ИНСТИТУТ ЯДЕРНОЙ ФИЗИКИ СО АН СССР

А.Г. Грозин

РЕШЕНИЕ ФИЗИЧЕСКИХ ЗАДАЧ НА ЯЗЫКЕ REDUCE.

1. Язык REDUCE

2. Классический нелинейный осциллятор

ПРЕПРИНТ 88-115

НОВОСИБИРСК

Решение физических задач на языке REDUCE.
1. Язык REDUCE
2. Классический нелинейный осциллятор

А.Г. Грозин

Институт ядерной физики
630090, Новосибирск 90, СССР

АННОТАЦИЯ

Настоящий препринт представляет собой первую часть пособия по применению языка аналитических вычислений REDUCE для решения физических задач. В нем приведено много примеров, которые могут быть полезны для конструирования программ решения физических задач самой различной природы. В этой части приведены примеры, иллюстрирующие язык REDUCE (п.1), и рассмотрена задача о классическом нелинейном осцилляторе (п.2).

Solving physical problems with REDUCE.
1. REDUCE language
2. Classical nonlinear oscillator

A.G. Grozin

Institute of Nuclear Physics
630090, Novosibirsk, USSR

ABSTRACT

This preprint is the first part of the problem book on using REDUCE in physics. It contains many examples useful for the construction of programs for solving physical problems of very different nature. This part contains examples illustrating REDUCE language (sect. 1) and the problem of classical nonlinear oscillator (sect. 2).

ВВЕДЕНИЕ

REDUCE — это язык программирования для аналитических вычислений. Обычные языки, такие, как ФОРТРАН или ПАСКАЛЬ, позволяют делать только численные расчеты. REDUCE не единственная система аналитических вычислений, но одна из наиболее мощных и широко распространенных. Она доступна на всех основных типах ЭВМ, включая IBM-370 (ЕС), PDP-11 (СМ, Электроника), VAX и персональных компьютерах типа IBM PC.

Аналитические вычисления на ЭВМ все более широко применяются в физике. Традиционными областями их приложения являются небесная механика, общая теория относительности и физика элементарных частиц. Так, при помощи ЭВМ вычислен вклад трехпетлевых диаграмм в аномальный магнитный момент электрона, что позволило достичь небывалой в физике точности согласия теории и эксперимента $\sim 10^{-10}$. Сейчас уже трудно назвать область физики, где бы не применялась машинная аналитика.

Имеются два основных класса применений аналитических вычислений на ЭВМ. С одной стороны, это рекордно трудоемкие вычисления, в которых обрабатываемые выражения содержат миллионы членов. Они требуют многих дней работы крупного быстродействующего компьютера, и никогда не могли бы быть проведены вручную. С другой стороны, с ростом доступности компьютеров все большее число физиков получают возможность проводить не очень сложные выкладки, и быстро (и главное безошибочно) получать результаты, на вывод (и проверку) которых вручную было бы затрачено довольно много времени.

С наступлением эры персональных компьютеров, для каждого физика становится необходимым уметь использовать их не только для численных, но и для аналитических вычислений. Скоро проведение алгебраических или тригонометрических преобразований ручкой на бумажке станет таким же анахронизмом, как умножение в столбик в век калькуляторов. Кстати, уже существуют карманные калькуляторы со встроенной системой аналитических вычислений. Конечно, они могут применяться только для решения небольших задач. Но современный (и тем более будущий) персональный компьютер дает такие возможности, которые в недалеком прошлом были доступны только на наиболее крупных ЭВМ.

Активно работающему физику, как правило, некогда и неинтересно штудировать подробные описания систем аналитических вычислений. Обычно, к тому же, они написаны довольно формально, и чтение первого десятка страниц способно только нагнать тоску. Вместо этого физик действует по прецедентам: если он уже встречался с подобной задачей, или слышал, что кто-то что-то такое решал, то он смотрит на свою или чужую готовую программу и сооружает новую по ее образу и подобию. Поэтому ему более полезен задачник с решениями, чем учебник со строгим и полным описанием языка программирования.

Здесь я попытался собрать коллекцию задач из различных областей физики с решениями на языке REDUCE. Внимательно просматривая эти решения, наверняка можно обнаружить отдельные куски программ или просто полезные приемы, которые могут пригодиться и в других ситуациях. Конкретная область физики при этом почти несущественна: работа с отрезками рядов Тэйлора или Фурье или решение линейных систем уравнений могут встретиться в задачах любой природы.

В первом параграфе этого задачника собраны примеры с пояснениями, по которым можно ознакомиться с языком REDUCE. Другие возможности разбросаны по решениям всех физических задач, и иногда поясняются комментариями. Впрочем, в большинстве случаев они рассчитаны скорее на возможность прямо использовать их в других программах, и даются без пояснений (например, измерение времени, затраченного на решение задачи).

Хорошее описание системы REDUCE дано ее автором [1]. Имеется также несколько учебников [2—4]. Особенно полезны интерактивные уроки по языку REDUCE [5], содержащие много примеров. Описание [1] и уроки [5] распространяются на магнитной ленте вместе с системой, и поэтому обычно доступны везде,

где есть REDUCE. На ленте имеются также стандартные тесты системы и ее частей, из которых можно извлечь много интересных примеров.

Я не ставил своей целью сделать читателя специалистом по системе REDUCE. Поэтому совершенно не рассматриваются специфические для этой системы вопросы типа символьной моды, устройства системы и распределения памяти. Кроме того, разные версии REDUCE заметно отличаются друг от друга, и нет гарантии, что приведенные в тексте программы (отлаженные автором) будут без изменений работать на других версиях.

Вместо этого, я старался сконцентрировать внимание на самих аналитических алгоритмах. Почти все системы аналитических вычислений довольно похожи, и приводимые алгоритмы могут быть без особого труда записаны на языке любой из них. Так что, если Вам доступна какая-нибудь другая хорошая система, этот задачник не будет для Вас абсолютно бесполезен. Правда, если какие-нибудь возможности REDUCE, используемые в программах, недоступны в другой системе, то их придется моделировать, например, написав соответствующий набор процедур.

Мне приятно поблагодарить В.П. Гердта, А.П. Крюкова, С.Л. Панфиля, А.Я. Родионова, В.А. Ростовцева за многочисленные обсуждения различных вопросов аналитических вычислений на ЭВМ в физике; Д.Л. Шепелянского за чтение и обсуждение рукописи п.2; студентов НГУ — слушателей спецкурса за помощь и участие в решении ряда задач; Н.А. Грозину за помощь в подготовке рукописи к печати.

1. ЯЗЫК REDUCE

Первые примеры. Для того, чтобы начать работать с системой REDUCE, необходимо сначала в нее войти. Способ входа различен на разных машинах, в разных операционных системах, а иногда и на разных экземплярах одинаковых машин с одинаковой системой (это зависит от местных традиций). Часто для того, чтобы вызвать эту систему (если она вообще есть), достаточно просто набрать

```
REDUCE
```

Если это не помогает, обратитесь к местным экспертам.

После того, как REDUCE загрузился, он представляется:

пишет свою версию и дату создания. Затем вы можете вводить операторы языка REDUCE, заканчивая их точкой с запятой. Когда вам надоест, попрощайтесь с REDUCEом, набрав команду

BYE;

Далее в этом параграфе приведен протокол диалога с системой REDUCE, демонстрирующий многие ее возможности. Введенные пользователем операторы оканчиваются символом ; , а ответы REDUCEа — нет. Приведены также необходимые пояснения.

Простейшее, что можно попросить сделать систему, — это упростить выражение. Для этого нужно просто ввести это выражение, закончив его знаком ; . В ответ REDUCE выведет упрощенное выражение. Как видно из примеров, он раскрывает скобки (с приведением подобных) и приводит к общему знаменателю.

Однако было бы обидно, если бы полученные выражения просто пропадали. Чтобы иметь возможность использовать выражение в дальнейшем, нужно присвоить его какой-нибудь переменной. После этого вы можете применять эту переменную для построения новых выражений. Вместо нее будет подставляться ее значение.

```
(X+Y)**2;
```

$X^2+2*X*Y+Y^2$

```
X/(X+Y)+Y/(X-Y);
```

$(X^2+Y^2)/(X^2-Y^2)$

```
A:=(X+Y)**2;
```

A := $X^2+2*X*Y+Y^2$

```
A-(X-Y)**2;
```

```
4*X*Y
```

Свободные и связанные переменные. Как видно, есть переменные двух типов. Одним из них не присвоено никакое значение, и они обозначают сами себя. Такие переменные называются свободными, и могут встречаться в выводимых REDUCEом ответах.

Другим переменным присвоены значения — выражения. Они называются связанными. Когда связанная переменная встречается в выражении, она заменяется на свое значение. Поэтому она не может входить в ответ.

Если присвоить значение свободной переменной, она превратится в связанную. Если присвоить его связанной переменной, то ее старое значение пропадет и заменится новым.

А можно ли превратить связанную переменную обратно в свободную? Да, для этого служит команда CLEAR. Она уничтожает

значение переменной и делает ее свободной. Следует уничтожать все выражения, как только они стали ненужными, поскольку при этом освобождается память.

В приводимом примере связанная переменная A выражается через свободные переменные X и Y. Что будет, если потом присвоить значение переменной X? Тогда при выводе значения A оно будет упрощено, и вместо X будет подставлено его значение. Но само значение A при этом не изменится, что легко проверить, сделав X свободной переменной при помощи команды CLEAR. Однако если выполнить присваивание A := A, то новое, упрощенное в текущей среде значение переменной A заменит старое. Такие присваивания, непривычно выглядящие для программиста, знакомого с обычными численными языками, в языке REDUCE часто полезны и могут выполнять большую работу. Например, можно присвоить переменной A выражение с неопределенными коэффициентами, затем вычислить эти коэффициенты, а потом сделать A := A.

Что будет, если свободной переменной присвоить выражение, содержащее эту переменную? Такое присваивание пройдет нормально. Но затем при любой попытке использовать эту переменную (хотя бы просто вывести ее значение) произойдет зацикливание. В выражении, являющемся значением X, переменная X будет заменяться на это выражение. В получившемся результате опять будет производиться такая замена, и так до бесконечности. Для каждой замены требуется память в стеке, что приводит к его переполнению.

Кстати, если нас не интересует значение оператора, мы можем вместо ; завершить его символом $, который подавляет вывод значения.

```
% FREE AND BOUND VARIABLES ;
X:=Z+1$ A;
```

$Y^2+2*Y*Z+2*Y+Z^2+2*Z+1$

```
X:=Z-1$ A;
```

$Y^2+2*Y*Z-2*Y+Z^2-2*Z+1$

```
CLEAR X; X;
X
A;
```

$X^2+2*X*Y+Y^2$

```
X:=2*Z$ A:=A;
```

A := $Y^2+4*Y*Z+4*Z^2$

```
CLEAR X; A;
```
$Y^2+4*Y*Z+4*Z^2$
```
CLEAR A; A;
A
% THIS WILL LEAD TO INFINITE LOOP ;
X:=X+1;
```
$X := X+1$
```
X;
*** PUSHDOWN STACK OVERFLOW
#000000000X
CLEAR X; X;
X
```

Флаги. Можно управлять процессом упрощения выражений с помощью флагов. Если выключить флаг MCD (Making Common Denominator), то REDUCE не будет приводить к общему знаменателю. Если выключить флаг EXP (EXPansion), то он не будет раскрывать скобки. Обычно эти флаги включены.

