

Анализ существующих программ показал, что они насыщены условными операторами, которые, в основном, выполняют следующие функции:

1) обеспечивают надежную работу программы при тривиальных значениях переменных (операторы типа if A then B);

2) обеспечивают прохождение нужного пути в зависимости от характера данных (операторы типа CASE V of ...).

Практика показывает, что основная масса ошибок, и при этом самых неприятных, связана именно с условными операторами. Хорошее структурирование программы позволяет уменьшить количество ошибок, но не снимает полностью проблемы – разработчику не под силу рассмотреть все возможные варианты. Реальный выход заключается в исключении условных операторов из программ высокого уровня. Ведь логические значения – это характеристики данных, т.е. они сами принадлежат к данным. Поэтому при достаточной независимости данных и программ возможно исключить почти все логические операторы из прикладных программ, т.е. соответствующее управление осуществлять системными средствами.

2. Управление вычислениями

Если исходить из предположения, что вычислительный процесс есть модификация некоторой системы данных, которая является отображением конкретной ситуации в предметной области, то естественно возникает проблема осуществления целенаправленного поиска последовательности действий над данными, которая привела бы к определению необходимых величин. Средства, которые позволяют реализовать такой поиск и обеспечить выполнение соответствующих процедур (процессов) мы и называем управлением вычислительным процессом.

Таким образом мы представляем решение задачи как выполнение множества в некотором смысле стандартных действий, которые образуют иерархическую структуру – действие более высокого уровня распадается на более простые поддействия. Поэтому можно управление вычислительным процессом рассматривать как построение такой структуры и реализацию соответствующих действий. Существует много способов реализации этого процесса, среди которых нужно отметить как наиболее четкие структурное программирование [4, 5, 6] и вычислительные модели [7].

По существу можно управление вычислительным процессом представлять сочетанием двух этапов: планирование и выполнение. Разумный компромисс при проведении границы между

этими этапами в зависимости от характера задач является, может быть, самым существенным фактором обеспечения эффективности конечного продукта.

Мы ниже будем классифицировать планирование только по двум уровням:

1) жесткий план, составляемый вручную (на этапе программирования);

2) динамический план, составляемый автоматически, в зависимости от характера конкретной задачи.

Хотя такая классификация является весьма условной и нестрогой, она все же позволяет нам более подробно рассматривать сам процесс планирования.

Жесткий план определяет, в основном, границы класса задач, которые могут быть решены данной прикладной программой. Для расширения возможностей такого плана мы должны предвидеть различные ситуации при выполнении и обеспечить их нормальное завершение. Это приводит к необходимости применения условных операторов в том или ином виде. Отметим, что жесткий план может иметь самые различные формы: 1) программа на языке с процедурной ориентацией; 2) программа на каком-либо макроязыке, которая должна обрабатываться препроцессором или интерпретироваться; 3) таблица решений; 4) вычислительная модель и т.п., которые могут комбинироваться в различных сочетаниях.

Характерным для жесткого плана является фиксированность структуры данных и отношений между данными, которые являются отправным пунктом при разработке прикладной программы. Поэтому и класс задач, которые могут быть решены по этому плану, определяется на этапе проектирования, и любые модификации требуют существенной доработки.

Но в то же время жесткий план позволяет создать эффективные программы, так как есть возможность скомпилировать программу, которая учитывает все особенности плана.

Типичным примером динамического планирования является решение сильно разреженных систем уравнений. Благодаря планированию хода вычислений для конкретной системы уравнений стало возможным повысить скорость решения и размерность задач на порядок.

Другим примером может быть система ПРИЗ, в которой планирование выполняется на вычислительных моделях, а результатом является рабочая программа [8].

Динамическое планирование можно считать средством, действительно обеспечивающим эффективность, так как оно позволяет автоматизировать самый сложный и трудоемкий этап в разработке средств автоматизации проектирования.

Однако следует отметить, что пока удается планировать решение лишь при достаточно существенных ограничениях [8, 9, 10], в частности, чрезвычайно большие трудности возникают при планировании рекуррентных и итерационных процессов. В то же время именно в этих процессах заключаются основные резервы повышения эффективности решения прикладных задач.

Но даже возможность частичного планирования позволяет существенно упростить прикладную программу, которая преобразуется в относительно простую организующую программу, обладающую высокой независимостью от данных и модулей низкого уровня.

3. Отношения и управление в системе SPADE

Система данных, которая преобразуется в ходе решения задачи (база данных задачи) состоит из двух компонентов: собственно данные и связи. Любая связь между данными представляет потенциально возможную операцию (действие, процесс), выполнение которого зависит от конкретной ситуации. Реализацию операции, соответствующей данной связи, мы называем ее активацией.

Здесь необходимо сделать терминологическое замечание. Связи называются также отношениями [7], но при этом может возникнуть противоречие с общеизвестным определением отношения как подмножества прямого произведения (так понимается отношение и в теории баз данных). Мы ниже будем придерживаться термина отношение, уточняя при необходимости, что именно имеется в виду.

Мы будем называть конкретным отношением связи, определенные для конкретных объектов и атрибутов. Примеры: "полусы передаточной функции TFI суть корни ее знаменателя",

"проводимость резистора R_{i4} есть обратная величина его сопротивления" и т.д.

Под абстрактным отношением мы понимаем связь, абстрагированную от конкретных объектов, т.е. определенную на уровне типов и атрибутов. Например: "проводимость резистора есть обратная величина его сопротивления", "частотная характеристика есть Фурье-преобразование импульсной характеристики" и т.д.

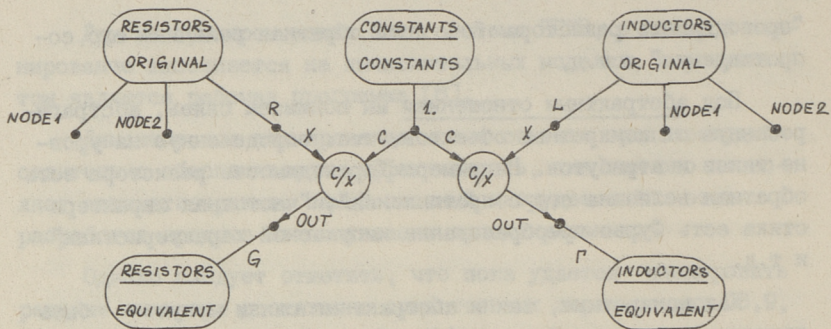
Как конкретные, так и абстрактные связи могут быть сгруппированы по действиям (процедурам), которые необходимо выполнить при активации соответствующих связей. Любая такая группа может быть рассмотрена как отношение в классическом смысле и представлена, например, в виде таблицы Кодда (фиг. 1). Поэтому конкретные связи ниже будут называться также элементами отношений.

Ввиду того, что отношение соответствует определенной вычислительной процедуре, мы будем его называть также оператором (отношением-оператором). Примеры операторов: "коэффициенты $\rightarrow$ корни", "слагаемые $\rightarrow$ сумма", "функция $\rightarrow$ ее Фурье-преобразование", "множество векторов $\rightarrow$ вектор их первых компонентов" и т.д.

Соотношения между введенными понятиями можно характеризовать следующим образом. Абстрактные отношения - это формулы, "законы природы", действующие в данной предметной области. При решении конкретной задачи они порождают множество конкретных отношений, которые группируются в отношения-операторы, т.е. становятся элементами последних. Совершенно ясно, что различные абстрактные отношения могут генерировать элементы одного оператора.

Задачу планирования вычислительного процесса можно ставить как задачу построения последовательности конкретных отношений так, что их последовательная активация приводит к определению необходимых значений.

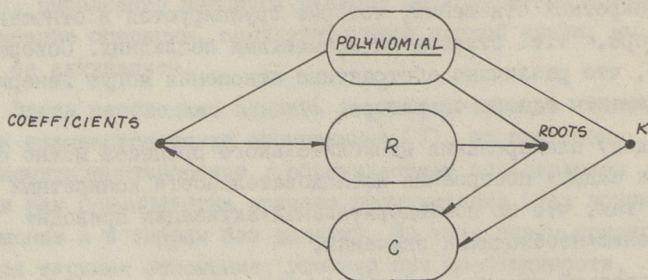
В третьей версии системы SPADE используется жесткое планирование с определением всех конкретных отношений, которые могут понадобиться при решении задач данной прикладной программой. Организация активации их возложена на прикладную программу, которая обеспечивает и управление итерациями и рекурсиями.



C/X

KEY	OUT	DISP	C	X
RADM	G	MOD	1.	R
LGA	Γ	MDD	1.	L

Фиг. 1. Два элемента отношения-оператора C/X из предметной области "электронная схема" в графическом представлении и в таблице Кодда.



Фиг. 2. Отношение R означает вычисление корней полинома, отношение C — вычисление коэффициентов из корней и множителя K.

При этом на отношения накладывалось довольно жесткое условие: оно может иметь лишь один выход, которым может быть только терминальное значение (в смысле, определенном в [II]). Другими словами, операторы должны быть своеобразными "функциями". Такое ограничение существенно облегчает планирование вычислений, так как обеспечивает однозначное соответствие "конкретное отношение — атрибут" (см. также [II]). Следует иметь в виду, что отношение является понятием логического уровня, а физическая реализация модулей может быть различной и необязательно однозначно соответствовать отношениям.

Таким образом, набор отношений является множеством конкретных отношений, каждый из которых содержит следующую информацию:

- 1) идентификация выходного атрибута;
- 2) идентификация входных атрибутов;
- 3) тип отношения (обычно совпадает с именем модуля, выполняющего соответствующее действие);
- 4) диспозиция.

Последний параметр введен для управления активацией отношений. Диспозиция может иметь три значения:

- 1) OLD (выход не обновляется);
- 2) MOD (значение выхода обновляется);
- 3) NEW (создается новый выходной вектор).

В описании отношений задается исходное значение диспозиции, которое может быть увеличено при активации отношения прикладной программой.

Следует отметить, что хотя диспозиция имеет сходство с понятием применяемости отношения [7], ее функции все же другие. При жестком плане применяемость отношения определяется на двух этапах: при составлении плана и при выполнении его. Если в плане определяется принципиальная возможность некоторого действия, то при выполнении соответствующий элемент отношения активируется только тогда, когда диспозиция имеет значение MOD или NEW и существуют все входные атрибуты.

Простейшая прикладная программа состоит по существу только из вызова программы, активирующей подряд все отно-

шения некоторого набора. Если жесткий план не позволяет решать задачу полностью, то прикладная программа содержит несколько таких вызовов, между которыми выполняются нестандартные действия (организация циклов, итераций и т.п.).

Жесткий план должен обеспечить активацию как можно большего количества отношений с тем, чтобы дать возможность решения как можно более широкого класса задач. Например, если имеется объект "полином" с атрибутами "коэффициенты" и "корни", то необходимо определить отношения, задающие связи в обоих направлениях (фиг. 2).

4. Динамическое планирование и управление

Под динамическим планированием мы понимаем планирование решения в зависимости от конкретного задания, т.е. от того, какие атрибуты являются исходными, а какие вычисляемыми.

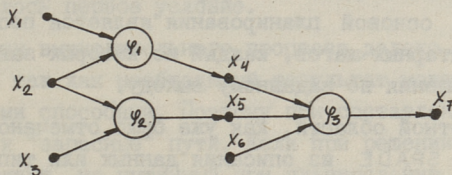
Активация отношений зависит от наличия значений входных атрибутов, которые могли быть, в свою очередь, вычислены с помощью другого отношения и т.д., т.е. в каждом конкретном случае мы можем сказать, что одни элементы отношений должны предшествовать другим. В сумме все отношения, подлежащие активации при решении данной задачи, образуют частично упорядоченное множество и могут быть представлены графом как на фиг. 3.

Задачей динамического планирования и является автоматическое построение этого графа или плана. В ходе решения задачи частичный порядок преобразовывается в линейный. Это преобразование может быть осуществлено несколькими способами, создающими дополнительные возможности для ускорения вычислительного процесса.

Способ построения плана в значительной степени зависит от характера данных, используемых в задаче. С этой точки зрения для класса задач, решаемых в системе SPADE, характерно то, что количество исходных данных сильно превышает количество вычисляемых.

Кроме этого вычисление различных величин из одних и тех же исходных данных предполагает наличие значительно отличающихся друг от друга планов. В подобных задачах планирование целесообразно осуществлять методом обратной волны, т.е.

а.



$\{\varphi_1, \varphi_2, \varphi_3\}$

или

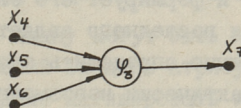
$\{\varphi_2, \varphi_1, \varphi_3\}$

б.

0.

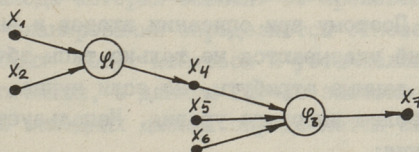
X_7

1.



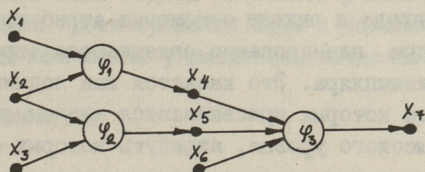
$\{\varphi_3\}$

2.



$\{\varphi_1, \varphi_3\}$

3.



$\{\varphi_2, \varphi_1, \varphi_3\}$

Фиг. 3. Готовый план (а) и процесс планирования (б).

сперва найти отношение, выходом которого является необходимый результат, затем отношения, вычисляющие входы этого отношения и т.д., пока входами не будут исходные данные [8, 9]. Таким образом, основой планирования является последовательность элементарных шагов, каждый из которых заключается в поиске отношения по заданному выходу.

