SCRATCHPAD: A Capsule View

by

James H. Griesmer
Richard D. Jenks
Mathematical Sciences Department
T. J. Watson Research Center
Yorktown Heights, New York

ABSTRACT

SCRATCHPAD is an interactive system for algebraic manipulation available under the CP/CMS time-sharing system at Yorktown Heights. It features an extensible declarative language for the interactive formulation of symbolic computations. The system is a large and complex body of LISP programs incorporating significant portions of other symbolic systems.

Here we present a capsule view of SCRATCHPAD, its language and its capabilities. This is followed by an example which illustrates its use in an application involving the solution of an integral equation.

I. INTRODUCTION

SCRATCHPAD is an experimental system for interactive symbolic mathematics implemented under the CP/CMS time-sharing system at the IBM Research Center, Yorktown Heights, New York. The current system [Griesmer et al, 1972b] has evolved from earlier versions [Blair et al, 1970] and [Griesmer et al, 1971] and has been used on some 50 applications over the past year. SCRATCHPAD is a large and complex body of LISP programs incorporating significant portions of other systems such as REDUCE2 [Hearn, 1971], MATHLAB [Engelman, 1971a], CHARYBDIS [Millen, 1968], SIN [Moses, 1967], META/LISP [Jenks, 1970], and META/PLUS [Jenks, 1971]. The SCRATCHPAD system currently has facilities for the manipulation of multi-indexed variables and functions, finite and infinite sums, products, and sequences, and polynomials of arrays. Specific capabilities include symbolic differentiation, integration, simplification, polynomial factoriza-

tion over the integers, solutions of linear equations and unlimited precision rational arithmetic.

Recent additions to SCRATCHPAD include a "backtracking" capability, an array facility similar to APL, and a variety of extension facilities which enable the user to tailor the basic system to his individual problem area.

II. LANGUAGE OVERVIEW

The SCRATCHPAD language is a special-purpose high-level language designed for use by a non-programming scientist for interactive formulation and solution of problems involving symbolic manipulation. Its design is an attempt to provide a rich, expressive language capturing the style and succinctness of mathematics while at the same time remaining simple, uniform, and easily extendable.

One key objective in the design of the language was to retain the definitional style of the mathematician. This is the concept that the "main

thought" comes first with qualifiers following. The result is an interactive language where all user commands have the form of relational declarations. For the purpose of comparison, in Figure 1 we give two programs to compute the first 11 terms of a recurrence relation, one written in the form of a procedure, the other written in the SCRATCHPAD language.

(a) PROC GENERATE;
 BEGIN M,P,Q;
 M:=2;
 DISPLAY P:=1;
 DISPLAY Q:=1+x;
 DO UNTIL M > 10;
 DISPLAY V:=x*Q-P;
 P:=Q; Q:=V; M:=M+1;
 END; END; END;
 GENERATE;

(b) $P_0(x) = 1$
 $P_1(x) = 1+x$
 $P_n(x) = x \cdot P_{n-1}(x) - P_{n-2}(x), n \geq 2$
 $P_n(x) \mid n \in [0, 1, \dots, 10]$

Figure 1. Generation of Polynomial Sequence by a Recurrence Relation (a) as a procedure, and (b) in SCRATCHPAD. In the SCRATCHPAD program, the first three commands define the sequence $P_0(x), P_1(x), \dots$; the fourth command causes $P_0(x)$ through $P_{10}(x)$ to be successively evaluated and displayed.

The language has been designed for eventual use with graphical input and output. It is two-dimensional in the sense that general sub- and super-scripting is allowed, and forms involving Σ , Π , ∂ , and $\int$ are given in "standard" mathematical notation, e. g. all of the following are legitimate SCRATCHPAD forms:

$$x_{i,j}^k(u,v) \quad \sum_{i=1}^n \prod_{j=0}^m u_i^j \quad \int_{x=a}^b f(x) \quad (\text{sic!})$$

These forms are linearized on input to the current system by means of a few simply stated rules [Griesmer et al, 1972].

2.1 Syntax

The SCRATCHPAD language has four major parts of speech: (in order of increasing generality) PRIMITIVE, EXPRESSION, STATEMENT, and COMMAND.

