Biquinary decimal error detection codes with one, two and three check digits

D. A. H. Brown

Royal Radar Establishment, St Andrews Road, Malvern, Worcestershire WR14 3PS

The biquinary system of representing the decimal integers 0 to 9 is combined with polynomial coding to produce true decimal codes having any required number of check digits added to an integer of any length.

(Received July 1973)

1. Introduction

Modulus p systems, with p a prime number, can be used to add check digits to a decimal number (Wild, 1968; Briggs, 1970). Modulus 11 is often used but unless additional restrictions are introduced it requires the use of a special symbol to represent the decimal number 10, which may be unacceptable in practice. Various attempts have been made to circumvent this difficulty (Andrew, 1970; Campbell, 1970) but they restrict the length of the integer which can be coded. The extensive literature on polynomial codes (Berlekamp, 1968; Peterson and Weldon, 1972) can be applied to decimal systems by using polynomials from GF(11) (Brown, 1974) but these have the same disadvantages as ordinary modulus 11 systems.

A method for constructing true decimal codes based on polynomial codes for a biquinary system has recently been proposed (Brown, 1973) and this paper explores its application to decimal error detection.

2. Terminology for errors in decimal numbers

The following terms are used in this paper for the errors described.

1. Single error: a single digit is altered, $a \rightarrow b$
2. Insertion error: a single extra digit is inserted somewhere in the digit string.
3. Deletion error: a single digit is omitted
4. Transposition error: two adjacent digits are interchanged, $ab \rightarrow ba$
5. Double error: two unrelated digits are altered
6. Double adjacent error: two adjacent digits are changed in no special way, $ab \rightarrow cd$
7. Double repeated error: a pair of equal digits in adjacent positions are changed to another pair, $aa \rightarrow bb$
8. Multiple errors: error types (5), (6) and (7) can occur in more than two positions. A multiple error is a set of unrelated errors in unrelated digits. A multiple consecutive error is a set of errors spread over t consecutive digits (an error burst of length t). A multiple repeated error occurs when a sequence of repeated equal digits is copied with the same sequence length with a different digit, e.g. ....333.... becomes ....555....
9. Shift errors: are insertion or deletion errors associated with the special case of a repeated digit being copied either too few (right shift error) or too many (left shift error) times, e.g. 2655557 becomes 26557 (right shift error) or 265555557 (left shift error)

3. Biquinary decimal codes

The biquinary coding of decimal digits uses the correspondences

		0	1	2	3	4	5	6	7	8	9
Decimal		0	1	2	3	4	5	6	7	8	9
Code	Binary	0	0	0	0	0	1	1	1	1	1
	Quinary	0	1	2	3	4	0	1	2	3	4

which converts each decimal digit into an ordered pair of one binary digit and one quinary digit, e.g. 7 decimal is equivalent to (1, 2) biquinary. It is then possible to consider the data digits of the message as two digit streams, one a binary stream and the other a quinary stream. The standard theory of polynomial coding can be applied independently to these streams using a polynomial with binary coefficients in GF(2) for the binary stream and a polynomial with quinary coefficients in GF(5) for the quinary. The check digits so produced can be recombined by reversing the biquinary form to give true decimal digits. It is not necessary that there should be the same number of binary and quinary check digits and for error correction codes they will usually be different but in this paper we consider only error detection codes with an equal number of binary and quinary check digits.

4. Outline of polynomial coding theory

Throughout this paper discussion is restricted to bases which are a prime p so that the elements are the integers modulo p namely, $0, 1, \ldots, p-2, p-1$. The k message digits to be encoded are $m_{k-1}, m_{k-2}, \ldots, m_1, m_0$ and from these a message polynomial $M(x)$ is formed where