Модель предметной области, как уже было отмечено выше, состоит в системе SPADE из описания данных или типов объектов, которые могут быть использованы в описаниях задачи и описания отношений. Отношения здесь представляют собой связи, которые существуют между атрибутами данных типов объектов.

При построении плана решения конкретной задачи нас интересуют не абстрактные отношения, а отношения, привязанные к атрибутам конкретных экземпляров объектов (элементы отношений). На каждом шаге планировщик, во-первых, по заданному выходному атрибуту находит отношение, выход которого является атрибутом соответствующего типа объекта, во-вторых, определяет атрибуты каких объектов могут быть выходами найденного отношения и формирует его элемент (конкретное отношение). Реализация последнего является довольно сложной проблемой. Однозначно определить экземпляр объекта для входа по его типу практически невозможно, так как такого типа объектов может быть в исходных данных несколько, а может и не быть совсем. Поэтому при описании входов и выходов абстрактных отношений указываются не только типы объектов, которым принадлежат данные атрибуты, но если нужно, то и объекты и атрибуты более высокого уровня. Используется два основных ограничения:

1) если на входе и выходе отношения атрибуты объекта одного типа, то при планировании связываются атрибуты одного и того же экземпляра. Это касается как подчиненных объектов, атрибуты которых присваиваются значениям, так и объектов более высокого уровня, атрибуты которых ссылаются на другие объекты;

2) если на входе и на выходе отношения указаны атрибуты разнотипных объектов, то необходимо задавать объекты, которые используют данные объекты как подчиненные. Уровней ссылок, т.е. уровней объектов, должно быть столько, сколько

необходимо для того, чтобы на самом высоком уровне на входе и выходе отношения оказался один тип объекта, и тем самым выполнилось первое условие.

План вычислительного процесса далеко не всегда однозначен, так как необходимый результат может быть получен несколькими способами. Поэтому при составлении плана можно отметить и "запасные" пути. Если при решении задачи возникает конфликт, не учтенный при планировании, происходит повторное обращение к планировщику, который пытается построить новый путь.

Вопрос выбора основного пути остается в системе SPADE пока открытым. Очевидно критериев может быть несколько, например, наименьшее количество элементов отношений или наименьшее количество используемых атрибутов и т.д.

Построенный один раз план целесообразно сохранить, так как он может быть частично или полностью использован и при вычислении других атрибутов из того же набора исходных данных. Таким образом, при последующих обращениях к планировщику план лишь дополняется недостающими ветвями.

По одному и тому же плану могут быть проведены вычисления для различных значений входных атрибутов. Как уже было показано выше, необходимо активировать лишь элементы отношений, выходы которых зависят от произведенных изменений. При ручном планировании определяются отношения, значения входов которых могут меняться и устанавливается соответствующая диспозиция, в данном случае зависимость отношений от изменений в исходных данных определяется автоматически.

Описанный выше подход позволяет учитывать основные особенности задач проектирования схем и называется в системе SPADE динамическим управлением вычислительным процессом.

6. Заключение

Рассмотренный в настоящей статье подход к управлению вычислениями является ничем иным, как реализацией основного принципа, используемого в системе: замены программ управляющими данными. Здесь такими данными являются описания отношений. При разработке пакета прикладных программ повышается роль изучения предметной области и составления ее

модели. Объем программирования в классическом смысле сводится к минимуму. Каждый новый пакет может использовать ранее созданные отношения, так что приходится программировать лишь модули, реализующие принципиально новые связи.

Определение вычислительного процесса как модификации некоторой совокупности данных имеет то существенное достоинство, что отладка всего пакета практически сводится лишь к отладке модулей, реализующих отдельные отношения. Их совместную работу гарантирует унифицированное представление входных и выходных параметров.

Л и т е р а т у р а

1. И л ь и н В.Н. Основы автоматизации схмотехнического проектирования, М., Энергия, 1979. 392 с.

2. Т и й г и м я з М. Алгоритмы и структуры данных при анализе нелинейных цепей. См. наст. сб. с. 83.

3. Л е п п и к с о н В., Р ю с т е р н Э. Модели линейных систем. См. наст. сб. с. 57.

4. Х ь ю з Дж., М и ч т о м Дж. Структурный подход к программированию. М., Мир, 1980. 279 с.

5. J e n s e n R.W., T o n i e s C.C. Software engineering. Englewood Cliffs, New Jersey, Prentice Hall, 1979. 530 p.

6. Y o u r d o n E., C o n s t a n t i n e L.L. Structured design: fundamentals of a discipline of computer program and systems design. Englewood Cliffs, New Jersey, Prentice Hall, 1979. 473 p.

7. Т ы у г у Э.Х. Решение задач на вычислительных моделях. - ЖВМ и МФ, т. 10, 1970, № 3, с. 716-733.

8. Т ы у г у Э.Х. Решатель вычислительных задач. - ЖВМ и МФ, т. 11, 1971, № 4, с. 992-1004.

9. К а р з а н о в А.В., Ф а р а д ж е в И.А. Планирование вычислений при решении задач на вычислительных моделях. - Программирование, 1975, № 4, с. 19-24.

Ю. Бухштаб Ю.А., Горлин А.И., Камынин С.С., Корягин Д.А., Любимский Э.З.
Об одном методе планирования расчетных цепочек. - Программирование, 1981, № 3, с. 34-38.

II. Кукк В. Принципы функционирования системы SPADE.
См. наст. сб. с. 3.

V. Kukk,
E. Vendelin

Computations Control

Summary

Computation process is supposed to be a sequence of modifications of the "task data base". Any modification can be considered as activation of a relation defined on the data base. Computations control is reduced to planning the sequence which guarantees calculation of the desired values. Different planning strategies are described and compared.

ФОРМИРОВАНИЕ ВЫХОДНЫХ ДОКУМЕНТОВ

I. Введение

По мнению многих, и может быть, даже большинства разработчиков систем машинного проектирования проблемы оформления выходных документов не заслуживают особого внимания и весь вывод данных можно организовать при помощи двух-трех довольно простых программ. На наш взгляд, непопулярность многих систем машинного проектирования можно объяснить именно тем, что их пользователи получают от ЭВМ не готовые рабочие документы, а рулон бумаги с непродуманно размещенными длинными таблицами и трудночитаемыми графиками. Опыт показывает, что "дешифровка" таких выходных материалов и составление по ним новых рабочих документов, по которым можно сделать выводы, сравнить друг с другом различные варианты и т.д., потребует немало времени и считается среди инженеров-проектировщиков крайне неприятной работой. Чтобы еще раз подчеркнуть важность проблемы оформления результатов, напомним, что очень часто инженер-разработчик обращается к ЭВМ не сам, а через специалиста по системе машинного проектирования. Тот кодирует для него начальные данные и сдает их в вычислительный центр, но разумеется, уже не может анализировать полученные результаты. Это значит, что пользователь может и не знать входной язык, но несмотря на это, распечатка выходных данных должна быть для него понятна.

Исходя из вышеуказанных причин, при разработке системы SPADE много внимания было уделено именно вопросам оформления выходных документов. Целью настоящей статьи не является описание того, как построена и как работает часть системы, которая собирает выводимые данные в одну структуру, готовит их к выводу и наконец организует их непосредственный вывод

на внешние носители. Основное внимание уделяется на то, по каким общим принципам организовано оформление выходных документов в системе SPADE, чтобы именно пользователь может получить от ЭВМ и каким образом он может выбирать себе подходящие ему формы представления. Читатель, конечно, довольно быстро заметит, что большинство сформулированных здесь правил кажутся совсем естественными и тривиальными. Тем удивительнее, что разработчики систем машинного проектирования этих вопросов до сих пор серьезно не продумали. Имеющийся у нас опыт показал, что хорошо оформленными выходными документами можно резко повысить эффективность использования системы и поднять ее авторитет среди пользователей.

2. Принцип организации вывода данных в системе SPADE

По основным принципам построения системы SPADE [1] все данные из базы данных доступны пользователю. Другими словами, система обладает средствами, при помощи которых пользователь может заказать вывод на внешние носители любых данных, указывая их идентификаторы и требования к оформлению. Для этой цели в состав системы входит специальный комплекс программ, который в дальнейшем будем называть системой оформления выходных документов или просто системой вывода. Благодаря своей универсальности система вывода, как и транслятор, не требует никакой переделки при расширении системы на новые области применения. Можно добавить, что возможность заказать вывод абсолютно всех, в том числе и промежуточных данных, дает разработчику системы дополнительные средства отладки. Пользователь, в свою очередь, довольно часто интересуется промежуточными данными, например, для того, чтобы убедиться в правильности полученных им окончательных результатов.

Выходные документы системы SPADE могут быть в виде графиков или таблиц. Заметим, что в виде таблицы можно оформлять и результат, состоящий всего лишь из одного единственного числа или слова. В системе SPADE принято, что один выходной документ (одна таблица или один график) связан только с одним объектом, т.е. его содержанием могут быть величины, являющиеся значениями атрибутов одного и того же объекта. Ясно, что это не противоречит основному принципу вывода, так

как в системе SPADE число выходных документов ничем не ограничивается. Но в то же время отсюда уже видно, что пользователь может действительно заказать вывод любых данных, но не в любой форме. Другими словами, возможности пользователя управлять оформлением выходных документов ограничены. Основной причиной этого является то простое обстоятельство, что система вывода не может быть слишком громоздкой. На практике, конечно, трудно решить, какими именно должны быть ограничения и чем должен управлять пользователь. Многое зависит здесь от опыта разработчиков системы в той области техники, для которой система машинного проектирования предназначена. Но прежде всего мы должны выяснить, для какой цели будут использованы выходные документы. Ясно, что если мы хотим получить от ЭВМ готовую рабочую документацию, оформленную со соблюдением всех соответствующих стандартов, то пользователь должен иметь возможность детально описывать свои выходные документы. На наш взгляд, составление окончательной проектной документации является специфической и самостоятельной работой, для проведения которой лучше всего разрабатывать специальные системы программ. Те системы машинного проектирования, которые созданы для проведения различных конструкторских расчетов, должны обеспечить пользователя прежде всего хорошими рабочими документами, не требующими специальных объяснений, "дешифровки", перерисовывания графиков и т.д. Никаких особых требований к оформлению рабочих документов обычно не выдвигается и пользователь готов принимать выходные документы любого вида, если он только может их легко понять и непосредственно использовать на следующих этапах проектирования. Можно даже сказать, что пользователь как инженер-проектировщик, как правило, не хочет сам заниматься подробностями оформления и радуется, если возможности управления выводом сведены к необходимому минимуму.

Пользователь системы SPADE не может управлять выводом таблиц. Это означает, что заказ на вывод таблицы может содержать лишь идентификаторы выводимых величин (напомним, что они могут быть связаны только с одним объектом), а все остальное, как, например, расположение таблицы на внешнем носителе, надписи, интервалы между столбцами и т.д., определяется автоматически соответствующими программами системы

вывода. При изображении графиков пользователь может выбрать формат рисунка (из соответствующего набора форматов, способ масштабирования осей (имеется несколько различных методов) и некоторые другие характеристики. Расположение и общий вид рисунка, размеры координатной сетки, число масштабных делений, надписи и т.д. определены так называемыми макетами рисунка. Для каждого формата на каждом внешнем устройстве (АЦПУ, графопостроитель) создан свой макет, по которым работают программы вывода. Заметим, что все рисунки, независимо от внешних устройств и форматов, похожи друг на друга. Все эти вопросы будут более подробно рассмотрены в следующих пунктах настоящей статьи.

3. Основные правила оформления выходных документов

Как уже отмечено, выходные документы системы машинного проектирования должны быть легко понятными и удобными для использования их на последующих этапах работы. Для достижения этой цели выходные документы системы SPADE оформляются по следующим основным правилам:

1. Распечатка разделена на пронумерованные страницы формата I2 (297x420 мм). Благодаря этому пользователь, разрезая распечатку, получает стандартные листы, которыми удобно обращаться и которые можно переплести в отчет. Графики можно изображать и на страницах с форматами I3, I4 и I5. Лист бумаги графопостроителя разделяется на страницы формата I2 или 22.

2. Все величины, имеющие единицу измерения, как, например, время, напряжения и т.д. выводятся вместе с ними.

3. Надписи и объяснительные заметки к выходным документам составлены так, чтобы документ был понятен и тем, кто не знает входного языка системы.

4. Расчеты с различными вариантами начальных данных производятся, как правило, для того, чтобы потом сравнить результаты. Чтобы соответствующие графики были сравнимыми, их изображают или на одном рисунке, или каждый отдельно, но в последнем случае шкалы осей всех графиков совпадают.

4. Описание выходных документов

Для получения одного выходного документа пользователь должен оформить один заказ или, другими словами, описать нужный ему выходной документ. Предложения для таких описаний пишутся по стандартным правилам входного языка и сам выходной документ является всего лишь одним объектом из совокупности объектов, принадлежащей данной задаче [1, 2]. Не следует, однако, забывать, что указанием идентификаторов выводимых величин мы не только описываем выходной документ, но и представляем машине конкретную задачу для расчета. Другими словами, описания выходных документов являются одновременно и командами. Из-за этого выходные документы описываются в специальном разделе исходного текста, куда кроме них входят и предложения для организации циклов [2].

Ясно, что при выводе типовых величин, как, например, переходной характеристики, многие атрибуты в описании выходного документа имеют всегда одни и те же значения. Для облегчения труда пользователя целесообразно составить типовые заказы в виде макромоделей [2].

