MODULO CHECKING

Report R-4

June 1951

Prepared by:

S. M. Dancoff

M. Gold-Hearn

CONTROL SYSTEMS LABORATORY
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS

Contract DA-11-022-ORD-174
In coding for computing one desires to add several digits, called the **signature**, to each binary coded number. This signature is extracted by the machine itself from the number and inserted at a provided location. The signature has the following properties:

1) Any change in a single digit of the number produces a different signature.

2) In addition, the sum of the signatures is the signature of the sum.

3) In multiplication, the product of the signatures is the signature of the product.

4) If a single mistake is made either in addition or multiplication the signature no longer agrees with the result.

The type of signature that will be discussed here is the number system based on modulus of any odd number. The simplest case to discuss, which will illustrate most of the features, is the mod three system. Two extra digits are added and in these the number is inserted mod three. Any single change in the full number represents the addition or subtraction of a power of two. This must change the mod three number and therefore change the signature. So condition (1) is satisfied.
The validity of (2) and (3) is known from elementary number theory. The sum or product of numbers given for a certain modulus is the same as the corresponding modulo expression for the sum or product of the full numbers.

Condition (4) is trivial for addition. Any single error always produces a change by a power of two, thereby producing disagreement with the signature.

Condition (4) is less straightforward for multiplication. If either number has signature zero then any arbitrary error may be made which involves changes in the second number only. The signature of the product would remain zero and the error would be undetectible.

To guard against this exceptional case the simplest method is to avoid signature zero completely. If it ever appears upon the extraction of the signature the machine is instructed to add one digit in the position furthest to the right, thereby increasing the signature to one and modifying the actual number by a very tiny percentage.

A scheme for the extraction of the signature with modulus three is the following:
The pair of squares on the right is the signature register. The remaining squares register the actual number (information register); only the right hand end of this register is shown.

Digits are designated $a, b, a, b, a, b, \ldots$ etc., starting from the right. All the $a$ digits (class $a$) are connected to the $a$ digit of the signature register; likewise for the $b$ digits (class $b$). The number is then scanned and searched for ones (The direction of the scanning makes no difference). Each time a one is encountered it is added into the signature register at the indicated point. When all the ones have been so added the signature register gives the number mod three as required.

During this series of additions the ones are added into the signature register in a normal fashion with carry-over; carry over from the $a$ digit goes into $b$, while carry over from the $b$ digit goes into $a$.

If one is willing to devote three digits to the signature register some considerable improvement in checking can be
gotten by going to a mod 7 system. The general discussion is the same as for modulus 3, but the exceptional case of zero mod 7 occurs statistically less frequently, making necessary less tampering with the numbers on this account. The signature is extracted as follows:

The digits are divided into classes \(a, b, c\), as shown and scanned for ones. These are then added into the signature register as indicated, with carry-over from \(a \rightarrow b; b \rightarrow c; c \rightarrow a\).

The extraction of the signature for higher moduli of the class \((2^n-1)\) is a straightforward generalization of this scheme.

The mod 7 system has the advantage over mod 3 that it offers a considerable amount of checking against two and multiple errors.

Two errors could leave the signature unchanged only if they added zero-mod 7 to the signature. If the two errors
were both of the same sign this couldn't happen at all, as one sees from a brief study of the above system whereby the signature mod 7 is extracted. Two errors of the same sign must produce a changed signature and are therefore detectible.

Two errors of different sign (one into zero in one digit, zero into one at another) will almost always be detectible too. The exception is the case when both occur in the same class (a, b, or a); then the signature will be unchanged by the errors.

Given that there are two errors, the probability that they will fall in the same class is (1/3). The probability that the digits were originally different—one a zero the other a one—is (1/2). Thus the fraction of the time that two errors are undetectible is $\frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$.

Higher numbers of errors will usually be detected. A likely generalization is that of all possible multiple errors, only 1/7 will escape detection since these represent the proportion of cases when zero mod 7 has been added or subtracted.

So far we have seen how to detect one error for certain, and higher numbers of errors with high probability. Our system so far is incomplete in two respects, even as far as the detection of a single error is concerned.

1) It makes no provision for locating and correcting this error.