Умеет ли REDUCE сокращать числитель и знаменатель на общий множитель? Видно, что в простых случаях умеет (например, когда числитель нацело делится на знаменатель). Он, однако, не находит общий множитель в более сложных случаях. Если включить флаг GCD (Greatest Common Divisor), то REDUCE будет находить и сокращать общие множители везде, где они есть. Обычно он выключен, поскольку поиски наибольшего общего делителя могут занять много времени. Его нужно включать на тех участках программы, где сокращения необходимы.

```
% MCD—MAKING COMMON DENOMENATOR ;
OFF MCD; B:=X/(X+Y)+Y/(X-Y);
```
$B := (X+Y)^{(-1)}*X+(X-Y)^{(-1)}*Y$
```
ON MCD; B;
```
$(X^2+Y^2)/(X^2-Y^2)$
```
% EXP—EXPANSION ;
OFF EXP; A:=(X+Y)**2;
```
$A := (X+Y)^2$
```
ON EXP; A;
```
$X^2+2*X*Y+Y^2$
```
% GCD—GREATEST COMMON DIVISOR ;
```



```
(X**2-Y**2)/(X-Y);
```
$X+Y$
```
(X**3-Y**3)/(X-Y);
```
$X^2+X*Y+Y^2$
```
C:=(X**3-Y**3)/(X**2-Y**2);
```
$C := (X^3-Y^3)/(X^2-Y^2)$
```
ON GCD; C:=C;
```
$C := (X^2+X*Y+Y^2)/(X+Y)$
```
OFF GCD;
```

Малые переменные. Часто в физических задачах некоторые переменные являются малыми параметрами. Они могут иметь разные порядки малости, приписываемые командой WEIGHT. Команда WTLEVEL дает указание системе сохранять члены только до указанного порядка малости. Так производится работа с отрезками степенных рядов по малым параметрам. Порядок малости переменной уничтожается командой CLEAR.

```
% SMALL VARIABLES—TRUNCATED POWER SERIES ;
WEIGHT X=1,Y=2; WTLEVEL 4;
P:=(X+Y+Z)**10;
```
$P := Z^6*(210*X^4+120*X^3*Z+360*X^2*Y*Z+45*X^2*Z^2$
$+90*X*Y*Z^2+10*X*Z^3+45*Y^2*Z^2+10*Y*Z^3+Z^4)$
```
CLEAR X,Y;
```

Анализ структуры выражения. Числитель рационального выражения выделяется функцией NUM (NUMerator), а знаменатель — функцией DEN (DENominator).

```
% NUMERATOR AND DENOMINATOR ;
% OF A RATIONAL EXPRESSION ;
C;
```
$(X^2+X*Y+Y^2)/(X+Y)$
```
NUM(C);
```
$X^2+X*Y+Y^2$
```
DEN(C);
```
$X+Y$

Для дальнейшего анализа нужно сначала познакомиться с массивами и циклами. Массив описывается командой ARRAY. Его индекс может пробегать значения от 0 до заданного размера массива. Размер может задаваться любым выражением; оно вычисля-

ется в момент описания, и должно иметь целое значение.

Элемент массива, пока ему ничего не присвоено, не является свободной переменной. Вместо этого он имеет значение 0. Если необходим массив свободных переменных, нужно использовать операторы, как это описано дальше.

Работа с массивами обычно проводится при помощи циклов.

```
% ARRAYS AND LOOPS ;
N:=5$ ARRAY AR(N);
FOR I:=0:N DO AR(I):=X**I;
FOR I:=0:N DO WRITE AR(I);
1
X
X^2
X^3
X^4
X^5
```

Теперь мы готовы анализировать структуру полиномов, которые стоят в числителе и знаменателе выражения общего вида (REDUCE приводит его к общему знаменателю и рассматривает как рациональное). А именно, полином можно разложить по степеням некоторой переменной X (или функции с параметрами). Это делает процедура COEFF. Она возвращает максимальную степень X в полиноме. Но главная польза от нее состоит не в этом значении, а в побочном эффекте. Она помещает коэффициент при X**I в I-ый элемент массива AR. В новых версиях REDUCEa она автоматически подправляет размер массива, делая его равным максимальной степени.

Если умножить I-ый элемент массива на X**I и все их сложить, то мы восстановим исходный полином. Для суммирования удобно использовать цикл с SUM. Он является аналогом математического символа $\sum_{i=0}^{n}$.

Другой полезный способ анализа полинома — попытаться разложить его на множители. Если факторизация на множители с целыми коэффициентами возможна, то REDUCE может ее произвести. Это делает процедура FACTORIZE. Она возвращает число найденных множителей, а в качестве побочного эффекта складывает эти множители в элементы массива. Нулевой элемент содержит

численный общий множитель. Эта процедура также подстраивает размер массива. Правда, для длинных полиномов, особенно от нескольких переменных, она может работать очень долго.

Чтобы проверить правильность факторизации, можно перемножить все элементы массива. Для перемножения удобно использовать цикл с PRODUCT. Он является аналогом математического символа $\prod_{i=0}^{n}$.

```
% COEFFICIENTS AT DIFFERENT POWERS OF A VARIABLE ;
% IN A POLYNOMIAL ;
N:=COEFF(P,X,AR);
N := 4
% N IS NOW A MAXIMUM POWER, DIMENSION OF AR IS NOW N ;
FOR I:=0:N DO WRITE AR(I);
Z^8*(45*Y^2+10*Y*Z+Z^2)
10*Z^8*(9*Y+Z)
45*Z^7*(8*Y+Z)
120*Z^7
210*Z^6
% SUMMING ALL POWERS AGAIN GIVES A POLINOMIAL ;
% Q=P ;
Q:=FOR I:=0:N SUM AR(I)*X**I$
IF Q=P THEN WRITE "OK" ELSE WRITE "ERROR";
OK
CLEAR Q,P;

% FACTORIZATION OF A POLYNOMIAL ;
P:=X**2-Y**2$ N:=FACTORIZE(P,AR);
N := 2
FOR I:=0:N DO WRITE AR(I);
1
X+Y
X-Y
% PRODUCT OF ALL FACTORS GIVES A POLYNOMIAL Q=P ;
Q:=FOR I:=0:N PRODUCT AR(I);
Q := X^2-Y^2
IF Q=P THEN WRITE "OK" ELSE WRITE "ERROR";
```

```
OK
CLEAR P,Q; CLEAR AR;
```

Дифференцирование. В примерах вычислены производные C по X; по Y; вторая производная по X; и наконец, первая производная по X и вторая по Y.

Иногда не все используемые переменные независимы. Пусть, например, Y зависит от X. Можно сообщить об этом REDUCEy командой DEPEND. Зависимость отменяется командой NODEPEND.

```
% DIFFERENTIATION ;
% ---------- ;
DF(C,X);
```

$$(X*(X+2*Y))/(X^2+2*X*Y+Y^2)$$

```
DF(C,Y);
```

$$(Y*(2*X+Y))/(X^2+2*X*Y+Y^2)$$

```
DF(C,X,2);
```

$$(2*Y^2*(X+Y))/(X^4+4*X^3*Y+6*X^2*Y^2+4*X*Y^3+Y^4)$$

```
DF(C,X,Y,2);
```

$$(2*X*(-X+2*Y))/(X^4+4*X^3*Y+6*X^2*Y^2+4*X*Y^3+Y^4)$$

```
CLEAR C;
% DECLARING Y AS DEPENDENT ON X ;
DEPEND Y,X; A:=Y/X$ DF(A,X);
```

$$(DF(Y,X)*X-Y)/X^2$$

```
% REMOVING THIS DEPENDENCE ;
NODEPEND Y,X; DF(A,X);
```

$$(-Y)/X^2$$

Подстановки. Как увеличить X на 1 в выражении A? Это делается при помощи функции SUB (SUBstitute). Она упрощает свой аргумент, заменяет переменную на выражение, и затем упрощает результат. Можно заменять несколько переменных одновременно. Например, так можно поменять 2 переменные местами.

Команда LET вводит правило подстановки. Отныне и до тех пор, пока подстановка не будет уничтожена командой CLEAR, при упрощении всех выражений левая часть подстановки будет заменяться на правую. REDUCE заменяет наибольшую степень левой части, какую может выделить.

После проведения замены, выражение упрощается, и к нему опять применяются все активные подстановки. Поэтому, если

задать зацикленную систему подстановок, то произойдет переполнение стека.

Подстановки, ставшие ненужными, следует сразу удалять командой CLEAR, иначе они будут замедлять дальнейшую работу.

Подчеркнем еще раз различия функции SUB и LET-подстановки:

1. Функция SUB производит замену в одном заданном выражении. LET-подстановка применяется ко всем выражениям, пока не будет уничтожена.
2. Функция SUB производит однократную замену. LET-подстановка применяется повторно, пока это возможно.
3. Функция SUB может заменять только переменную (или функцию с аргументами). LET-подстановка допускает широкий класс выражений в левой части.

Если возможности функции SUB достаточны, то лучше применять ее, а не LET-подстановку.

```
% SUBSTITUTING AN EXPRESSION FOR A VARIABLE ;
% ------------------------ ;
A:=(X+Y)**2;
```

$$A := X^2+2*X*Y+Y^2$$

```
A:=SUB(X=X+1,A);
```

$$A := X^2+2*X*Y+2*X+Y^2+2*Y+1$$

```
A:=SUB(X=1-X,Y=Y-1,A);
```

$$A := X^2-2*X*Y-2*X+Y^2+2*Y+1$$

```
% EXCHANGE OF TWO VARIABLES ;
A:=SUB(X=Y,Y=X,A);
```

$$A := X^2-2*X*Y+2*X+Y^2-2*Y+1$$

```
CLEAR A;

% DECLARING A SUBSTITUTION RULE ;
% ---------------------- ;
LET X*Y**2=Z; X**2*Y**4;
```

$$Z^2$$

```
X**3*Y**5;
```

$$X*Y*Z^2$$

```
% REMOVING THIS RULE ;
CLEAR X*Y**2; X**2*Y**4;
```

$$X^2*Y^4$$

```
X**3*Y**5;
```

$X^{3}*Y^{5}$

```
% THIS SHOULD LEAD TO AN INFINITE LOOP ;
LET X=Y+1,Y=2*X; X;
*** PUSHDOWN STACK OVERFLOW
#000000000X
CLEAR X,Y;
```

Комплексная алгебра. В REDUCEe можно работать с комплексными выражениями; I означает мнимую единицу. Комплексное сопряжение выполняется при помощи функции SUB.

```
% COMPLEX ALGEBRA ;
% ---------- ;
Z:=X+I*Y;
Z := I*Y+X
Z:=Z**2;
```

$Z := 2*I*X*Y+X^{2}-Y^{2}$

```
% COMPLEX CONJUGATE ;
SUB(I=-I,Z);
```

$-2*I*X*Y+X^{2}-Y^{2}$

```
CLEAR Z;
```

Управление форматом вывода. Можно до некоторой степени влиять на форму, в которой REDUCE выводит выражения. Это может значительно облегчить понимание результатов.

REDUCE выводит члены выражения, а также переменные в каждом члене, в каком-то своем порядке. Команда ORDER Y,X; заставляет его выводить сначала Y, затем X, а потом уже все остальные переменные. Команда ORDER NIL; отменяет это указание.

Флаг ALLFAC (ALL FACtors), обычно включенный, требует выносить за скобки все простые общие множители (числа и степени переменных). Флаг DIV (DIVide), обычно выключенный, заставляет делить каждый член в числителе на простые общие множители знаменателя.

Команда FACTOR X; заставляет группировать члены по степеням X, и выводить выражение в виде этих степеней с коэффициентами. Если дополнительно включить флаг RAT (RATional), то каждая такая группа членов будет делиться на знаменатель, так что получатся степени X с рациональными коэффициентами. Действие этой команды отменяется командой REMFAC X;.

Флаг LIST, обычно выключенный, заставляет REDUCE выводить каждый член выражения (начиная с + или —) с новой строчки.

Флаг NAT (NATural), обычно включенный, заставляет производить вывод в естественной математической форме, т. е. показатели степени размещаются строчкой выше. Конечно, REDUCE не сможет потом прочитать такое выражение. Если выключить этот флаг, то степени будут выводиться через **, т. е. в виде, пригодном для ввода. Кроме того, в этом случае любой вывод заканчивается символом $.

```
% CONTROLLING OUTPUT FORMAT ;
% ---------------- ;
W:=(X**2*(Y**2+2*Y)+X*(Y**2+Z)/(2*A))/(Y+2);
```

$W := (X*(2*A*X*Y^{2}+4*A*X*Y+Y^{2}+Z))/(2*A*(Y+2))$

```
% ORDER OF VARIABLES IN OUTPUT—Y, X, ;
% AND THEN ALL OTHER VARIABLES ;
ORDER Y,X; W;
```

$(X*(2*Y^{2}*X*A+Y^{2}+4*Y*X*A+Z))/(2*A*(Y+2))$

```
ORDER NIL;

% ALLFAC—ALL SIMPLE FACTORS—OUT OF BRACKETS ;
OFF ALLFAC; W;
```

$(2*A*X^{2}*Y^{2}+4*A*X^{2}*Y+X*Y^{2}+X*Z)/(2*A*Y+4*A)$

```
ON ALLFAC;

% DIVIDE ALL TERMS IN THE NUMERATOR ;
% BY SIMPLE FACTORS OF DENOMINATOR ;
ON DIV; W;
```

$X*(1/2*A^{(-1)}*Y^{2}+1/2*A^{(-1)}*Z+X*Y^{2}+2*X*Y)/(Y+2)$

```
OFF DIV;

% FACTOR—PUT POWERS OF X OUT OF BRACKETS ;
FACTOR X; W;
```

$(2*X^{2}*A*Y*(Y+2)+X*(Y^{2}+Z))/(2*A*(Y+2))$

```
% RAT—DIVIDE ALL FACTORED PARTS ;
% OF THE NUMERATOR BY THE DENOMINATOR ;
% POWERS OF X WITH RATIONAL COEFFICIENTS ;
ON RAT; W;
```

```
X^2*Y+X*(Y^2+Z)/(2*A*Y+4*A)
ON DIV; W;
X^2*Y+1/2*X*A^(-1)*(Y^2+Z)/(Y+2)
OFF DIV,ALLFAC; W;
X^2*Y+X*(Y^2+Z)/(2*A*Y+4*A)
OFF RAT; ON ALLFAC; REMFAC X;

% LIST—EACH TERM ON A SEPARATE LINE ;
ON LIST; W;
(X*(2*A*X*Y^2
    +4*A*X*Y
    +Y^2
    +Z))/(2*A*(Y
    +2))
OFF LIST;

% NAT—NATURAL OUTPUT ;
OFF NAT; W;
(X*(2*A*X*Y**2+4*A*X*Y+Y**2+Z))/(2*A*(Y+2))$
WRITE "ABCD";
ABCD$
ON NAT;
```

Работа с файлами. Команда IN INFILE; заставляет REDUCE вместо ввода с терминала читать далее программу из файла INFILE. Этот файл должен заканчиваться командой END;.