5. Вывод таблиц

Принято, что все таблицы, независимо от их размеров, оформляются по столбцам, т.е. значения одного атрибута сосредоточены в один столбец. Размеры таблиц ничем не ограничены. При необходимости программа размещения разделяет таблицу на подтаблицы, которые в зависимости от их размеров размещаются на поле страницы или рядом друг с другом или одна за другой. Не касаясь подробностей заметим, что основным критерием при разработке алгоритма размещения было получение хорошо обозреваемых таблиц, сосредоточенных, если это только возможно, на одной странице.

Для иллюстрации приведем следующий пример описания таблицы:

```
DOC TABLE OBJECT=TF1 LIST=ZEROES, POLES;
```

Данным предложением описывается выходной документ с именем DOC и типом TABLE. Значениями двух атрибутов OBJECT и

LIST определяются идентификаторы выводимых величин. В результате выполнения данного заказа печатается таблица, содержащая значения атрибутов ZEROES и POLES объекта TFI. Наконец заметим, что слова ZEROES и POLES фигурируют в таблице в качестве надписей столбцов.

6. Вывод графиков

Опыт показывает, что разработчики электронной аппаратуры и средств автоматики пользуются в своей работе главным образом только двумя основными видами графиков:

- 1) графики функций одной переменной;
- 2) параметрические кривые (например, годограф частотной функции).

Различие между этими графиками с точки зрения их вывода состоит в том, что при изображении параметрических кривых масштабные коэффициенты обеих осей должны быть равными. Кроме того, для работы с параметрическими кривыми потребуется и знание значений параметра (например, частоты) в определенных точках графика.

На практике часто для решения какого-нибудь вопроса требуется одновременное рассмотрение нескольких (как правило, двух) различных графиков, имеющих один и тот же диапазон изменения аргументов. В качестве примера можно назвать амплитудную и фазовую частотную характеристику. Такие кривые следует разместить в одном поле.

При выводе на графопостроитель следует различать графики непрерывных и дискретных функций (на АЦПУ этой проблемы, конечно, не существует). Анализ показывает, что никаких особых трудностей здесь не возникает и всю проблему удастся решить введением дополнительного атрибута в описание графика и расширением системы вывода двумя-тремя подпрограммами.

Особым видом графиков являются корневые годографы. С точки зрения вывода корневой годограф можно рассматривать как семейство параметрических кривых, размещенных в одном поле. Как было отмечено, система SPADE обеспечивает пользователя возможностью разместить в одном поле кривые, полученные в результате расчетов с различными данными. Отсюда

вытекает, что для решения проблемы вывода корневых годографов необходимо организовать их расчет так, чтобы каждая ветвь оказалась результатом расчета какого-то отдельного варианта данных.

Анализ проблем, связанных с масштабированием и непосредственным выводом (вычерчиванием или печатью) различных графиков показал, что целесообразно ввести три типа графических выходных документов:

1. Кривая функции одной переменной. Обе оси таких графиков масштабируются независимо друг от друга. Способы масштабирования определяются значениями соответствующих атрибутов описания (вообще в распоряжении пользователя имеется 5 различных способов масштабирования). Шкалы обеих осей могут быть равномерными или логарифмическими.

2. Пара кривых двух функций одной переменной, имеющих один и тот же диапазон изменения аргумента. Рисунок данного типа имеет две независимые шкалы для значений функции. Все атрибуты описания такого рисунка аналогичны соответствующим атрибутам предыдущего типа.

3. Параметрическая кривая. Обе оси имеют один и тот же масштабный коэффициент, который можно определить двумя различными методами. При желании пользователя в каждой j -ой точке кривой наносится маркер, а к рисунку присоединяется таблица значений параметра в отмеченных точках (целое число j определяется в описании рисунка).

Для иллюстрации приведем пример описания кривой функции одной переменной:

```
FIGURE FUN_PLOT OBJECT=SR1 X_LIST=TIME  
Y_LIST=STEP-RESPONSE DEVICE=PRINTER FORMAT=I2  
X-AXIS=LINEAR Y-AXIS=LINEAR  
X-SCALE=NOT-ROUNDED Y-SCALE=FREE;
```

На описанном здесь рисунке с именем FIGURE и типом FUN_PLOT изображается функциональная зависимость между атрибутом TIME и STEP-RESPONSE объекта SR1. Рисунок выводится на АЦПУ на стандартной странице формата I2 (см. значения атрибутов DEVICE и FORMAT). Обе шкалы должны быть линейными (см. атрибуты X-AXIS и Y-AXIS). Значения атрибутов X-SCALE и Y-SCALE служат для определения способов масштабирования обеих осей (см. следующий пункт).

7. Алгоритмы масштабирования

В системе SPADE размеры рисунка и число масштабных делений шкалы предопределены макетом рисунка. Из-за этого задача выбора масштаба сводится к определению граничных точек шкалы.

Для масштабирования рисунков первых двух типов (см. предыдущий пункт) разработаны следующие режимы масштабирования:

1. Автоматическое масштабирование без округления. В этом случае крайними точками шкалы будут точные значения максимума и минимума соответствующей величины (т.е. аргумента или значения функции).

2. Автоматическое масштабирование с округлением. В этом случае крайние точки шкалы определяются по округленным значениям максимума и минимума. В результате этого диапазон шкалы будет немного шире диапазона рассматриваемой величины, т.е. часть поля рисунка остается неиспользованным. Точность из-за этого обычно меньше, чем в режиме без округления, но читаемость рисунка заметно улучшается. Режим масштабирования с округлением является основным при изображении графиков в начальной стадии проектирования, когда еще мало информации о поведении функции.

3. Автоматическое масштабирование с "красивой" маркировкой шкалы. В этом режиме пользователь может задавать в описании рисунка допустимые значения для мантиссы расстояний между двумя соседними делениями шкалы. Такими мантиссами можно выбирать, например, числа 0.1, 0.15, 0.2, 0.25, 0.3333, 0.5 и 0.75. В результате получается шкала с начальной точкой $k_1 \cdot \alpha \cdot 10^{k_2}$ и расстоянием между делениями $\alpha \cdot 10^{k_2}$, где α одна из допустимых мантисс, а k_1 и k_2 — целые числа. Полученные по этому методу графики обладают хорошей читаемостью, но их точность иногда невысока.

4. Автоматическое масштабирование с заданными значениями для одного или двух делений шкалы. Как и в режиме с округлением, так и в данном режиме граничные значения шкалы определяются исходя из округленных значений экстремумов рассматриваемой величины, но с учетом дополнитель-

ных требований для шкалы. Неудачный выбор заданных значений может существенно повлиять на точность графика или даже оставить часть графика за пределами рисунка.

5. Масштабирование с заданными граничными точками шкалы. Значения граничных точек задаются соответствующими атрибутами описания рисунка.

Пример: пусть экстремумы равны - 1.789 и 6.245. Шкала разделена на 10 интервалов. Тогда маркировка оси будет следующей:

1. При автоматическом масштабировании без округления: -1.789, -0.9856, -0.1822, 0.6212, 1.425, 2.228, 3.031, 3.835, 4.638, 5.442, 6.245.

2. При автоматическом масштабировании с округлением: -2.7, -1.8, -0.9, 0.0, 0.9, 1.8, 2.7, 3.6, 4.5, 5.4, 6.3.

3. При автоматическом масштабировании с "красивой" маркировкой: -3.0, -2.0, -1.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0 (значением допустимой мантиссы было выбрано 0.1).

4. При автоматическом масштабировании с заданным значением 5.0: -6.7, -5.4, -4.1, -2.8, -1.5, -0.2, 1.1, 2.4, 3.7, 5.0, 6.3.

5. При масштабировании с заданными граничными точками -2.0 и 8.0: -2.0, -1.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0.

Для масштабирования параметрических кривых имеется два режима:

1. Автоматическое масштабирование с округлением. Этот метод аналогичен одноименному методу масштабирования графиков функции одной переменной. Единственным различием является то, что обе оси масштабируются одинаково.

2. Масштабирование по заданной окружности. Координаты центра и радиус окружности задаются пользователем при помощи соответствующих атрибутов описания рисунков. Граничные точки шкалы выбираются так, чтобы окружность точно поместилась на поле рисунка.

Л и т е р а т у р а

1. К у к к В. Принципы функционирования системы SPADE. См. наст. сб. с. 13.

2. Венделин Е. Структуры данных и входной язык
См. наст. сб. с. 19.

V. Leppikson

Representation of Output Data

Summary

The computer-aided design system SPADE contains a multi-purpose output program for presenting the output data in the tabular form or in graphical representations. The paper discusses the basic demands for the output documents of a CAD system and describes the standard documents established in the SPADE. Finally, a classification of graphical representations is given and some methods for choosing the scaling factor are presented (2 ref.).

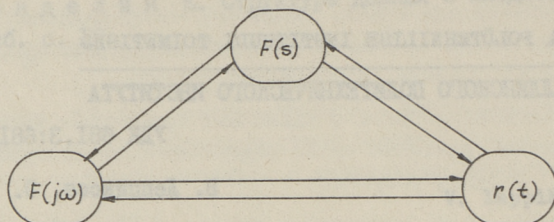
МОДЕЛИ ЛИНЕЙНЫХ СИСТЕМ

I. Введение

Основные принципы построения системы SPADE изложены в [1, 2]. Настоящая статья посвящена вопросам применения системы SPADE для анализа линейных систем. При этом предполагается, что система является одномерной, непрерывной и стационарной. Разработка прикладной программы для решения таких задач является, конечно, всего лишь первым и самым простым шагом машинного анализа систем. Но в то же время нельзя забывать, что большинство практических задач происходит и сейчас именно из этой ограниченной области и данную работу можно рассматривать как создание базы для решения более сложных проблем. Непосредственной целью настоящей статьи является исследование предметной области задачи анализа линейных систем и построение моделей для проведения расчетов. Немного упрощая можно сказать, что нас интересует, как выглядит эта хорошо известная задача в свете идей из [1] и [2]. Таким образом настоящая статья является в том числе и иллюстрацией к названным материалам.

2. Основная задача машинного анализа линейных одномерных систем

Как известно, линейная одномерная система может быть описана уравнениями состояния, передаточной функцией $F(s)$, частотной функцией $F(j\omega)$, импульсной функцией $h(t)$ и переходной функцией $r(t)$. Но при решении прикладных задач в большинстве случаев используются лишь функции $F(s)$, $F(j\omega)$ и $r(t)$. Из-за этого основной задачей машинного анализа являются всевозможные преобразования между этими тремя представлениями. Схема анализа изображена на фиг. 1.



Фиг. 1. Общая схема машинного анализа линейных систем.

Для уточнения данной задачи рассмотрим, каким образом могут быть представлены названные три функции. Дробнорациональную функцию $F(s)$ можно записать в виде

$$F(s) = \frac{n(s)}{d(s)},$$

где

$$n(s) = \sum_{i=0}^m b_i s^i = k_n \prod_{i=1}^m (s - z_i)$$

и

$$d(s) = \sum_{i=0}^n a_i s^i = k_d \prod_{i=1}^n (s - p_i).$$

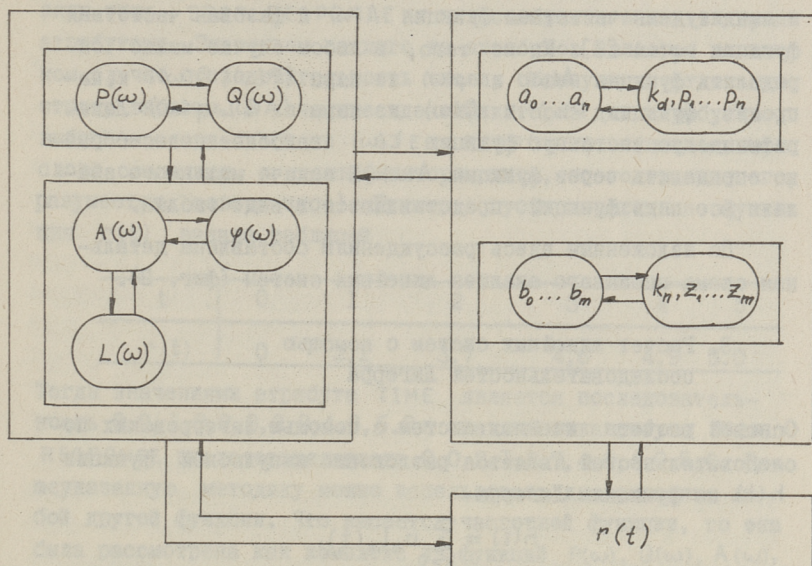
Другими словами, передаточная функция $F(s)$ представляется двумя многочленами $n(s)$ и $d(s)$, а оба многочлена, в свою очередь, могут быть заданы или коэффициентами, или корнями и множителем. В ходе решения задачи довольно часто приходится переходить от одного вида представления к другому.

Аналитическое выражение переходной функции $r(t)$ инженера-проектировщика, как правило, не интересует. Отсюда следует, что в задаче машинного анализа линейных систем функция $r(t)$ всегда представляется в виде таблицы, а вне памяти ЭВМ обычно в виде графика.