1. PRIMITIVES are of five types:

INTEGERS	10	1234567890
VARIABLEs	x	alpha
FORMs	f(x)	x_i
ARRAYs	[1, 2, 3]	[1, 2, ..., n]
SETs	{x f(x) > 0}	{y y ∈ S ∧ y > 0}

2. EXPRESSIONs represent the objects manipulated by the user. An EXPRESSION is either a PRIMITIVE or a syntactically allowed combination of PRIMITIVES, parentheses, quote marks, and operators, e. g.

$$t^{-1} * (1 - u(t))^2 \quad f(t) - \int_{x=0}^{\infty} \frac{J_1(x*t) * \sin(x)}{x} .$$

3. STATEMENTs are the fundamental semantic constructs in the language. They have the uniform format: L r E, where

L: the left-part, is usually a VARIABLE or FORM;

r: the relator, is one of =, ≠, <, ≤, >, ≥, ∈, ∉; and,

E: the right-part, is an EXPRESSION; e. g.,

$$f_i(x) = x^i \quad x > 0 \quad i \in [1, 2, \dots, n] .$$

4. COMMANDs (user commands) consist of either a single STATEMENT, or a STATEMENT followed by one or more qualifiers preceded by commas. Each qualifier is again a STATEMENT and thus has the format: L r E, e. g.

$$f_i(x) = x^i, \quad x > 0, \quad i \in [1, 2, \dots, n] \quad (1)$$

2.2 Semantics

The effect of a STATEMENT, L r E, is to create a replacement rule. For the case L = E, the replacement rule takes the form:

"replace l by e when c"

where:

l and e are representations of the left- and right-parts of the STATEMENT in terms of formal parameters; and,

c is a conditional expression which describes the condition under which the replacement rule is applicable.

Every user command creates exactly one permanent replacement rule; each rule as created is pushed onto the "rule stack". This ordered set of replacement rules is initially empty and at every instant in time defines the environment of evaluation. Rules are effectively cleared when reissued with an empty right part, e. g. " $x =$ ".

We now outline the substitution and simplification phase of evaluation. The evaluation of VARIABLES is carried out by searching the rule stack from top to bottom for an appropriate substitution. If a non-empty substitution is found, the substitution is itself evaluated; the resulting EXPRESSION, called the "value" of the VARIABLE, then replaces the VARIABLE in the original EXPRESSION. If no substitution or the "empty" substitution is found, the value of the VARIABLE is the VARIABLE itself.

The evaluation of FORMS is similar except that the actual parameters of the FORM are first evaluated, then substituted for the formal parameters of appropriate rules.

The evaluation of EXPRESSIONS is similar to that in LISP. For example, the evaluation of $x + y$ consists of first evaluating x and y to obtain values x' and y' , then applying the system function associated with the infix operator "+" to produce a value.

A STATEMENT is evaluated for the purpose of creating a replacement rule. This involves first evaluating its right-part, an EXPRESSION, then evaluating the parameters (if any) of the left-part to determine the condition c . As an example, the replacement rule created for (1) is:

"replace $f_b(a)$ by a^b when $a > 0$ and $b \in [1, 2, \dots, n]$ ".

A double evaluation of the right-part of a STATEMENT is thus normally done: once when

the replacement rule is created, and then again whenever the rule is applied. Quote marks (') are used to cause an EXPRESSION to evaluate to itself, and thus to delay or inhibit evaluation; e. g. given the command " $y = 2$ ", the command: $x = 'y'$ creates the replacement rule: "replace x by y " so that on later application, x evaluates to 2.

In the evaluation of COMMANDS, qualifying STATEMENTS are evaluated from right to left. Each in turn creates a temporary replacement rule which is pushed onto the rule stack. These temporary rules are removed following the evaluation of the left-most STATEMENT.

Iteration Qualifiers. Qualifiers preceded by a bar ("|"), instead of a comma, cause STATEMENTS to their left to be iterated. Iteration qualifiers are restricted to those of the form " $x \in u$ " where x is a VARIABLE and u evaluates to an ARRAY or SET.

2.3 System Commands

In addition to the above user COMMANDS, a number of system commands may be given; these provide miscellaneous services to the user, such as to read or write files, to modify or record the current state of the system, or to backtrack (see Section IV). Also provided is a procedural language facility and commands for operator, semantic, and syntax extension (see [Griesmer et al, 1972] for details).