$$M(x) = m_{k-1}\,x^{k-1} + m_{k-2}\,x^{k-2} \ldots m_1\,x + m_0$$

A checking polynomial $g(x)$ of degree t is used where

$$g(x) = x^t + g_{t-1}\,x^{t-1} + g_{t-2}\,x^{t-2} \ldots g_1\,x + g_0$$

The data digits m_i and coefficients g_i are all members of the set of integers modulo p. The conditions governing the choice of the coefficients of $g(x)$ will be considered later. The t check digits to be added to the message are now formed by dividing $x^t\,M(x)$ by $g(x)$ to give a remainder $R(x)$ which is a polynomial of degree $t-1$ with t coefficients. All the arithmetic operations on the coefficients during this division are modulo p. These t coefficients are used to derive the t check digits but it is an advantage to use the complements of the coefficients of $R(x)$ taken modulo p as the actual check digits. The reason for this is that the division process can be represented as

$$x^t\,M(x) = g(x)\,Q(x) + R(x)$$

where $Q(x)$ is the quotient which is not used. If $R(x)$ is subtracted from both sides we obtain

$$x^t\,M(x) - R(x) = g(x)\,Q(x)$$

which shows that when the final message polynomial of k data digits plus t check digits is formed in this way it is exactly divisible by $g(x)$. This property is used at the receiver to check the received digits; if the remainder is zero then the message is accepted as correct.

The division process is easily implemented by a repetitive process which takes the message digits one at a time starting with the most significant m_{k-1}. Let R_i be the column vector $\{r_0, r_1, \ldots, r_{t-2}, r_{t-1}\}_i$ of t elements which are the t coefficients of the remainder $R(x)$ after $i-1$ message digits have been processed and let the ith message digit be m_i. Then R_{i+1} is formed by

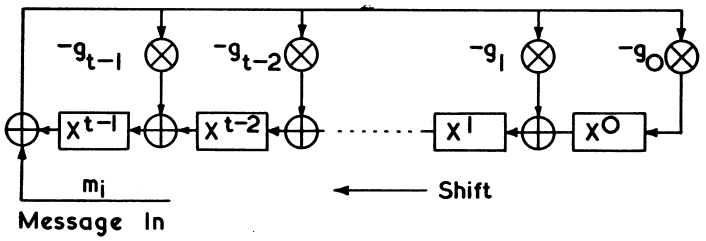

Fig. 1 Shift register

$$R_{i+1} = \begin{bmatrix} r_0 \\ r_1 \\ \\ r_{t-2} \\ r_{t-1} \end{bmatrix}_{i+1} = \begin{bmatrix} 0 & 0 \ldots 0 & -g_0 \\ 1 & 0 \ldots 0 & -g_1 \\ & & \\ 0 & 0 \ldots 0 & -g_{t-2} \\ 0 & 0 \ldots 1 & -g_{t-1} \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \\ \\ r_{t-2} \\ r_{t-1} + m_i \end{bmatrix}_i$$

This mathematical description is represented exactly by the shift register configuration of **Fig. 1.** In Fig. 1 the t-stage shift register holds the coefficients of $R_i(x)$. At the input to the jth stage of the shift register there is a modulo p adder with one input from the previous stage $(j-1)$ and its other input the product modulo p of $-g_j$ times the sum of m_i and the most significant remainder digit; g_j is the coefficient of x^j in the code polynomial. As a shift pulse is applied this forms R_{i+1}.

Once the formation of the check digits is understood as a polynomial division giving a remainder the chief error detection properties become obvious. Any error in the received message can be considered as the addition to the correct message of an error polynomial $E(x)$ multiplied by some power of x which simply shifts $E(x)$ to the appropriate part of the message. The important part of this error is $E(x)$. If $E(x)$ is not divisible exactly by $g(x)$ there will be a remainder after division and an error will be detected. If $E(x)$ is a polynomial of degree $t-1$ or less then it certainly cannot be divided by $g(x)$. This means that any error spread over t adjacent digits (an error burst of length t) will be detected with certainty by the code. For bursts of length greater than t some of the errors will be undetected but, assuming a 'random' set of errors, the chance of this happening is 1 in p^t which is the chance that a random $E(x)$ will be divisible by $g(x)$.