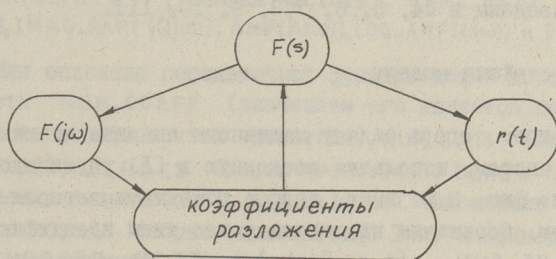
Частотную функцию

$$F(j\omega) = P(\omega) + jQ(\omega) = A(\omega)e^{j\varphi(\omega)}$$

можно рассматривать как комплекс, состоящий из пяти отдельных функций — $P(\omega)$, $Q(\omega)$, $A(\omega)$, $\varphi(\omega)$ и $L(\omega) = \log A(\omega)$. Известно, что в случае устойчивых систем действительная частотная функция $P(\omega)$ и мнимая частотная функция $Q(\omega)$ однозначно связаны, т.е. по любой из них можно определить другую. В случае минимально-фазовых систем однозначно связаны



Фиг. 2. Детальная схема машинного анализа линейных систем.



Фиг. 3. Общая схема машинного анализа с помощью последовательностей Лагерра.

и амплитудная частотная функция $A(\omega)$ и фазовая частотная функция $\varphi(\omega)$ [3]. Кроме того, в любом случае можно определить функции $A(\omega)$ и $\varphi(\omega)$ из пары $(P(\omega), Q(\omega))$ и, наоборот, функции $P(\omega)$ и $Q(\omega)$ из пары $(A(\omega), \varphi(\omega))$. Логарифмическую частотную функцию $L(\omega)$ наиболее целесообразно определить через функцию $A(\omega)$. В задаче машинного анализа все пять функций представляются в виде таблиц.

По изложенным здесь рассуждениям составлена детальная схема машинного анализа линейных систем (фиг. 2).

3. Расчет линейных систем с помощью последовательностей Лагерра

Основой расчета линейных систем с помощью лагеровских последовательностей является разложение импульсной функции $h(t)$ по функциям Лагерра

$$h(t) = \sum_{k=0}^{\infty} a_k l_k(t).$$

Общая схема машинного анализа с помощью последовательностей Лагерра показана на фиг. 3. Отметим, что приведенная здесь схема не является единственной возможностью организации вычислений на базе лагеровских разложений и была выбрана, главным образом, из-за своей простоты. Основы данного метода и алгоритмы для изображенных на фигуре переходов приведены в [4, 5, 6, 7, 8, 9, 10, 11].

4. Простейшие модели

Рассмотрим теперь задачу машинного анализа линейных одномерных систем, используя введенную в [II] терминологию. Как видно из фиг. 3, мы имеем дело с объектами четырех различных типов. Обозначим имеющиеся у нас типы идентификаторами TRANS_FUN (т.е. $F(s)$), FREQ_RESPONSE ($F(j\omega)$), STEP_RESPONSE ($r(t)$) и SERIES (последовательность Лагерра) и выясним, какие атрибуты необходимы для описания объектов каждого типа.

Как уже было сказано, в задачах анализа функция $r(t)$ представлена всегда таблицей, или по-другому, двумя последовательностями чисел. Отсюда можно сделать вывод, что обь-

екты типа STEP_RESPONSE целесообразно описать двумя атрибутами. Значением каждого из атрибутов является в таком случае последовательность чисел, образующая отдельную строку таблицы. В качестве идентификаторов для атрибутов выбираем слова TIME и STEP_RESPONSE (последнее слово совпадает с идентификатором типа, но содержание его, разумеется, совсем иное). Пример: пусть переходная функция $r(t)$ задана таблицей

t	0	1	2	3	4	5
r(t)	0	2.1	3.7	4.6	4.9	5.0

Тогда значениями атрибута TIME является последовательность 0.0, 1.0, 2.0, 3.0, 4.0, 5.0 и значениями атрибута STEP_RESPONSE последовательность 0.0, 2.1, 3.7, 4.6, 4.9, 5.0. Вышеуказанную методику можно использовать для описания любой другой функции. Что касается частотной функции, то она была рассмотрена как комплекс из функций $P(\omega)$, $Q(\omega)$, $A(\omega)$, $L(\omega)$ и $\varphi(\omega)$. Для опережения излишних трудностей как в описаниях, так и в вычислениях, потребуем, чтобы все пять функций были бы определены на одних и тех же значениях частоты. В таком случае полная таблица частотной функции будет состоять из шести строк и для ее описания нам необходимо ввести шесть атрибутов. Обозначим их идентификаторами ANG_FREQ (угловая частота), REAL_PART (значения $P(\omega)$), IMAG_PART ($Q(\omega)$), AMP ($A(\omega)$), LOG_AMP ($L(\omega)$) и PHASE ($\varphi(\omega)$).

Для описания передаточной функции можно использовать атрибуты NUM_COEFF (значением его является последовательность коэффициентов числителя), NUM_CONST (множитель числителя), ZEROES (корни числителя), DENOM_COEFF (коэффициенты знаменателя), DENOM_CONST (множитель знаменателя) и POLES (корни знаменателя). Так, например, для описания передаточной функции

$$F(s) = \frac{s+1}{s^2+5s+6}$$

можно составить предложение:

FS TRANS_FUN NUM_COEFF = 1., 1.

DENOM_COEFF = 1., 5., 6;

Но та же функция может быть описана и предложением:


```
FS TRANS_FUN NUM_CONST=1. ZEROES=-1.+J*φ.
```

```
DENOM_COEFF=1.,5.,6;
```

Отсюда видно, что числитель может быть в исходном тексте задан или атрибутом NUM_COEFF или атрибутами NUM_CONST и ZEROES. Аналогичное правило относится и к знаменателю. Другими словами, для полного описания передаточной функции в исходном тексте задачи имеется несколько возможностей. Но это значит, что, проверяя лишь наличие или отсутствие отдельных атрибутов и не обращая внимания на взаимные связи между ними, мы уже не можем обнаружить всевозможные ошибки в исходном тексте [12]. Так, например, транслятор не может считать ошибкой отсутствие атрибута NUM_COEFF, так как числитель может быть задан и множителем и корнями. По аналогичной причине не считается ошибкой отсутствие атрибута ZEROES. Следовательно, транслятор пропускает и предложение, в котором числитель вообще не описан. Таким же образом пропускается предложение, содержащее все три атрибута числителя. В последнем случае данные заданы избытком. Так как избыточность в начальных данных мешает организации вычислительного процесса, то и такой исходный текст следует считать ошибочным. Но в то же время ясно, что составить универсальную программу для обнаружения всех таких неформальных ошибок не удастся (нетрудно заметить, что аналогичные проблемы возникают и при описании частотной функции). Поэтому при составлении вычислительных алгоритмов следует всегда считаться с тем, что исходные данные могут быть неполными, заданы с избытком или вообще отсутствовать.

Отметим, что описание передаточной функции может быть организовано и по-другому. Выведем многочлен как новый тип объектов. Атрибутами многочлена (POLYNOMIAL) являются коэффициенты (COEFF), множитель (CONST) и корни (ROOTS). Числитель и знаменатель приведенного выше примера задаются теперь предложениями

```
P1 POLYNOMIAL CONST=1. ROOTS=-1.+J*φ;
```

и

```
P2 POLYNOMIAL COEFF=1.,5.,6;
```


Объекты типа TRANS_FUN имеют теперь атрибуты NUM и DENOM, значениями которых являются имена соответствующих многочленов:

```
FS TRANS_FUN NUM=P1 DENOM=P2;
```

Последний способ описания кажется более естественным, так как мы все время рассматривали оба многочлена отдельно друг от друга. Но теперь для описания передаточной функции потребуется уже три предложения, а в условиях жесткого планирования увеличение числа объектов заметно затрудняет организацию вычислительного процесса [2]. Кроме того, оказывается невозможным вывод всей функции $F(s)$ в виде одной комплексной таблицы [13]. Из-за этого в настоящей версии системы SPADE от данного способа описания передаточной функции пришлось отказаться.

Наконец, для описания объектов типа SERIES необходимо иметь только один атрибут SER, значением которого является сама последовательность Лагерра.

Приведенные здесь атрибуты можно считать основными, так как их значениями являются величины, занимающие центральное место в задачах анализа линейных систем. Но как мы увидим в дальнейшем, для решения задач на ЭВМ придется ввести целый ряд дополнительных атрибутов и даже расширить имеющийся набор типов объектов.

5. Расширение основной задачи

Во многих практических задачах анализа линейных систем фигурируют и такие величины и понятия, о которых мы до сих пор еще не упоминали. Некоторые из них можно встретить довольно часто и поэтому любая система машинного проектирования должна давать своему пользователю возможность оперировать с ними. Рассмотрим теперь, каким образом нам следовало бы дополнить имеющиеся наборы атрибутов (без приведения новых типов объектов).

В некоторых задачах аргументом частотной функции является не угловая частота ω , а ее логарифм. В других задачах в качестве аргумента частотной функции выбрана частота f (в герцах). Поэтому целесообразно включать в состав атрибутов объектов типа FREQ_RESPONSE еще и атрибуты

LOG_FREQ и FREQ. Значения функции $\varphi(\omega)$ могут быть в градусах или в радианах. Для нас это значит, что имеется две фазовые функции и вместо одного атрибута PHASE необходимо использовать атрибуты PHASE_IN_DEG и PHASE_IN_RAD.

Для решения некоторых проблем используются рисунки, на которых изображены нули и полюсы на комплексной плоскости переменной s . Чтобы описывать такие рисунки, мы должны ввести атрибуты для вещественных и мнимых частей как для полюсов, так и для нулей [13]. Отметим, что значениями атрибутов ZEROES и POLES являются последовательности комплексных чисел. Итак, набор атрибутов объектов типа TRANS_FUN следует дополнить атрибутами POLE_REAL-PART, POLE_IMAG-PART и двумя аналогичными атрибутами для нулей. Кроме того, для решения некоторых задач потребуются полярные координаты корней, вычеты и т.д. Детальный обзор этих вопросов выходит за рамки настоящей статьи.

6. Определение значений аргументов

Предположим, что целью вычислений является определение переходной функции. Совершенно ясно, что в большинстве случаев неизвестным является не только значение атрибута STEP_RESPONSE, но и значение атрибута TIME. Для определения последнего значения мы должны знать начальный и конечный моменты времени и длину шага (или числа точек на заданном интервале). Отсюда вытекает, что для описания переходной характеристики мы должны ввести еще дополнительные атрибуты для задания параметров, характеризующих интересующий нас промежуток времени. Обозначим их идентификаторами INITIAL_TIME, STEP, FINAL_TIME и NP.

Пример: предложением

```
RT STEP_RESPONSE INITIAL_TIME = 0.
```

```
FINAL_TIME = 10. NP = 100;
```

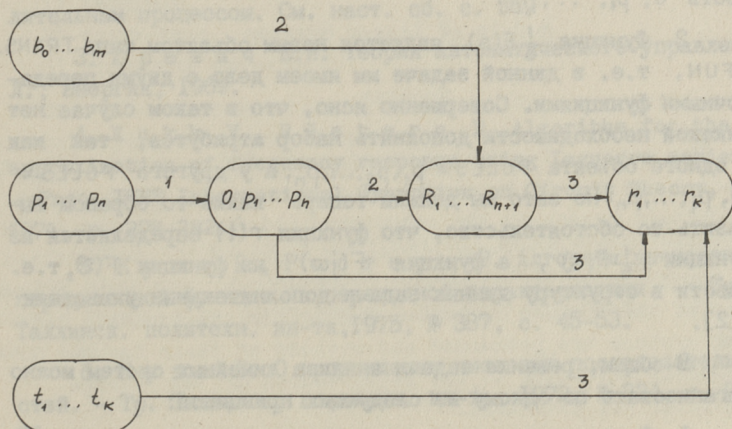
описывается переходная функция в промежутке времени от нуля до десяти секунд. Число точек - 100. Обратим внимание на то, что в данном предложении отсутствуют атрибуты TIME и STEP_RESPONSE. Зато они оба перечислены в соответствующем заказе [13].

Аналогичным образом следует дополнить набор атрибутов объектов типа `FREQ_RESPONSE`. Не касаясь подробностей отметим, что для описания диапазона частоты имеется десять атрибутов. Объясняется это прежде всего тем, что аргументом частотной функции кроме ω может быть и f или $\log \omega$ (см. выше).

7. Принципы дальнейшего развития моделей

До сих пор мы рассматривали задачу анализа линейных систем извне, т.е. мы перечисляли все величины, которые могут быть или начальными, или конечными данными расчета, но не занимались вычислительным процессом. Как уже было сказано, любой вычислительный процесс в системе `SPADE` разделен на элементарные операции [2]. Результатом каждой такой операции (кроме последней) является величина, которую с точки зрения данной задачи можно считать промежуточной (напомним, что для какой-то другой задачи та же величина может оказаться исходной или конечной). В качестве примера рассмотрим определение переходной функции $r(t)$ из передаточной функции $F(s)$. Как известно,

$$r(t) = \mathcal{L}^{-1} \left\{ \frac{1}{s} F(s) \right\},$$



Фиг. 4. Схема расчета переходной функции из известной передаточной функции.