2) It assumes that errors may occur only in the information register, not in the signature register.
We will now show how to remove these defects.

Let us define the rank of a signature as the number of binary digits, or number of classes, it contains. Thus the mod 3 system has rank 2, the mod 7 system has rank 3. In general, mod \(2^n-1\) has rank \(n\).

What could we gain by using two signatures of different rank? Let us call them left and right signatures. Consider the case where a single error is present in the information register. One now finds that both left and right signatures indicate the presence of an error. Now proceed along the information register, testing the effect of changing each one into a zero and v.v. The left register will be brought into agreement when the digit is reached where the error resides. But it will also be brought into agreement when the corresponding change is made at any other digit of the same class as the guilty one.

Likewise the right signature is brought into agreement when the faulty digit is corrected; but it is also brought into agreement if a corresponding change is made elsewhere, but in the same class.
The two signatures will simultaneously be brought into agreement when the error is corrected; they could also simultaneously be brought into agreement if any other digit existed which was in the same class for the left signature and for the right signature simultaneously. However, if the lowest common multiple of the two ranks is greater than the number of digits in the information register, then no such second location exists. The two signatures will simultaneously agree with the number when and only when the single error is corrected.

Thus we have a procedure for locating and correcting errors in the information register. It requires a total expenditure of signature digits of order $2^N$, where $N$ is the number of digits in the information register.

Consider now the case where the single error is located in one of the signatures. Then the situation consists of agreement between the number and only one of its signatures. But we know that no single change in the number can leave it in agreement with either of its signatures. Therefore this situation is invariably and uniquely interpreted as an error in the one signature that is not in agreement. The instruction is then to erase this signature and extract a new one to replace it.

Therefore we have now a complete system for locating and correcting a single error, whether in information or signature digits.
The system has the further property that two errors are always detected. If both errors are in the information register they can escape detection only by being in the same class and in opposite direction. But, as pointed out before, there exists no pair of locations which are in the same class for both left and right signatures.

If one error is in the information register and the other in one of the signatures, then the remaining signature will disagree with the number.

If both errors are in one of the signatures, this signature is changed and brought into disagreement (an exceptional case is that of mod 3 when the addition of one in each digit leaves the signature unchanged.)

If one error is in each signature register, then each will disagree with the number.

Therefore we see that, except for a special situation in the case of mod 3, two errors are always detected.

Higher numbers of errors will generally be detected. The error in the number must be a multiple of the ranks of left and right signatures. Only in this way will both signatures be left in agreement. Roughly speaking the fraction of the time this situation will arise is the reciprocal of the product of the two ranks.

An example of such a system would be a 41 digit information register with signature registers of rank 6 and 7 respectively.
The same general considerations would apply if one wished to consider three (or more) signatures. The basic condition is that the lowest common multiple of the ranks is greater than the length of the information register. Three signatures, as compared with two, would allow some saving in the number of digits carried and some added protection against triple errors, but at the expense of more circuitry.
A. Rounding off.

If there are $M$ digits in the information register then a product of two such numbers will in general result in a number with $2M$ digits which may be different from zero. The product signature is applicable to this number of length $2M$. In the course of a calculation it is usually required to cut down this expanded number to a length of $M$ once again, keeping only the first $M$ significant digits. Whatever the detailed character of the round off instructions furnished to the machine, it is clear that corresponding instructions must be given to the signature extracting system, so that the contracted number and its signature are in agreement.

Before discussing how this adjustment of signature is achieved, we will describe the process of transposition of the signature. The problem is the following: we have an $R$ digit number and its signature is known. It happens that the right hand digit of the number is zero. We want also to consider the $(R-1)$ digit number formed by selecting the right hand zero. How is the signature of this contracted number related to that of the original one?
To put it more conventionally, we want to know the remainder upon division of a number by a certain divisor, given only that the remainder would be for twice the number and the same divisor. In general this problem is not soluble. For our special class of divisors (moduli), however, the solution is remarkably simple.

The property of interest is the cyclical relationship among the digits of the signature. It is probably most appropriate to draw the signature register in circular form as shown in the figure. The case illustrated is for a signature of rank eight. Each digit is connected to all digits of the same class in the information register, as previously described.