5. Decimal codes using one check digit

These are formed by using first degree polynomials. There is only one such binary polynomial $X+1$ and division of the binary digits by $X+1$ is a complex way of describing the formation of a simple parity check giving even parity over the message and single check digit.

There are four quinary polynomials of the first degree $X+1$, $X+2$, $X+3$, $X+4$. $X+1$ is rejected because it will not detect double repeated errors, type $aa \rightarrow bb$. The error $E(x)$ in this case is a polynomial of the form $c\,(X+1)$ where c is 1, 2, 3 or 4 and this is divisible by $X+1$. Likewise $X+4$ is rejected because it will not detect a transposition error which always has an $E(x)$ of the form $c\,(X+4)$. Either of the two remaining polynomials $X+2$ or $X+3$ can be chosen for the quinary check; neither seems to have particular advantages.

With then a parity check over the binary digits and a quinary check digit formed by using $X+2$ or $X+3$ as the check polynomial over the quinary digits a decimal code can be constructed which will have the following error detection properties over a string of decimal digits of any length. It will detect a single error with certainty and most double repeated and transposition errors. It will not detect double repeated errors of the form $aa \rightarrow 5+a$, $5+a$, e.g. $33 \rightarrow 88$ which alter both binary digits so leaving the parity check correct and leave the quinary digits unaltered. Similarly it will not detect transposition errors of the form a, $a+5 \rightarrow a+5$, a; this reverses the binary digits leaving the parity check unaffected and leaves the quinary digits unchanged.

A numerical example based on $X+1$ (binary) and $X+3$ (quinary) of a coding for 11 data digits is:

	data	check	polynomial
	2 6 0 3 5 7 9 2 8 2 8	1	
binary	0 1 0 0 1 1 1 0 1 0 1	0	$X+1$
quinary	2 1 0 3 0 2 4 2 3 2 3	1	$X+3$

6. Decimal codes using two check digits

There are two binary polynomials of the second degree X^2+1 and X^2+X+1. The first of these X^2+1 generates two parity checks, one over the odd digits and the other over the even digits. It will therefore detect all errors in the binary stream in two adjacent digits as well as three consecutive errors. In this it is superior to X^2+X+1 which will not detect three consecutive errors which correspond to $E(x) = X^2+X+1$. In the same manner X^2+2 or X^2+3 is a reasonable choice for the quinary check polynomial. Coding based on these choices will detect the following errors; (*a*) a single error; (*b*) a double repeated error; (*c*) tranposition error; (*d*) most errors in three consecutive digits except for certain special cases the derivation of which should be clear from the previous discussion.

7. Decimal codes using three check digits

The same principles can obviously be extended to three or more decimal check digits. The binary polynomial X^3+1 will generate three independent parity checks over the digits spaced three apart and it will detect most error bursts of length 4 in the binary stream except those corresponding to $E(x) = X^3+1$.

Any third degree quinary polynomial unlikely in itself to occur as an error polynomial can be used. X^3+3 is convenient to implement and is used here for a final numerical example of the same eleven data digits as in Section 5 but coded with three check digits.

	data	checks	polynomial
	2 6 0 3 5 7 9 2 8 2 8	1 9 5	
binary	0 1 0 0 1 1 1 0 1 0 1	0 1 1	X^3+1
quinary	2 1 0 3 0 2 4 2 3 2 3	1 4 0	X^3+3

8. Insertion, deletion and shift errors

Polynomial codes will not detect insertion or deletion errors with certainty. This is because the difference between the true and erroneous digits may be an error polynomial $E(x)$ which is divisible by the code polynomial. An example is:

$M_1(x)$ (correct)	1 4 4 4 5 7 3
$M_2(x)$ (erroneous)	1 4 4 5 7 3
$E(x) = M_1(x) - M_2(x)$	1 3 0 0 0 0 0

If the quinary check polynomial is $X+3$ this is still divisible by $X+3$ and so passes undetected.

