# COMPOUND ALGORITHMS FOR DIGIT ONLINE ARITHMETIC

Robert Michael Owens

Department of Computer Science
The Pennsylvania State University
University Park, Pennsylvania 16802

## Abstract

This paper describes a systematic method which has been successfully used to create several digit online algorithms. Basically, the method entails converting in a systematic way a known continued sums/products algorithm and combining the converted form of the continued sums/product algorithm with a generalized digitization algorithm. Not only does the method seem to have wide applicability in the creation of digit online algorithms for many elementary functions but the algorithms which have resulted from this method themselves have several desirable properties.

## Introduction

Continued Sums/Products, CS/P, algorithms are a family of iterative algorithms which generate an increasingly better approximation to either the additive or multiplicative inverse of each independent variable and, thereby, generate an increasingly better approximation to each dependent variable[8]. CS/P algorithms can be used, in a natural way, as the basis for the implementation of segmented arithmetic units (e.g. pipelines)[9]. Since the segmented arithmetic units so implemented possess several desirable properties, CS/P algorithms are an important family of iterative algorithms. Furthermore, since CSP/P algorithms have been formulated for addition, subtraction, multiplication, division, logarithm, exponentiation, sine, cosine, hyperbolic sine, and hyperbolic cosine[3,2,11,12,16,4,5], CS/P algorithms enjoy wide applicability.

Digit Online Continued Sums/Products, DOCS/P, algorithms are yet another family of iterative algorithms. DOCS/P algorithms differ from CS/P algorithms in that each independent variable is supplied to and each dependent variable is generated by the algorithm in a digit online manner. That is, at the j'th iteration of the algorithms, only the first (j + k) digits of each independent variable must be supplied to the algorithm, where k is a small predefined constant. An algorithm which possesses this property is digit online with respect to its input(s). Also, at the j'th iteration at least the first (j - $\hat{k}$) digits of each dependent variable has been generated by the algorithm, where $\hat{k}$ is a small predefined constant. An algorithm which possesses this property is digit online with respect to its output(s). The sum of the constants k and $\hat{k}$ corresponds to the digit online delay of the algorithm as a whole. Furthermore, these constants are dependent for the most part on the specific algorithm and number system being used.

DOCS/P algorithms can be used, in a natural way, as the basis for the implementation of segmented arithmetic units which possess a similar digit online property. Since this property implies that segmented arithmetic units with high throughput and minimal online delay can be implemented, DOCS/P algorithms are an important family of iterative algorithms. While DOCS/P algorithms have been formulated for addition, subtraction, multiplication, division, square root, logarithm, exponentiation, sine, and cosine[5,6,7,9,13,14,15], the algorithms did not at least on the surface appear to share a common, exploitable structure. Furthermore, given a CS/P algorithm there did not appear to be a systematic way to go about formulating an equivalent (in the sense that they both evaluate the same function) DOCS/P algorithm. This paper explores a method which has been used to formulate several DOCS/P algorithms. Furthermore, the algorithms so formulated possess several advantages over the equivalent algorithms previously known.

The second part of this paper covers the method which has been successfully used to convert several known CS/P algorithms. The converted form of the CS/P algorithm will be digit online with respect to its input(s) but will not be digit online with respect to its output(s). The converted form of the CS/P will, however, generate a sequence of better approximations to each dependent variable. Supplied with a sequence of approximations to a dependent variable, a digitization algorithm will generate that dependent variable in a digit online manner. This digitization algorithm will be covered in the third section. The fourth covers a new DOCS/P divide algorithm and describes the advantages of the new algorithm over the previously known equivalent algorithms.

## Conversion

Unless otherwise stated, this paper implicitly assumed that a fixed point number system is being used. That is, the sequence of digits $d_{i_1}, d_{i_2}, \ldots, d_{i_n}$ represents the value v, if

$$v = \sum_{j=1}^{n} d_{i_j} r^{p-j},$$

where $d_{i_j}$, $j = 1, 2, \ldots, n$, is an element of the digit set of the number system, $r$ is the radix of the number system, and $p$ is some predefined constant. Also $(v)_j$, $j = 1, 2, \ldots, n$, will represent the j'th digit, $d_{i_j}$, of the implied representation of $v$.

A CS/P algorithm can be expressed in such a way that each iterative equation within the algorithm serves one of two purposes. That is, each iterative equation is either being used to help generate the approximations to the inverse of each independent variable or being used to help generate the increasingly better approximations to each dependent variable. Furthermore, the iterative equations being used to help generate the approximations to each dependent variable are not directly dependent on the independent variable(s). Hence, attention can be turned to only those iterative equations being used to help generate the approximations to the inverse of each independent variable.