The **a** class contains the digit furthest to the right. In adding, carry over goes in the sense **a** → **b**; **b** → **g**; etc., up to **g** → **b**; **b** → **a**.

We now imagine the signature extracted for the **R** digit number, according to the procedure of the earlier note. Now the right hand zero is lopped off and the signature extracted for the (**R-1**) digit number. It is clear that the only new thing
that has happened is that digits which were in class b in the
first case have been relabelled class a; class a digits have
been relabelled class h, etc. The numerical calculation
of the signature proceeds altogether unchanged from one case
to the second, except for this transposition of the signature
by one unit.

To state the rule: Given the signature of an R digit
number whose right hand digit is zero. To obtain the signature
of the (R-1) digit number with the zero deleted, simply trans-
pose the signature by one unit clockwise (b → a; c → b; etc.).

Let's return now to the problem of the contraction of
the 2M digit number obtained on multiplication. The process
can be pictured in the following steps:

1. All the ones are changed into zeros in the right hand,
or "expendable", M digits. Each time a one is so changed the
instruction must be to subtract * one in the signature register
at that digit which represents the class of the one which was
changed. At the end of this procedure one has M significant
digits followed by M zeros, the signature being correct for the
whole.

---

* We will not go into detail about the nature of the subtraction
process as applied to the signature. Subtraction is pro-
grammed differently in various existing computing systems,
and each system would require a different subtraction pro-
gram for the signature also. We note however, that subtrac-
tion is really nothing more than an implied addition, that it
could in principle be carried out without ever doing anything
except addition. We therefore feel free to talk about sub-
traction in the signature register without further discussion.
II The expendable M zeros are now detached and the resulting contracted number dealt with. What is the signature of this number? What we have here is nothing but a series of M transpositions, as discussed above. The new signature is just the old one transferred cyclically M times.

III It may or may not be required to add a one to the right hand digit of the contracted number. If this is required then a corresponding one should of course be added to the signature in the appropriate class.

As outlined above the steps appear to require the display of the full 2M digit number at some point in the proceedings. Actually this is by no means necessary. The expendable digits can be lopped off one by one as the multiplication progresses, with appropriate subtractions in the signature register as described in (I).

In practice one would of course not perform as many as M transpositions, since a number of transpositions equal to the rank leaves the signature unchanged. All such complete cycles of transpositions would not be carried out; only the remainder in excess of complete cycles need be performed. Stated otherwise, the number of transpositions to be performed is M mod (rank). If M happens to be a multiple of the rank, then no transposing at all is necessary. Still another way to state the case is the following: Consider the digit that will become the right hand digit of the number after contraction. This digit falls in a certain class α. The required
operation is to rotate the signature register (as pictured in Fig. 1) until the digit originally in the \( a \) position moves into the \( a \) position. The rotation may be in either sense.

B. **Multiple error detection**

So far we have paid particular attention to the case where only an error in a single digit was to be anticipated. We have shown how to locate and correct this error and, moreover, how to detect whether two errors were present. Our system therefore seems designed to deal especially with cases where the probability for error is very small, the occurrence of one error is rare, and the occurrence of more than two quite negligible.

It is well to question whether this preoccupation with small numbers of errors is justified. In the first place, even though errors are rare, their occurrence may not be statistically independent; they may tend to occur in batches, each batch associated with some transient weakness in a circuit element.

In the second place a single error need not be manifested only at a single digit. Many kinds of single errors (as far as circuit operation is concerned) will simultaneously change a large number, perhaps most, of the digits. This is particularly the case for single errors which occur in the instructions themselves.

But finally, a more general or strategic question is involved. It is true that if we check at sufficiently narrow intervals during the course of a calculation then we can make
the occurrence of errors as rare as we like between check points. A checking system based on rare errors could then be made to work. The disadvantage would be that the amount of machine effort expended in checking alone might become a larger fraction of the total effort that we would like to devote.

On the contrary, as we expand the interval between check points, we make more and more efficient use of machine time. The expected probability of error (one or more) increases meanwhile, but if our checking system is prepared to deal with multiple errors also, then the optimum condition may well be found to lie toward the wide interval side.

It is this last question of strategy in programming of checks which interests us particularly in this section.