Команда OUT OUTFILE; направляет весь дальнейший вывод вместо терминала в файл OUTFILE. Команда SHUT OUTFILE; закрывает файл, и возвращает вывод на терминал.

Связь имен, используемых в командах IN и OUT, с реальными файлами на диске обычно производится средствами операционной системы. Так, в системе CMS для установления связи имени INFILE с файлом FNAME RED A (здесь FNAME—имя файла, RED—его тип, а A—имя диска CMS, на котором он находится) нужно перед запуском REDUCEa выполнить команду

FILEDEF INFILE DISK FNAME RED A

Однако если вы забыли это сделать, не расстраивайтесь—вы можете выполнить команду CMS из REDUCEa:

CMS "FILEDEF INFILE DISK FNAME RED A";

Как сохранить некоторые выражения для дальнейшего использования? Для этого их нужно записать в файл в таком виде, чтобы REDUCE мог их потом прочесть. Направьте вывод в файл: OUT OUTFILE;. Выключите флаг NAT; для экономии времени лучше выключить и флаг PRI (PRInting), что приводит к выключению всех украшающих преобразований и к выводу выражений в форме, близкой к внутренней. Присваивание A:=A; выведет A:=⟨выражение⟩$, поэтому при вводе этого файла значение A восстановится. Вы можете записать выражение и под другим именем: WRITE "B:=",A;. Не забудьте записать в файл END$: WRITE "END";. Затем закройте файл: SHUT OUTFILE;. Теперь он готов; в следующий раз вы можете ввести его командой IN. Для нормального продолжения работы включите флаги NAT и PRI.

В примере именно это и сделано. После команды OUT OUTFILE; следуют команды W:=W;WRITE "END";SHUT OUTFILE;. Их не видно на распечатке, поскольку вывод на терминал в этот момент не производится. После команды IN INFILE; следуют две команды: W:=⟨выражение⟩$ END$. Они введены не с терминала, а из файла.

```
% WORKING WITH FILES ;
% ---------- ;
CMS "FILEDEF OUTFILE DISK FNAME RED A";
0
OFF NAT,PRI; OUT OUTFILE;

CLEAR W;
% INPUT FROM THE FILE ;
CMS "FILEDEF INFILE DISK FNAME RED A";
0
IN INFILE;
W := ((2*Y**2+4*Y)*X**2*A+(Y**2+Z)*X)/((2*Y+4)*A)$
END$
W;

(X*(2*A*X*Y^2+4*A*X*Y+Y^2+Z))/(2*A*(Y+2))
CLEAR W;
```

Элементарные функции. REDUCE знает функции SIN, COS, LOG, числа PI и E и их простейшие свойства. Экспоненту можно записывать как EXP(X) или E**X, корень—как SQRT(X) или X**(1/2). Он умеет дифференцировать выражения с элементарными функциями. REDUCE умеет также вычислять очень многие неопределенные интегралы.

```
% ELEMENTARY FUNCTIONS ;
% ---------------- ;
SIN(-X);
-SIN(X)
COS(-X);
COS(X)
SIN(0);
0
COS(0);
1
SIN(PI);
0
COS(PI);
(-1)
SIN(PI/2);
1
LOG(E);
1
LOG(1);
0
SQRT(X**2);
X
SQRT(4);
2

% DIFFERENTIATION ;
% --------- ;
A:=SIN(X)/X$ DF(A,X);
(COS(X)*X-SIN(X))/X**2
DF(A,X,2);
(-2*COS(X)*X-SIN(X)*X**2+2*SIN(X))/X**3
DF(E**X*LOG(X),X);
(E**X*(LOG(X)*X+1))/X
A:=X**X;
A := X**X
DF(A,X);
X**X*(LOG(X)+1)
DF(A,X,2);
```



```
(X**X*(LOG(X)**2*X+2*LOG(X)*X+X+1))/X
% INTEGRATION ;
% ---------- ;
INT(1/X,X);
LOG(X)
INT(LOG(X),X);
X*(LOG(X)-1)
B;
(X**2+Y**2)/(X**2-Y**2)
B:=INT(B,X);
B := LOG(X-Y)*Y-LOG(X+Y)*Y+X
DF(B,X);
(X**2+Y**2)/(X**2-Y**2)
CLEAR B;
INT(X*SIN(X),X);
-COS(X)*X+SIN(X)
INT(1/SIN(X),X);
LOG(TAN(X/2))
```

Функции пользователя. Вы можете вводить свои функции при помощи описания OPERATOR. Их свойства определяются LET-подстановками.

Если мы хотим ввести функцию F(X)=X**2, то подстановка LET F(X)=X**2; для этого недостаточна — она действует на F(X), но не на F(X+Y). Нужно приписать перед ней FOR ALL X.

При дифференцировании REDUCE оставляет производные от функций пользователя невычисленными. Если Вы знаете, чему равна производная от Вашей функции, то можете сообщить это REDUCEу при помощи LET-подстановки.

Как определить четную или нечетную функцию? Легко сообщить REDUCEу, что для нечетной функции F(0)=0. Однако он должен сам решать, приводить F(X—Y) и F(Y—X) к первой или второй форме. Для этого используются подстановки, в которых кроме произвольных переменных (FOR ALL) задано условие на них (SUCH THAT). В данном случае нужно такое условие, чтобы для любого выражения X либо F(X) приводилось к F(—X), либо наоборот.

Почти подходит условие ORDP(X,Y), которое истинно, если X предшествует Y в смысле некоторого внутреннего отношения порядка, которое REDUCE устанавливает между всеми выражени-

ями. Но ORDP(X,X) истинно, и если не застраховаться от этого случая, то вычисление F(0) приведет к зацикливанию.

Точно так же можно определить симметричную или антисимметричную функцию двух аргументов.

```
% USER FUNCTIONS ;
% ---------- ;
OPERATOR F,G;
LET F(X)=X**2; F(X);
X^2
F(X+Y);
F(X+Y)
CLEAR F(X);
FOR ALL X LET F(X)=X**2; F(X);
X^2
F(X+Y);
X^2+2*X*Y+Y^2
FOR ALL X CLEAR F(X);
A:=(X*F(X))**2;
A := F(X)^2*X^2
DF(A,X);
2*F(X)*X*(F(X)+DF(F(X),X)*X)
FOR ALL X LET DF(F(X),X)=G(X); DF(A,X);
2*F(X)*X*(F(X)+G(X)*X)
FOR ALL X CLEAR DF(F(X),X); CLEAR A;

% DEFINING AN ODD FUNCTION ;
LET F(0)=0;
FOR ALL X SUCH THAT X NEQ 0 AND ORDP(X,-X) LET F(X)=-F(-X);
F(X);
F(X)
F(-X);
-F(X)
F(X-Y);
F(X-Y)
F(Y-X);
-F(X-Y)
F(X-Y)+F(Y-X);
0
CLEAR F(0);
```



```
FOR ALL X SUCH THAT X NEQ 0 AND ORDP(X,-X) CLEAR F(X);
% EVEN FUNCTIONS ARE DEFINED ANALOGOUSLY ;

% DEFINING (ANTI-) SYMMETRIC FUNCTIONS OF TWO ARGUMENTS ;
FOR ALL X LET F(X,X)=0;
FOR ALL X,Y SUCH THAT X NEQ Y AND ORDP(X,Y) LET F(X,Y)=-F(Y,X);
F(X,Y);
-F(Y,X)
F(Y,X);
F(Y,X)
F(X+Y,X+Y);
0
F(X+Y,X-Y)+2*F(X-Y,X+Y);
-F(X+Y,X-Y)
FOR ALL X CLEAR F(X,X);
FOR ALL X,Y SUCH THAT X NEQ Y AND ORDP(X,Y) CLEAR F(X,Y);
```

Определение свойств функций при помощи правил подстановки. REDUCE не знает многих обычных тождеств для элементарных функций, потому что заранее неизвестно, в какую сторону их применять — выражать ли, например, произведения тригонометрических функций через функции от сумм и разностей или наоборот. Когда нужно, вы можете сами сообщить это REDUCEу, задав соответствующие правила подстановки.

Первый пример показывает, как заставить REDUCE выражать логарифмы произведений и дробей через суммы и разности логарифмов. REDUCE сначала упрощает аргумент, а потом применяет подстановки. Аргумент приводится к общему знаменателю, и в общем случае имеет вид дроби. Наши подстановки умеют справляться с дробями. С одночленами тоже все в порядке, но в общем случае в числителе и знаменателе стоят суммы. Даже если они содержат общий множитель, REDUCE рассматривает их не как X*(Y+Z), а как X*Y+X*Z. Поэтому введем еще 2 подстановки: одну для случая X+Y, где Y делится на X (надо застраховаться от случая X=1, так как любой Y делится на 1, и произойдет зацикливание), и другую для X*Y+Z, где Z делится на X.

```
% DEFINING PROPERTIES OF FUNCTIONS ;
% USING SUBSTITUTION RULES;
% ---------- ;
% 1) PROPERTIES OF LOG ;
FOR ALL X,Y LET LOG(X/Y)=LOG(X)-LOG(Y),
        LOG(X*Y)=LOG(X)+LOG(Y),
```

```
      LOG(X**Y)=Y*LOG(X);
LOG(X*E**Y);
LOG(X)+Y
LOG(X/2+Y);
LOG(X+2*Y)-LOG(2)
LOG(X*(A+1));
LOG(A*X+X)
FOR ALL X,Y SUCH THAT X NEQ 1 AND DEN(Y/X)=1 LET
   LOG(X+Y)=LOG(X)+LOG(1+Y/X);
FOR ALL X,Y,Z SUCH THAT DEN(Z/X)=1 LET
   LOG(X*Y+Z)=LOG(X)+LOG(Y+Z/X);
LOG(X*(A+1));
LOG(A+1)+LOG(X)
LOG(X*(A+B+C)/Y);
LOG(A+B+C)+LOG(X)-LOG(Y)
FOR ALL X,Y CLEAR LOG(X/Y),LOG(X*Y),LOG(X**Y);
FOR ALL X,Y SUCH THAT DEN(Y/X)=1 CLEAR LOG(X+Y);
FOR ALL X,Y,Z SUCH THAT DEN(Z/X)=1 CLEAR LOG(X*Y+Z);
```

Второй пример заставляет REDUCE выражать произведения и степени синусов и косинусов через линейные по ним выражения. Это очень полезный набор подстановок, который всегда применяется при работе с отрезками рядов Фурье. Произведения и степени их являются снова отрезками рядов Фурье. Работа с ними нужна в задачах о колебаниях и волнах, о движении планет вокруг Солнца и т. д.