In a similar way shift errors alter the block length of the coded numeral so that the difference, the error polynomial $E(x)$, has no special form and may be divisible by the code polynomial.

9. Comparison with previous codes

An attempt has been made to summarise the performance of various codes in **Tables 1 and 2** which refer to the use of one and two check digits respectively. In these tables an entry of x means that x per cent of the errors in that class will be detected so that an entry of 100 (per cent) represents certainty of detection. The entries are mainly the writer's interpretation of the

Table 1 Codes with one check digit

Error type	*Mod* 11 *Type* 1	*Mod* 11 *Type* 2	*Andrew* (1970)	*Biquinary*
Single	100	100	100	100
Insertion	100	91	100	87
Deletion	100	91	100	87
Transposition	100	100	<100	89
Double repeated	<100	0	<100	89
Double adjacent	<100	91	?	87

Table 2 Codes with two check digits

Error type	*Mod* 97 *Type* 1	*Mod* 97 *Type* 2	*Andrew* (1972)	*Tang and Lum* (1970)	*Biquinary*
Single	100	100	100	100	100
Insertion	100	99	100	100	99
Deletion	100	99	100	100	99
Transposition	100	100	100	100	100
Double Repeated	100	100	100	95	100
Double Adjacent	100	99	?	?	100
Triple Repeated	100	100	?	?	89
Triple Consecutive	99	99	?	?	99

various papers referred to and may therefore be open to dispute because, apart from claiming 100 per cent. coverage of certain errors, authors seldom seem to give exact probabilities for coverage of other errors.

The reference to type 1 and type 2 systems is taken from Wild (1968) who has described two principal ways in which modulus p (p is a prime) systems are used.

His description is:

Type 1: the maximum number of digits, k, in a code is predetermined and a weight w_i ($i = 1$ to k) is associated with each of the digits n_i in the code number. The sum $\sum_{i=1}^{k} (w_i n_i)$ is divided by p and the remainder is used as the check digit(s).

Type 2: the code, which from the check digit point of view may be of any length, is divided by p and the remainder is used as the check digit(s).

$p = 11$ gives a modulus 11 system with one check digit (but remainder 10 needs special treatment). $p = 97$ gives a modulus 97 system with two check digits. However, Andrew (1972) has devised a modified type 1 modulus 11 system with two sets of weights to generate two check digits.

It should be remembered that polynomial codes do not restrict the integer length and so they should more properly be compared with type 2 codes only.

These comparative tables cannot do justice to codes having special properties. An example is the extension made by Briggs (1971) to a type 1 modulus 97 system which will detect multiple transcription errors of identical digits in any position, e.g., 7273774 copied erroneously as 1213114. This also illustrates the importance of choosing a code with properties selected to detect the errors most likely to occur. It seems unlikely that a human operator would copy 7273774 as 1213114 but a faulty automatic reader might well consistently mistake one digit for another.

The figure of 87 per cent error coverage for insertion and deletion errors in the biquinary column is estimated as follows. There is a chance of one half that the error will be detected by the binary parity check alone. For those errors not so detected there is a $\frac{3}{4}$ chance that they will be detected by the quinary check. The total chance of detecting the error is therefore $0{\cdot}5 + 0{\cdot}5 * 0{\cdot}75 = 0{\cdot}875$. The biquinary system is slightly inferior to a type 2 modulus 11 system on insertion, deletion and transposition errors but gives 89 per cent. coverage of double repeated errors which are completely undetected by the modulus 11 system.

In this case of two check digits the biquinary system is slightly inferior to a type 2 modulus 97 system which is certainly easier to implement as a straightforward division by 97.

Although no work has been reported on three check digit systems it appears that a biquinary code would be slightly inferior to a modulus 997 type 2 system.

10. Implementation of biquinary coding

For only a few check digits generated by polynomials of low degree there is no difficulty in the matrix multiplication of Section 3 for obtaining R_{i+1} from R_i.