First it is to be pointed out that the modulus check system gives powerful protection against any arbitrary error or combination of errors. Imagine an arbitrary disarrangement of the information register and of the signature register as well. What is the probability that the disarranged signature will agree with the disarranged number? Since my signature has equal a priori probability, its expectation of chance agreement is \(1/(2^R - 1)\), where \(R\) is the rank.

When we use two independent signatures of ranks \(R_1\) and \(R_2\), the probability for chance agreement among all three is

\[
\frac{1}{(2^{R_1} - 1)} \cdot \frac{1}{(2^{R_2} - 1)}
\]

When dealing with ranks of 5 or higher, these formulae can be approximated by the simpler one \(2^{-s}\).
which is now the probability for chance agreement, where $S$ is
the total number of signature digits, whether located in one
signature or several. With 13 signature digits, an arbitrary
error would escape detection not oftener than once in
ten thousand, on the average.

For purposes of definiteness in discussion we will set
down rough estimates of some significant constants for a typical
high speed computer.

| Time for one elementary operation: | $10^{-7}$ sec. |
| Operations per multiplication: | $10^4$ operations |
| Multiplications per problem: | $10^7$ multiplications |
| Time per problem: | $10^4$ sec. |
| Operations per problem: | $10^{11}$ operations |

Operations devoted to addition are not included since they
will be much fewer than those devoted to multiplication, if the
numbers of additions and multiplications are comparable.

If this machine operates without checking, depending for
the accuracy of the result upon the fidelity of each elementary
operation, then it follows that a fidelity of $10^{-13}$ (one error
in $10^{13}$ elementary operations) would lead to a false final answer
one percent of the time.

Let us compare now the situation in which a modulus check
is used. Assume that the numbers dealt with are of length 27
binary digits and the signatures of length 13 digits (whether
in one or more signatures). Thus a third of the arithmetic register
is filled with signature digits. The length of time, or number
of operations, devoted to arithmetic is about the same as if all 40 digits were information and no signature digits were used. (The figure of $10^4$ operations is estimated to apply to multiplication in a system of 27 plus 13 signature digits, with rounding off as described above.) What one has done is to decrease the number of significant figures carried from about twelve, on the decimal scale, to about eight.

This last concerns (a) the amount of space appropriated by the signature on the arithmetic register and (b) the amount of extra time necessary in arithmetic because of the presence of the signature digits. Similar considerations must now be made concerning the memory and general programming.

The memory must contain instructions for extracting the signature, for making digit-by-digit comparison at intervals between the signature being carried along and that newly extracted, and for the procedure to be adopted in case this latter comparison fails to show an agreement.

We will not attempt to estimate how much of the memory will be tied up in these functions. Because of the simplicity of extraction of this kind of signature we speculate that the amount will be small percentage wise. The amount of memory involved will depend largely on the nature of the instructions devoted to the case where the signature check shows a disagreement. The simplest possible way to deal with this case is just to instruct the machine to return to the last previous check point and repeat the calculation from that point. This would mean, for example, that in a given multiplication the
multiplier, multiplicand, and the instruction to multiply these two numbers are not erased from the machine until after the product has satisfied the modulus check.

To use the modulus check in this latter way would mean to give up the feature discussed in the earlier note whereby errors at single digits can be rectified without reference to any information except the output number and the output signature. It would, however, emphasize the ability of the modulus check to detect (without correcting) any arbitrary error or combination of errors. Whether the modulus check should be used so as to exploit one or both of these properties is a question that may not be answerable without reference to specific details in each special case. For the rest of this note we are going to assume, for definiteness, that the correcting of errors at single digits is to be ignored; that the modulus check is to be used altogether as a means for detecting the presence of arbitrary errors; and that any detection of an error is followed by the instruction to return to the problem as constituted at the last previous check point.

We are left now with the question of the extra time, or extra number of operations, over and above the arithmetic operations, necessary because of the check system.

We estimate that to extract the signature of a number in this system requires 500 operations. Suppose that signature checks are made after every addition and multiplication throughout the problem. If we include the time necessary to carry out these checks we arrive at an estimate of several times $10^{10}$ additional
operations per problem. This does not include the operations involved in repetition when the test is not satisfied, however these will be comparatively few since the overwhelming majority of the individual additions and multiplications will carry no error.