REDUCE не считает SIN(X)**2 частным случаем SIN(X)*SIN(Y) при X=Y, поэтому нужны отдельные подстановки для квадратов. Однако для кубов и т. д. они уже не нужны — мы уже говорили, что REDUCE выделяет левую часть подстановки в наибольшей степени, так что он будет рассматривать SIN(X)**7 как (SIN(X)**2)**3*SIN(X). Точно так же не нужны подстановки для тройных и т. д. произведений: если в члене есть более двух синусов или косинусов, то REDUCE применит к двум из них подстановку, при упрощении результата снова натолкнется на произведения тригонометрических функций и применит ее опять, и так до тех пор, пока таких произведений не останется.

```
% 2) CONVERTING SIN, COS PRODUCTS TO SUMS ;
% -TRUNCATED FOURIER SERIES ;
FOR ALL X LET COS(X)**2=(1+COS(2*X))/2,
   SIN(X)**2=(1-COS(2*X))/2;
FOR ALL X,Y LET
```



```
   COS(X)*COS(Y)=(COS(X-Y)+COS(X+Y))/2,
   SIN(X)*SIN(Y)=(COS(X-Y)-COS(X+Y))/2,
   SIN(X)*COS(Y)=(SIN(X-Y)+SIN(X+Y))/2;
FACTOR COS,SIN;
A:=A1*COS(X)+A2*COS(2*X)+B1*SIN(X)+B2*SIN(2*X);
A := COS(2*X)*A2+COS(X)*A1+SIN(2*X)*B2+SIN(X)*B1
A**2;
(COS(4*X)*(-B2^2+A2^2)+2*COS(3*X)*(-B2*B1+A2*A1)
+COS(2*X)*(-B1^2+A1^2)+2*COS(X)*(B2*B1+A2*A1)
+2*SIN(4*X)*B2*A2+2*SIN(3*X)*(B2*A1+B1*A2)
+2*SIN(2*X)*B1*A1+2*SIN(X)*(B2*A1-B1*A2)+B2^2+B1^2A2^2+A1^2)/2
CLEAR A;
FOR ALL X CLEAR COS(X)**2,SIN(X)**2;
FOR ALL X,Y CLEAR COS(X)*COS(Y),SIN(X)*SIN(Y),SIN(X)*COS(Y);
REMFAC COS,SIN;
```

Третий пример делает в некотором смысле обратную работу — выражает синусы и косинусы сумм и кратных аргументов через произведения и степени синусов и косинусов. При этом из-за тождества SIN(X)**2+COS(X)**2=1 возникает неоднозначность в форме записи выражений. Чтобы ее избежать, договоримся, например, выражать COS(X)**2 через SIN(X)**2.

REDUCE рассматривает X-Y как X+(-Y), а сумму многих членов — как первый член плюс остальные, поэтому для таких случаев специальные подстановки не нужны. Вспомним, что он знает о нечетности синуса и четности косинуса: если бы не этот факт, нам бы следовало задать подстановки для SIN(-X) и COS(-X). Как справиться с привычкой REDUCEа приводить все к общему знаменателю и не рассматривать дробь как сумму, мы уже знаем.

Для кратных углов, мы заставляем REDUCE сводить N*X к (N-1)*X и X. Он будет многократно применять эту подстановку, пока не доберется до N=1. Проблема знаменателя решается так же. Если мы хотим рассматривать N как кратный угол от 1, то этот случай нужно предусмотреть отдельно, потому что REDUCE не считает N частным случаем N*X при X=1.

```
% 3) EXPANSION OF SIN, COS OF SUMS AND MULTIPLE ANGLES ;
FOR ALL X LET COS(X)**2=1-SIN(X)**2;
FOR ALL X,Y LET SIN(X+Y)=SIN(X)*COS(Y)+COS(X)*SIN(Y),
   COS(X+Y)=COS(X)*COS(Y)-SIN(X)*SIN(Y);
```

```
SIN(U+V);
COS(U)*SIN(V)+COS(V)*SIN(U)
SIN(U-V);
-COS(U)*SIN(V)+COS(V)*SIN(U)
SIN(X+Y+Z);
COS(X)*COS(Y)*SIN(Z)+COS(X)*COS(Z)*SIN(Y)
+COS(Y)*COS(Z)*SIN(X)-SIN(X)*SIN(Y)*SIN(Z)
SIN(X/2+Y);
SIN((X+2*Y)/2)

FOR ALL X,Y,Z LET
   SIN((X+Y)/Z)=SIN(X/Z)*COS(Y/Z)+COS(X/Z)*SIN(Y/Z),
   COS((X+Y)/Z)=COS(X/Z)*COS(Y/Z)-SIN(X/Z)*SIN(Y/Z);
SIN(X/2+Y);
COS(X/2)*SIN(Y)+COS(Y)*SIN(X/2)
SIN(PI/2+X);
COS(X)

FOR ALL N,X SUCH THAT FIXP(N) AND N>1 LET
   SIN(N*X)=SIN(X)*COS((N-1)*X)+COS(X)*SIN((N-1)*X),
   COS(N*X)=COS(X)*COS((N-1)*X)-SIN(X)*SIN((N-1)*X);
SIN(2*X);
2*COS(X)*SIN(X)
SIN(4*X);

4*COS(X)*SIN(X)*(-2*SIN(X)^2+1)
SIN(4*X/U);
SIN((4*X)/U)

FOR ALL N,X,Z SUCH THAT FIXP(N) AND N>1 LET
   SIN(N*X/Z)=SIN(X/Z)*COS((N-1)*X/Z)+COS(X/Z)*SIN((N-1)*X/Z),
   COS(N*X/Z)=COS(X/Z)*COS((N-1)*X/Z)-SIN(X/Z)*SIN((N-1)*X/Z);
SIN(4*X/U);

4*COS(X/U)*SIN(X/U)*(-2*SIN(X/U)^2+1)
SIN(4*X/3);
COS(X/3)*SIN(X)+COS(X)*SIN(X/3)
SIN(4);
SIN(4)

FOR ALL N SUCH THAT FIXP(N) AND N>1 LET
   SIN(N)=SIN(1)*COS(N-1)+COS(1)*SIN(N-1),
   COS(N)=COS(1)*COS(N-1)-SIN(1)*SIN(N-1);
```



```
SIN(4);
4*COS(1)*SIN(1)*(-2*SIN(1)^2+1)
FOR ALL X CLEAR COS(X)**2;
FOR ALL X,Y CLEAR SIN(X+Y),COS(X+Y);
FOR ALL X,Y,Z CLEAR SIN((X+Y)/Z),COS((X+Y)/Z);
FOR ALL N,X SUCH THAT FIXP(N) AND N>1 CLEAR SIN(N*X),COS(N*X);
FOR ALL N,X,Z SUCH THAT FIXP(N) AND N>1
   CLEAR SIN(N*X/Z),COS(N*X/Z);
FOR ALL N SUCH THAT FIXP(N) AND N>1 CLEAR SIN(N),COS(N);
```

Вы сами легко можете написать и проверить наборы подстановок для выражения комплексных экспонент через синусы и косинусы, для выражения тригонометрических функций через комплексные экспоненты и т. д. Не забудьте, что E**I и E**(I*X/Y) не являются частными случаями E**(I*X)!

Операторы как массивы свободных переменных. Часто бывают необходимы массивы свободных переменных. Например, пусть мы хотим использовать координаты x_i, где $i=1,2,3$. При этом должна быть возможность использовать x_i как простые переменные: дифференцировать по ним, присваивать им значения, уничтожать эти значения командой CLEAR и т. д. Массивы для этого не годятся, так как их элементы не бывают свободными. Поэтому необходимо использовать описание OPERATOR.

```
% OPERATORS AS ARRAYS OF FREE VARIABLES ;
% ---------------------- ;
OPERATOR X; N:=3$
R:=SQRT(FOR I:=1:N SUM X(I)**2);

R := SQRT(X(3)^2+X(2)^2+X(1)^2)
FOR I:=1:N DO WRITE DF(R,X(I));

X(1)/SQRT(X(3)^2+X(2)^2+X(1)^2)

X(2)/SQRT(X(3)^2+X(2)^2+X(1)^2)

X(3)/SQRT(X(3)^2+X(2)^2+X(1)^2)

% ASSIGNING VALUES TO THESE VARIABLES ;
X(1):=B$ X(2):=0$ X(3):=V*T$ R;

SQRT(B^2+T^2*V^2)
CLEAR X(1),X(2),X(3); R;

SQRT(X(3)^2+X(2)^2+X(1)^2)
CLEAR X; CLEAR N,R;
```

Матрицы. REDUCE умеет работать с матрицами. Их надо описывать командой MATRIX A(N,M);. Индексы A могут пробегать значения от 1 до N и от 1 до M (не от 0, как у массивов!). Начальные значения элементов матрицы равны 0. Из матриц и скаляров можно строить выражения, подчиняющиеся обычным правилам. Не обязательно указывать размерность матрицы в описании; такой безразмерной матрице можно присвоить матричное выражение, тогда ее размеры и определятся. Присваивание X := A**(-1)*B — удобный способ решения системы линейных уравнений A*X=B. Функция MAT строит матрицу; каждая ее строчка заключается в дополнительные скобки.

```
% MATRICES ;
% ------ ;
MATRIX A(2,2),B(2,1),C(2,2),X,Y; OPERATOR AA,BB,CC;
FOR I:=1:2 DO FOR J:=1:2 DO A(I,J):=AA(I,J);
FOR I:=1:2 DO B(I,1):=BB(I);
FOR I:=1:2 DO FOR J:=1:2 DO C(I,J):=CC(I,J);
DET(A);
AA(2,2)*AA(1,1)-AA(2,1)*AA(1,2)
TRACE(A);
AA(2,2)+AA(1,1)
TP(A);
MAT(1,1) := AA(1,1)
MAT(1,2) := AA(2,1)
MAT(2,1) := AA(1,2)
MAT(2,2) := AA(2,2)
A*C;
MAT(1,1) := CC(2,1)*AA(1,2)+CC(1,1)*AA(1,1)
MAT(1,2) := CC(2,2)*AA(1,2)+CC(1,2)*AA(1,1)
MAT(2,1) := CC(2,1)*AA(2,2)+CC(1,1)*AA(2,1)
MAT(2,2) := CC(2,2)*AA(2,2)+CC(1,2)*AA(2,1)
1/A;
MAT(1,1) := AA(2,2)/(AA(2,2)*AA(1,1)-AA(2,1)*AA(1,2))
MAT(1,2) := (-AA(1,2))/(AA(2,2)*AA(1,1)-AA(2,1)*AA(1,2))
MAT(2,1) := (-AA(2,1))/(AA(2,2)*AA(1,1)-AA(2,1)*AA(1,2))
MAT(2,2) := AA(1,1)/(AA(2,2)*AA(1,1)-AA(2,1)*AA(1,2))
X:=A**(-1)*B;
X(1,1) := (-BB(2)*AA(1,2)+BB(1)*AA(2,2))/(AA(2,2)*AA(1,1)
    -AA(2,1)*AA(1,2))
X(2,1) := (BB(2)*AA(1,1)-BB(1)*AA(2,1))/(AA(2,2)*AA(1,1)
    -AA(2,1)*AA(1,2))
```



```
Y:=MAT((U,-Z),(Z, U));
Y(1,1) := Z
Y(1,2) := U
Y(2,1) := U
Y(2,2) := -Z
DET(Y);
-(U^2+Z^2)
CLEAR A,B,C,X,Y; CLEAR AA,BB,CC;
```

Векторы и тензоры. Векторы описываются командой VECTOR. Команда VECDIM устанавливает размерность пространства. Скалярное произведение обозначается точкой. Из векторов и скаляров можно строить векторные и скалярные выражения, подчиняющиеся обычным правилам. Обычно удобно задать квадраты и скалярные произведения векторов задачи командой LET.

REDUCE рассматривает векторы и индексы единым образом. Именно, вместо индексов X, Y, Z используются единичные векторы EX, EY, EZ вдоль координатных осей. Тогда, например, X-компонента вектора V будет V.EX. Поэтому присутствующие в задаче индексы следует описывать как векторы. Из векторов и индексов можно строить тензоры. Скалярное произведение двух индексов (с точки зрения REDUCEа они векторы) M.N имеет смысл единичного (метрического) тензора δ_{mn}. Действительно, если M и N направлены вдоль одной оси, их скалярное произведение равно единице, а если вдоль разных — нулю.

При перемножении тензоров должна производиться свертка по повторяющимся индексам. Если M — такой индекс, то нужно суммировать по M=EX, EY, EZ. При этом U.M*V.M=U.V, а M.M=размерности пространства (так как это сумма единиц в количестве, равном числу осей координат). Индексы, по которым нужно суммировать, описываются командой INDEX. Они должны встречаться в каждом члене выражения ровно 2 раза, иначе REDUCE выдаст сообщение «неспаренный индекс». Поэтому, перемножив тензоры X и Y со сверткой по индексам M и N, мы не сможем даже просто вывести X — в нем эти индексы неспарены! Вернуть индексы в состояние векторов можно командой REMIND.

Как продифференцировать тензор S по вектору U с индексом M? Трактовка индекса как одного из единичных координатных векторов подсказывает решение. Именно, надо менять M-ю компоненту U, т. е. прибавить к U бесконечно малый вектор X*M, и продифференцировать по X в точке X=0.

