International Journal of Advanced Computer and Mathematical Sciences
ISSN 2230-9624. Vol 4, Issue 1, 2013, pp58-62
http://bipublication.com

APPLICATIONS OF COMPUTER ALGEBRA

***Ravindra Pawar, #R Dhaigude, Kottawar C. S. and Walke Ragini R.**

*MKT college, Malegaon, Maharashtra, India
#Department of Mathematics, Dr. Babasaheb Ambedkar Marathwada University, AURANGABAD, India

[Received-05/12/2012, Accepted-10/01/2013]

ABSTRACT
Algebra is defined as the manipulation of symbols. Today, algebra is commonly known as abstract algebra or modern algebra. Algebra is mainly used to describe the symmetries or patterns in the mathematics. Operations that made on algebraic structures are basic thing in the computer algebra algorithms. Set, functions (arithmetic, numerical, symbolic), field, group, semi-group, abelian group and rings are the basic things in the basic algebraic systems. Computer algebra is used in various applications and fields like computer science, chemistry, physics, and mechanics and so on.

Index Terms— Algebra, Algorithms

I. INTRODUCTION TO COMPUTER ALGEBRA

Computer algebra is the research field and its driving force mainly comes from various applications. These applications range from computer science and mathematics over physics and engineering. Computer algebra system is simply a software program which facilitates symbolic mathematics. Computer Algebra is also called as Algebraic Manipulation, Formula Manipulation or Symbolic Computation. Computer Algebra is the scientific field in mathematics and computer science. Here, the computation is performed on the symbols and it represents mathematical objects other than their numeric values. According to Winkler [4], computer algebra is a part of computer science and it helps in analyzing, designing, implementation, and applied in algebraic algorithms.

The data mining can be gained by classification and prediction, association discovery, predictions, clustering and sequential Patterns.

a) ALGEBRAIC CALCULATIONS*:*

Algebraic computation or computer algebra or symbol manipulation is a field of scientific computation that implements, analyzes, develops and uses the algebraic algorithms. Typical features of the algebraic computation are:

- computation with the functions (e.g., $\sin, \cos$)
- symbolic operations (Integration, differentiation, factorization, etc)
- computation with variables and symbols (e.g., x)
- computation with the arbitrary precision numbers (no rounding)
- manipulation of formulas

There are two types of algebraic computation: Numerical Computation and Symbolic Computation.

Numerical Computation:

- Computations are done only with the numbers
- It produces approximate results
- It produces only numeric results
- Numeric evaluation will be performed

Symbolic Computation:

- Computations are done in algebraic structures (finite fields, polynomial rings, etc…) and also it computes with the numbers (algebraic number domains).
- It produces exact results
- It produces only symbolic results
- It has symbolic simplification

b) ARITHMETIC:

Arithmetic is the oldest and most elementary branch of the mathematics. Arithmetic is used by everyone and ranging from day-to-day counting to advanced business and science calculations [1]. There are four basic operations in arithmetic and they are: addition (+), subtraction (–), multiplication (×) and division (÷). In addition to these, arithmetic operation includes more advanced operations such as square roots, exponentiation, percentages and logarithmic functions.
The following table illustrates the numeric versus symbolic arithmetic.

Numeric Arithmetic	Symbolic Arithmetic
• Normally inexact • Usually produces some sort of answer (may suffer loss of accuracy, fail to converge, etc.) • Arithmetic relatively cheap • Zeroes usually not given special treatment except to avoid computational singularities • Zero detection: number sufficiently small? • Maximize size of pivots to promote stability	• Always exact • Frequently cannot produce an answer (problem is intractable, expression swell, etc.) • Arithmetic often expensive • Zeroes can greatly simplify calculations by making some of them unnecessary • Zero detection: expression equivalent to zero? • Minimize complexity of pivots to stunt expression growth.