III. WORKSPACE

The design of SCRATCHPAD for interactive manipulation of symbolic expressions centers around the use of the workspace, denoted by the special variable: "ws". The statement "E" is shorthand for " $ws = E$ " and causes the expression E to be evaluated, displayed, and placed in the workspace. A complete history of workspace values is continuously recorded on secondary storage as the user session progresses. Successive workspace values are assigned consecutive ex-

pression numbers beginning with 1. At any later time, expression n may be retrieved by evaluation of the form "ws(n)".

Expressions are manipulated by their evaluation in a dynamically varying environment of replacement rules. This is normally done by first loading an expression into the workspace, then operating on it in subsequent commands; e. g. ,

$$\int_{x=0}^1 ws, \quad t = 0$$

causes the current expression in the workspace to be evaluated at $t=0$ then integrated from 0 to 1 with respect to x : the result is then displayed and becomes the next value of the workspace.

A workspace expression may either be a scalar expression such as a polynomial, power series, and an equation -- or may be an array of arbitrary dimension and structure. A number of system provided functions may be used to manipulate workspace expressions. These result in workspace transformations of four types:

(1) scalar-to-scalar, e. g. "FACTOR(ws)" replaces a polynomial with integer coefficients in the workspace by a product of its irreducible factors. These functions when applied to an array-valued argument are mapped onto the array in component-by-component fashion as in APL, e. g. , "FACTOR([e_1, e_2])" is equivalent to "[FACTOR(e_1), FACTOR(e_2)]".

(2) scalar-to-array, e. g. "COVEC(x, ws)" will replace a polynomial in x of degree n in the workspace by an array of $n + 1$ coefficients.

(3) array-to-array, e. g. , "COEFF([$x_1, x_2, \dots, x_n$], ws)", will replace a system of n equations in n unknowns x_i by a matrix of coefficients (an array of arrays of coefficients); and,

(4) array-to-scalar, e. g. "DETERMINANT(ws)" will replace a square matrix in the workspace by its determinant. The use of the latter three of these commands is illustrated in the example in Section VI.

IV. BACKTRACKING; HISTORY FILE

An important facility which greatly enhances the interactive use of SCRATCHPAD is that of backtracking. By the use of two system commands REVERT and REJECT, a user may "backup" to any previous stage of his interactive dialogue.

The backtracking facility is made possible by the maintenance of a complete record of the user session. This record, called the "history file", also provides certain other obvious user aids such as the ability to obtain a hard copy of all previously issued commands and workspace values. Furthermore, the file permits recovery following a system malfunction.

V. PRESENT IMPLEMENTATION

The system currently runs on the S/360 Model 67 under the CP/CMS time sharing system in a 768 byte virtual machine. A batch version (called BATCHPAD) is also available under OS/360. Communication to SCRATCHPAD is via an IBM 2741 terminal with either an EBCDIC or APL type ball.

VI. AN EXAMPLE

The following example presents a partial solution to one of three problems proposed by W. M. Gentleman in the SIGSAM Bulletin (Bulletin for the Special Interest Group on Symbolic and Algebraic Manipulation), No. 18, April 1971. The problem is to find the largest eigenvalue β^2 of the following integral equation:

$$\int_{x=0}^1 \left(\frac{1-m^3}{3} + (1-m) \left(xy - \frac{(x+y)(1+m)}{2} \right) \right) f(x) = \beta^2 f(y) \quad (I)$$

where $m = \max(x, y)$. The method of solution is as follows. The integral equation is converted to a differential equation by repeatedly differentiating the integral equation, and eliminating (if necessary) the various integrals between the new equations formed. Four derivatives suffice. The resulting

fourth derivative is a linear differential equation with constant coefficients depending on β^2 . Its solution may then be written down immediately as a linear combination of exponentials:

$$f(x) = \sum_{i=1}^4 r_i e^{s_i x} \quad (\text{II})$$

where the s_i are specific functions of β^2 . The coefficients r_i are then determined by substituting general solution (II) of the differential equation back into the original integral equation (I) and noting that it must be an identity for all y . This then requires that a set of four linear combinations of the r_i simultaneously vanish. For not all of the r_i to be zero, the determinant of a 4 by 4 matrix formed from these linear combinations of r_i must itself vanish. The determinant is a scalar expression involving β^2 and is set equal to zero in order to solve for β^2 .

