бы к необходимости участить сетку минимум в пять раз, что сделает процедуру расчета непозволительно долгой. Поэтому сильно локализованные возмущения следует моделировать на основе отдельной краевой задачи для подобласти, заимствуя граничные условия из глобальной задачи.

Библиографический список

1. **Возможный** подход к проблеме возбуждения электрических полей и токов, обусловленных B_y-компонентой ММП / В. М. Уваров // Геомагнетизм и аэрономия. – 1981. – № 1. Т. 21. – С. 114–120.

2. **Электрические** поля в ионосфере Земли. Численные модели / В. М. Уваров, Б. А. Самокиш. – Санкт-Петербург : Петербургский гос. ун-т путей сообщения, 2009. – 63 с.

3. **Модифицированная** численная модель глобального распределения электрического потенциала. UT-эффект обращения ионосферной конвекции / А. Б. Кондаков, Б. А. Самокиш, В. М. Уваров // Геомагнетизм и аэрономия. – 1999. – № 6. Т. 39. – С. 50–55.

УДК 609.6(075.8)

С. Г. Подклетнов
Петербургский государственный университет путей сообщения

АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ С ПОМОЩЬЮ ЯЗЫКА AUTOLISP

Показана возможность применения языка AutoLISP для создания параметрических чертежей в AutoCAD. Рассмотрены основные понятия и области применения широко известного языка программирования AutoLISP, используемого в инженерных расчетах, позволяющего разрабатывать макропрограммы и функции, хорошо сочетающего с прикладной графикой AutoCAD. Приведен текст программы для создания блока построения чертежа параметрической детали. Сделан вывод о возможности применения данного языка для создания параметрических чертежей уже более сложных моделей.

автоматизированное проектирование, язык AutoLISP, параметрические чертежи.

Введение

Особого внимания среди *CAD*-систем заслуживает программный комплекс *AutoCAD*. Эта система автоматизированного проектирования не только предоставляет пользователю удобный графический интерфейс и обеспечивает получение высококачественной графики, но и располагает широким набором инструментов программирования.

К средствам программирования *AutoCAD* относятся следующие языки: *AutoLISP*, *Visual C++*, *Visual Basic for Application* (*VBA*, *ARX*, *DCL*) [1, 2, 3, 4]. Будучи частью *AutoCAD*, *AutoLISP* позволяет оперировать переменными различных типов и передавать их значения командам *AutoCAD* при вводе данных. При ответах на запросы команд *AutoCAD* существует возможность использования выражений *AutoLISP*, в которых могут выполняться различные арифметические и условные операции с числовыми значениями и значениями определенных переменных.

Возможности, которые обеспечивает *AutoLISP*:

– использование переменных и выражений при ответах на запросы команд *AutoCAD*;

– чтение и создание внешних файлов (таким образом осуществляется обмен информацией с внешними программами, которые можно запускать из *AutoCAD*);

– создание различных функций и новых команд *AutoCAD*, что обеспечивает настройку и расширение графических возможностей системы;

– программный доступ (чтение и редактирование) к данным, которые относятся к объектам проектирования, а также к таблицам *AutoCAD*, содержащим информацию о блоках, слоях, видах, стилях и типах линий;

– программное управление графическим экраном *AutoCAD*, а также вводом/выводом из различных устройств;

– параметрическое моделирование.

1 Параметрическое проектирование

Сущность параметрического проектирования состоит в создании математической модели класса конструктивно однородных изделий, а затем в генерации изображений этих изделий по набору задаваемых размерных параметров. При параметрическом проектировании конструктор запускает программу, рассчитанную на определенный класс изделий, и вводит требуемые размеры. Программа зарисовывает на экране чертеж детали. Конструктор оценивает его и при необходимости вводит размеры снова, до достижения требуемого результата. Особо следует отметить два типа функций, обеспечивающих создание параметрических моделей:

– функции ввода данных и указания объектов;

– функции, работающие со списками.

Функции ввода данных и указания объектов дают возможность пользователю вводить данные в интерактивном режиме. Основные функции:

Функция INITGET:

(initget [<флаг>] [<строка]) – задание ключевых слов и ограничений ввода для функций getpoint, getreal и др.

Функция GETPOINT:

(getpoint [<точка1>] [<запрос]) – ввод точки с помощью клавиатуры или мыши.

Функция GETREAL:

(getreal [<запрос>]).

Функции, работающие со списками.