Table: Numeric versus Symbolic Arithmetic
Source: Liska et al [3], COMPUTER ALGEBRA, Algorithms, Systems and Applications, February 11, 1999.

c) TYPES OF EXPRESSIONS:

The expressions that manipulated in computer algebra typically include standard functions of expressions (exponential, sine, etc.); polynomials in multiple variables; arbitrary functions of expressions; various special functions (Bessel functions, etc.); derivatives, sums, optimization; integrals; simplifications; and products of expressions; matrices of expressions; truncated series with expressions and so on. In addition to these, numeric domains supported may include complex, real, rational, interval and algebraic.
Arithmetic operations will be performed according to the order of the operations. Many programming languages are using the precedence levels like in mathematics. Smalltalk and APL have no operator precedence rules. All programming languages are borrowed the precedence rules from C which have an ordered logical bitwise operators, for e.g. PHP, C++ and Perl.
The relative precedence levels of the operators that found in most of the languages are as follows:

1	() [] -> . ::	Grouping, scope, array/member access
2	! ~ - + * & sizeof *type cast* ++x --x	(most) unary operations, sizeof and type casts
3	* / %	Multiplication, division, modulo
4	+ -	Addition and subtraction
5	<< >>	Bitwise shift left and right
6	< <= > >=	Comparisons: less-than, ...
7	== !=	Comparisons: equal and not equal
8	&	Bitwise AND
9	^	Bitwise exclusive OR
10	|	Bitwise inclusive (normal) OR
11	&&	Logical AND
12	||	Logical OR
13	?: = += -= *= /= %= &= |= ^= <<= >>=	Conditional expression (ternary) and assignment operators
14	,	Comma operator

Table: Precedence Levels of the Operators
Source: Rithche D [2]: The development of C language, In History of Programming Languages, 2nd ed., ACM Press 1996.

I. THE EUCLIDEAN ALGORITHM

The Euclidean algorithm is the basic algorithm in computing the greatest common divisors (GCDS) and it is extremely frequent operation in the computer algebra computation.

Let be the ring and $a, b, c \in$. Then here, is the gcd or greatest common divisor of and if:

- $c \mid a$ and $c \mid b$ (c divides both a and b),
- if $d \mid a$ and $d \mid b$, then $d \mid c$, for all $d \in R$.

In the Euclidean domain , it is possible to divide any $a, b \in R, b \neq$ with the remainder , and quotient , so that: $a = qb +$. Then the used notation is $q = a\ div$ and $r = a\ rem$.

- The unit $u \in R$ will be any element with the multiplicative inverse $v \in R$, then $uv = 1$.
- Two elements a, b are associate if $a = ub$ for the unit $b \in R$, and then it is denoted as: $a \sim b$.

In general, greatest common divisors need not be unique. All greatest common divisors of a and b are associates of one of them. For example: the only units in Z are 1 and -1. The greatest common divisors of 12 and 15 in Z are 3 and -3 and these are associates [5].

a) THE CLASSICAL EUCLIDEAN ALGORITHM:

The following algorithm computes gcd (greatest common divisors) in the arbitrary Euclidean domain.

Input: $f, g \in R$, where R is a Euclidean domain.

Output: gcd of f and g.

$r_0 := f, r_1 := g$

$i := 1;$

while $r_i \neq 0$ **do** $r_{i+1} := r_{i-1}$ rem r_i, $i := i+1$

return r_{i-1} .

For example:

Tracing the algorithm with given input $f = 1$ and $g =$; gives

i	r_{i-1}	r_i	$r_{i+1} = r_{i-1}$ rem r_i
1	126	35	21
2	35	21	14
3	21	14	7
4	14	7	0

b) THE EXTENDED EUCLIDEAN ALGORITHM

The classical Euclidean algorithm always works with the computes and normalized remainders and not only with the greatest common divisor, but also representation of it as the linear combination of inputs.

A function $normal($ is used. Then $R =$, $normal(a) = |$, and $R = F[$, $normal(a) = a/lc($, here lc is leading coefficient of (by convention, $lc(0) =$).

Input: $f, g \in R$, (R is a Euclidean domain with normal form).

Output: gcd(f,g), $s, t \in R$, such that $sf + tg$ = gcd(f,g), l as computed below.