Each of these iterative equations has one of two forms depending on whether an additive or multiplicative inverse is being approximated. The case where the additive inverse is being approximated will be considered first. Let $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_n$ each be some small set of predefined constants. These sets are dependent on the number system being used and the CS/P algorithm itself. Then, an iterative equation in this case has the form

$$i = 0$$
$$X_i = x$$
$$i = 1, 2, \ldots, n$$
$$X_i = X_{i-1} + \hat{s}_i,$$

where $x$ is an independent variable and $\hat{s}_i$, $i = 1, 2, \ldots, n$, is an element of $\hat{S}_i$. Furthermore, the constant $\hat{s}_i$, $i = 1, 2, \ldots, n$, selected from $\hat{S}_i$ is such that the quantity

$$|X_{i-1} + \hat{s}_i|$$

is minimized.

The sets $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_n$ are so defined that it is always possible, given some $x$ in the domain of the algorithm, to select $\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_i$, $i = 1, 2, \ldots, n$, so that

$$|X_i| \leq \hat{K}\, r^{-i},$$

where $\hat{K}$ is some predefine constant and $r$ equals the radix of the number system being used. Note that $r$ need not necessarily be equal to the radix but consideration of this generalization is not necessary for the purposes of this paper. Note that

$$i = 1, 2, \ldots, n$$
$$\left|x + \sum_{j=1}^{i} \hat{s}_j\right| \leq \hat{K}\, r^{-i}.$$

Hence the sum of $\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_i$ becomes a better approximation to the additive inverse of $x$ as $i$ gets larger. To make the process of selecting the $\hat{s}_i$'s almost independent of $i$, a scaling

$$U_i = r^i X_i$$

of $X_i$ is used. This scaling transforms the original iterative equation into

$$i = 0$$
$$U_i = x$$
$$i = 1, 2, \ldots, n$$
$$U_i = r\, U_{i-1} + r^i \hat{s}_i$$

Note that

$$i = 1, 2, \ldots, n$$
$$|X_i| \leq \hat{K}\, r^{-i}$$

implies that

$$|U_i| \leq \hat{K}.$$

If a redundant number system[1] is being used, a random error $e_i$, $i = 1, 2, \ldots, n$, can be added to $U_{i-1}$ while allowing the selection of $\hat{s}_i$'s whose sum still sufficiently approximates the additive inverse of $x$. The only restriction on $e_i$, $i = 1, 2, \ldots, n$, is that it be bound by some constant $e$. The constant $e$ is dependent on the sets $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_n$ and, therefore, on the number system being used and the algorithm itself. In particular, the constant $e$ is dependent on the redundancy of the number system being used. For example, if a non-redundant number system is being used, the constant $e$ equals zero. Note that only the first

$$\lceil \log_r(\hat{K}) \rceil - \lfloor \log_r(2e) \rfloor$$

digits of $U_{i-1}$, $i = 1, 2, \ldots, n$, are needed to select $\hat{s}_i$.

The converted form of this iterative equation is

$$i = 0$$
$$\hat{X}_i = \sum_{j=1}^{k} (x)_j\, r^{p-j}$$
$$i = 1, 2, \ldots, n$$
$$\hat{X}_i = \hat{X}_{i-1} + (x)_{i+k}\, r^{p-i-k} + \hat{s}_i.$$

Note that this iterative equation is, as desired, digit online with respect to $x$.

Again, when scaling is used to make the process of selecting the $s_i$'s almost independent of $i$, the converted iterative equation is transformed into

$i = 0$

$$\hat{U}_i = \sum_{j=1}^{k} (x)_j \, r^{p-j}$$

$i = 1, 2, \ldots, n$

$$\hat{U}_i = r \, \hat{U}_{i-1} + (x)_{i+k} \, r^{p-k} + r^i \, \hat{s}_i .$$

Equivalently, this iterative equation can be written as follows.

$i = 0$

$$\hat{U}_i = x$$

$i = 1, 2, \ldots, n$

$$\hat{U}_i = r(\hat{U}_{i-1} + e_i) + r^i \, \hat{s}_i ,$$

where

$i = 1, 2, \ldots, n$

$$|e_i| = \left| \sum_{j=i+k+1}^{n} (x)_j \, r^{p+i-j-1} \right| \leq r^{p-k} .$$

Assume that only the first $j$ digits of $\hat{U}_{j-1}$, $i = 1, 2, \ldots, n$, are used to select $\hat{s}_i$. Therefore, if

$$r^{p-k} + .5 \, pow_r(\hat{K}) \, r^{-j} \leq e ,$$

where

$$pow_r(x) = r^{\lceil \log_r(x) \rceil} ,$$

$\hat{s}_i$'s whose sum sufficiently approximates the additive inverse of $x$ will still be selected.