Функция LIST – это основная функция, работающая со списками:

(list [<элемент 1> [<элемент 2> … [<элементN>] …]]))

Функция NTH применяется для извлечения из списка элемента по порядковому номеру (нумерация элементов списка осуществляется слева направо и начинается с нуля):

(nth <номер><список>)

2 Использование AutoLISP для создания параметрических чертежей в AutoCAD

Напишем программу параметрического моделирования детали, изображенной на рис. 1.

Для начала нарисуем параметрический эскиз детали. Указываются:

– размеры детали, которые являются входными параметрами к программе (*nD*1, *nD*2, *nD*3, *H*1, *H*2);

– базовая точка;

– точки, которые будут рассчитываться при программировании (*p*1, *p*2, *p*3, *p*4, *p*5, *p*6, *p*7, *p*8).

Эскиз должен использоваться на этапе программирования как пособие для верного понимания детали и как руководство к расчету точек, необходимых для отображения детали (рис. 2).

Программа может быть создана с помощью любого редактора (например, Блокнота), но предпочтительно использовать специально разработанный для этой цели встроенный в *AutoCAD* редактор *VisualLISP* (рис. 3).

3 Создание программы на AutoLISP

Программа включает три блока (функции пользователя):

– блок создания слоев (выполняется вначале);

– блок ввода данных;

– блок построения чертежа параметрической детали.

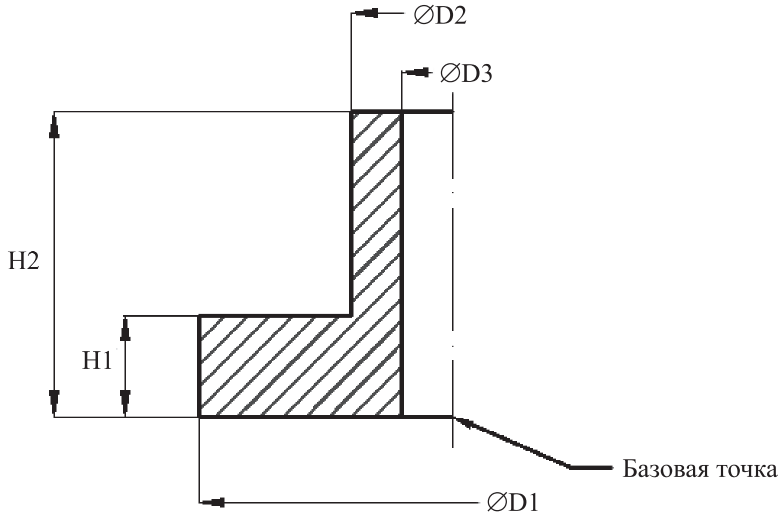

Рис. 1. Задание для создания параметрической модели

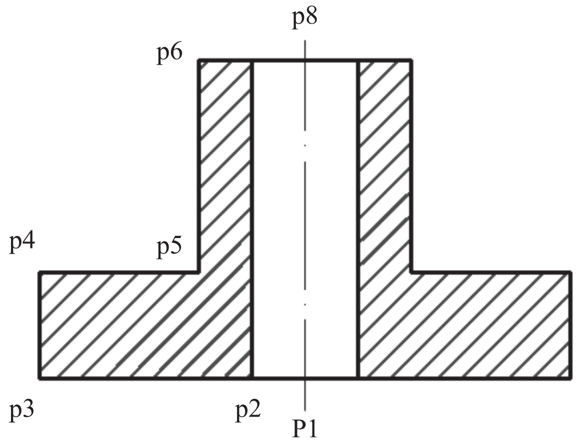

Рис. 2. Эскиз

Блок создания слоев *param*:

(defun c: param)

(command "_layer" "_m" "Baseline" "_LW" "0.5" "" "_c" "_white" "" "")

(command "_layer" "_m" "Lines" "_LW" "0.5" "" "_c" "_blue" "" "_LT" "Осевая" "" "")

(command "_layer" "_m" "Hatch" "_LW" "0.2" "" "_c" "_blue" "" "")

Здесь используется команда *AutoCAD layer* (слой) с параметрами *M* (*name* – наименование слоя), *LW* (*line width* – ширина линии), *LT* (*line type* – тип линии), *C* (*color* – цвет).

Этот блок запускается первым, поэтому перед названием функции стоит префикс *c:*. Создаются три слоя *Baseline* (базовая линия), *Lines* (основные линии) и *Hatch* (штриховка). Слои различаются цветом (белый, синий), толщиной линии (0.2, 0.5) и типом линии (непрерывная, штрихпунктирная).