10.1. *One check digit*

Parity formation over the binary digit stream is simple and only the quinary check calls for comment. If $X + e$ is the quinary check polynomial and q_i is the quinary check digit value when message digit m_i is to be processed then, with $f = 5 - e$,

$$q_{i+1} = f * (q_i + m_i) \text{ modulo } 5$$

Because f is a known constant this expression can be evaluated conveniently by setting up a 5×5 integer array giving q_{i+1} for all input combinations of q_i and m_i. Then a single ALGOL 60 statement $q :=$ array $[q, m]$ gives the new value of q. Using this array the following ALGOL 60 procedure gives the new values of the binary check digit b and quinary check digit q when message digit m is processed.

```
procedure next remainder (m, b, q);
value m; integer m, b, q;
begin if m ⩾ 5
  then
  begin b := if b > 0 then 0 else 1;
    m := m − 5
  end;
  q := array [q, m]
end next remainder;
```

10.2. *Two and three check digits*

Consider first the case of two check digits. The formation of binary parity checks over odd and even digits using $X^2 + 1$ is simple. For the quinary checks it is convenient in the general case to set up two 5×5 arrays called sum and times such that sum $[i, j] = i + j$ modulo 5 and times $[i, j] = i * j$ modulo 5 where i and j take the values 0, 1, 2, 3, 4. Let the quinary checking polynomial be $X^2 + cX + d$ and let e and f be the complements modulo 5 of c and d respectively. Let $b0$ and $b1$ be the two binary check digits and $q0$ and $q1$ be the two quinary check digits and let m be the message digit to be processed. Then an ALGOL 60 procedure to form the new digits is:

```
procedure next 2 remainder (m, b0, b1, q0, q1);
value m; integer m, b0, b1, q0, q1;
begin integer k, l;
  k := b1; b1 := b0; l := q1;
  if m ⩾ 5
  then
  begin b0 := if k > 0 then 0 else 1;
        m := m − 5
  end
  else b0 := k;
```

```
    k := sum [l, m];
    q1 := sum [q0, times [e, k]];
    q0 := times [f, k]
  end next 2 remainder;
```

If a quinary check polynomial of the form $X^2 + d$ is used the procedure can be simplified further. Then the next $q0 = f^*(q1 + m)$ and this can be found directly from the same type of 5×5 integer array as was used in the one digit case.

These methods can obviously be extended quite easily to three or more check digits.

11. Conclusion

Biquinary coding is a systematic method of constructing true decimal error detecting codes with as many check digits as may be required over integers of any length. The error detecting properties can be deduced from the method of formation of the check digits as a polynomial division process.

References

ANDREW, A. M. (1970). A Variant of Modulus 11 Checking, *The Computer Bulletin*, Vol. 14, No. 8, pp. 261-266.

ANDREW, A. M. (1972). Decimal Numbers with Two Check Digits, *The Computer Bulletin*, Vol. 16, No. 3, pp. 156-159.

BERLEKAMP, E. R. (1968). *Algebraic Coding Theory*, McGraw-Hill.

BRIGGS, T. (1970). Modulus 11 Check Digit Systems, *The Computer Bulletin*, Vol. 14, No. 8, pp. 266-270.

BRIGGS, T. (1971). Weights for Modulus 97 Systems, *The Computer Bulletin*, Vol. 15, No. 2, p. 79.

BROWN, D. A. H. (1974). Some Error Correcting Codes for Certain Transposition and Transcription Errors in Decimal Integers, *The Computer Journal*, Vol. 17, No. 1, pp. 9-12.

BROWN, D. A. H. (1973). Construction of Error Detection and Correction Codes to any Base, *Electronics Letters*, Vol. 9, No. 13, 28th June, p. 290.

CAMPBELL, D. V. A. (1970). A Modulus 11 Check Digit System for a given system of Codes, *The Computer Bulletin*, Vol. 14, No. 1, pp. 12-13.