The case where the multiplicative inverse is being approximated will now be considered. Again, let $\bar{S}_1, \bar{S}_2, \ldots, \bar{S}_n$ each be some small set of predefined constants. These sets are dependent on the number system being used and the algorithm itself. Then, the iterative equation in this case has the form

$i = 0$

$$X_i = x$$

$i = 1, 2, \ldots, n$

$$X_i = X_{i-1} \, \bar{s}_i$$

where $x$ is an independent variable whose multiplicative inverse is being approximated and $\bar{s}_i$, $i = 1, 2, \ldots, n$, is an element of $\bar{S}_i$. The constant $\bar{s}_i$, $i = 1, 2, \ldots, n$, is selected from $\bar{S}_i$ such that the quantity

$$|X_{i-1} \, \bar{s}_i - 1|$$

is minimized. The sets $\bar{S}_1, \bar{S}_2, \ldots, \bar{S}_n$ are so defined that it is always possible, given some $x$ in the domain of the algorithm, to select $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_i$, $i = 1, 2, \ldots, n$, so that

$$|X_i - 1| \leq \bar{K} \, r^{-i} ,$$

where $\bar{K}$ is some predefined constant and $r$ equals the radix of the number system being used. Note, therefore, that

$i = 1, 2, \ldots, n$

$$\left| x \prod_{j=1}^{i} \bar{s}_j - 1 \right| \leq \bar{K} \, r^{-i} .$$

Hence, the product of $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_i$ becomes a better approximation to the multiplication inverse of $x$ as $i$ gets larger.

To make the process of selecting the $\bar{s}_i$'s almost independent of $i$, a scaling

$$U_i = r^i (X_i - 1)$$

of $X_i$ is used. This scaling transforms the original iterative equation into

$i = 0$

$$U_i = x - 1$$

$i = 1, 2, \ldots, n$

$$U_i = r \, U_{i-1} \, \bar{s}_i + r^i (\bar{s}_i - 1) .$$

Note that

$i = 1, 2, \ldots, n$

$$|X_i - 1| \leq \bar{K} \, r^{-i}$$

implies that

$$|U_i| \leq \bar{K} .$$

Again, if a redundant number system is being used, a random error $e_i$, $i = 1, 2, \ldots, n$, can be added to $U_{i-1}$ while allowing the selection of $\bar{s}_i$'s whose product still sufficiently approximates the multiplicative inverse of $x$. The only restriction on $e_i$, $i = 1, 2, \ldots, n$, is that it be bounded by some constant $e$. The constant $e$ is dependent on the sets $\bar{S}_1, \bar{S}_2, \ldots, \bar{S}_n$ and, therefore, on the number system being used and the algorithm itself. Note, therefore, that only the first

$$\lceil \log_r(\bar{K}) \rceil - \lfloor \log_r(2e) \rfloor$$

digits of $U_{i-1}$, $i = 1, 2, \ldots, n$, are needed to select $\bar{s}_i$.

The converted form of this iterative equation is

$i = 0$

$$T_i = 1$$

$$\hat{X}_i = \sum_{j=1}^{k} (x)_j \, r^{p-j} - 1$$

$i = 1, 2, \ldots, n$

$$T_i = T_{i-1} \, \bar{s}_i$$

$$\hat{X}_i = \hat{X}_{i-1} \, \bar{s}_i + T_i (x)_{i+k} \, r^{p-i-k} .$$

Note that this iterative equation is, as desired,

digit on line with respect to $x$.

Again, when a scaling of the $\hat{X}_i$'s is used to make the process of selecting the $\bar{s}_i$'s almost independent of $i$, the converted iterative equation is transformed into

$i = 0$

$T_i = 1$

$\hat{U}_i = \sum_{j=1}^{k} (x)_j \, r^{p-j} - 1$

$i = 1, 2, \ldots, n$

$T_i = T_{i-1} \, \bar{s}_i$

$\hat{U}_i = r \, \hat{U}_{i-1} \, \bar{s}_i + r^i(\bar{s}_i - 1) + T_i(x)_{i+k} \, r^{p-k}.$

Equivalently, this iterative equation can be written as follows.