В конце блока – вызов блока ввода данных *Try_param*.

Блок ввода данных *Try_param*:

(defun Try_param (/ P1 D1 D2 D3 H1 H2)

(command "_layer" "_s" "Lines" "")

;Параметрическое моделирование – точки вводятся пользователем во время работы программы с помощью функций ввода данных и указания объектов


```
;*********************
;Блок ввода данных
;*********************
(defun Try_param(/ P1 D1 D2 D3 H1 H2)
(initget 1)
(setq P1 (Getpoint "Укажите базовую точку детали:"))
(initget 6)
(setq D1 (Getreal "\nD1 <100.0>:"))
        (if (not D1)(setq D1 100.0))
(setq D2 (Getreal "\nD2 <40.0>:"))
        (if (not D2)(setq D2 40.0))
(setq D3 (Getreal "\nD3 <10.0>:"))
        (if (not D3)(setq D3 100.0))
(setq H1 (Getreal "\nH1 <20.0>:"))
        (if (not H1)(setq H1 20.0))
(setq H2 (Getreal "\nH2 <60.0>:"))
        (if (not H2)(setq H2 60.0))
(DrawDetail P1 D1 D2 D3 H1 H2)
)
;*********************
;Блок построения чертежа детали
;*********************
(defun DrawDetail (P1 D1 D2 D3 H1 H2 / p2 p3 p4 p5 p6 p7 x y det)
;Обходим деталь по часовой стрелке
(setq
x (nth 0 P1)
y (nth 1 P1)
p2 (list (- x (/ D3 2.0)) y)
p3 (list  (- x (/ D1 2.0)) y )
p4 (list  (- x (/ D1 2.0)) (+ y H1))
p5 (list  (- x (/ D2 2.0)) (+ y H1))
p6 (list  (- x (/ D2 2.0)) (+ y H2))
p7 (list  (- x (/ D3 2.0)) (+ y H2))
p8 (list  x  (+ y H2) )
)
;Отключение привязок
        (command "_osnap" "_none")
; рисование
        (command "_pline" p2 "_w" "0" "" p3 p4 p5 p6 p7 "_c")

; сохранение нарисованного примитива
        (setq det (entlast))
;Зеркальное отображение нарисованной детали
        (command "_mirror" det "" P1 P8 "_N")

;рисование отрезков
(command "_pline" p2 (list (+ x (/ D3 2)) y ) ""  )
(command "_pline" p7 (list (+ x (/ D3 2)) (+ y H2) ) ""   )
;установка белого цвета
        (command "_layer" "_s" "Lines" "")
;Ось
        (command "_line" (list x y ) (list x (+ y H2 )) "" )
;Штриховка
;       (command "_layer" "_s" "Hatch" "")
;( command " bhatch" " p" " ansi31" "0.5"  "0"  p9    p10 "" )
```

Рис. 3. Программа из *AutoLISP*

(initget 1)
(setq P1 (Getpoint “Укажите базовую точку детали:”))
(initget 6)
(setq D1 (Getreal «\nD1 <100.0>:»))
(if (not D1) (setq D1 100.0))
(setq D2 (Getreal «\nD2 <00.0>:»))
(if (not D2) (setq D2 40.0))
(setq D3 (Getreal «\nD3 <20.0>:»))
(if (not D3) (setq D3 20.0))
(setq H1 (Getreal «\nH1 <20.0>:»))
(if (not H1) (setq H1 20.0))
(setq H2 (Getreal «\nH2 <60.0>:»))
(if (not H2) (setq H2 60.0))
(prompt «Try_param»)
(DrawDetail P1 D1 D2 D3 H1 H2)

В конце блока – вызов блока построения чертежа параметрической детали *DrawDetail*.

Здесь используется команда *AutoCAD layer* (слой) с параметром *S* (*set* – установить слой).