Consequently the number of machine operations has been increased by perhaps ten to thirty percent, a modest amount. Spacing the checks more widely will decrease the number of machine operations and make it approach more closely the value for unchecked operation.

Now what about the residual error in a machine equipped with a modulus checking system? Residual errors arise from two causes:

1) The modulus check has a limited efficiency. As discussed above there is, with 13 signature digits, a probability of $10^{-4}$ that an error will be such as to leave the modulus check satisfied. This error therefore goes undetected.

2) The modulus check is itself subject to error. If there are about $10^3$ operations in each check and if each operation has a chance, $p$, to be incorrect, then the entire check will be wrong with a probability $10^3 p$. This will have the result in the first place, that perfectly valid multiplications and additions are rejected and caused to be repeated; this aspect is unimportant if $p$ is reasonably small. In the second place, when actual mistakes in arithmetic are present, there is superimposed on this a probability of $10^3 p$ that the check is
incorrectly made. Now it does not follow that in this fraction $10^3 \ p$ of the cases, a false answer is mistakenly interpreted as a correct one; in fact there is, in turn, only a very small fraction of $10^3 \ p$ in which this will happen. Without analyzing this case too closely, it is sufficient for our purposes to say that this probability of accepting a mistake on checking is not smaller than, and probably of the order of $p$ itself. This lower limit is arrived at by noting that the checking system must at some point pass the final decision (yes or no) that the multiplication is in error. Since no element in the system can perform a single act with reliability better than $p$, this figure must serve as the lower limit of the probability for falsely approving a mistaken result. Other simple considerations show that this limit must be fairly closely approached for the kind of checking scheme being studied.

We have pointed out that an elementary error rate of $10^{-13}$ per operation, with unchecked performance, yields an overall probability of one percent for the answer to be wrong. With the checking system described here, the overall error probability is

$$\left[10^4 \ p\right] \left[2^{-3} + p\right] \left[10^7\right]$$

(1)

The first factor is the probability for error in a single multiplication. The second factor is the probability for this error to go undetected, due to causes (1) and (2) above. The member $2^{-3}$ amounts to $10^{-4}$ for 13 signature digits; the member $p$ has
been only very roughly estimated, as discussed above. The
third factor is the number of multiplications per problem.
If \( p \) is much less than \( 2^{-8} \), then the overall error probability is
\[
(10^{11} p)(2^{-8})
\]
One sees that this probability has simply been reduced by the
factor \( 2^{-8} \) as against the unchecked case. Thus the checked
machine can operate with \( p = 10^{-9} \) and be just as reliable
as was the unchecked machine with \( p = 10^{-13} \). Alternatively
if \( p = 10^{-13} \) is retained, the checked machine is in error only
\( 10^{-6} \) of the time, as against \( 10^{-2} \) for the unchecked.

We have included only the errors in multiplication, since
those in addition and in signature extraction will be fewer,
the number of operations involved being fewer. We have imagined
that a check is made after each multiplication. But if the check
were made after, say, 10 multiplications the only effect would be
to increase the first factor to \( 10^5 p \) and decrease the last one to
\( 10^6 \), leaving the overall error probability unchanged. In fact
one could wait a thousand, or ten thousand multiplications be-
fore checking (if \( p = 10^{-9} \)) with no loss of reliability. Under
these conditions the number of extra operations devoted to
signature extraction and checking is utterly negligible.

One should not, however, increase the interval between
checks indefinitely. If \( N \) is the number of multiplications be-
tween checks then the number \( 10^{11} pN \) measures the probability
of error in the interval, or more exactly
\[
\exp \left( -10^4 pN \right)
\]
is the probability to avoid error. One desires to keep the ex-
One may appreciably less than one to avoid excessive repetitions. If one fixes the allowable repetition probability at 10^{-5}, then the corresponding number of multiplications between checks is

\[ N_{\max} = \left( 10^{-5}/p \right) \]

A possible elaboration is to have the machine monitor itself to the extent of noticing how frequent the repetitions due to error are. As the parts wear out and the error rate becomes larger this would be indicated by an elevated repetition rate. The response would then be to reduce \( N_{\max} \) until the average repetition rate again reached a tolerable value.