$i = 0$

$T_i = 1$

$\hat{U}_i = x - 1$

$i = 1, 2, \ldots, n$

$T_i = T_{i-1} \, \bar{s}_i$

$\hat{U}_i = r(\hat{U}_{i-1} + e_i)\bar{s}_i + r^i(\bar{s}_i - 1)$,

where

$i = 1, 2, \ldots, n$

$|e_i| = |T_{i-1} \sum_{j=i+k+1}^{n} (x)_j \, r^{p+i-j-1}| \leq$

$|T_{i-1}| \, r^{p-k}$

$|T_{i-1}| \leq \frac{1}{|x|} (1 + \bar{K} \, r^{1-i})$.

When the properties of the number system being used are known, a tighter bound on $T_{i-1}$ can be found.

Assume that only the first $j$ digits of $\hat{U}_{i-1}$, $i = 1, 2, \ldots, n$, are used to select $\bar{s}_i$. Therefore, if

$\max_{i=1}^{n}(|T_{i-1}|)r^{p-k} + .5 \, \text{pow}_r(\bar{K})r^{-j} \leq e$,

$\bar{s}_i$'s whose product sufficiently approximates the multiplicative inverse of $x$ will still be selected.

In the way of an example, the following illustrates the material covered in this section. Let $z$ equal the quotient of $y$, $|y| < 1$, the dividend, and $x$, $1/r \leq |x| < 1$, the divisor. Then supplied with $x$ and $y$, the following, previously known CS/P algorithm[3,2,11] generates increasingly better approximations to $z$.

Algorithm ODIV

$i = 0$

$X_i = x$

$Q_i = y$

$i = 1, 2, \ldots, n$

$X_i = X_{i-1} \, \bar{s}_i$

$Q_i = Q_{i-1} \, \bar{s}_i$,

where $Q_i$ and the product of $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_i$ are respectively the i'th approximation to the dependent variable $z$ and the multiplicative inverse of the independent variable $x$. Observe, however, that the iterative equation

$i = 0$

$Q_i = y$

$i = 1, 2, \ldots, n$

$Q_i = Q_{i-1} \, \bar{s}_i$,

which is used to generate the approximations to the dependent variable $z$ is directly dependent on the independent variable $y$. Rewriting algorithm ODIV to remove this dependence produces the following algorithm.

Algorithm BDIV

$i = 0$

$T_i = 1$

$X_i = x$

$Y_i = y$

$Q_i = 0$

$i = 1, 2, \ldots, n$

$T_i = T_{i-1} \, \bar{s}_i$

$X_i = X_{i-1} \, \bar{s}_i$

$Y_i = Y_{i-1} + \hat{s}_i$

$Q_i = Q_{i-1} \, \bar{s}_i - T_i \, \hat{s}_i$,

where $Q_i$, the product of $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_i$, and the sum of $\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_i$ are respectively the i'th approximation to the dependent variable $z$, the multiplicative inverse of the independent variable $x$, and the additive inverse of the independent variable $y$. Furthermore, $\bar{s}_i$ and $\hat{s}_i$, $i = 1, 2, \ldots, n$, are respectively elements of $\bar{S}_i$ and $\hat{S}_i$.

To make the process of selecting the $\bar{s}_i$'s and the $\hat{s}_i$'s almost independent of $i$, two scaling

$U_i = r^i(X_i - 1)$

and

$V_i = r^i Y_i$

are used. These scalings transform algorithm BDIV into the following algorithm.

Algorithm FDIV

$i = 0$

$T_i = 1$

$U_i = x - 1$

$V_i = y$

$Q_i = 0$

$i = 1, 2, \ldots, n$

$T_i = T_{i-1}\,\bar{s}_i$

$U_i = r\,U_{i-1}\,\bar{s}_i + r^i(\bar{s}_i - 1)$

$V_i = r\,V_{i-1} + r^i\,\hat{s}_i$

$Q_i = Q_{i-1}\,\bar{s}_i - T_i\,\hat{s}_i.$

To complete this example, assume that a fully redundant, quaternary, fractional number system is being used. Furthermore, assume that