Как усреднить тензор S по направлениям вектора U? Если в каждый член S этот вектор входит 2 раза (или не входит), то достаточно заменить U.M*U.N на U.U/3*M.N для всех M и N (здесь 3—размерность пространства). Средние от произведений 4, 6 и т. д. U выражаются более сложными формулами. В этом примере мы столкнулись с ошибкой в имеющейся версии REDUCEa: он не всегда учитывает коммутативность скалярного произведения. Чтобы добиться упрощения такого выражения, достаточно передать его какой-нибудь процедуре, даже такой, которая ничего не делает.

```
% VECTORS AND TENSORS ;
% ---------------;
VECTOR U,V,W,M,N; VECDIM 3;
W:=U+V;
W := U+V
W.W;
U.U+2*U.V+V.V
CLEAR W;

% CONTRACTION OVER REPEATED INDICES ;
LET U.U=1,V.V=1,U.V=COS(T);
X:=U.M*V.N+M.N;
X := M.N+M.U*N.V
Y:=M.N;
Y := M.N
X*Y;
M.N*(M.N+M.U*N.V)
INDEX M,N; X*Y;
COS(T)+3
X;
***** UNMATCHED INDEX M N
REMIND M,N; X;
M.N+M.U*N.V
CLEAR X,Y; CLEAR U.U,V.V,U.V;

% DIFFERENTIATION OVER VECTOR U WITH INDEX ;
% M OR N ;
S:=1/U.U;
S := 1/U.U
S:=SUB(X=0,DF(SUB(U=U+X*M,S),X));
S := (-2*M.U)/U.U^2
```



```
S:=SUB(X=0,DF(SUB(U=U+X*N,S),X));
S := (2*(-M.N*U.U+4*M.U*N.U))/U.U^3

% AVERAGING OVER U DIRECTIONS ;
LET U.U=U2; S:=S;
S := (2*(-M.N*U2+4*M.U*N.U))/U2^3
CLEAR U.U;
FOR ALL M,N LET U.M*U.N=U2/3*M.N;
*** =M DECLARED VECTOR
*** =N DECLARED VECTOR
S:=S;
S := (2*M.N*(4*M.U-3*U.M))/(3*U.M*U2^2)
% THIS IS A BUG IN REDUCE. ;
% TO OBTAIN A CORRECT ANSWER, WE DO: ;
PROCEDURE DUMMY(X);X;
DUMMY
S:=DUMMY(S);
S := (2*M.N)/(3*U2^2)
FOR ALL M,N CLEAR U.M*U.N; CLEAR S; CLEAR U,V,M,N;
```

Процедуры. Процедура состоит из заголовка, за которым следует выражение. Параметры передаются по значению, т. е. в процедуре вводятся локальные переменные, и в них копируются фактические параметры. Исключением являются левые части LET- и SUB-подстановок—там используются не эти локальные переменные (что было бы бессмысленно), а значения параметров. Процедура вычисляет выражение и возвращает его значение.

Телом процедуры может быть любое выражение. Так, в примере—процедуре вычисления факториала это цикл PRODUCT. Кстати, мы видим, что REDUCE умеет работать со сколь угодно большими целыми числами.

Процедура может вызывать сама себя. Это называется рекурсией. В простейшем случае рекурсивная процедура непосредственно выдает результат, а в более сложных—рекурсивно вызывает себя для более простого случая, так что рекурсия когда-то завершится.

```
% PROCEDURES AND RECURSION ;
% ----------------;
PROCEDURE FAC(N);
```

```
FOR I:=1:N PRODUCT I;
FAC

FAC(100);
933262154439441526816992388562667004907159682643816214685929
638952175999932299156089414639761565182862536979208272237582
511852109168640000000000000000000000000

% RECURSIVE VERSION OF FACTORIAL ;
PROCEDURE FACR(N);
IF N=0 THEN 1 ELSE N*FACR(N-1);
FACR

IF FACR(10)=FAC(10) THEN WRITE "OK" ELSE WRITE "ERROR";
OK
CLEAR FACR;
```

Обычно процедуры выполняют достаточно сложные действия. Поэтому должны существовать способы сгруппировать много операторов в единое выражение, являющееся телом процедуры. Мы уже встречались с условным выражением (или оператором) IF ... THEN ... ELSE ... Можно образовать групповой оператор, заключив последовательность операторов в скобки ⟨⟨...⟩⟩. Это часто нужно в условных операторах, циклах и процедурах — везде, где формально требуется один оператор, а нужно подставить несколько. Значение группового оператора (если оно требуется, например, в случае X := ⟨⟨...⟩⟩ или в процедуре) — это значение последнего стоящего в нем выражения.

Более богатые возможности предоставляет группировка операторов в блок BEGIN ... END. В нем можно описать локальные переменные командой SCALAR. Они не имеют ничего общего с переменными с теми же именами, которые могут существовать вне блока, и исчезают при выходе из него. Их начальное значение равно нулю. Если при исполнении блока встретится оператор RETURN, то оно прекращается, а значение стоящего в этом операторе выражения становится значением блока. Если в операторе RETURN не стоит выражение, или исполнение дойдет до конца, не наткнувшись на RETURN, то возвращается 0. Блоки часто используются в процедурах.

В примерах процедура EXP1(X,N) дает N членов разложения EXP(X). Более удобной является процедура EXP2, обрезающая ряд по WTLEVEL. Она позволяет находить разложения экспонент от выражений, члены которых имеют разный порядок малости. Аналогично, процедура BINOM строит разложение (1+X)**N. Эти процедуры могли бы быть совсем однотипными; в них проиллюстрированы 2 вида циклов: WHILE ... DO ... и REPEAT ... UNTIL ... Первый из них проверяет условие в начале и, если оно истинно, делает очередной оборот. Второй проверяет условие в конце и, если оно истинно, кончает работу. Кстати, как видно из примера, присваивание имеет значение, и может использоваться в выражении.


```
% TAYLOR SERIES FOR EXP ;
PROCEDURE EXP1(X,N);
BEGIN SCALAR S,T;S:=1;T:=1;
   FOR I:=1:N DO ⟨⟨ T:=T*X/I;S:=S+T ⟩⟩;
   RETURN S
END;
EXP1

FACTOR X; ON DIV,RAT; EXP1(X,10);
1/3628800*X^10+1/362880*X^9+1/40320*X^8+1/5040*X^7
+1/720*X^6+1/120*X^5+1/24*X^4+1/6*X^3+1/2*X^2+X+1
CLEAR EXP1;

% A METHOD USING SMALL VARIABLES ;
PROCEDURE EXP2(X);
BEGIN SCALAR S,T,I;S:=1;T:=X;I:=1;
   REPEAT ⟨⟨ S:=S+T;I:=I+1;T:=T*X/I ⟩⟩ UNTIL T=0;
   RETURN S
END;
EXP2

WEIGHT X=1; WTLEVEL 10; EXP2(X);
1/3628800*X^10+1/362880*X^9+1/40320*X^8+1/5040*X^7
+1/720*X^6+1/120*X^5+1/24*X^4+1/6*X^3+1/2*X^2+X+1

EXP2(X+X**2);
19093/518400*X^10+5357/72576*X^9+1979/13440*X^8+1303/5040*X^7
+331/720*X^6+27/40*X^5+25/24*X^4+7/6*X^3+3/2*X^2+X+1

% BINOMIAL EXPANSION ;
```

```
PROCEDURE BINOM(X,N);
BEGIN SCALAR S,T,I,J;S:=1;T:=1;I:=1;J:=N;
   WHILE (T:=T*X*J/I) NEQ 0 DO <<S:=S+T;I:=I+1;J:=J-1>>;
   RETURN S
END;
BINOM

BINOM(Y,4);
```

$$Y^4+4*Y^3+6*Y^2+4*Y+1$$

```
BINOM(X,1/2);
```

$$-2431/262144*X^{10}+715/65536*X^9-429/32768*X^8+33/2048*X^7$$
$$-21/1024*X^6+7/256*X^5-5/128*X^4+1/16*X^3-1/8*X^2+1/2*X+1$$

```
BINOM(X,-1);
```

$$X^{10}-X^9+X^8-X^7+X^6-X^5+X^4-X^3+X^2-X+1$$

```
BINOM(X+X**2,-1);
```

$$-X^{10}+X^9-X^7+X^6-X^4+X^3-X+1$$

```
WTLEVEL 4; BINOM(X,N);
```

$$X^4*N*(1/24*N^3-1/4*N^2+11/24*N-1/4)+X^3*N*(1/6*N^2$$
$$-1/2*N+1/3)+X^2*N*(1/2*N-1/2)+X*N+1$$

```
CLEAR X; REMFAC X; OFF DIV,RAT;
```

Полиномы Лежандра. В этом примере мы приводим 5 способов получения первых 10 полиномов Лежандра. Первый — это процедура, вычисляющая их по формуле Родригеса.

```
% LEGENDRE POLYNOMIALS ;
% ---------------- ;
N:=10$ ARRAY P1(N),P2(N);

% 1) RODRIGUES FORMULA ;
PROCEDURE P(N);
DF((X**2-1)**N,X,N)/(2**N*FAC(N));
P

P1(0):=1;
P1(0) := 1
FOR I:=1:N DO WRITE P1(I):=P(I);
```



```
P1(1) := X
```

$$P1(2) := (3*X^2-1)/2$$
$$P1(3) := (X*(5*X^2-3))/2$$
$$P1(4) := (35*X^4-30*X^2+3)/8$$
$$P1(5) := (X*(63*X^4-70*X^2+15))/8$$
$$P1(6) := (231*X^6-315*X^4+105*X^2-5)/16$$
$$P1(7) := (X*(429*X^6-693*X^4+315*X^2-35))/16$$
$$P1(8) := (6435*X^8-12012*X^6+6930*X^4-1260*X^2+35)/128$$
$$P1(9) := (X*(12155*X^8-25740*X^6+18018*X^4-4620*X^2+315))/128$$
$$P1(10) := (46189*X^{10}-109395*X+90090*X^6-30030*X^4+3465*X^2-63)/256$$

```
CLEAR P;
```

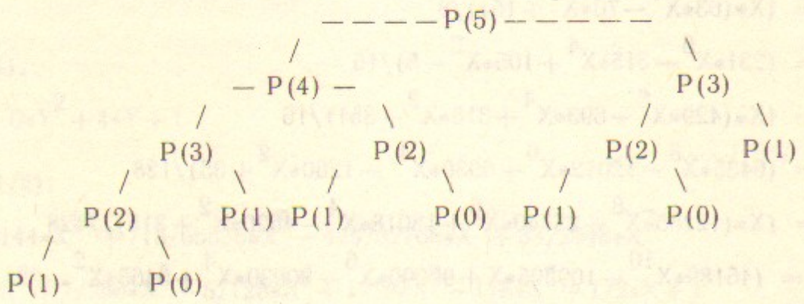
Второй — через дифференцирование производящей функции. Он работает очень медленно, так как высшие производные выражений с корнями вычислять сложно даже REDUCEу. Поэтому лучше него третий метод — вместо многократного дифференцирования получить сразу разложение в ряд по Y при помощи процедуры BINOM. Вообще, получать ряд Тейлора любой функции по определению через ее производные — это почти всегда самый плохой путь. Лучше комбинировать известные ряды — биномиальный, для экспоненты, логарифма и т. д.

```
% 2) GENERATING FUNCTION ;
A:=(1-2*X*Y+Y**2)**(-1/2)$
FOR I:=1:4 DO
<< A:=DF(A,Y)/I;P:=SUB(Y=0,A);
   IF P NEQ P1(I) THEN WRITE "ERROR",I >>;
CLEAR A,P;

% 3) BETTER WAY TO DO THE SAME ;
WEIGHT Y=1; WTLEVEL 10;
A:=BINOM(-2*X*Y+Y**2,-1/2)$
CLEAR Y; D:=DEN(A)$ COEFF(NUM(A),Y,P2)$
FOR I:=0:N DO
<< P2(I):=P2(I)/D;
   IF P2(I) NEQ P1(I) THEN WRITE "ERROR",I >>;
CLEAR A,D; CLEAR BINOM;
```

Четвертый способ — это введение оператора и описание рекуррентных соотношений для него при помощи LET-подстановки. Эти

подстановки рекурсивны — при вычислении P(5) получится выражение, содержащее P(4) и P(3), к нему снова будет применяться подстановка, и так до тех пор, пока не получатся P(1) и P(0), значения которых известны:



Этот способ очень неэффективен: при вычислении P(4) опять возникает P(3), и оно вычисляется совершенно независимо от того P(3), которое уже было.

Пятый способ — это просто применение рекуррентных соотношений в цикле, он очень эффективен.

```
% 4) RECURRENT RELATIONS ;
OPERATOR P; P(0):=1$ P(1):=X$
FOR ALL I SUCH THAT FIXP(I) AND I>1 LET
   P(I)=(2-1/I)*X*P(I-1)-(1-1/I)*P(I-2);
FOR I:=1:5 DO
<< PP:=P(I);
   IF PP NEQ P1(I) THEN WRITE "ERROR",I >>;
CLEAR P; CLEAR PP;

% 5) BETTER WAY TO DO THE SAME ;
P2(0):=1$ P2(1):=X$
FOR I:=2:N DO
<< P2(I):=(2-1/I)*X*P2(I-1)-(1-1/I)*P2(I-2);
   IF P2(I) NEQ P1(I) THEN WRITE "ERROR",I >>;
CLEAR P1,P2;
```

Функции Бесселя. В этом примере продемонстрирована работа с функциями Бесселя. Он начинается с процедуры, проверяющей, удовлетворяет ли функция уравнению Бесселя.