A SCRATCHPAD conversation for carrying out the solution to this problem through the calculation of the determinant is given in Figure 2. Its description is as follows:

(a) Initialization. The user enters SCRATCHPAD from LISP by executing the LISP function "spad" applied to an empty list of arguments. Once in SCRATCHPAD, the user issues the system command ")clear" to clear all previous values and rules. The flag EXP (meaning "expand") is then set to 0 to keep all expressions in unexpanded form. The integrand is next entered as a function u of m . Finally, the left-hand side of (I), expressed as a sum of two integrals, is loaded into the workspace (expression (1)).

(b) Differentiation. We now reduce the integral equation to a differential equation. First, we give a statement which causes any integral of the form $\int_y^1$ to be replaced by $\int_0^1 - \int_0^y$. This turns out to be necessary in order to eliminate various integrals which will occur on differentiation. We then let e_n denote the n th derivative of the left- minus the right-hand side of the inte-

gral equation. Next, the values of e_1, e_2, e_3 , and e_4 are successively computed and displayed as (2), (3), (4) and (5), respectively. The "EXP=1" allows products and powers to expand out, etc. in order that simplification may occur.

Expression (5) is the linear differential equation which has the solution (II) with $s_k = i^k / \sqrt{\beta}$ ($1 \leq k \leq 4$), the solutions D to $-\beta^2 D^4 + 1 = 0$.

(c) Substitution. We now substitute general solution (II) back into (I). We delay however the substitution for s_k until step (e) below.

Workspace expression (1) is retrieved from secondary storage and becomes (6). The general solution (II) is next entered into the system. Re-evaluating (6) then causes the substitution for $f(x)$ to be made (expression (7)). With EXP=0 (our global setting), no expansion occurs. Note that $\int$ of the Σ in each case becomes the Σ of the $\int$ and the common Σ is pulled out in front.

(d) Performing the integration. Here we illustrate the concept of "windowing" and the use of the pattern matching facility to carry out the integration inside the summation sign in expression (7). The flag XOP (meaning "expand operator forms") is kept at 0 (its default setting) in order to keep the outer sum from expanding. To do the integration, we "window" in on the sum of integrals in the numerator of the workspace. The designation of this subexpression in the workspace is given by a "Dewey-decimal" denotation [1,1], meaning "the first part of the first part". With the window set on the workspace, all accesses of or stores into the workspace affect only the selected window. Evaluating the workspace with EXP=1 yields expression (8), and we find that the system is unable to carry out the various integrations. All that is needed here, however, is for the user to supply two integration rules;

$$\int_x e^{\alpha x} = \frac{e^{\alpha x}}{\alpha} \quad (\text{III})$$

$$\int_x x^n e^{\alpha x} = \frac{x^n e^{\alpha x}}{\alpha} - \frac{n}{\alpha} \int_x x^{n-1} e^{\alpha x}, \quad n \geq 1 \quad (\text{IV})$$

(Note: The above statements are called "pattern statements"; they have the form: $L \text{ r } E$ where L is not a primitive. Here, the left-part L denotes a "template" to be tested against every expression during subsequent evaluation. Variables placed in the template define the set of expressions which are to match the pattern. Variables are quoted to stand for themselves; if unquoted, they stand for expressions satisfying a preassigned condition or else for arbitrary expressions. For example, the effect of pattern matching statement (IV) is to create the following replacement rule:

"replace $\int_b^c e^{a*b}$ by ... when $c \geq 1$
(for all a and b)".

Once the integration rules have been declared, the integration is performed simply by re-evaluating the workspace (expression (9)). (Note: The integration could have been carried out automatically using the SIN option [Griesmer et al, 1972b]).

(e) Determining the linear equations. Having finished the integration, we now substitute for the s_k . Complex arithmetic is required and so we must issue the pattern statement: " $i^2 = -1$ ". Next the window is removed, expansion is turned on, and XOP is set to 1 (to allow the Σ to ex-

pand). Expression (9) minus the right-hand side of the integral equation is then evaluated with the substitution for s_k performed (expression (10)).

Expression (10) is a polynomial of degree 3 in y which must be 0 for all y in the range 0 to 1. This is true only if each of the coefficients of y^k , $0 \leq k \leq 3$, is identically 0. The command "COVEC(y,ws)" replaces the workspace by an array of length 4 whose k -th component is the coefficient of y^k in (10). (Note: If an array can be displayed horizontally or in block form, the system puts it out that way; otherwise, it outputs the array component-by-component as in (11) with the k -th component denoted by " $(ws)_k$ ".)