$i = 1$

$\bar{S}_i = \{j \mid j = -3, -2, \ldots, 2, 3\}$

$i = 2, 3, \ldots, n$

$\bar{S}_i = \{1 + j\,4^{1-i} \mid j = -3, -2, \ldots, 2, 3\}$

$i = 1, 2, \ldots, n$

$\hat{S}_i = \{j\,4^{-i} \mid j = -3, -2, \ldots, 2, 3\}.$

Note that if

$i = 3, 4, \ldots, n$

$|U_{i-1}| \leq 3.011$

then $\bar{s}_i$ can be selected so that

$|r\,U_{i-1}\,\bar{s}_i + r^i(\bar{s}_i - 1)| \leq 2.371.$

Hence

$|U_i| \leq 2.371.$

Therefore, an error, $e_i$, $i = 1, 2, \ldots, n$, which is bounded by .640, can be added to $U_{i-1}$ while allowing the selection of $\bar{s}_i$'s whose product still sufficiently approximates the multiplicative inverse of $x$. This implies that a digit online delay of

$0 - \lfloor \log_4(.640/4.60) \rfloor = 2$

with respect to the independent variable $x$ is possible.

Also note that as $i$ becomes large, if

$|U_{i-1}| \leq 3.5$

then $\bar{s}_i$ can be selected so that

$|r\,U_{i-1}\,\bar{s}_i + r^i(s_i - 1)| \leq 2.0.$

Hence,

$|U_i| \leq 2.0.$

Therefore, an error, $e_i$, $i$ sufficiently large, which is bounded by 1.5, can be added to $U_{i-1}$ while allowing the selection of $\bar{s}_i$'s whose product still sufficiently approximates the multiplicative inverse of $x$. This implies that an asymptotic digit online delay of

$0 - \lfloor \log_4(1.5/4.0) \rfloor = 1$

with respect to the independent variable $x$ is possible.

Furthermore, note that if

$i = 1, 2, \ldots, n$

$|V_{i-1}| \leq .875$

then $\hat{s}_i$ can be selected so that

$|r\,V_{i-1} + r^i\,\hat{s}_i| \leq .5.$

Hence

$|V_i| \leq .5.$

Therefore, an error, $\hat{e}_i$ $i = 1, 2, \ldots, n$, which is bounded by .375, can be added to $V_{i-1}$ while allowing the selection of $\hat{s}_i$'s whose sum still sufficiently approximate the additive inverse of $y$. This implies that a digit online delay of

$0 - \lfloor \log_4(.375) \rfloor = 1$

with respect to the independent variable $y$ is possible.

Converting algorithm FDIV so that it is digit online with respect to its inputs produces the following algorithm.

Algorithm RDIV

$i = 0$

$T_i = 1$

$\hat{U}_i = \sum_{j=1}^{2} (x)_j\,r^{-j} - 1$

$\hat{V}_i = \sum_{j=1}^{2} (y)_j\,r^{-j}$

$Q_i = 0$

$i = 1, 2, \ldots, n$

$T_i = T_{i-1}\,\bar{s}_i$

$\hat{U}_i = r\,\hat{U}_{i-1}\,\bar{s}_i + r^i(\bar{s}_i - 1) + T_i(x)_{i+2}\,r^{-2}$

$\hat{V}_i = r\,\hat{V}_{i-1} + (y)_{i+2}\,r^{-2} + r^i\,\hat{s}_i$

$Q_i = Q_{i-1}\,\bar{s}_i - T_i\,\hat{s}_i.$

Note that at the i'th step of the algorithm only the first (i + 2) digits of each independent variable must be supplied to the algorithm. Hence, the algorithm is, as desired, digit online with respect to its inputs. Also, note that only

the first two digits of $U_{i-1}$ and the first digit of $V_{i-1}$, $i = 1, 2, \ldots, n$, are required to select respectively $\bar{s}_i$ and $\hat{s}_i$.

The following is given as an example for algorithm FDIV. Note that $\bar{1}$, $\bar{2}$, and $\bar{3}$ are used to represent respectively the values -1, -2, and -3. Also, note the all number are given in base four.

$x = .2\bar{1}21 \quad y = .11\bar{2}\bar{1}$

| | $\bar{s}_i$ | $T_i$ | $\hat{U}_i$ | $\hat{s}_i$ | $\hat{v}_i$ | $Q_i$ |
|---|---|---|---|---|---|---|
| i=0 | NA | 1.000000 | $0.\bar{2}\bar{1}000$ | NA | .1100 | .000000 |
| i=1 | 2.0000 | 2.000000 | $0.\bar{1}0000$ | $.\bar{1}000$ | $.1\bar{2}00$ | .200000 |
| i=2 | 1.0000 | 2.000000 | $\bar{1}.02000$ | $.0\bar{1}00$ | $.\bar{2}\bar{1}00$ | .220000 |
| i=3 | 1.0100 | 2.020000 | 0.10200 | $.00\bar{2}0$ | $.\bar{1}000$ | $.212\bar{1}00$ |
| i=4 | 1.0000 | 2.020000 | 1.02000 | .0001 | .0000 | $.212\bar{1}\bar{0}\bar{2}$ |
| i=5 | $1.000\bar{1}$ | $2.020\bar{2}\bar{0}\bar{2}$ | $0.20\bar{1}\bar{0}\bar{2}$ | .0000 | .0000 | $.212\bar{2}11$ |
| i=6 | 1.0000 | $2.020\bar{2}\bar{0}\bar{2}$ | $2.0\bar{1}\bar{0}\bar{2}0$ | .0000 | .0000 | $.212\bar{2}11$ |