т.е. переходная функция равна обратному преобразованию Лапласа из передаточной функции, дополненной полюсом $s=0$. Схема расчета изображена на фиг. 4. Как видно, вычислительный процесс разделен на три операции: образование нового знаменателя, определение вычетов $R_1, R_2, \dots, R_{n+1}$ и определение значений временной функции $r_1, r_2, \dots, r_k$ в моментах времени $t_1, t_2, \dots, t_k$. Целесообразность такого разбиения вычислительного процесса очевидна. Полюсы и вычеты дробно-рациональной функции $\frac{1}{s} F(s)$ можно считать промежуточными результатами. Но как известно, результат любой элементарной операции включается в базу данных задачи или, другими словами, является значением некоторого атрибута какого-то объекта [2]. Значит, разрабатывая схему вычислений мы вводим в совокупность данных задачи все новые и новые величины и естественно возникает вопрос, как их интерпретировать, т.е. каким объектам они принадлежат. Возвратимся еще раз к нашему примеру. Мы можем выбрать один из следующих подходов:

1. Полюсы и вычеты функции $\frac{1}{s} F(s)$ принадлежат к исходной функции $F(s)$. В таком случае объекты типа TRANS-FUN должны иметь, например, два атрибута для полюсов: один из них (POLES) имеет своим значением последовательность $p_1, p_2, \dots, p_n$, а другой (ADD_POLES) последовательность $0, p_1, \dots, p_n$.

2. Функция $\frac{1}{s} F(s)$ является новым объектом типа TRANS-FUN, т.е. в данной задаче мы имеем дело с двумя передаточными функциями. Совершенно ясно, что в таком случае нет никакой необходимости дополнять набор атрибутов, так как у одного объекта POLES = $p_1, p_2, \dots, p_n$, а у другого POLES = $0, p_1, \dots, p_n$. Но зато мы должны теперь каким-то образом выразить то обстоятельство, что функция $r(t)$ определяется из функции $\frac{1}{s} F(s)$, а функция $F(j\omega)$ из функции $F(s)$, т.е. ввести в структуру данных задачи дополнительные указатели [12].

В общем, решение задачи анализа линейных систем можно организовать по одному из следующих принципов:

1. В совокупности объектов, принадлежащих к одной задаче, из каждого типа может быть только один объект. Объекты могут быть описаны в исходном тексте задачи или введены

в базу данных соответствующими элементарными операциями [2]. В ходе вычислительного процесса определяются значения все новых и новых атрибутов одних и тех же объектов.

2. В ходе вычислительного процесса образуется несколько объектов одного и того же типа. В этом случае нет необходимости вводить новые атрибуты, но зато усложняется структура данных.

При реализации первой версии прикладной программы анализа линейных систем было решено организовать вычислительный процесс по первому принципу. Объясняется это прежде всего тем, что в условиях жесткого планирования решения необходимо иметь как возможно простую структуру данных [2].

Наконец отметим, что в ходе разработки конкретных алгоритмов иногда выясняется, что для управления вычислениями следует ввести еще некоторые дополнительные величины и параметры.

Л и т е р а т у р а

1. К у к к В. Принципы функционирования системы SPADE. См. наст. сб. с. 3.

2. К у к к В., В е н д е л и н Е. Управление вычислительным процессом. См. наст. сб. с. 33.

3. Р е в и ч Е.И. Теория автоматического управления. Л., Энергия, 1969.

4. К у к к V., R ü s t e r n E. Algorithm for the approximation of frequency response using Laguerre series. - Proc. IEEE International Symposium on Circuit Theory, 1974, p. 278-282.

5. К у к к В., Р ю с т е р н Э. Алгоритм вычисления передаточной функции из переходной характеристики. - Тр. Таллинск. политехн. ин-та, 1975, № 387, с. 45-53.

6. К у к к В. О продолжении числовых последовательностей. - Тр. Таллинск. политехн. ин-та, 1972, № 334, с. 77-81.

7. К у к к В., Р ю с т е р н Э. Новый метод идентификации времени чистого запаздывания. - Тр. Таллинск. политехн. ин-та, 1973, № 350, с. II-17.

8. К у к к В. Машинный метод обработки трансцендентных передаточных функций. - Изв. вузов СССР, 1974. Радио-электроника, т. XVII, № 6, с. 26-29.

9. К у к к V. An improved algorithm for rational approximation of transfer functions. - Int. J. of Circuit Theory and Applications, 1977, vol. 5, p. 149-155.

10. К у к к V. State variables and Laguerre series. - Electronics Letters, 1971, vol. 7, p. 269-270.

11. К у к к V. On the use of number series in computer-aided analysis. - European Conference on Circuit Theory and Design, 1974, p. 269-274.

12. В е н д е л и н Е. Структуры данных и входной язык. См. наст. сб. с. 19.

13. Л е п п и к с о н В. Формирование выходных документов. См. наст. сб. с. 47.

V. Leppikson,
E. Rüstern

Models of Linear Systems

Summary

The design of application programs for analysing linear control systems is discussed. The approach is based on the techniques proper to the computer-aided design system SPADE. A linear one-dimensional system is supposed to be described by its transfer function, step response or Laguerre expansion of the impulse response. Models for the analysis of such systems are presented and methods of their further development according to the requirements of the system scheduler are described.

МОДЕЛИ ЭЛЕКТРОННЫХ СХЕМ

I. Введение

Основная задача настоящей статьи — показать применение системы SPADE для анализа электронных схем. Основные принципы построения системы SPADE изложены в [1, 2]. Настоящая статья должна показать, как выглядит предметная область "электронные схемы" в рамках системы SPADE и как строятся модели для проведения расчетов.

Под моделью электронной схемы мы понимаем набор данных и отношений, которые понадобятся для расчета электронной схемы.

В данной статье рассматриваются следующие типы элементов:

- 1) линейные,
- 2) нелинейные,
- 3) источники сигналов.

Это — базисные компоненты, набор которых в действительности увеличивается. Например, ради удобства постановки задачи можно определить такие компоненты, как вольтметры и амперметры.

В данной статье не рассматриваются также такие объекты, как вторичные характеристики схемы, сигналы и т.п., которые должны быть согласованы с другими областями (см. также [3]).

2. Базисные компоненты электронной схемы

Целесообразно рассматривать компоненты электронной схемы в виде отдельных объектов.

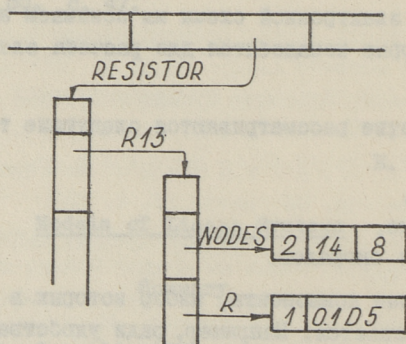
В третьей версии системы SPADE основными объектами выбраны:

- 1) резистор,
- 2) конденсатор,
- 3) катушка индуктивности,
- 4) транзистор,
- 5) идеальный диод,
- 6) идеальный транзистор.

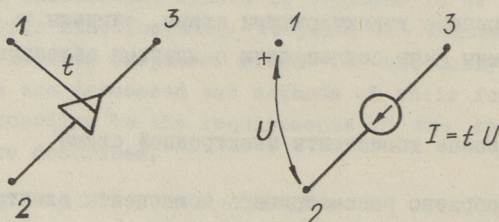
Каждый элемент описывается на входном языке отдельным предложением, которое транслятор преобразует в набор данных в структуру определенной формы [4]. Например, резистор может быть представлен предложением

RI3 RESISTOR INODES=14,8 RESISTANCE=10.КОМ;

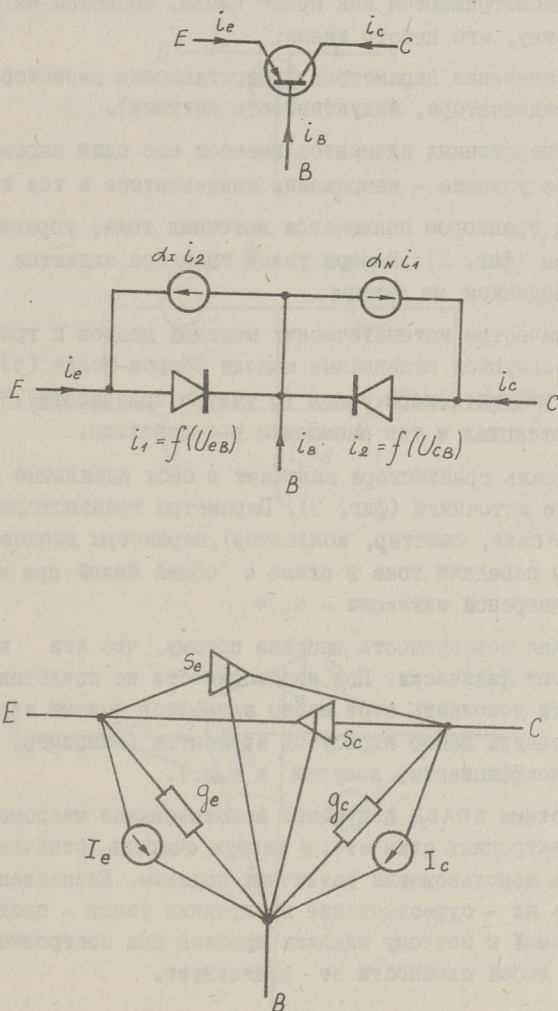
Это означает, что резистор с именем RI3 и значением 10 КОм, расположен между узлами I4 и 8 (см. фиг. 1).



Фиг. 1. Структура данных резистора.



Фиг. 2. Транзистор.



Фиг. 3. Модель Эберса-Молла и линейная эквивалентная схема в статическом режиме транзистора.

Все указанные выше компоненты имеют следующие атрибуты:

1) номера узлов – по традиции и в системе SPADE номера узлов рассматриваются как целые числа, несмотря на то что, по существу, это просто имена;

2) значения параметров (сопротивление резистора, емкость конденсатора, индуктивность катушки).

У инерционных элементов имеется еще один параметр – начальное условие – напряжение конденсатора и ток катушки.

Под транзором понимается источник тока, управляемый напряжением (фиг. 2). Номера узлов транзора задаются в порядке, приведенном на фигуре.

В качестве математических моделей диодов и транзисторов используются нелинейные модели Эберса-Молла [5]. Для диода имеем атрибуты: номера узлов (с учетом полярности), φ_T и I_S – тепловой потенциал и ток насыщения р-п-перехода.

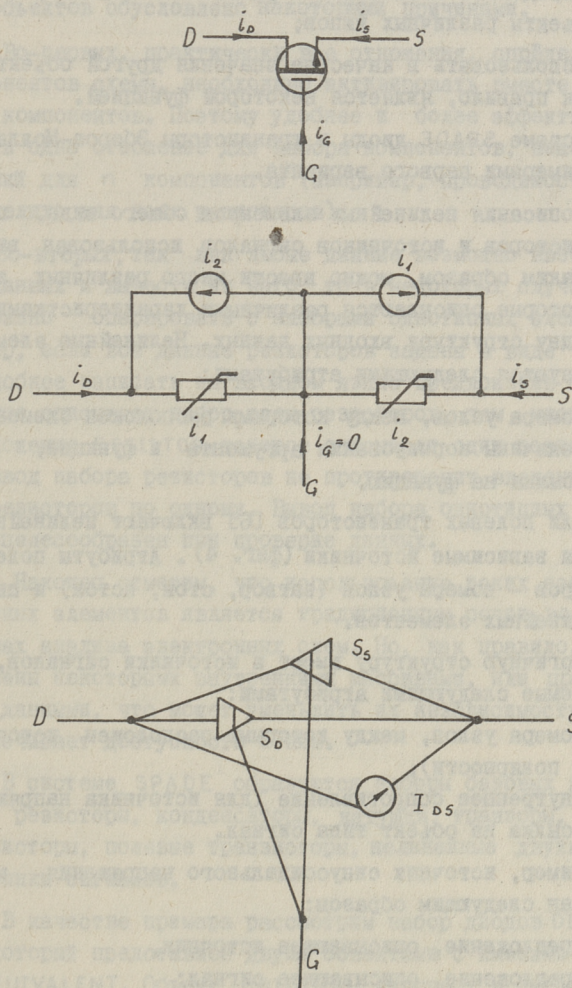
Модель транзистора включает в себя идеальные диоды и зависимые источники (фиг. 3). Параметры транзисторов – номера узлов (база, эмиттер, коллектор), параметры диодов и коэффициенты передачи тока в схеме с общей базой при нормальной и инверсной инжекции – α_N, α_I .

Такая совокупность выбрана потому, что эти элементы существуют физически. При необходимости не представляет трудности дополнить этот набор элементов новыми элементами или дополнить набор атрибутов элементов (например, температурные коэффициенты, допуски и т.д.).

Система SPADE допускает использование макромоделей [4]. Для электронных схем это, в первую очередь, означает возможность использования различных подсхем. Единственная особенность их – существование внутренних узлов – предусмотрено системой и поэтому никаких проблем при построении макромоделей любой сложности не существует.

3. Более сложные объекты

В отличие от базисных компонентов, атрибуты которых могут иметь одно значение, в системе SPADE есть компоненты (нелинейные элементы и источники сигналов), которые требуют более сложной структуры данных, так как их атрибуты – функции.



Фиг. 4. Модель Эберса-Молла и линейная эквивалентная схема в статическом режиме полевого транзистора.

Имеется два принципиально различных решения, имеющих свои особенности и преимущества:

1) определить компоненты с различными характеристиками как объекты различных типов;

2) использовать в качестве значения другой объект, который, как правило, является некоторой функцией.

В системе SPADE диоды и транзисторы Эберса-Молла являются примерами первого варианта.

Для описания нелинейных элементов общего вида, полевых транзисторов и источников сигналов использован второй подход. Таким образом, можно ввести много различных элементов, которые описываются различными характеристиками, используя одну структуру входных данных. Нелинейные элементы характеризуются следующими атрибутами:

- 1) номера узлов, между которыми расположен элемент;
- 2) величины нормирования аргумента и функции;
- 3) ссылка на функцию.