(f) Calculating the determinant. The expressions in (11) give four different linear combinations of the r_i which must all vanish. For the r_i not all to be zero, the determinant of the matrix of coefficients of $[r_1, r_2, r_3, r_4]$ in (11) must vanish. This matrix is extracted from (11) by means of the COEFF function and is given as expression (12).

We now note that matrix (12) is unnecessarily complicated and may be simplified by row and column operations. This is done in the four commands following (12). The simplified matrix is then displayed (expression (13)). Finally, the determinant is computed (expression (14)) and the result simplified (expression (15)).

Expression (15), our final result, is a transcendental expression in β^2 which may be equated to zero and solved numerically.

Figure 2. Example Conversation with SCRATCHPAD for Obtaining a Solution to an Integral Equation Problem. User input in lower case begins at column 8; system output in upper case begins in column 1. The following linearization rules are needed for understanding the input commands: arithmetic expressions are linearized as in FORTRAN, e. g., "***" is used for exponentiation; scripts are enclosed in angle brackets with subscripts given first; multiple scripts at the same location are separated by commas; superscripts are given after subscripts with a semi-colon separating the two groups; limits on operator forms are given as superscripts and subscripts.

In addition, the following character substitutions are necessary for transliterating notations appearing in Section 2: ( ) for [ ]; IN for ϵ ; < = and > = for $\leq$ and $\geq$; SUM, INT, and DF for Σ , $\int$, and ∂ . An example:

$\sum_{j=1}^n a_{ij}^k x_j^2$ is linearized as : sum<j=1;n>a<i,j;k>*x<j>**2.

```

spad()
)clear
exp=0
u(m)=((1-m**3)/3+(1-m)*(x*y-(x+y)*(1+m)/2))*f(x)
int<x=0;y>u(y)+int<x=y;1>u(x)
(1)
Y
/
- | F(X)(2Y^3 + (Y - 1)(6Y*X - (3Y + 3)(Y + X)) - 2)
/
X=0
+
1
/
- | F(X)(2X^3 + (6Y*X - (3Y + 3X)(X + 1))(X - 1) - 2)
/
X=Y
/
6

int<x='y';1>=int<x=0;1>-int<x=0;y>
e<n>=df<y;n>(ws - beta**2*f(y))
e<n> | n in (1,2,3,4), exp=1

- 2BETA^2 ( d F(Y)) + Y^2 (| F(X)) - 2Y(| X*F(X)) + 2(| X*F(X)) + | X^2 F(X) - | X^2 F(X) - | F(X)
Y X=0 X=0 X=0 X=0 X=0 X=0
(2) -----
2

(3) - BETA^2 ( d F(Y)) + Y(| F(X)) - | X*F(X)
Y X=0 X=0

(4) - BETA^2 ( d F(Y)) + | F(X)
Y X=0

(5) - BETA^2 ( d F(Y)) + F(Y)
Y

(6) ws(1)
Y
/
- | F(X)(2Y^3 + (Y - 1)(6Y*X - (3Y + 3)(Y + X)) - 2)
/
X=0
+
1
/
- | F(X)(2X^3 + (6Y*X - (3Y + 3X)(X + 1))(X - 1) - 2)
/
X=Y
/
6