Далее определяются свойства функций Бесселя с целым индексом: функции с отрицательным индексом выражаются через функции с положительным, а они через J_0 и J_1 при помощи рекурсивной подстановки. Для этих функций заданы их свойства четности

и значения в нуле. Точно так же, как и в случае полиномов Лежандра, подстановка, выражающая J_N через J_{N-1} и J_{N-2}, приводит к очень неэффективному вычислению при больших N. Поэтому далее приведена другая подстановка, применяющая рекурентное соотношение в цикле. Затем задано правило дифференцирования функций Бесселя.

Как известно, функции Бесселя с полуцелым индексом выражаются через синусы и косинусы. Выражения для положительных и отрицательных индексов различны, и содержат многократное применение оператора $1/x\ d/dx$. В примере приведена соответствующая подстановка с весьма сложным выражением в правой части.

```
% BESSEL FUNCTIONS ;
% ---------- ;
OPERATOR J;
% 1) A PROCEDURE TO CHECK IF A FUNCTION OBEYS ;
%    THE BESSEL EQUATION ;
PROCEDURE BESEQ(J,N,X);
X**2*DF(J,X,2)+X*DF(J,X)+(X**2-N**2)*J;
BESEQ

% 2) BESSEL FUNCTIONS OF INTEGER INDICES ;
FOR ALL N,X SUCH THAT FIXP(N) AND N<0 LET
   J(N,X)=(-1)**(-N)*J(-N,X);
FOR ALL N,X SUCH THAT FIXP(N) AND N>1 LET
   J(N,X)=2*(N-1)/X*J(N-1,X)-J(N-2,X);
LET J(0,0)=1,J(1,0)=0;
FOR ALL X SUCH THAT X NEQ 0 AND ORDP(X,-X) LET
   J(0,X)=J(0,-X),J(1,X)=-J(1,-X);
J(2,X);
(2*J(1,X)-J(0,X)*X)/X
J(3,X);
(-J(1,X)*X^2+8*J(1,X)-4*J(0,X)*X)/X^2
J(-3,X);
(J(1,X)*X^2-8*J(1,X)+4*J(0,X)*X)/X^2
J(-3,-X);
(-J(1,X)*X^2+8*J(1,X)-4*J(0,X)*X)/X^2

% 3) BETTER WAY TO DO THE SAME ;
FOR ALL N,X SUCH THAT FIXP(N) AND N>1 CLEAR J(N,X);
FOR ALL N,X SUCH THAT FIXP(N) AND N>1 LET J(N,X)=
```

```
   BEGIN SCALAR J0,J1,J2;J0:=J(0,X);J1:=J(1,X);
      FOR I:=2:N DO
      ⟨⟨J2:=2*(I-1)/X*J1-J0;J0:=J1;J1:=J2⟩⟩;
      RETURN J2
   END;
J(10,X);
```

$$(50*J(1,X)*X^8-19200*J(1,X)*X^6+1693440*J(1,X)*X^4-41287680*J(1,X)*X^2$$
$$+185794560*J(1,X)-J(0,X)*X^9+1200*J(0,X)*X^7-201600*J(0,X)*X^5$$
$$+9031680*J(0,X)*X^3-92897280*J(0,X)*X)/X^9$$

```
% 4) DIFFERENTIATION RULE ;
FOR ALL N,X LET DF(J(N,X),X)=(J(N-1,X)-J(N+1,X))/2;
DF(J(1,X),X);
```

$$(-J(1,X)+J(0,X)*X)/X$$

```
DF(J(2,X),X);
```

$$(J(1,X)*X^2-4*J(1,X)+2*J(0,X)*X)/X^2$$

```
DF(J(3,X),X);
```

$$(5*J(1,X)*X^2-24*J(1,X)-J(0,X)*X^3+12*J(0,X)*X)/X^3$$

```
BESEQ(J(5,X),5,X);
0

% 5) BESSEL FUNCTIONS OF HALF-INTEGER INDICES ;
FOR ALL N,X SUCH THAT NOT FIXP(N) AND FIXP(2*N)
   LET J(N,X)=SQRT(2/PI)*
   IF 2*N>0
   THEN X**N*
      BEGIN SCALAR A;A:=SIN(X)/X;
         FOR I:=1:N-1/2 DO A:=-DF(A,X)/X;
         RETURN A
      END
   ELSE X**(-N)*
      BEGIN SCALAR A;A:=COS(X)/X;
         FOR I:=1:-N-1/2 DO A:=DF(A,X)/X;
         RETURN A
      END;
J(3/2,X);
```

$$(SQRT(2)*SQRT(X)*(-COS(X)*X+SIN(X)))/(SQRT(PI)*X^2)$$

```
J(5/2,X);
```


$$(SQRT(2)*SQRT(X)*(-3*COS(X)*X-SIN(X)*X^2+3*SIN(X)))/(SQRT(PI)*X^3)$$

```
J(-3/2,X);
```

$$(-(SQRT(2)*SQRT(X))*(COS(X)+SIN(X)*X))/(SQRT(PI)*X^2)$$

```
BESEQ(J(5/2,X),5/2,X);
0
BESEQ(J(-5/2,X),5/2,X);
0
FOR ALL N,X SUCH THAT FIXP(N) AND N<0 CLEAR J(N,X);
FOR ALL N,X SUCH THAT FIXP(N) AND N>1 CLEAR J(N,X);
CLEAR J(0,0),J(1,0);
FOR ALL X SUCH THAT X NEQ 0 AND ORDP(X,-X)CLEAR J(0,X),J(1,X);
FOR ALL N,X CLEAR DF(J(N,X),X);
FOR ALL N,X SUCH THAT NOT FIXP(N) AND FIXP(2*N)CLEAR J(N,X);
CLEAR BESEQ;
```

2. КЛАССИЧЕСКИЙ НЕЛИНЕЙНЫЙ ОСЦИЛЛЯТОР

Введение. В физике ни одна задача не решается абсолютно точно: поскольку все явления в природе взаимосвязаны, для этого пришлось бы учесть бесчисленное множество факторов. Поэтому необходимо выделить из них самые существенные, менее существенные и практически несущественные. Если повезет, то идеализированная задача, в которой учтены только наиболее существенные факторы, решается точно. Влияние менее важных факторов в таком случае учитывается по теории возмущений, т. е. в виде рядов по степеням безразмерных малых параметров, характеризующих влияние этих факторов на рассматриваемое явление.

Не всегда поправки теории возмущений приводят просто к непринципиальным количественным изменениям в решении невозмущенной задачи: они могут приводить к качественно новым явлениям. Например, в задаче о нелинейном осцилляторе во втором порядке теории возмущений появляется зависимость частоты колебаний от амплитуды. В квантовомеханических задачах может возникнуть расщепление уровней, которые были вырожденными в отсутствие возмущения. Наконец, в физике элементарных частиц начинают со случая невзаимодействующих частиц, и рассматривают взаимодействие как возмущение. Тогда физически интересные процессы рассеяния и распада частиц появляются в различных порядках теории возмущений.

Обычно сложность вычислений по теории возмущений стремительно возрастает с ростом порядка, однако правила вычисления членов ряда относительно просты и легко алгоритмизуются. Поэтому теория возмущений является идеальным местом для применения машинной аналитики. Она позволяет продвинуться на несколько порядков дальше, чем это возможно при ручном счете. Мы рассмотрим большое число примеров теории возмущений в разных областях физики, и начнем с простейшей задачи — классического одномерного нелинейного осциллятора.

Эта задача рассматривается в параграфе 28 учебника [6], где приведено решение с точностью до членов $\sim a^3$ (где a — амплитуда колебаний). При помощи программы на REDUCEe мы без труда достигнем точности a^7. Мы будем использовать тот же простейший метод решения, что и в [6]. Он изложен также в п.2.1а книги [7], по которой можно изучить более современные алгоритмы теории возмущений в классической механике.

Теория. Рассмотрим одномерное движение частицы массы m вблизи минимума произвольного гладкого потенциала $U(x)$. Приближенно можно заменить $U(x)$ вблизи минимума (считаем, что он находится в точке $x=0$) на параболу $U_0(x)=kx^2/2$. Тогда уравнение движения $m\ddot{x}=-dU/dx$ принимает вид $m\ddot{x}+kx=0$. Его решение очевидно: $x(t)=a\cos\omega_0 t+b\sin\omega_0 t$, $\omega_0=\sqrt{k/m}$. Выбирая начало отсчета времени в максимуме $x(t)$, мы всегда можем записать его в виде $a\cos\omega_0 t$.

Мы будем интересоваться влиянием остальных членов разложения потенциала: $U(x)=U_0(x)+V(x)$, $V(x)=\sum_i c_i x^{i+2}$. Размерная оценка дает $c_i\sim k/L^i$, где L — характерная длина, на которой существенно меняется поведение потенциала $U(x)$ (рис. 1). При колебаниях с малой амплитудой $a\ll L$, $x\sim a$, и вклад члена $c_i x^{i+2}$ содержит малый множитель $\sim(a/L)^i$ по сравнению с характерной энергией колебаний $\sim ka^2$. Поэтому решение задачи можно строить в виде ряда по степеням безразмерного малого параметра a/L.

В задаче имеются три независимых размерности (время, длина, масса), поэтому мы могли бы выбрать три параметра задачи с независимыми размерностями за единицу и тем самым зафиксировать единицы измерения. Мы выбираем $m=1$ и $k=1$, что приводит к $\omega_0=1$. Единицу длины оставим незафиксированной. Если считать, что она $\sim L$, то все коэффициенты $c_i\sim 1$, а амплитуда a становится безразмерным малым параметром, по которому производится разложение.

Итак, мы ищем решение уравнения движения

$$\ddot{x}+x=R(x)\equiv -dV/dx \qquad (1)$$

в виде ряда по степеням a: $x(t)=\sum_{i=1}^{\infty} a^i x_i(t)$. Уравнение должно выполняться в каждом порядке по a. Правая часть R содержит степени x, начиная со второй. Член с a^2 в R содержит $\cos^2\omega_0 t$, т. е. нулевую и вторую гармоники. Поэтому $x_2(t)$ содержит те же гармоники, что и вынуждающая сила, т. е. нулевую и вторую. Член с a^3 в R получается из $x_1^3(t)$ или $x_2(t)x_1(t)$, и содержит первую и третью гармоники. Вообще, легко проверить, что разные члены в R содержат следующие гармоники:

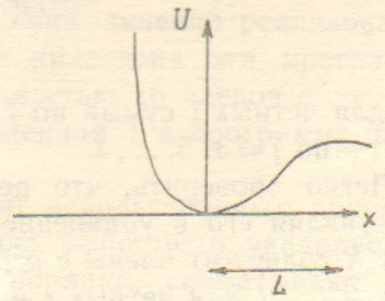


Рис. 1.

a^2	0		2				
a^3		1		3			
a^4	0		2		4		
a^5		1		3		5	
a^6	0		2		4		6
a^7		1		3		5	7

Все члены с нечетными степенями a содержат резонансный член — первую гармонику. Это привело бы к неограниченно растущим решениям для $x_i(t)$, что не имеет смысла. Значит, мы чего-то не учли. А именно, мы знаем, что одномерное движение в потенциальной яме всегда периодично, однако его частота зависит от амплитуды (если только потенциал не параболический). Поэтому решение надо искать в виде рядов Фурье по $\cos\omega t$ и $\sin\omega t$, где $\omega=\omega(a)\neq 1$. Неограниченно растущие члены в решении получались из-за разложения ω в ряд по a под знаком cos или sin и последующего разложения этих функций. Разумеется, лучше найти $\omega(a)$ и $x(t)$ в виде ряда Фурье с этой правильной частотой.

Мы условились выбрать начало отсчета времени в максимуме $x(t)$. Поскольку движение обратимо во времени, $x(t)$ должно быть четной функцией, и ряд Фурье содержит только косинусы. Далее, мы можем считать по определению, что первая гармоника целиком относится к невозмущенному движению $a\cos\omega t$, т. е. мы определяем амплитуду a как коэффициент при $\cos\omega t$ в $x(t)$. Таким образом, мы ищем решение в виде $x(t)=a\cos\omega t+x'(t)$,

$$x'(t) = \sum_{i=2}^{\infty} a^i \sum_j b_{ij} \cos j\omega t, \qquad (2)$$

где для четных i сумма по j берется по $j=0, 2, \ldots, i$, а для нечетных i — по $j=3, 5, \ldots, i$.