## Digitization

The following algorithm was originally used[9] to create a digit online algorithm for exponentiation. There was, at that time, some question concerning the wider applicability of this algorithm. The following, albeit tersely, addresses these questions. A more complete discussion can be found[10].

Let $Z_1, Z_2, \ldots, Z_n$ be any suitable approximations to some value $z$. That is, there exists a constant $\hat{C}$ such that

$i = 1, 2, \ldots, n$

$$|z - Z_i| \leq \hat{C}\, r^{-i},$$

where $r$ again equals the radix of the number system being used. Furthermore, let $D$ be the digit set of this number system. Also, assume that $z$ is bounded. That is, there exists a constant $\bar{C}$ such that

$$|z| \leq \bar{C}.$$

Supplied with $Z_i$ at its i'th iteration the following algorithm generates $z$ in a digit online manner. That is, at the i'th iteration, the algorithm generates the (i - k) digit of $z$.

Algorithm DISC

$i = 0$

$A_i = 0$

$i = 1, 2, \ldots, k$

$A_i = A_{i-1} + Z_i - Z_{i-1}$

$i = k+1, k+2, \ldots, k+n$

$$A_i = A_{i-1} + Z_i - Z_{i-1} - s_{i-k}\, r^{k+p-i},$$

where $s_{i-k}$ is selected from $D$ so that the value $A_i$ is minimized and the constant $k$ is yet to be defined. Then assuming only that the number system being used is fully redundant and the original premise, that is there exists constants $\hat{C}$ and $\bar{C}$ such that

$i = 1, 2, \ldots, n$

$$|z - Z_i| \leq \hat{C}\, r^{i}$$

and

$$|z| \leq \bar{C},$$

it is claimed that there exists a constant $C$ such that

$i = 1, 2, \ldots, n$

$$\left| \sum_{j=1}^{i} s_j\, r^{p-j} - z \right| \leq C\, r^{-i}.$$

That is, the value formed by concatenating together the digits $s_1, s_2, \ldots, s_i$ becomes an increasingly better approximation to $z$ as $i$ gets larger.

Let the constant $k$ be such that

$i = k+2, k+3, \ldots, k+n$

$$|Z_i - Z_{i-1}| \leq .5(r - 1)r^{k+p-i}$$

$i = k+1$

$$|Z_i| \leq (r - .5)r^{k+p-i}.$$

The existence of the constant $k$ is assured by the original premise. Observe that if

$i = k+1, k+2, \ldots, k+n$

$$|A_{i-1} + Z_i - Z_{i-1}| \leq (r - .5)r^{k+p-i},$$

then $s_{i-k}$ can be selected so that

$$|A_i| \leq .5\, r^{k+p-i}.$$

Furthermore, observe that

$i = 1, 2, \ldots, n$

$$\left| \sum_{j=1}^{i} s_j\, r^{p-j} - z \right| = |Z_{i+k} - z - A_{i+k}| \leq$$

$$|Z_{i+k} - z| + |A_{i+k}| \leq \hat{C}\, r^{-i-k} + |A_{i+k}|.$$

Therefore, if it can be shown that there exists a constant $B$ such that

$i = k+1, k+2, \ldots, k+n$

$$|A_i| \leq B\, r^{-i},$$

then the claim can be proven via a corollary.

An inductive proof for the second claim,

$i = 1, 2, \ldots, n$

$$|A_i| \leq B\, r^{-i},$$

is given by the following.

Base $i = k+1$

Note that

$$|Z_i| \leq (r - .5)r^{k+p-i}$$

implies that

$$|A_{i-1} + Z_i - Z_{i-1}| = |Z_i| \leq (r - .5)r^{k+p-i}.$$

But this implies that

$$|A_i| \leq .5\ r^{k+p-i}.$$

Step $i = k+2, k+3, \ldots, k+n$

Note that

$$|A_{i+1}| \leq .5\ r^{k+p-i-1}$$

implies that

$$|A_{i-1} + Z_i - Z_{i-1}| \leq |A_{i-1}| + |Z_i - Z_{i-1}| \leq$$
$$.5\ r\ r^{k+p-i} + .5(r-1)r^{k+p-i} =$$
$$(r - .5)r^{k+p-i}.$$

But this implies that

$$|A_i| \leq .5\ r^{k+p-i}.$$

Both claims are now justified.