```



```

+
  S 1 3 - 3E 1 2 + 6E 1 1 - 6E 1 1 + 6)
R (E S 1 1
/
S 1 4
S 1

'1**2'=-1
window(ws)=
exp=1
xop=1
ws-beta**2*f(y), s(k)=1**k*c

(10)
- 1*C Y R 3 3 + 1*C Y R 3 3 + 31*C Y R 3 3 - 31*C Y R 3 3 - 21*C R 3 3 + 21*C R 3 3 - 31*C Y R 2 2 + 31*C Y R 2 2
+
- 31*C Y R 2 2 + 31*C Y R 2 2 + 61*C Y R 2 2 - 61*C Y R 2 2 + 61*C Y R 2 2 - 61*C Y R 2 2 - 31*C R 2 2 + 31*C R 2 2
+
- 31*C R 2 2 + 31*C R 2 2 - 61*C*Y*E R 2 2 + 61*C*Y*E R 2 2 + 61*C*E R 2 2 - 61*C*E R 2 2 - 61*E R 2 2
+
- 61*E R 1 1 - 61*E R 2 2 - 61*E R 3 3 - 1*C R 3 3 + C Y R 3 3 - C Y R 3 3 - 3C Y R 3 3 + 3C Y R 3 3 + 2C R 3 3 - 2C R 3 3
+
6C*Y*E R 1 1 - 6C*Y*E R 3 3 - 1*C R 3 3 + 6C*E R 1 1 - 1*C R 3 3
/
61*C 4

covec(y,ws)

(11)
(Ws) 0
:
- 21*C R 3 3 + 21*C R 3 3 - 31*C R 3 3 + 31*C R 3 3 - 31*C R 3 3 + 31*C R 3 3 + 61*C*E R 3 3 - 61*C*E R 3 3 - 61*E R 3 3
+
- 61*E R 1 1 - 61*E R 2 2 - 61*E R 3 3 + 2C R 3 3 - 2C R 3 3 - 6C*E R 1 1 + 6C*E R 3 3 - 1*C R 3 3
/
61*C 4

(Ws) 1
:
1*C R 2 2 - 1*C R 2 2 + 21*C*R 2 2 - 21*C*R 2 2 + 21*C*R 2 2 - 21*C*R 2 2 - 21*E R 2 2 + 21*E R 2 2 - C R 2 2 + C R 2 2
+
2E R 1 1 - 2E R 3 3 - 1*C R 3 3
/
21*C 3

(Ws) 2 :
- R 4 4 + R 3 3 - R 2 2 + R 1 1
-----
2C 2

(Ws) 3 :
- 1*R 4 4 + 1*R 2 2 + R 3 3 - R 1 1
-----
61*C

```

(12) coeff((r<1>,...,r<4>), ws)

$$(WS)_{1,1} : \frac{31C^2 - 61E^2 - 2C - 6C^3E}{61C^4}$$

$$(WS)_{1,2} : \frac{2C^3 - 3C^2 - 6C^3E - 6E^2}{6C^4}$$

$$(WS)_{1,3} : \frac{31C^2 - 61E^2 - 1C^3 + 2C^3 + 6C^3E - 1C^4}{61C^4}$$

$$(WS)_{1,4} : \frac{-2C^3 - 3C^2 + 6C^3E - 6E^2}{6C^4}$$

$$(WS)_{2,1} : \frac{-21C^2 + C^2 + 2E^2}{21C^3}$$

$$(WS)_{2,2} : \frac{-C^2 + 2C + 2E^2}{2C^3}$$

$$(WS)_{2,3} : \frac{-21C^2 - C^2 - 2E^2}{21C^3}$$

$$(WS)_{2,4} : \frac{C^2 + 2C - 2E^2}{2C^3}$$

$$(WS)_{3,1} : \frac{1}{2C^2}$$

$$(WS)_{3,2} : \frac{-1}{2C^2}$$

$$(WS)_{3,3} : \frac{1}{2C^2}$$

$$(WS)_{3,4} : \frac{-1}{2C^2}$$

$$(WS)_{4,1} : \frac{-1}{61C^2}$$

$$(WS)_{4,2} : \frac{1}{6C^2}$$

$$(WS)_{4,3} : \frac{1}{61C^2}$$

$$(WS)_{4,4} : \frac{-1}{6C^2}$$

(ws)<4>=6*1*c*(ws)<4>
 (ws)<3>=2*c**2*(ws)<3>
 (ws)<2>=1*c**3*((ws)<2>+(ws)<3>/c**2+(ws)<4>/(2*1*c))
 (ws)<1>=c**4*((ws)<1>+1/c**3*(ws)<2>+1/(2*c**2)*(ws)<3>-1/(3*c)*(ws)<4>)
 ws

(13)

$\frac{1C}{E}$	$\frac{-C}{E}$	$\frac{-1C}{E}$	$\frac{C}{E}$
$\frac{1C}{E}$	$\frac{-C}{1E}$	$\frac{-1C}{-E}$	$\frac{C}{-1E}$
1	-1	1	-1
-1	1	1	-1

determinant(ws)

(14)

$$21E^2 \frac{1C + C}{E} + 21E^2 \frac{1C - C}{E} + 21E^2 \frac{-1C + C}{E} + 21E^2 \frac{-1C - C}{E} + 81$$

ws*(-1)/2

(15)

$$E \frac{1C + C}{E} + E \frac{1C - C}{E} + E \frac{-1C + C}{E} + E \frac{-1C - C}{E} + 4$$