Модели полевых транзисторов [6] включают нелинейные элементы и зависимые источники (фиг. 4). Атрибуты полевых транзисторов — номера узлов (затвор, сток, исток) и параметры нелинейных элементов.

Аналогичную структуру имеют и источники сигналов, характеризующиеся следующими атрибутами:

- 1) номера узлов, между которыми расположен источник (с учетом полярности);
- 2) внутреннее сопротивление (для источника напряжения);
- 3) ссылка на объект типа сигнал.

Например, источник синусоидального напряжения может быть описан следующим образом:

- 1) предложение, описывающее источник,
- 2) предложение, описывающее сигнал:

```
VS VOLTAGE-SOURCE NP=1 NM=3 RI=1.0HM  
VALUE=HS;
```

```
HS HARMONIC AMPLITUDE= $\phi$ . $\phi$ 5 FREQUENCY=314.16  
TYPE=SIN PHASE= $\phi$ ;
```


4. Наборы однотипных элементов

Использование наборов элементов в качестве самостоятельных объектов обусловлено некоторыми причинами.

Во-первых, практически все отношения, определённые для компонентов схемы, необходимо активировать вместе, т.е. для всех компонентов. Поэтому удобнее и более эффективно определить одно отношение для набора компонентов, нежели n отношений для n компонентов (например, проводимости следует определить для всех резисторов).

Во-вторых, так как любые данные возможно ввести в набор данных и вывести из него, то в некоторых случаях целесообразно оперировать с наборами однотипных элементов. Например, если все данные резисторов заданы в виде таблицы, то удобнее написать на входном языке предложение RESISTORS, которое описывает набор всех резисторов, чем многократно предложение RESISTOR, которое описывает один резистор. Кстати, ввод набора резисторов не противоречит введению остальных резисторов по одному. Вывод набора однотипных элементов целесообразен при проверке данных.

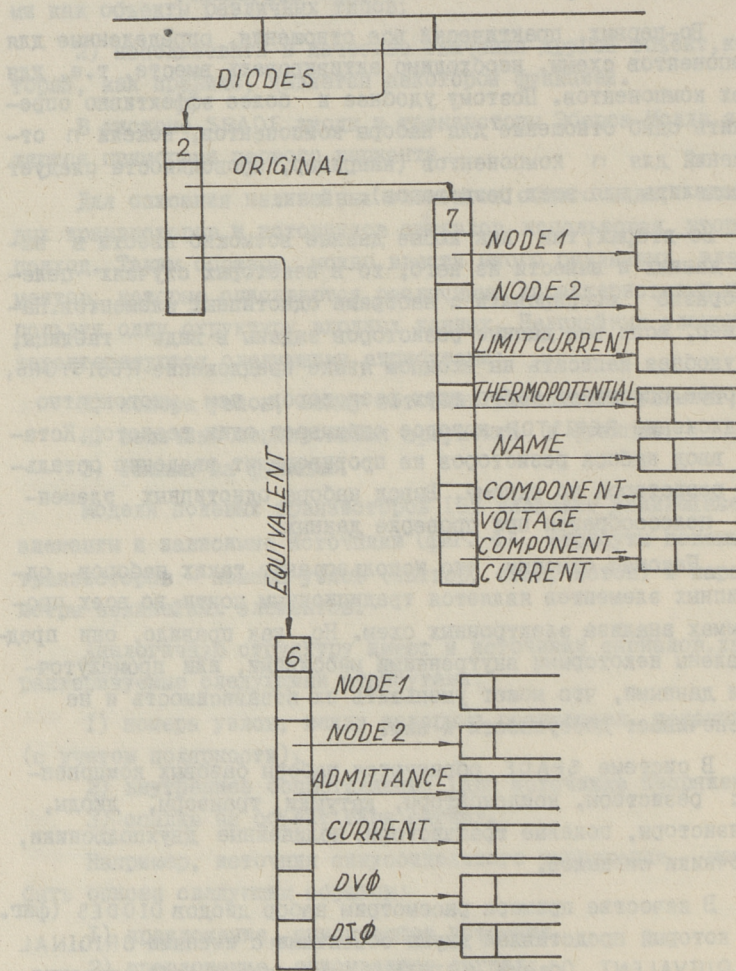
Наконец отметим, что использование таких наборов однотипных элементов является традиционным почти во всех программах анализа электронных схем. Но, как правило, они представлены некоторыми внутренними массивами, или промежуточными данными, что может уменьшить их независимость и не обеспечивает доступности к ним.

В системе SPADE образуются наборы базовых компонентов: резисторы, конденсаторы, катушки, транзисторы, диоды, транзисторы, полевые транзисторы, нелинейные двухполюсники, источники сигналов.

В качестве примера рассмотрим набор диодов DIODES (фиг. 5), который представлен двумя объектами с именами ORIGINAL и EQUIVALENT. Объект ORIGINAL включает в себя значения атрибутов, взятые из объектов типа DIODE:

1) из значения атрибута NODES объекта DIODE: атрибуты NODE1 — первые значения, NODE2 — вторые значения;

2) векторы, содержащие значения параметров компонентов: LIMITCURRENT — значения теплового тока перехода,



Фиг. 5. Набор диодов.

THERMOPOTENTIAL — значения теплового потенциала;

3) вектор NAME состоит из имен объектов типа DIODE,

4) векторы COMPONENT_CURRENT и COMPONENT_VOLTAGE состоят из значений тока и напряжения элемента.

У других наборов компонентов объект ORIGINAL аналогичный:

1) номера узлов в различных векторах,

2) характеристические параметры (сопротивление, емкость и т.д.),

3) вектор имен,

4) векторы токов и напряжений элементов.

Объект с именем EQUIVALENT включает в себя параметры схем замещения компонентов. Векторы NODE1 и NODE2 получаются из векторов NODE1 и NODE2 объекта ORIGINAL перенумерацией узлов с целью получения непрерывной нумерации.

Атрибутами объекта EQUIVALENT являются параметры схемы замещения компонентов или векторы, характеризующие электронную схему в соответствующий момент анализа. Например, для набора инерционных элементов объект EQUIVALENT включает в себя и момент времени, для которого рассматривается эквивалентная схема [7], а также параметры эквивалентной схемы в предыдущий момент времени.

Для набора диодов (фиг. 5) атрибуты объекта EQUIVALENT следующие:

ADMITTANCE и CURRENT — проводимость и ток линейной схемы замещения диода (фиг. 3 [7]), $DV\phi$ и $DI\phi$ — напряжение и ток рабочей точки диода.

Простая линейная модель биполярного транзистора изображена на фиг. 3. Параметры модели определяются следующим образом:

проводимости транзисторов

$$g_c = (1 - \alpha_I) G_{oc},$$

$$g_e = (1 - \alpha_N) G_{oe},$$

проводимости транзисторов

$$S_c = \alpha_I G_{oc},$$

$$S_e = \alpha_N G_{oe},$$

ТОКИ ИСТОЧНИКОВ

$$I_c = -\alpha_N I_{oe} + I_{oc},$$

$$I_e = -\alpha_I I_{oe} + I_{oe}.$$

где G_{oc} , G_{oe} и I_{oc} , I_{oe} получаются при линеаризации диодов, входящих в модель транзистора.

Перечисленные параметры схем замещения транзистора и являются атрибутами объекта EQUIVALENT типа TRANSISTORS.

Аналогично для полевого транзистора параметры линейной модели (фиг. 4) следующие:

проводимости транзисторов

$$S_D = G_{od}$$

$$S_S = G_{os}$$

ток источника

$$I_{DS} = I_{od} - I_{os},$$

где G_{od} , G_{os} , I_{od} , I_{os} получены при линеаризации нелинейных элементов модели Эберса-Молла.

5. Дополнительные данные

Для решения электронных схем необходимо иметь еще набор данных, характеризующий электронную схему в целом:

- сопротивление и частота нормирования,
- базисный узел,
- точность для проведения итерационного процесса,
- метод интегрирования (в третьей версии SPADE метод Эйлера или метод трапеции),
- интервал времени и начальный временной шаг для интегрирования и т.п.

6. Y-матрица

В рассматриваемой версии системы SPADE при решении уравнений используется адмитансная модель (метод узловых потенциалов). Для составления матрицы узловых проводимостей Y желательно определить специальные объекты, которые отображают схему в любых ситуациях, т.е. содержат описание эквивалентной схемы с минимальным набором базисных элементов. Такой набор образуют обратимые двухполюсники (дисторы) и транзоры.

Для этого служат два объекта типа EQ_CONDUCTANCES: DISTORS и TRANSORS. Оба объекта включают в себя вектора номеров всех узлов. Вектор проводимостей в обоих объектах заполняется постепенно во время анализа [7].

Рассмотрим, например, объект DISTORS – набор всех проводимостей (фиг. 6). В начале анализа в вектор дублируются проводимости резисторов G_C . На каждом временном шаге определяются эквивалентные схемы для инерционных элементов, проводимости эквивалентных схем G_T включаются в вектор ADMITTANCE. Во время итерационного процесса в эту структуру добавляются еще проводимости линейных схем замещения всех нелинейных компонентов G_N .

Аналогично заполняется вектор CONDUCTANCES – атрибут объекта TRANSORS.

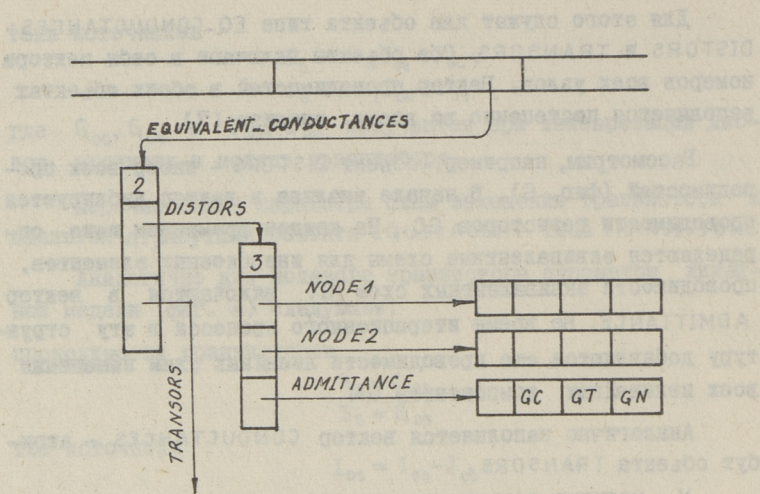
У – матрица заполняется исходя из объектов типа EQ_CONDUCTANCES.

При составлении У-матрицы надо учитывать следующее. С одной стороны, как показано в [8], целесообразно применять при решении уравнения метод LU-разложения. С другой стороны, известно, что У-матрица является разреженной и ее структурная несимметрия в смысле расположения ненулевых элементов относительно главной диагонали незначительна. Поэтому для описания У-матрицы выбрана схема представления симметричных разреженных матриц с ненулевыми элементами главной диагонали [8]. Тогда У-матрица описывается с помощью 5 векторов (атрибутов):

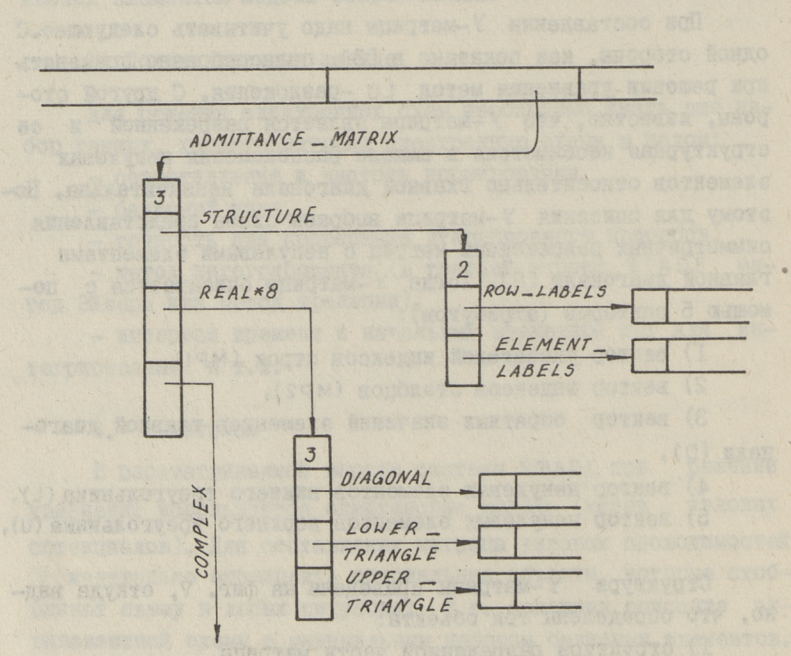
- 1) вектор указателей индексов строк (MP1),
- 2) вектор индексов столбцов (MP2),
- 3) вектор обратных значений элементов главной диагонали (D),
- 4) вектор ненулевых элементов нижнего треугольника (L),
- 5) вектор ненулевых элементов верхнего треугольника (U),

Структура У-матрицы приведена на фиг. 7, откуда видно, что определены три объекта:

- 1) структура разреженной части матрицы,
- 2) вещественная матрица,
- 3) комплексная матрица.



Фиг. 6. Набор проводимостей.



Фиг. 7. Структура Y-матрицы.

7. Заключение

В настоящей статье рассмотрены проблемы создания информационной модели электронных схем на базе системы SPADE. Показано, что целесообразно рассматривать в виде отдельных объектов компоненты схемы, а также наборы однотипных объектов. Сложные объекты, как источники сигналов и нелинейные элементы, следует представлять двухуровневыми, где подчиненный объект описывает нелинейную функцию.