4 Блок построения чертежа параметрической детали DrawDetail

Блок построения чертежа:
(defun DrawDetail (P1 D1 D2 D3 H1 H2 / p2 p3 p4 p5 p6 p7 x y det)
Обходим деталь по часовой стрелке
Параметрическое моделирование – рассчитываются координаты введенных пользователем точек с использованием функций работы со списками
(setq
x (nth 0 P1)
y (nth 1 P1)
p2 (list (- x (/ D3 2.0)) y)
p3 (list (- x (/ D1 2.0)) y)
p4 (list (- x (/ D1 2.0)) (+ y H1))
p5 (list (- x (/ D2 2.0)) (+ y H1))
p6 (list (- x (/ D2 2.0)) (+ y H2))
p7 (list (- x (/ D3 2.0)) (+ y H2))
p8 (list x (+ y H2))
Отключение привязок
(command «_osnap» «_none»)
Слой Lines – текущий
(command “_layer” “_s” “BaseLine” “”)
рисование
(command «_pline» p2 «_w» «0» «» p3 p4 p5 p6 p7 «_c»)
Сохранение нарисованного примитива
(setq det (entlast))
Зеркальное отображение нарисованной детали
(command “_mirror” det “” P1 P8 “_N”)
Рисование отрезков
(command «_pline» p2 (list (+ x (/ D3 2)) y) «»)
(command «_pline» p7 (list (+ x (/ D3 2)) (+ y H2)) «»)
установка белого цвета
(command «_layer» «_s» «Lines» «»)
Ось
(command «_line» (list x y) (list x (+ y H2)) «»)
Штриховка
(command «_layer» «_s» «Hatch» «»)
(command «_bhatch» «_p» «_ansi31» (/ (+ D1 H2) 70.0) «0» (list (- x (/ D2 2.0)) (+ y (/ H1 2.0))) (list (+ x (/ D2 2.0)) (+ y (/ H1 2.0)) «»))
(prompt “DrawDetail”)
DrawDetail

Здесь производится расчет координат *x* и *y* базовой точки *P*1 (базовая точка указывается с помощью мыши или вводится с клавиатуры). Рассчитываются координаты точек *p1*, *p2*, *p3*, *p4*, *p5*, *p6*, *p7*, *p8*. Отключаются объектные привязки. Устанавливается текущий слой и рисуется по часовой стрелке контур детали (рис. 1). Нарисованному примитиву присваивается имя *det*. Выполняется зеркальное отображение и рисуются два дополнительных отрезка.

На слое белого (черного) цвета рисуется штрихпунктирная ось. В заключение на слое *Hatch* выполняется штриховка (см. рис. 2).

Используются команды *AutoCAD osnap* (объектная привязка), *layer* (слой) с параметром *S* (*set* – установить слой), *pline* (полилиния), *line* (отрезок), *bhatch* (штриховка) с параметром *P* (*prototype* – тип).

Сохраним файл, состоящий из трех функций, под именем *param.lsp*.

Загрузить этот файл можно одним из способов: меню → сервис \ *AutoLISP* \ Приложения (команда *appload*), выводится окно, в котором можно указать файл для загрузки (рис. 4).

При фиксированном расположении загружаемого модуля нужно записать строку для его загрузки в командной строке *AutoCAD*: (load «диск:\\папка\\param.lsp»).

пользованием *AutoLISP* не требуется вводить каких-либо изменений в текст программы. Достаточно лишь запустить программу для выполнения, в процессе диалога указать базовую точку и ввести новые данные. Программа в процессе выполнения сделает все перерасчеты.