The following continues the example started for algorithm FDIV.

| | $s_i$ | $Z_i$ | $A_i$ |
|---|---|---|---|
| i=0 | NA | .000000 | .000000 |
| i=1 | NA | .200000 | .200000 |
| i=2 | NA | .220000 | .220000 |
| i=3 | 2 | .212100 | .012100 |
| i=4 | 1 | $.212\bar{1}0\bar{2}$ | $.002\bar{1}0\bar{2}$ |
| i=5 | 2 | $.212\bar{2}11$ | $.000\bar{2}11$ |
| i=6 | $\bar{2}$ | $.212\bar{2}11$ | $.000011$ |

Hence

$$y/x = .212\bar{2}.$$

## Division

The example started in the second section will now be completed. To create a single fully digit online algorithm from algorithms RDIV and DISC it must be shown that there exists constants $\hat{C}$ and $\bar{C}$ such that

$$i = 1, 2, \ldots, n$$
$$|z - Q_i| \leq \hat{C}\ r^{-i}$$

and

$$|z| \leq \bar{C}.$$

The following shows the existence of these constants. First, note that

$$\max(|z|) \leq \frac{\max(|y|)}{\min(|x|)} = r.$$

However, if

$$|z| \geq 1,$$

then overflow has occurred. That is, $z$ cannot be represented by the number system being used. Second, note that

$$i = 1, 2, \ldots, n$$

$$|z - Q_j| = |z - (\prod_{j=1}^{i} s_j)(\sum_{j=1}^{i} \hat{s}_j)| =$$

$$\frac{|y - (x \prod_{j=1}^{i} \bar{s}_j - 1 + 1) \sum_{j=1}^{i} \hat{s}_j|}{|x|} =$$

$$\frac{|y - \sum_{i=1}^{i} \hat{s}_j|}{|x|} + \frac{|(x \prod_{j=1}^{i} \bar{s}_j - 1) \sum_{j=1}^{i} \hat{s}_j)|}{|x|} \leq$$

$$\frac{\hat{K}\ r^{-i}}{|x|} + \frac{\bar{K}\ r^{-i}}{|x|} =$$

$$\frac{r^{-i}}{|x|}(\hat{K} + \bar{K}).$$

The necessary premises for the use of algorithm DISC with algorithm RDIV have now been shown.

The following finds a smaller $k$ than would be found by directly using the constants $\hat{K}$ and $\bar{K}$. Observe that

$$i = 1, 2, \ldots, n$$
$$|z - Q_i| \leq \frac{r^{-i}}{|x|}(\hat{K} + \bar{K})$$

implies that

$$|Q_i| \leq |z| + \frac{r^{-i}}{|x|}(\hat{K} + \bar{K}).$$

Furthermore, observe that

$$i = 2, 3, \ldots, n$$

$$|Q^i - Q_{i-1}| =$$

$$|(\prod_{j=1}^{i} \bar{s}_j)(\sum_{j=1}^{i} \hat{s}_j) - (\prod_{j=1}^{i-1} \bar{s}_j)(\sum_{j=1}^{i-1} \hat{s}_j)| =$$

$$|s_i \prod_{j=1}^{i} \bar{s}_j + (\bar{s}_i - 1)(\prod_{j=1}^{i-1} \bar{s}_j)(\sum_{j=1}^{i-1} \hat{s}_j)| \leq$$

$$\frac{(r-1)r^{-i}}{|x|}|(\hat{K}\ r^{-i} + 1) +$$

$$(\hat{K}\ r^{1-i} + y)(\bar{K}\ r^{1-i} + 1)| \leq$$

$$r(r-1)r^{-i}|1.149 + (1.125)(1.592)| =$$

$$.5(r-1)r^{2-i}.$$

Hence, with a suitable restriction on the quotient of $y$ and $x$, $k$ equals 2. Again it can be shown that, for large $i$, $k$ can be made to equal 1.