Легко проверить, что решение действительно имеет вид (2). Подставляя его в уравнение (1), мы получим, что вынуждающая сила R содержит члены с $a^2, a^3, \ldots$, причем при них стоят гармоники $0, 2, \ldots, i$ для четных i и $3, 5, \ldots, i$ для нечетных. Действительно, при перемножении членов $a^{i_1} \cos j_1\omega t$ и $a^{i_2} \cos j_2\omega t$ получаем члены $a^{i_1+i_2} \cos(j_1 \pm j_2)\omega t$. При $j_1 \leqslant i_1$, $j_2 \leqslant i_2$ получим $j \leqslant i$. При условии что четность j_1 и i_1, а также j_2 и i_2, одинакова, получим, что четность j и i тоже одинакова. Поэтому утверждение про R доказано по индукции. Далее, каждый член в вынуждающей силе R приводит к члену с той же степенью a и с теми же гармониками в колебании $x(t)$. Кроме вынужденных колебаний, существуют свободные — первая гармоника, коэффициент при ней мы по определению назвали амплитудой a. Значит, мы пришли к форме решения (2).

Алгоритм. Теперь путь действий становится понятным: надо подставить решение (2) в уравнение (1), и найти неопределенные коэффициенты b_{ij} и частоту ω.

Первая гармоника требует особого рассмотрения. В левой части уравнения (1) она содержится только в невозмущенном члене $a\cos\omega t$. Пусть $\omega^2 = 1 + w$, где w — малая поправка (она начинается с a^2). Тогда член с первой гармоникой в левой части имеет вид $-wa\cos\omega t$. Выделяя такой член и в правой части $R = R_1 \cos\omega t + R'$, мы найдем поправку к частоте $w = -R_1/a$. Она является рядом по a, коэффициенты которого выражаются через b_{ij}.

После этого уравнение (1) можно переписать в виде

$$\frac{d^2}{d(\omega t)^2} x' + x' = R'' \equiv R' - w \frac{d^2}{d(\omega t)^2} x'. \qquad (3)$$

В нем уже не содержится первой гармоники. Его можно последовательно решать, приравняв сначала члены при a^2, затем при a^3 и т. д. Когда мы решаем его в порядке a^i, слева стоит $a^i \sum_j (1 - j^2) b_{ij} \cos j\omega t$, а справа в R'' некое выражение, содержащее уже вычисленные ранее коэффициенты $b_{i'j'}$ с $i' < i$. Приравнивая отдельные гармоники, найдем все b_{ij}.

Программа. Таким образом, мы получили алгоритм, который позволяет получить любое желаемое число членов разложения ω^2 и $x(t)$ по a. Этот алгоритм довольно прямолинейно реализован в виде программы на REDUCE. Здесь приведена эта программа вместе с результатами ее работы с точностью до членов a^7, т. е. в 6-м порядке теории возмущений. Переменная T в программе означает ωt.

Следует сказать несколько слов об использованных приемах. Описание A как величины 1-го порядка малости, и указание сохранять величины вплоть до $N+1$ порядка, срабатывают при вычислении правой части R и позже R''. Они позволяют отбросить огромное количество ненужных членов, которые появились бы при возведении x в различные степени. При вычислении R происходит также преобразование степеней и произведений косинусов в суммы. Соответствующие подстановки дальше не нужны, и поэтому они уничтожены командой CLEAR. Дальше такой командой уничтожен также порядок малости переменной A. В существующей версии REDUCE по переменной с порядком малости нельзя дифференцировать и разлагать в ряд процедурой COEFF.

Выделение коэффициента при COS(T) в R реализовано так. Сначала R умножается на свободную переменную Z при действии подстановки Z*COS(T)=1. В членах, не содержащих COS(T), при этом остается Z. Затем подстановка Z=0 уничтожает такие члены.

Для разложения выражений на гармоники, т. е. для выделения коэффициентов при COS(J*T), использован такой трюк. Есть процедура COEFF, дающая коэффициенты при степенях переменной. Поэтому сначала мы заменяем COS(J*T) на Z**J, а затем применяем COEFF. Однако COEFF не работает с выражениями, содержащими знаменатель, даже если это просто число. Для обхода этой трудности написана процедура COEF1, делающая то же, что COEFF, но переносящая знаменатель во все коэффициенты.

```
N:=6$
% CLASSICAL ONE-DIMENSIONAL NONLINEAR OSCILLATOR ;
% PARTICLE WITH MASS=1 IN THE POTENTIAL ;
% U(X)=X**2/2+V(X), ;
% WHERE V(X)=FOR I:=1:INFINITY SUM C(I)*X**(I+2) ;
% PERTURBATION THEORY TO THE ORDER N IN THE AMPLITUDE A ;
% THE VARIABLE T IN THE PROGRAM MEANS REALLY OMEGA*T ;
% WHERE OMEGA**2=1+W. EQUATION OF MOTION IS ;
%    (1+W)*DF(X,T,2)+X=R ;
```

```
% WHERE R=-DF(V(X),X). ;
SYMBOLIC OPERATOR TIME; TT:=TIME()$
OPERATOR B,C;
WEIGHT A=1; WTLEVEL N+1;

% GENERAL FORM OF SOLUTION ;
X:=A*COS(T)
   +FOR I:=2:N+1 SUM
      A**I*(FOR J:=I STEP-2 UNTIL 0 SUM
      IF J=1 THEN 0 ELSE B(I,J)*COS(J*T))$
FOR ALL X LET COS(X)**2=(1+COS(2*X))/2;
FOR ALL X,Y LET COS(X)*COS(Y)=(COS(X+Y)+COS(X-Y))/2;
% NONLINEAR TERM R ;
R:=-FOR I:=1:N SUM C(I)*(I+2)*X**(I+1)$
FOR ALL X CLEAR COS(X)**2;
FOR ALL X,Y CLEAR COS(X)*COS(Y);
% THESE SUBSTITUTIONS WILL NOT BE USED LATER ;

% TERMS WITH COS(T) IN THE EQUATION OF MOTION. ;
% THEY GIVE US W ;
LET Z*COS(T)=1; W:=Z*R$ CLEAR Z*COS(T);
W:=-SUB(Z=0,W)/A$

% EQUATION OF MOTION IS NOW DF(X,T,2)+X=R, WHERE R IS: ;
R:=R-W*DF(X,T,2)$

% WE CAN USE COEFF TO SEPATATE COEFFICIENTS AT COS(J*T), ;
% IF WE DO: ;
FOR ALL J LET COS(J*T)=Z**J; R:=R$
FOR ALL J CLEAR COS(J*T);

% THIS PROCEDURE DOES THE SAME AS COEFF, ;
% BUT ACCEPTS DENOMINATORS ;
PROCEDURE COEF1(P,X,A);
BEGIN SCALAR N,D,M;
   N:=NUM(P);D:=DEN(P);M:=COEFF(N,X,A);
   FOR I:=0:M DO A(I):=A(I)/D;CLEAR P;RETURN M
END;
COEF1

% SIZES OF ARRAYS WILL BE CHANGED DINAMICALLY BY COEFF ;
ARRAY RA(1),RZ(1);
```



```
% THE WEIGHT OF A WILL NOT BE USED LATER. ;
% IT IS DELETED BY CLEAR A ;
% IN THE CURRENT VERSION OF REDUCE IT IS IMPOSSIBLE TO USE ;
% VARIABLES WITH WEIGHT IN COEFF AND DF ;
CLEAR A;

COEF1(R,A,RA)$
FOR I:=2:N+1 DO
<< COEF1(RA(I),Z,RZ);
   FOR J:=I STEP-2 UNTIL 0 DO
   IF J NEQ 1 THEN B(I,J):=RZ(J)/(1-J**2);
>>;

% RESULTS ARE: ;
FACTOR A; ON DIV,RAT;
% OMEGA**2= ;
W:=1+W;
W := A^6*(35/8*C(6) - 315/8*C(5)*C(1) + 15/8*C(4)*C(2) + 1935/16*C(4)*C(1)^2
   - 315/16*C(3)^2 + 1235/8*C(3)*C(2)*C(1) - 4705/16*C(3)*C(1)^3
   - 57/64*C(2)^3 - 41529/128*C(2)^2*C(1)^2 + 171021/256*C(2)*C(1)^4
   - 145755/512*C(1)^6) + A^4я(15/4*C(4) - 105/4*C(3)*C(1) + 3/8*C(2)^2
   + 429/8*C(2)*C(1)^2 - 1005/32*C(1)^4) + A^2*(3*C(2) - 15/2*C(1)^2) + 1

% MOTION OF THE PARTICLE ;
X:=X;

X := A^7*(1/384*C(6)*COS(7*T) + 7/192*C(6)*COS(5*T)21/64*C(6)*COS(3*T)
   + 9/640*C(5)*C(1)*COS(7*T) - 1/64*C(5)*C(1)*COS(5*T)
   - 357/320*C(5)*C(1)*COS(3*T) + 3/512*C(4)*C(2)*COS(7*T)
   + 3/128*C(4)*C(2)*COS(5*T) - 123/128*C(4)*C(2)*COS(3*T)
   + 149/5120*C(4)*C(1)^2*COS(7*T) - 43/256*C(4)*C(1)^2*COS(5*T)
   - 1401/1280*C(4)*C(1)^2*COS(3*T) + 5/2304*C(3)^2*COS(7*T)
   + 115/1152*C(3)^2*COS(5*T) - 45/128*C(3)^2*COS(3*T)
   + 181/7680*C(3)*C(2)*C(1)*COS(7*T) - 199/384*C(3)*C(2)*C(1)*COS(5*T)
   - 263/640*C(3)*C(2)*C(1)*COS(3*T) + 91/3072*C(3)*C(1)^3*COS(7*T)
   + 125/256*C(3)*C(1)^3*COS(5*T) + 4083/256*C(3)*C(1)^3*COS(3*T)
   + 1/512*C(2)^3*COS(7*T) - 43/512*C(2)^3*COS(5*T)
```

```
+417/512*C(2)^3*COS(3*T)+21/1024*C(2)^2*C(1)^2*COS(7*T)
-1235/1024*C(2)^2*C(1)^2*COS(5*T)+17145/1024*C(2)^2*C(1)^2*COS(3*T)
+35/2048*C(2)*C(1)^4*COS(7*T)+2295/2048*C(2)*C(1)^4*COS(5*T)
-139845/2048*C(2)*C(1)^4*COS(3*T)+7/4096*C(1)^6*COS(7*T)
+2375/4096*C(1)^6*COS(5*T)+173547/4096*C(1)^6*COS(3*T))
+A^6*(1/160*C(5)*COS(6*T)+7/80*C(5)*COS(4*T)+35/32*C(5)*COS(2*T)
-35/16*C(5)+9/320*C(4)*C(1)*COS(6*T)-3/40*C(4)*C(1)*COS(4*T)
-485/64*C(4)*C(1)*COS(2*T)+105/8*C(4)*C(1)
+1/80*C(3)*C(2)*COS(6*T)+9/80*C(3)*C(2)*COS(4*T)
-81/16*C(3)*C(2)*COS(2*T)+135/16*C(3)*C(2)
+3/64*C(3)*C(1)^2*COS(6*T)-5/32*C(3)*C(1)^2*COS(4*T)
+5087/192*C(3)*C(1)^2COS(2*T)-1375/32*C(3)C(1)^2
+3/128*C(2)^2*C(1)*COS(6*T)-21/16*C(2)^2*C(1)*COS(4*T)
+1387/64*C(2)^2*C(1)*COS(2*T)-4323/128*C(2)^2C(1)
+5/128*C(2)*C(1)^3*COS(6*T)+3/8*C(2)*C(1)^3*COS(4*T)
-4813/64*C(2)*C(1)^3*COS(2*T)+15375/128*C(2)*C(1)^3
+3/512*C(1)^5*COS(6*T)+89/64*C(1)^5*COS(4*T)
+10863/256*C(1)^5*COS(2*T)-35691/512*C(1)^5)
+A^5*(1/64*C(4)*COS(5*T)+15/64*C(4)*COS(3*T)
+11/192*C(3)*C(1)*COS(5*T)-9/64*C(3)*C(1)*COS(3*T)
+1/64*C(2)^2*COS(5*T)-21/64*C(2)^2*COS(3*T)+5/64*C(2)*C(1)^2*COS(5*T)
-129/64*C(2)*C(1)^2*COS(3*T)+5/256*C(1)^4*COS(5*T)
+711/256*C(1)^4*COS(3*T))+A^4*(1/24*C(3)*COS(4*T)
+5/6*C(3)*COS(2*T)-15/8*C(3)+1/8*C(2)*C(1)*COS(4*T)
-31/8*C(2)*C(1)*COS(2*T)+15/2*C(2)*C(1)+1/16*C(1)^3*COS(4*T)
+59/16*C(1)^3*COS(2*T)-57/8*C(1)^3)+A^3*COS(3*T)*(1/8*C(2)
+3/16*C(1)^2)+A^2*C(1)*(1/2*COS(2*T)-3/2)+A*COS(T)

WRITE "TIME ",TIME()-TT," MSEC";
TIME 114753 MSEC
END;
```


Заключение. Задача об одномерном движении частицы вблизи минимума произвольного гладкого потенциала в принципе полностью решается построенным рядом. Любое движение является периодическим, и может быть найдено со сколь угодно высокой точностью. Нет никаких причин для того, чтобы этот ряд не сходился, за исключением тех случаев, когда амплитуда колебаний возрастает настолько, что это приводит к качественной перестройке движения (например, частица достигает соседнего максимума потенциала и убегает из потенциальной ямы, см. рис. 1).