To summarize: One has been able to maintain the reliability of machine operation unchanged while relaxing by a factor of 10^{-4} (for the parameter chosen) the requirement on fidelity of operation of individual parts. For this one has paid practically nothing in machine time, but one has reduced the number of significant figures carried from 12 decimal digits to 8 decimal digits.

A most interesting question is the one of how far one can go in relaxing the requirement on \( p \) while maintaining successful operation. Referring to formula (1), one would like to increase \( p \) while maintaining the product unchanged. This will be possible up to a point by increasing \( s \), the number of signature digits. The limiting condition will be reached when the second member of the second bracket equals the first. A quantitative solution for this case is not here possible because the second member is
not rigorously equal to $p$, but only of that order. Nevertheless we should get an approximate solution by solving the equations

$$2^{-8} = p$$

$$p \cdot 2^{-8} = \frac{1}{2} \left[ 10^{-9} \right] \left[ 10^{-4} \right]$$

or $p = 2^{-8} = 2 \times 10^{-7}$

Thus one has been able to relax the reliability requirement by still another factor of 200, or $2 \times 10^6$ altogether. One now has a total of 23 signature digits. This leaves 17 digits for information if one restricts himself to a total of 40 digits. This represents just 5 decimal digits, which may be insufficient for some problems, particularly in view of propagation of round-off errors. Adding 10 more binary digits, or 3 more decimal digits, to the information register should give enough significant figures for practically all purposes; this addition would not essentially change the numbers derived above.

C. **Checking of non-arithmetic information**

At any instant the machine has in it a sizable fraction of information which is not numerical, but consists of the orders

---

+ This degree of relaxation is to be considered a limit only if one requires that the same value of $p$ apply to all components of the machine. The critical factor is the second member of the second factor of (1). The numerical value of this factor is extremely sensitive to the reliability of a small number of components near the output stage of the modulus checking part of the machine. If these components are of high fidelity, then considerable further relaxation of fidelity requirement can be achieved for the overwhelming majority of the machine components.
necessary to carry out the numerical operations. What about the checking of such instructions?

Of course such orders are not compounded one with another according to the rules of addition or multiplication. They may be transferred bodily from one point in the machine to another. Or they may be used in the direction of arithmetic operations and then returned to storage for future use. Or, finally, they may be used and discarded.

When an order is transferred bodily, or when it is recovered and stored for future use, the modulus check is just as efficient in detecting errors in the transferred or recovered order as it is in detecting arithmetic errors. Since the order has been coded in a digital fashion it is for all practical purposes a number. The signature of this number may be extracted according to the methods described and signature checks instituted at appropriate intervals.

When the order is used up and discarded, mistakes made in its transference may escape direct detection. To circumvent this one may wish to recover the order and apply the signature check to it before discarding it.

In spite of the fact that signature checking of orders is feasible, it seems probable that this should not always be done and that many orders should carry no signature. For the purpose of this discussion, let us divide the orders into two categories which we will call "strategic" and "tactical". The strategic orders are the kind which specify what arithmetic operations are to be
performed, and when. These orders would be retained intact if the problem was recoded for an analogue computer. The tactical orders are the ones specific to the machine in question; they describe in detail how the arithmetic steps are accomplished.

The tactical orders tell step-by-step how two numbers, and their signatures, are to be added or multiplied. An error in such an order results in (a) a false product or sum of the two numbers (b) a false product or sum of the signatures, or (c) both. The situation here encountered is therefore essentially a mistake in arithmetic; it can be classed together with the mistakes in arithmetic considered earlier, in which the orders were not to blame. Since the modulus check based on the signatures carried by the numbers themselves can (with high probability) detect any such error, it should not be necessary to institute any further check to find out whether the order was correctly given.

One anticipates that most of the machine time, by a large fraction, is consumed in carrying out tactical operations rather than strategic ones; that is, the time spent in multiplying numbers is much greater than the time spent in selecting them out of the memory. Thus to be relieved of the necessity for checking tactical orders means that only a little additional machine time is involved in order checking, namely the time sufficient to check the strategic orders. This latter checking, however, cannot be avoided since an error in a strategic order would not be equivalent to a mistake in arithmetic.