В качестве самостоятельных объектов целесообразно рассматривать наборы эквивалентных элементов, включающих линейные и нелинейные схемы замещения нелинейных компонентов.

Разреженные матрицы узлов проводимостей могут быть также представлены отдельными объектами.

Проведенный анализ показывает, что средства системы SPADE позволяют создать весьма четкую модель предметной области "электронные схемы".

Л и т е р а т у р а

1. К у к к В. Принципы функционирования системы SPADE. См. наст. сб. с. 3.

2. К у к к В., В е н д н л и н Е. Управление вычислительным процессом. См. наст. сб. с. 33.

3. Л е п п и к с о н В., Р ю с т е р н Э. Модели линейных систем. См. наст. сб. с. 57.

4. В е н д е л и н Е. Структуры данных и входной язык. См. наст. сб. с. 19.

5. И л ь и н В.И. Машинное проектирование электронных схем. М., Энергия, 1972. 280 с.

6. А г а х а н я н Т.М. Основы транзисторной электроники. М., Энергия, 1974. 256 с.

7. Т и й г и м я э М. Алгоритмы и структуры данных при анализе нелинейных цепей. См. наст. сб. с. 83.

8. Т ь ю а р с о н Р. Разреженные матрицы. М., Мир, 1977, 192 с.

M. Kurm,
M. Tiigimäe

Models of Electrical Networks

Summary

Techniques of the computer-aided design system SPADE are used to design an application program for the analysis of linear and nonlinear circuits. The paper describes the models of different network components as well as equivalent circuits and nodal admittance matrices.

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ ПРИ АНАЛИЗЕ
НЕЛИНЕЙНЫХ ЦЕПЕЙ

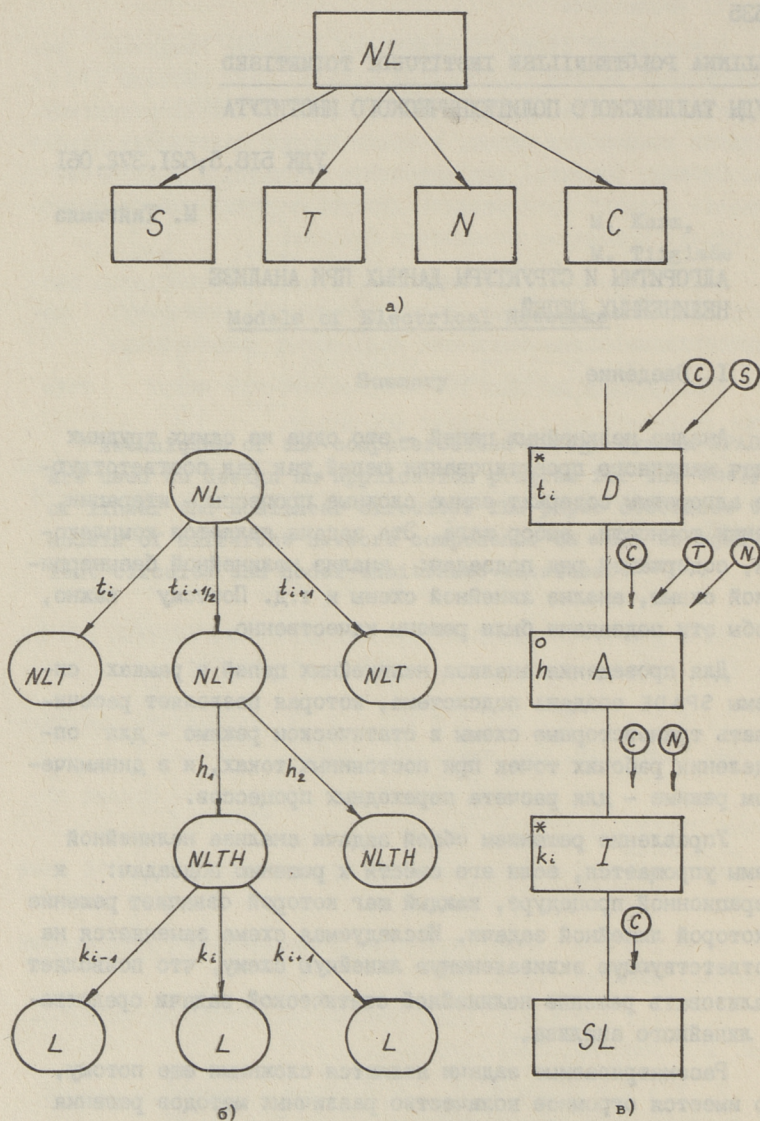
I. Введение

Анализ нелинейных цепей – это одна из самых трудных задач машинного проектирования цепей, так как соответствующие алгоритмы содержат самые сложные процессы – итерации, оценки точности, выбор шага. Эта задача является комплексной, содержащей ряд подзадач: анализ нелинейной безынерционной схемы, анализ линейной схемы и т.д. Поэтому важно, чтобы эти подзадачи были решены качественно.

Для проведения анализа нелинейных цепей в рамках системы SPADE создана подсистема, которая позволяет рассчитывать транзисторные схемы в статическом режиме – для определения рабочих точек при постоянных токах, и в динамическом режиме – для расчета переходных процессов.

Управление решением общей задачи анализа нелинейной схемы упрощается, если его свести к решению подзадач: к итерационной процедуре, каждый шаг которой означает решение некоторой линейной задачи. Исследуемая схема заменяется на соответствующую эквивалентную линейную схему, что позволяет реализовать решение нелинейной статической задачи средствами линейного анализа.

Рассматриваемые задачи являются сложными еще потому, что имеется огромное количество различных методов решения их. Поэтому особенно важно соблюдать независимость структуры данных, что упрощает разработку различных методов анализа. Подсистема анализа нелинейных цепей является частью системы SPADE основной принцип построения которой и заключается в том, что любые операции, выполняемые во время



Фиг. 1. Подсистема анализа нелинейных цепей
а - компоненты нелинейных цепей,
б - процесс анализа нелинейных цепей,,
в - модификация структуры данных.

расчетов, рассматриваются как модификация основной структуры данных.

Поэтому в данной статье сначала рассмотрим расширение модели электронных схем, которое необходимо для решения поставленных задач. В качестве примера построения пакета на этой модели рассмотрим реализацию анализа диодно-транзисторных схем.

2. Данные

Исходная нелинейная схема может содержать компоненты, перечисленные в [1]. Различие в функциональных характеристиках позволяет разбить все элементы на четыре группы (фиг. 1, а):

S - источники переменных токов и напряжений;

T - линейные и нелинейные инерционные элементы (конденсаторы, катушки);

N - нелинейные безынерционные элементы (диоды, транзисторы и т.д.);

C - постоянные источники сигналов и линейные безынерционные элементы (резисторы, транзисторы).

Решение задачи динамики можно рассматривать как обработку схем четырех уровней (фиг. 1, б):

1) исходная нелинейная электронная схема - NL,

2) схемы, соответствующие различным значениям t_i времени - NLT,

3) безынерционные схемы, соответствующие различным значениям шага h и порядка, а также методам интегрирования - NLTH,

4) линейные безынерционные схемы, соответствующие различным шагам итераций при решении - L.

Такая схема данных является универсальной и независимой от методов интегрирования и решения нелинейных и линейных уравнений. Независимо от процедур обработки можно построить и схему потока данных (фиг. 1, в).

Рассмотрим ее в соответствии с введенными выше классификациями:

- D - Анализ динамики нелинейной схемы производится для множества моментов времени. Для каждого момента составляется своя нелинейная схема, где переменные источники S получают соответствующие значения, которые включаются в группу данных C ;
- A - Инерционные элементы T заменяются эквивалентными схемами в зависимости от метода интегрирования, порядка и шага. Параметры линейных эквивалентных схем заносятся в группу данных C . Эквивалентные схемы нелинейных инерционных элементов нелинейны и включаются в структуры нелинейных элементов N . Получается нелинейная модель, для которой должен быть проведен анализ в режиме постоянного тока.
- I - Анализ статического режима производится до достижения необходимой точности в итерационном процессе. При этом нелинейные элементы N линеаризуются и проводимости и токи линейных схем заносятся в группу C .
- SL - Линейная модель решается программами линейного анализа. Линейный анализ линеаризованной нелинейной схемы производится на структуре, содержащей векторы проводимостей всех схем замещения и постоянных токов.

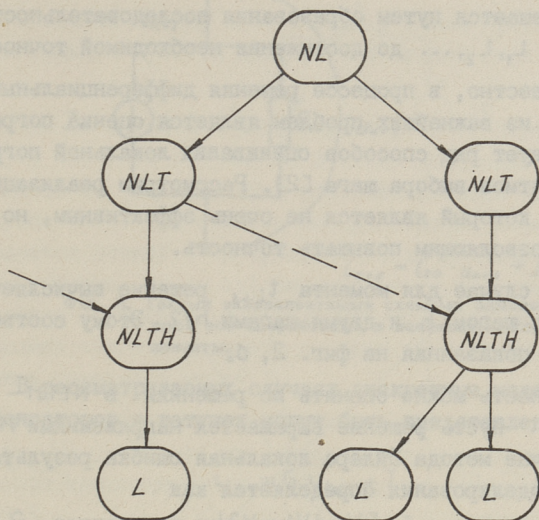
Таким образом, на каждом этапе имеем дело с некоторой физической электронной схемой, которая отображается в структуре данных.

3. Динамика

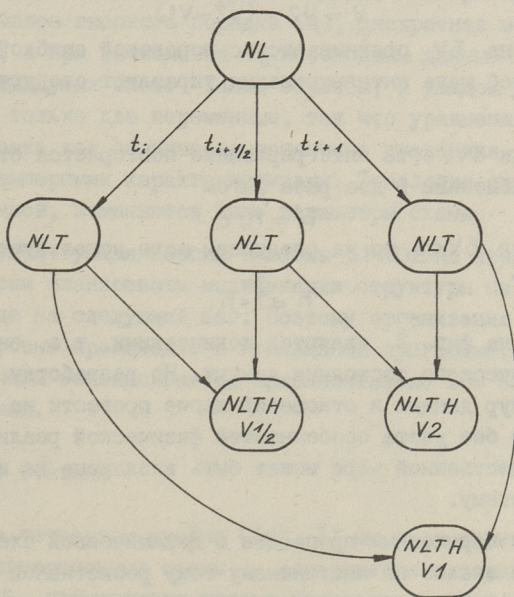
Рассмотрим, например, реализацию неявного метода Эйлера и трапеций для интегрирования уравнений нелинейной схемы. Оба метода используют на каждом шаге данные, соответствующие двум моментам времени - началу и концу шага t_i и t_{i+1} .

Соответственно мы имеем дело с двумя схемами второго уровня (фиг. 2, а), в которых зафиксированы значения источников.

На базе этих двух схем и в зависимости от метода решения образуется схема на третьем уровне NLTH, которая зависит от шага $h = t_{i+1} - t_i$. Полученная нелинейная безынерцион-



a)



б)

Фиг. 2. Метод выбора шага интегрирования:
а - процесс анализа,
б - выбор шага.

ная схема решается путем образования последовательности линейных схем $L_1, L_2, \dots$ до достижения необходимой точности.

Как известно, в процессе решения дифференциальных уравнений одной из важнейших проблем является оценка погрешности. Существует ряд способов оценивания локальной погрешности и алгоритмов выбора шага [2]. Рассмотрим реализацию метода Рунге, который является не очень эффективным, но зато простым и позволяющим повышать точность.

В этом случае для момента t_{i+1} решение вычисляется два раза: с шагом h и двумя шагами $h/2$. Этому соответствует схема, показанная на фиг. 2, б.

Погрешность можно оценить по решениям в $NLTH-V1$ и $NLTH-V2$. Пусть решение выражается напряжениями $V1$ и $V2$. Тогда в случае метода Эйлера локальная ошибка результатов цифрового моделирования определяется как

$$\delta V = |V1 - V2|,$$

а уточненное значение —

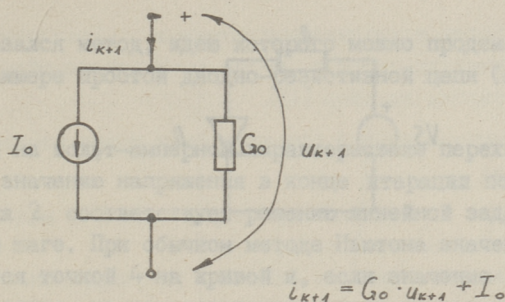
$$V' = V2 + (V2 - V1).$$

Величина δV сравнивается с пороговой ошибкой ε и затем величиной шага интегрирования управляют следующим образом:

- 1) если $\delta V > \varepsilon$, то интегрирование повторяется от точки t_i с уменьшенным в два раза шагом
 $h' = h/2,$
- 2) если $\delta V < \varepsilon$, то на следующем шаге используется вдвое больший шаг
 $h' = 2 * h.$

Схемы на фиг. 2 являются логическими, т.е. они не отражают физического состояния данных. Но разработку алгоритмов, структур данных и отношений легче провести на логическом уровне без учета особенностей физической реализации, которая в существенной мере может быть возложена на инструментальную систему.

Анализ переходных процессов в динамической схеме можно заменить на анализ по постоянному току резистивной схемы путем замены каждого конденсатора и каждой катушки индуктивности на резистивную дискретную модель цепи (фиг. 3), ассоциированную с заданным методом интегрирования [2, 3].



Фиг. 3. Такими эквивалентными схемами заменяются все инерционные и нелинейные компоненты.