Ситуация совсем не так проста уже в двумерном случае. Приближенно можно заменить потенциал $U(x)$ на квадратичную форму $U_0(x)$. Приводя ее к главным осям, получим $U_0(x) = (k_1x_1^2+k_2x_2^2)/2$. В этом приближении решение очевидно: $x_{1,2}(t)=a_{1,2}\cos\omega_{1,2}t+b_{1,2}\sin\omega_{1,2}t$, $\omega_{1,2}=\sqrt{k_{1,2}/m}$. Однако при рассмотрении нелинейных членов возникает множество приближенных резонансов $j_1\omega_1\approx j_2\omega_2$. Они приводят к малым знаменателям $1/(j_1\omega_1-j_2\omega_2)$ в членах ряда теории возмущений. В результате, члены сколь угодно высокого порядка малости могут иметь сколь угодно малые знаменатели, и ряд не сходится.

И это не формально-математическая трудность. При решении по теории возмущений функции $x_{1,2}(t)$ обязательно являются квазипериодическими, т. е. выражаются через $\cos(j_1\omega_1\pm j_2\omega_2)t$. Однако в действительности существуют движения, не представимые в таком виде. Они имеют очень запутанный вид, и называются стохастическими (или хаотическими). Теория возмущений не может ухватить этой качественной перестройки движения, и потому ее ряды не могут описывать реальные движения со сколь угодно высокой точностью (т. е. сходиться). Исследование стохастических движений в классической механике представляет собой в настоящее время весьма развитую и интересную область науки, см. книгу [7].

ЗАДАЧИ

Задача 1. Для движения частицы в кулоновском поле a/r, найти всевозможные скобки Пуассона гамильтониана, компонент радиус-вектора, импульса, момента импульса, вектора Рунге—Ленца (см. [8], задача 10.26).

Решение. Процедура POISS вычисляет скобки Пуассона. Компотенты момента импульса вычислены при помощи циклической перестановки индексов. Заметим, что REDUCE присваивает только

что вычисленное выражение переменной WS (Work Space), которую можно использовать для построения следующего выражения.

```
OPERATOR Q,P; ARRAY M(3),L(3);

% POISSON BRACKETS ;
PROCEDURE POISS(F,G);
FOR I:=1:3 SUM DF(F,P(I))*DF(G,Q(I))-DF(F,Q(I))*DF(G,P(I));
POISS

% SCALAR PRODUCT ;
PROCEDURE SCAL(A,B);
FOR I:=1:3 SUM A(I)*B(I);
SCAL

% HAMILTONIAN ;
H:=SCAL(P,P)/2+A*SCAL(Q,Q)**(-1/2)$

% ANGULAR MOMENTUM ;
J:=2$ K:=3$
FOR I:=1:3 DO << M(I):=Q(J)*P(K)-Q(K)*P(J);J:=K;K:=I >>;

% RUNGE-LENZ VECTOR ;
FOR I:=1:3 DO
L(I):=SCAL(P,P)*Q(I)-SCAL(P,Q)*P(I)+A*Q(I)*SCAL(Q,Q)**(-1/2);

POISS(M(1),Q(2));
-Q(3)
POISS(M(1),P(2));
-P(3)
POISS(M(1),M(2));
-P(2)*Q(1)+P(1)*Q(2)
POISS(M(1),L(2));
(SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*P(3)*P(2)*Q(3)^2*Q(2)+SQRT(Q(3)^2+Q(2)^2+
Q(1)^2)*P(3)*P(2)*Q(2)^3+SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*P(3)*P(2)*Q(2)*Q(1)^2
+SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*P(3)*P(1)*Q(3)^2*Q(1)+
SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*P(3)*P(1)*Q(2)^2*Q(1)+SQRT(Q(3)^2+Q(2)^2+
Q(1)^2)*P(3)*P(1)*Q(1)^3-SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*P(2)^2*Q(3)^3-
SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*P(2)^2*Q(3)*Q(2)^2-SQRT(Q(3)^2+Q(2)^2+
Q(1)^2)*P(2)^2*Q(3)*Q(1)^2-SQRT(Q(3)^2+Q(2)^2+
```



```
Q(1)^2)*P(1)^2*Q(3)^3-SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*P(1)^2*Q(3)*Q(2)^2-
SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*P(1)^2*Q(3)*Q(1)^2-Q(3)^3*A-Q(3)*Q(2)^2*A-
Q(3)*Q(1)^2*A)/(SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*(Q(3)^2+Q(2)^2+Q(1)^2))
WS+L(3);
0
POISS(H,Q(1));
P(1)
POISS(H,P(1));
(Q(1)*A)/(SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*(Q(3)^2+Q(2)^2+Q(1)^2))
POISS(H,M(1));
0
POISS(H,L(1));
0
ON GCD;
POISS(L(1),L(2));
(SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*P(3)^2*P(2)*Q(1)-SQRT(Q(3)^2+Q(2)^2+
Q(1)^2)*P(3)^2*P(1)*Q(2)SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*P(2)^3*Q(1)-
SQRT(Q(3)^2+Q(2)^2+Q(1)^2)*P(2)^2*P(1)*Q(2)SQRT(Q(3)^2+Q(2)^2+
Q(1)^2)*P(2)*P(1)^3*Q(1)+2*P(2)*Q(1)*SQRT(Q(3)^2+Q(2)^2+
Q(1)^2)*P(1)^3*Q(2)2*P(1)*Q(2)*A)/SQRT(Q(3)^2+Q(2)^2+Q(1)^2)
WS/H;
2*(P(2)*Q(1)-P(1)*Q(2))
M(3);
P(2)*Q(1)-P(1)*Q(2)
END;
```

Задача 2. Найти потенциал, создаваемый на больших расстояниях следующими системами зарядов (рис. 2): диполь, плоский и линейный квадруполь, пространственный, плоский и два линейных октуполя (см. [9], задачи 94—95).

Решение. Оно основано на разложении потенциала точечного заряда в точке AX, AY, AZ в ряд по этим величинам. Разложение производится процедурой BINOM из п.1.

```
% BINOMIAL EXPANSION ;
PROCEDURE BINOM(X,N);
BEGIN SCALAR S,T,I,J;S:=1;T:=1;I:=1;J:=N;
   WHILE (T:=T*X*J/I) NEQ 0 DO << S:=S+T;I:=I+1;J:=J-1 >>;
```

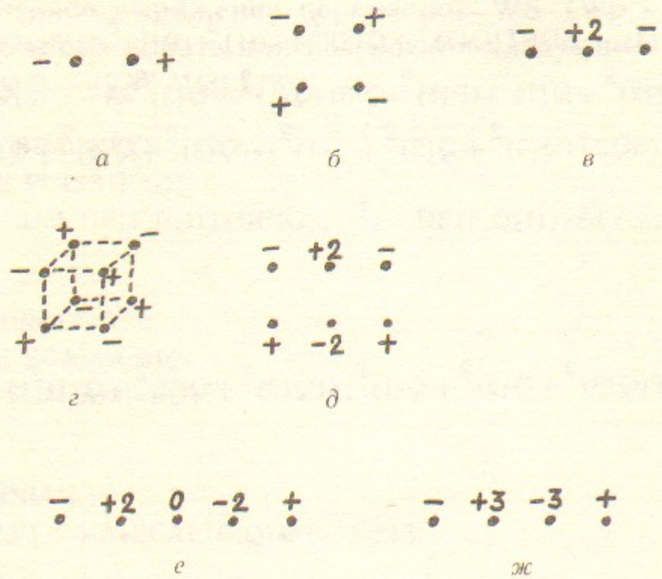


Рис. 2.

```
    RETURN S
END;
BINOM

% POTENTIAL OF UNIT CHARGE AT AX,AY,AZ ;
PROCEDURE F(AX,AY,AZ);
BINOM((AX**2+AY**2+AZ**2-2*(AX*X+AY*Y+AZ*Z))/R**2,-1/2)/R;
F
WEIGHT A=1; WTLEVEL 1;
% DIPOLE ;
FOR I:=-1 STEP 2 UNTIL 1 SUM
I*F(I*A,0,0);
(2*A*X)/R^3
WTLEVEL 2;
% QUADRUPOLE ;
FOR I:=-1 STEP 2 UNTIL 1 SUM
FOR J:=-1 STEP 2 UNTIL 1 SUM
I*J*F(I*A,J*A,0);
(12*A^2*X*Y)/R^5

% LINEAR QUADRUPOLE ;
2*F(0,0,0)-F(A,0,0)-F(-A,0,0);
```



```
(A^2*(R^2-3*X^2))/R^5

WTLEVEL 3;
% OCTUPOLE ;
FOR I:=-1 STEP 2 UNTIL 1 SUM
FOR J:=-1 STEP 2 UNTIL 1 SUM
FOR K:=-1 STEP 2 UNTIL 1 SUM
I*J*K*F(I*A,J*A,K*A);
(120*A^3*X*Y*Z)/R^7

% PLANAR OCTUPOLE ;
2*F(0,A/2,0)-F(A,A/2,0)-F(-A,A/2,0)
-2*F(0,-A/2,0)+F(A,-A/2,0)+F(-A,-A/2,0);
(3*A^3*Y*(R^2-5*X^2))/R^7

% LINEAR OCTUPOLE 1 ;
F(2*A,0,0)-2*F(A,0,0)+2*F(-A,0,0)-F(-2*A,0,0);
(6*A^3*X*(-3*R^2+5*X^2))/R^7

% LINEAR OCTUPOLE 2 ;
F(3/2*A,0,0)-3*F(1/2*A,0,0)+3*F(-1/2*A,0,0)-F(-3/2*A,0,0);
(3*A^3*X*(-3*R^2+5*X^2))/R^7
END;
```

ЛИТЕРАТУРА

1. *Hearn A.C.* REDUCE User's Manual. Rand Corporation, 1983.
2. *Еднерал В.Ф., Крюков А.П., Родионов А.Я.* Язык аналитических вычислений REDUCE.—Изд. МГУ, ч.I, 1983, ч.II, 1986. Готовится новое издание: Изд. МГУ, 1989 .
3. *Боголюбская А.А., Жидкова И.Е., Ростовцев В.А.* Система программирования REDUCE-2.—ОИЯИ Б1-11-83-512, Дубна, 1983.
4. *Rayna G.* REDUCE: Software for Algebraic Computation.—Springer-Verlag, 1987 .
5. *Stoutemayer D.* REDUCE Interactive Lessons.—REDUCE Newsletters, Univ. of Utah, 1978.
6. *Ландау Л.Д., Лифшиц Е.М.* Механика.—М.: Наука, 1988.
7. *Лихтенберг А., Либерман М.* Регулярная и стохастическая динамика.—М.: Мир, 1984.
8. *Коткин Г.Л., Сербо В.Г.* Сборник задач по классической механике.—М.: Наука, 1977.
9. *Батыгин В.В., Топтыгин И.Н.* Сборник задач по электродинамике.—М.: Наука, 1970.

А.Г. Грозин

Решение физических задач на языке REDUCE.
1. Язык REDUCE
2. Классический нелинейный осциллятор

Ответственный за выпуск С.Г.Попов

Работа поступила 5 августа 1988 г.
Подписано в печать 30.08.88 г. МН 05456
Формат бумаги 60×90 1/16 Объем 3,5 печ.л., 2,8 уч.-изд.л.
Тираж 290 экз. Бесплатно. Заказ № 115

Набрано в автоматизированной системе на базе фотонаборного автомата ФА1000 и ЭВМ «Электроника» и отпечатано на ротапринте Института ядерной физики СО АН СССР,
Новосибирск, 630090, пр. академика Лаврентьева, 11.