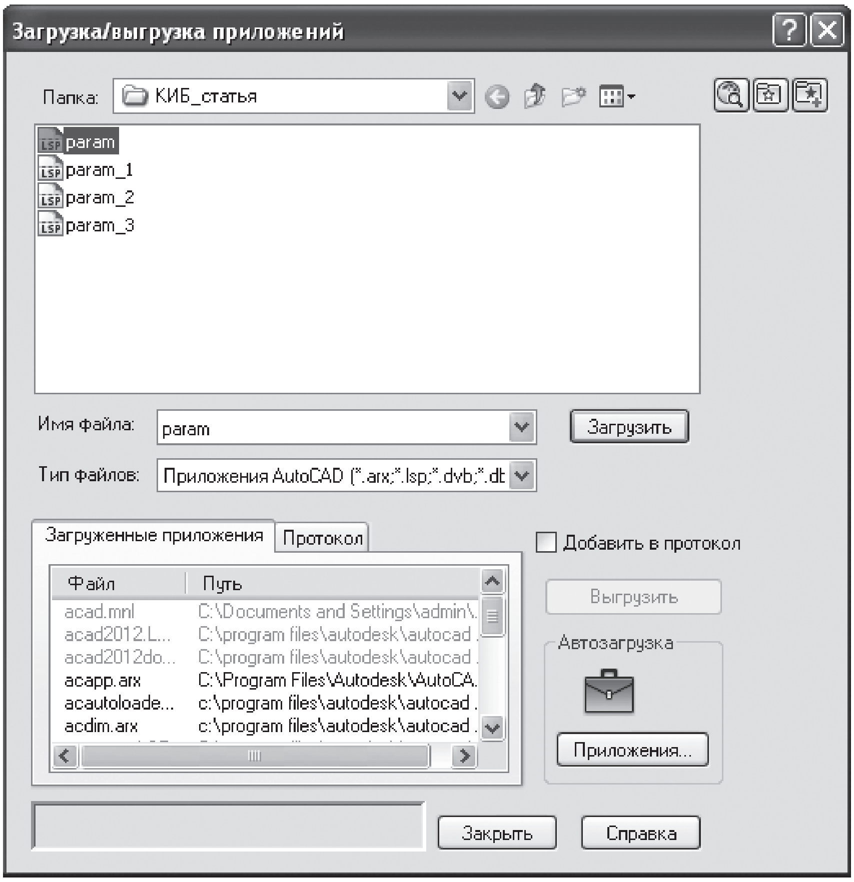

Рис. 4. Файл для загрузки

Для запуска программы для выполнения главная функция указывается в круглых скобках (при наличии префикса «*c:*» – без круглых скобок) в командной строке *AutoCAD*.

Если запуск должен быть автоматическим, нужные функции определяются в файле *ACAD.LSP* в функции *s:: startup*.

Заключение

Приведенная программа демонстрирует успешное применение языка для создания параметрической модели простой детали. При параметрическом моделировании с использованием *AutoLISP* не требуется вводить каких-либо изменений в текст программы.

В дальнейшем *AutoLISP* может быть рекомендован для создания чертежей значительно более сложных деталей, имеющих много размеров, которые поэтому нежелательно создавать путем обычного моделирования.

Библиографический список

1. **Мир** Лиспа / Э. Хювенен, Й. Сеппянен. – Москва : Мир, 1990.

2. **Среда** программирования на AutoLISP в графической системе AutoCAD : справочник / А. А. Тучков, А. М. Покровский. – Санкт-Петербург : Звезда, 1992.

3. **Автолисп** – язык графического программирования / Г. А. Бугрименко. – Москва : Машиностроение, 1992.

4. **AutoCAD**: программирование и адаптация / Ю. А. Кречко. – Москва : Диалог – МИФИ, 1996.

УДК 681.518.5:004.052.32

Вал. В. Сапожников, Вл. В. Сапожников, Д. В. Ефанов
Петербургский государственный университет путей сообщения

СИНТЕЗ КОНТРОЛЕПРИГОДНЫХ ДИСКРЕТНЫХ УСТРОЙСТВ ПУТЕМ ПРИМЕНЕНИЯ МОДУЛЬНЫХ КОДОВ С СУММИРОВАНИЕМ

При проектировании контролепригодных дискретных устройств часто используются коды с суммированием. Классический код с суммированием позволяет синтезировать дискретные устройства с обнаружением любых одиночных ошибок, возникающих в элементах внутренней структуры контролируемого устройства. При этом исходное устройство преобразуется и становится устройством с независимыми или монотонно-независимыми выходами. Показан подход, применение которого уменьшает сложность системы функционального контроля. Это достигается использованием в качестве основы системы контроля модульного кода с суммированием, а не кода Бергера.

функциональный контроль; код Бергера; модульный код с суммированием; информационные разряды; необнаруживаемая ошибка; однонаправленная необнаруживаемая ошибка; комбинационные схемы с монотонно-независимыми выходами.

Введение

При проектировании контролепригодных дискретных устройств [1–5] используют различные способы достижения высокого уровня надежности, при этом несомненно важное значение имеет техническое диагностирование. Под техническим диагностированием понимается процесс определения технического состояния контролируемого объекта с заданной полнотой и глубиной поиска [6].

Процесс диагностирования может осуществляться в двух режимах: тестовом и функциональном. При тестовом диагностировании устройство кратковременно отключается и на его входы подается тест, включающий в себя совокупность проверок [6, 7]. По завершении диагностирования устройство вновь включается в работу. Такой подход, например, требует при построении систем управления 100%-го резервирования. В отличие от тестового диагностирования, при функциональном, или рабочем, контроле определение состояния производится без отключения объекта [2]. В современных микропроцессорных системах управления оба метода диагностирования используются наравне.

Настоящая работа посвящена развитию теории функционального контроля комбинационных логических устройств, являющихся обязательными компонентами любой микропроцессорной и микроэлектронной системы управления.

1 Система функционального контроля

В системе функционального контроля (рис. 1) выделяются три блока: исходное устройство $f(x)$, в функции которого входит вычисление системы булевых функций $f_1(x)$,