PETERSON, W. W., and WELDON, E. J. (1972). *Error Correcting Codes* (2nd Edition), MIT Press.

TANG, D. T., and LUM, V. Y. (1970). Error Control for Terminals with Human Operators, *IBM Journal of Research and Development*, Vol. 14, No. 4, pp. 409-416.

WILD, W. G. (1968). The Theory of Modulus N Check Digit Systems, *The Computer Bulletin*, Vol. 12, No. 8, pp. 309-311.

Book reviews

Optimisation Techniques with FORTRAN, by J. L. Kuester and J. H. Mize, 1973; 500 pages. (*McGraw-Hill (UK)*, £2·90)

This book covers a wide variety of optimisation techniques and would be useful for teaching purposes as a supplementary or reference text. Unfortunately, the description of each technique is absolutely minimal so a further text book on the subject would be essential. This lack of detail would also be a grave handicap for research workers and others who attempted to use it as a 'handy reference book', as suggested by the authors.

The book is divided into two parts: Part One—Special Purpose Methods—includes algorithms for linear programming, quadratic programming, geometric programming, dynamic programming and for problems whose objective functions are sums of squared terms; Part Two—Search Methods—deals with the general non-linear problem and is subdivided into chapters on single variable unconstrained, single variable constrained, multivariable unconstrained and multivariable constrained optimisation methods. Each technique is described in five sections, namely, Purpose, Method, Program description, Text problem and Program listings with example output; the programs are coded in ASA FORTRAN, apart from one subroutine in IBM 360 Assembler language.

To a certain extent, one of the main reasons given for writing the book—the difficulty of obtaining computer programs for optimisation techniques—is not valid in Great Britain, owing to the ready availability of copies of routines from the Harwell library, from the Numerical Analysis and Computing Division of the NPL, from the Hatfield Optimisation Centre and from the CERN library; in addition, many universities now have access to the NAG library. The latter, in particular, provides more efficient and up to date routines for many of the topics covered here (particularly those in Part Two) in place of the rather limited versions of some programs included in this text. Some of the programs given are very close to the original versions without any acknowledgement being given to the person who provided the original coding—the routines for Powell's sum of squares method, pages 258-269, even have the same variable names and statement numbers as the original Harwell versions, VA02A and VD01A, although some comments and additional statements have been added to the former.

The choice of algorithms does not always reflect those which are currently considered the best available, particularly in the case of non-linear techniques. It would appear that the authors are somewhat unfamiliar with this area—they describe the methods as being available for small problems with typical limits on equations and variables being less than one hundred—this is certainly not always the situation; for example, the Fletcher Reeves algorithm has been applied with success to problems having of the order of a thousand variables, and some recent programs developed by Murray and Gill at the National Physical Laboratory have also been applied to large scale problems.

To summarise, this book could be fairly useful for teaching purposes in that it provides readily available algorithms for course work use, but has only limited application for other purposes.

HEATHER M. LIDDELL (London)

Elementary Numerical Analysis, by Conte de Boor, second edition, 1973; 396 pages. (*McGraw-Hill*, hard cover £4·80, paperback £2·70)

The book gives an introduction to computer arithmetic, computational linear algebra, the solution of non-linear equations, interpolation and approximation and the numerical solution of initial and boundary value problems for ordinary differential equations. The approach is, as the title indicates, algorithmic. For each of the topics discussed algorithms are presented in an ALGOL-like language and FORTRAN listings are given. Where appropriate, flow diagrams are included. The algorithms are clearly explained and the discussion generally includes the background Mathematics necessary for an understanding of the construction and analysis of the algorithms. Thus, for example, the chapter dealing with the solution of linear equations contains an introduction to matrix algebra and matrix norms. The limitations of the algorithms described are generally indicated and reference made to more powerful algorithms.

Evidently the author's intention is to give the reader an understanding of the basic ideas and techniques of numerical methods, and the fact the methods described are not always the best currently available does not detract from what is certainly an excellent introduction.

M. J. M. BERNAL (London)