В рассматриваемых случаях дискретные модели линейных конденсаторов и катушек могут быть представлены уравнением

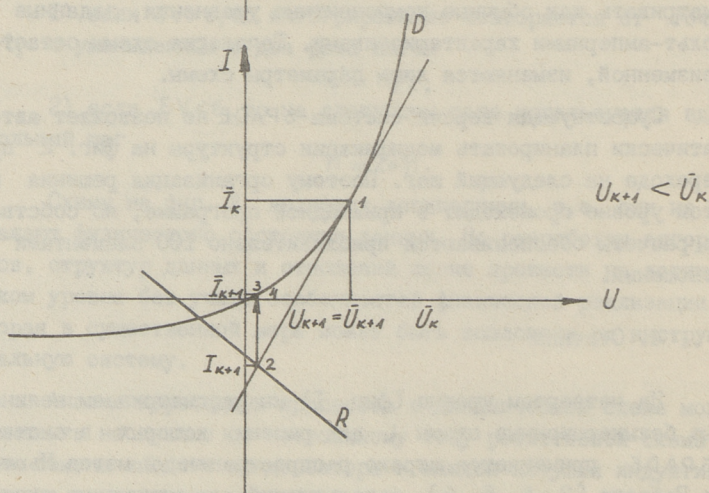
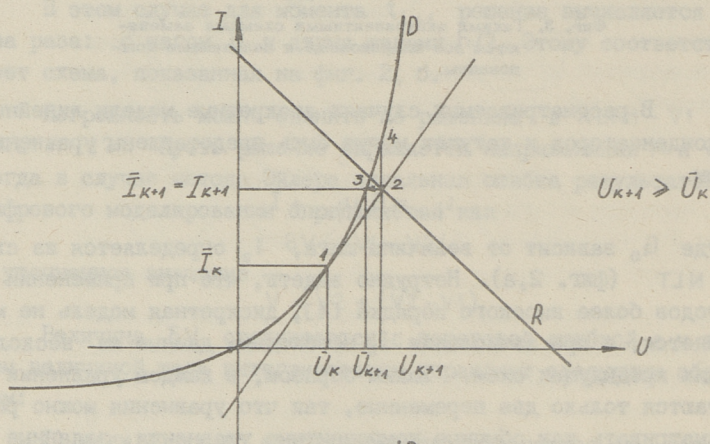
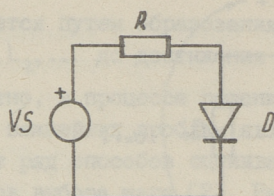
$$i_{i+1} = G_0 u_{i+1} + I_0,$$

где G_0 зависит от величины шага, I_0 определяется из схем NLT (фиг. 2, а). Нетрудно видеть, что при применении методов более высокого порядка [4], дискретная модель не меняется, а при вычислении I_0 необходимы данные из нескольких предыдущих схем. Таким образом, в каждом уравнении остаются только две переменные, так что уравнения можно рассматривать как обычные компонентные уравнения, заданные вольт-амперными характеристиками. Топология схемы остается неизменной, изменяются лишь параметры схемы.

Существующая версия системы SPADE не позволяет автоматически планировать модификации структуры на фиг. 2 при переходе на следующий шаг. Поэтому организация решения на этом уровне происходит в прикладной программе, но собственно расчеты обеспечиваются приблизительно 100 элементами 8 отношений.

4. Статика

На четвертом уровне (фиг. 1) мы рассматриваем нелинейные безынерционные схемы I для решения которых в системе SPADE применяется широко распространенный метод Ньютона-Рафсона [2, 3, 5, 6], использующий для улучшения сходимости особенности вольт-амперной характеристики р-п-перехода. После нескольких опытов с разными модификациями наи-



Фиг. 4. Определение напряжения на переходе диода при итерациях.

лучшим оказался метод, идею которого можно продемонстрировать на примере простой диодно-резистивной цепи (фиг. 4) [6].

Точки на вольт-амперной характеристике перехода соответствуют значению напряжения в конце итерации порядка k (U_k). Точка 2 соответствует решению линейной задачи на предыдущем шаге. При обычном методе Ньютона значение тока определяется точкой 4 на кривой и, если значение U_{k+1} достаточно велико, из-за экспоненциального характера кривой легко может возникнуть переполнение при расчете на ЭВМ. Чтобы избежать этого и ускорить процесс вычислений, лучше осуществлять переход в точку 3.

Такой переход следует применять тогда, когда $U_{k+1} > \bar{U}_k$. Если же $U_{k+1} < \bar{U}_k$, то предлагаемый прием становится невозможным или невыгодным и поэтому следует осуществить переход в точку 3.

С учетом этого соображения формулы для вычисления тока и напряжения диода принимают следующий вид:

если $U_{k+1} \geq \bar{U}_k$, то $\bar{I}_{k+1} = I_{k+1} = \bar{I}_k + G_0 \cdot \bar{U}_k$,

$$\bar{U}_{k+1} = \varphi_T \ln(\bar{I}_{k+1}/I_S + 1);$$

если $U_{k+1} < \bar{U}_k$, то $\bar{U}_{k+1} = U_{k+1}$,

$$\bar{I}_{k+1} = I_S \exp(\bar{U}_{k+1}/\varphi_T - 1).$$

Здесь $\bar{U}_k, \bar{I}_k$ - напряжение и ток в предыдущей итерационной точке на характеристике;

U_{k+1}, I_{k+1} - решения линейного уравнения;

$\bar{U}_{k+1}, \bar{I}_{k+1}$ - напряжение и ток в новой итерационной точке;

I_S - тепловой ток р-п-перехода;

φ_T - тепловой потенциал.

Параметры линейной модели диода (фиг. 3) следующие:

проводимость

$$G_0 = \frac{\bar{I}_{k+1} + I_S}{\varphi_T},$$

ток источника

$$I_0 = \bar{I}_{k+1} - G_0 \cdot \bar{U}_{k+1}.$$

Во многих случаях характеристики нелинейных элементов представляются такими кривыми, наклон которых изменяется плавно. Это позволяет аппроксимировать характеристики неко-

торым набором прямолинейных отрезков. Уравнение представляет нелинейную систему, но по отдельности области между точками излома линейны [2].

Кусочно-линейные аппроксимации имеют следующие преимущества:

- 1) для вычисления параметров линейной модели используются линейные отношения;
- 2) изменением шага селекции регулируется точность аппроксимации;
- 3) программа универсальная, так как ею можно воспользоваться для аппроксимации произвольной монотонной однозначной кривой.

В системе SPADE это позволяет ограничиваться одной программой линеаризации и одним типом элементов - кусочно-линейным элементом, который аппроксимируется соответственной кусочно-линейной кривой [1].

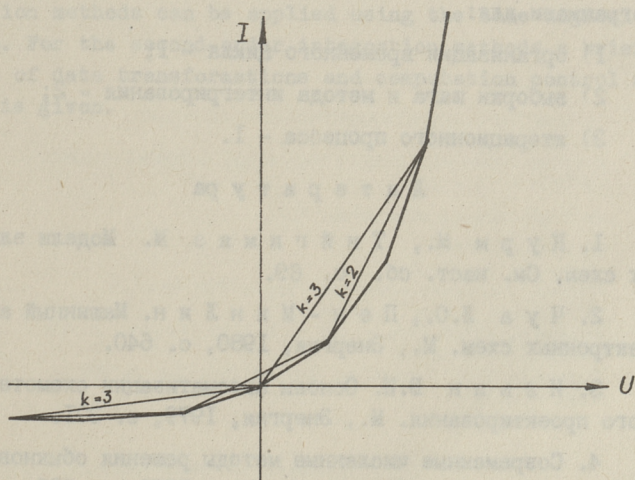
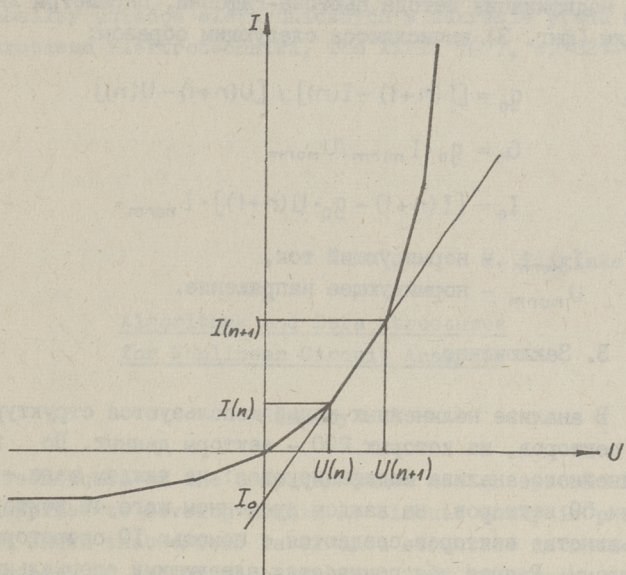
Кусочно-линейные кривые задаются двумя множествами. На них можно построить много кусочно-линейных характеристик, используя различные значения шага селекции k (фиг. 5). Это позволяет использовать массивы больших размерностей, но во время итераций работать лишь частью из них. Изменяя шаг селекции, можно ускорить процесс итераций. На первых итерациях можно использовать большой шаг селекции, т.е. невысокую точность аппроксимации, но простое и быстрое вычисление параметров линейной модели. При приближении к решению можно уменьшить шаг селекции до $k=1$. Основной выигрыш получается за счет того, что если не происходит перехода на другой отрезок, то не пересчитываются параметры линейной схемы замещения.

Отрезок n полностью определяется следующими соотношениями (фиг. 5):

$$I = G(n) \cdot U + I(n),$$

$$U(n) \leq U \leq U(n+1).$$

Геометрически параметры $G(n)$ и $I(n)$ представляют собой наклон прямой (проводимость) и точку пересечения этой прямой с осью I соответственно. Интервал $[U(n), U(n+1)]$ является областью определения отрезка n .



Фиг. 5. Кусочно-линейная аппроксимация вольт-амперной характеристики диода и различные кусочно-линейные характеристики.

Для нахождения нужного отрезка используется описанная выше модификация метода Ньютона-Рафсона. Параметры линейной модели (фиг. 3) вычисляются следующим образом:

$$g_0 = [I(n+1) - I(n)] / [U(n+1) - U(n)]$$

$$G_0 = g_0 \cdot I_{\text{norm}} / U_{\text{norm}}$$

$$I_0 = [I(n+1) - g_0 \cdot U(n+1)] \cdot I_{\text{norm}},$$

где I_{norm} - нормирующий ток,
 U_{norm} - нормирующее напряжение.

3. Заключение

В анализе нелинейных цепей используются структуры из 250 векторов, из которых 200 - векторы данных. Во время нелинейного анализа модифицируются: на каждом шаге - итерации 50 векторов, на каждом временном шаге 40 векторов. Большинство векторов создаются с помощью 10 операторных программ. Расчет обеспечивается следующими специальными программами для:

- 1) организации временного цикла - I;
- 2) выборки шага и метода интегрирования - 2;
- 3) итерационного процесса - I.

Л и т е р а т у р а

1. Курм М., Тийгимяз М. Модели электронных схем. См. наст. сб. с. 69.

2. Ч у а Л.О., П е н - М и н Л и н. Машинный анализ электронных схем. М., Энергия, 1980, с. 640.

3. И л ь и н В.И. Основы автоматизации схемотехнического проектирования. М., Энергия, 1979, с. 371.

4. Современные численные методы решения обыкновенных дифференциальных уравнений. М., Мир, 1979, с. 312.

5. Ч а х м а х с а з я н Е.А., Б а р м а к о в Ю.Н., Г о л ь д е н б е р г А.Э. Машинный анализ интегральных схем. М., Советское радио, 1974, с. 272.

6. B u d z i s z H. Metoda przyspieszenia Komputerowej analizy uhladow elektronicznych w zakresie pradu stalego. Archiwum Elektrotechniki, Tom XXVI, 1977, c. 621-633.

M. Tiigimäe

Algorithms and Data Structures
for Nonlinear Circuit Analysis

Summary

Techniques of the computer-aided design system SPADE are applied to develop nonlinear circuit analysis program. It is shown that a four level data structure is needed to solve both DC and AC problems, and that high order integration methods can be applied using the same logical structure. For the second-order integration methods a brief review of data transformations and computation control methods is given.

С о д е р ж а н и е

1.	В. Куку. Принципы функционирования системы SPADE.	3
2.	Е. Венделин. Структуры данных и входной язык.....	19
3.	В. Куку, Е. Венделин. Управление вычислительным процессом.....	33
4.	В. Леппиксон. Формирование выходных документов...	47
5.	В. Леппиксон, Э. Рюстерн. Модели линейных систем.	57
6.	М. Курм, М. Тийгимаяэ. Модели электронных схем....	69
7.	М. Тийгимаяэ. Алгоритмы и структуры данных при анализе нелинейных цепей.....	83

ТАЛЛИНСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Труды ТПИ № 535

АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ В ЭЛЕКТРОНИКЕ -
СИСТЕМА SPADE

Электротехника и автоматика XX

Редактор Р. Убар

Техн. ред. М. Тамме

Сборник утвержден коллегией Трудов ТПИ 06.05.82

Подписано к печати 29.10.82

Формат 80х90/16. Печ. л. 6,0 +0,25 прил. Уч.-изд. л. 5,0

Тираж 300. МВ-03783.

Ротапринт ТПИ, Таллин, ул. Коскла, 2/9. Зак. № 506

Цена 75 коп.



Цена 75 коп.