$a_0 := lc(f)$, $r_0 := normal(f)$, $s_0 := a_0^{-1}$, $t_0 := 0$,

$a_1 := lc(g)$, $r_1 := normal(g)$, $s_1 := 0$, $t_1 := a_1^{-1}$;

$i := 1;$

while $r_i \neq 0$ **do**

$q_i := r_{i-1}$ div r_i

$a_{i+1} := lc(r_{i-1}$ rem $r_i)$

$r_{i+1} := normal(r_{i-1}$ rem $r_i)$

$s_{i+1} := (s_{i-1} - q_i s_i) / a_{i+1}$

$t_{i+1} := (t_{i-1} - q_i t_i) / a_{i+1}$

$i := i + 1;$

$l := i - 1;$

return $r_{i-1}, s_{i-1}, t_{i-1}, l$.

In this process the chosen techniques of classification are concerned on similar techniques for prediction and classification specifically in data mining.

The variables used in the algorithm are
: *Euclidean length* of the pair
: the quotients; : the remainders;

and from th iteration satisfy: for all . Particularly, for , the output of algorithm and satisfy They are known as the Bézout coefficients of and .

c) APPLICATIONS OF THE EUCLIDEAN ALGORITHM

Modular arithmetic:

In the modular arithmetic, one computes with the remainders arithmetic expressions mod some nonzero integer. Taking the number , two integers and are equal modulo if they have same remainder on division by .

, i.e. is divisible by .

The rule for computing with the congruences is:

where and is one of .

Using the rule inductively, the expressions will be computed modulo efficiently by reducing all the integers modulo , and then by step by step, performing the arithmetic expression in and by reducing the result modulo again. Likewise, intermediate results never exceed .

Modular exponentiation:

An important tool for the modular exponentiation is the repeated squaring (multiply and square). This technique will works in any set with the associative multiplication. Here, it is used in the residue class ring. The modular exponentiation is required for computing modular inverses. The following algorithm is used for modular exponentiation:

Input: $a \in R$, a ring with identity, $n \in N\ (n \neq 0)$

Output: $a^n \in R$.

Find the binary representation of $n = 2^k + n_{k-1} \cdot 2^k + \ldots + n_1 \cdot 2 + n_0,\ n_i \in \{0, 1\}$

$b := a$;

for $i := k\text{-}1, k\text{-}2, \ldots, 0$ do

    if $n_i = 1$ then $b := b^2 a$ else $b := b^2$

return b

This procedure uses squaring plus at most multiplications by (if binary representation of is all 1's).

II. CONCLUSION

It is concluded that modern algebra is one of the sections of computer science and it helps in analyzes, applies, designs and implements algebraic algorithms. Algebra involves various operations and functions to find the solutions. In that, groups are the basic building blocks of modern algebra [6]. Theories and algorithms in modern algebra help to many applications and it is used in various fields and industries. Modern Algebra is the powerful component with applications and algorithms throughout mathematics [7].

Modern algebra is mainly used in the computer programming to find the solutions for complex programs. In addition to these, modern algebra is used in other fields like physics, chemistry, and mechanics and so on. It can be clearly understood that, today modern algebra is almost used in all the major fields and industries. Several programming languages are using modern algebra to find the results for their complex programs. Almost all researcher are considering that systems of modern algebra has tend to generate the solutions fast and also without mistakes.

REFERENCES

1) Cunnington, Susan, (1904): *The Story of Arithmetic: A Short History of Its Origin and Development*, Swan Sonnenschein, London, 1904

2) Rithche D (1996): The development of C language, In History of Programming Languages, 2nd ed., ACM Press 1996.
3) Liska et al (1999): COMPUTER ALGEBRA, Algorithms, Systems and Applications, February 11, 1999.
4) Winkler F (1996): Polynomial Algorithms in Computer Algebra, Springer-Verlag, 1996.
5) Gathen and Gerhard (1999): Modern Computer Algebra, Cambridge University Press, 1999
6) Malik D S, Mordeson J N and Sen M K (1997), Fundamentals of Abstract Algebra, McGraw Hill, New York.
7) Herstein I N (1975), Topics in Algebra, 2nd edition, Xerox College Publ., Lexington, MA.