In the algorithm which results from combining algorithms RDIV and DISC, three values must be selected. Two are due to algorithm RDIV and one is due to algorithm DISC. It is possible, however, using a different methodology[7,13,14,15] to formulate an algorithm such that the selection of only one value is necessary. There are, however, advantages in not doing so. The total digit online

delay of the algorithm which results from combining algorithms RDIV and DISC is four (asymptotically two). This is better than the digit online delay of five (asymptotically three) which is required by the single selection algorithm. Also a word length of at most only two digits are needed to select each of the three values in the algorithm which results from combining algorithms RDIV and DISC. Whereas, the single selected algorithm needs at most a word length of five digits to select its single value. Therefore, the algorithm which results from combining algorithms RDIV and DISC has a smaller online delay and a much smaller comparison table than the single selection algorithm. Finally, the other algorithms which have been formulated using the method described in this paper share a common structure. This characteristic can be used when the algorithms are implemented in hardware.

## Conclusion

This paper, as promised, presented a method which has been successfully used to create several digit online algorithms. In general, these algorithms have a smaller digit online delay and smaller comparison tables than the previously known algorithms. Furthermore, these algorithms share a common structure. Sufficient detail is given so that a reader familiar with CS/P algorithms should be able to apply the method.

The paper also presents a new digit online divide algorithm. For the case of a fully redundant, quaternary number system, correctness proofs are given.

## Bibliography


[1] Atkins, D. E., "Introduction to the Role of Redundancy in Computer Arithmetic," Computer, Vol. 8, No. 6, pp. 84-96, June 1975.

[2] Baker, P. W., "More Efficient Radix-2 Algorithms for Some Elementary Functions," IEEE Transactions on Computers, Vol. C-24, No. 11, Nov. 1975.

[3] DeLugish, B. G., "A Class of Algorithms for Automatic Evaluation of Certain Elementary Functions in a Binary Computer," Ph.D. Thesis, Report 399, Department of Computer Science, University of Illinois, Urbana, June 1970.

[4] Ercegovac, M. D., "Radix-16 Evaluation of Certain Elementary Functions," IEEE Transactions on Computers, Vol. C-22, No. 6, pp. 561-566, June 1973.

[5] Ercegovac, M. D., "A General Method for Evaluation of Functions in a Digital Computer," Ph.D. Thesis, Report No. 750, Department of Computer Science, University of Illinois, Urbana, Aug. 1975.

[6] Ercegovac, M. D., "An On-Line Square Rooting Algorithm," Proceedings of Fourth Symposium on Computer Arithmetic, Santa Monica, CA, Oct. 1978.

[7] Irwin, M. J., "An Arithmetic Unit for On-Line Computation," Ph.D. Thesis, Report No. UIUCDCS-R-77-873, Department of Computer Science, University of Illinois, Urbana, May 1977.

[8] Kuck, D. J., The Structure of Computers and Computations, Vol. I, John Wiley & Sons, Inc., 1978.

[9] Owens, R. M. and M. J. Irwin, "On-Line Algorithms for the Design of Pipeline Architectures," Proceedings of the Sixth Annual Symposium of Computer Architecture, Philadelphia, PA, pp. 12-19, April 1979.

[10] Owens, R. M., "Digit On-Line Algorithms for Pipeline Architectures," Ph.D. Thesis, Department of Computer Science, The Pennsylvania State University, University Park, PA, Aug. 1980.

[11] Robertson, J. E., "A New Class of Digital Division Methods," IRW Transactions on Electronic Computers, Vol. EC-7, pp. 218-222, Sept. 1958.

[12] Specker, W. H., "A Class of Algorithms for Ln X, Exp X, Sin X, Cos X, $\text{Tan}^{-1}$ X, $\text{Cot}^{-1}$," IEEE Transactions on Electronic Computers, Vol. EC-14, No. 1, pp. 85-86, Feb. 1965.

[13] Trivedi, K. S. and M. D. Ercegovac, "On-Line Algorithms for Division and Multiplication," Proceedings of the Third IEEE Symposium on Computer Arithmetic, Dallas, TX, Nov. 1975.

[14] Trivedi, K. S. and M. D. Ercegovac, "On-Line Algorithms for Division and Multiplication," IEEE Transactions on Computers, Vol. C-26, No. 7, pp. 681-687, July 1977.

[15] Trivedi, K. S. and J. G. Rusnak, "Higher Radix On-Line Division," Proceedings of the Fourth Symposium on Computer Arithmetic, Santa Monica, CA, Oct. 1978.

[16] Volder, J. E., "The CORDIC Trigonometric Computing Technique," IRE Transactions on Electronic Computers, Vol. EC-8, No. 5, pp. 330-334, Sept. 1959.