# A MODIFICATION TO THE HALF-INTERVAL SEARCH (BINARY SEARCH) METHOD

Louis F. Williams, Jr.

Abstract: This modification to the half-interval search (binary search) method finds the best computer zero within a fixed number of iterations of the half-interval search algorithm.

The paper outlines the modification for use with a computer where floating-point numbers are stored in a fixed length field of thirty-two bits. Also, the floating-point numbers are represented using a hexidecimal base and twenty-four bits to store the fraction.

The modification has to do with initially starting with a and b within consecutive powers of sixteen. That is, there exists an integer n such that

$$16^n \leq a < b \leq 16^{n+1} \quad \text{or} \quad -16^{n+1} \leq a < b \leq -16^n.$$

This determines the characteristic of $x_0$. Then a binary search for the fraction of $x_0$ can be completed within twenty-four iterations. If a computer zero of the function is not found in the search, then the a and b of the last iteration are consecutive computer numbers with f(a) and f(b) having opposite signs.

INTRODUCTION: The half-interval search (binary search) method is widely used to find the zero of a continuous function $y = f(x)$ on $[a,b]$ where $f(a)$ and $f(b)$ have opposite signs. The method is usually continued until:

a. $|f(x_0)| \leq \varepsilon$ ,

b. $|b - a| \leq \varepsilon$ , or

c. $x_0$ does not change.

This modification finds the best computer zero within twenty-four iterations of the half-interval search algorithm. There is no need for an epsilon in the modification.

This paper outlines the modification for use with a computer where floating-point numbers are stored in a fixed-length field of thirty-two bits. Also, the floating-point numbers are represented using a hexidecimal base and twenty-four bits to store the fraction. These different representations determine all the computer numbers.

MODIFICATION: The modification, see flow chart 1, has to do with initially starting with a and b within consecutive powers of sixteen. That is, there exists an integer n such that

$$16^n \leq a < b \leq 16^{n+1} \quad \text{or} \quad -16^{n+1} \leq a < b \leq -16^n.$$

This determines the characteristic of $x_0$. Then a binary

search for the fraction of $x_0$ can be completed within twenty-four iterations. If a computer zero of the function is not found in the search, then the a and b of the last iteration are consecutive computer numbers with f(a) and f(b) having opposite signs. Then $x_0$ can be selected from a and b such that

$$|f(x_0)| = \text{MINIMUM } \{|f(a)|, |f(b)|\}.$$

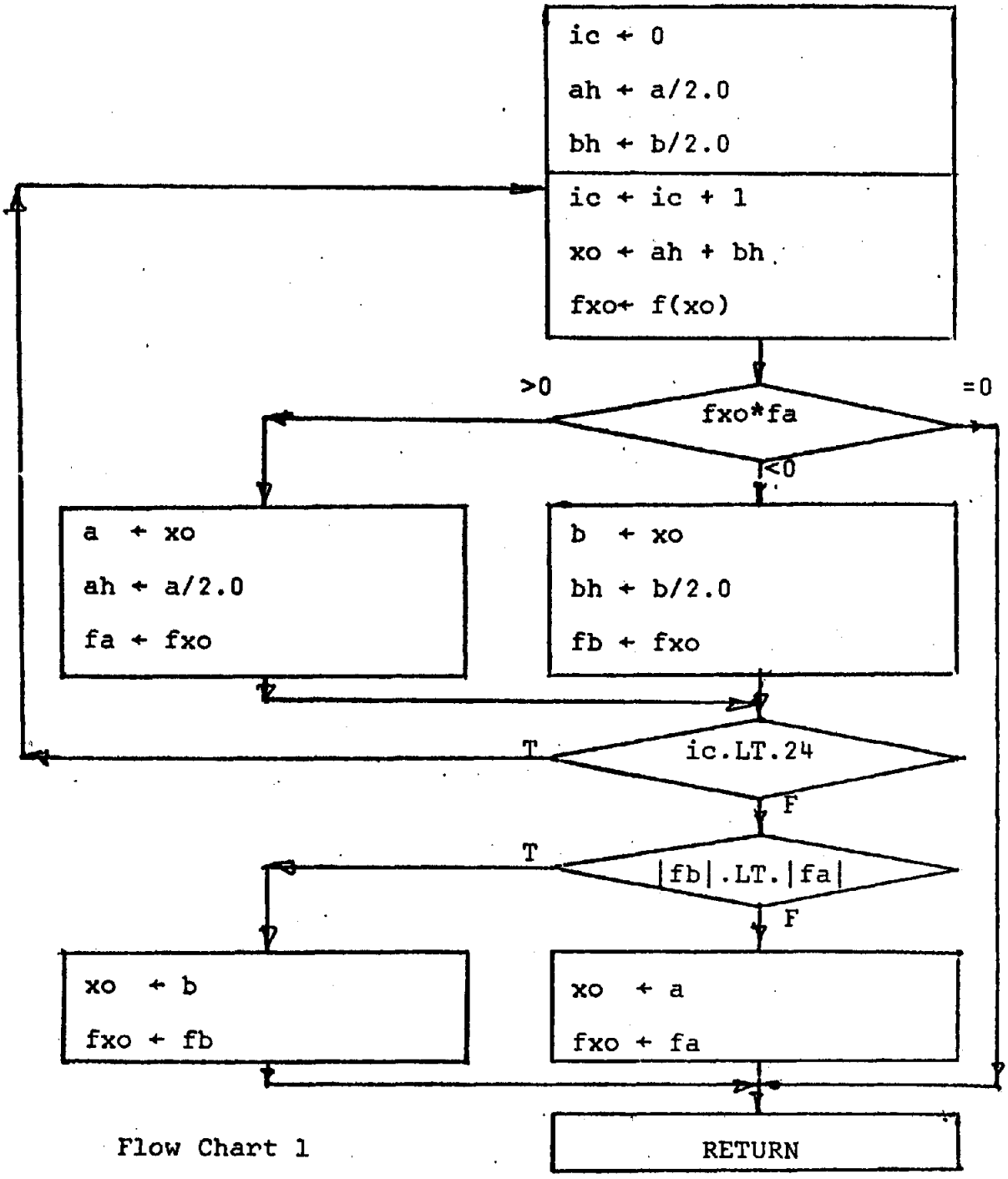
This $x_0$ is considered the best computer zero of the function for this modification.

APPLICATION: There is usually no major problem in finding the initial values for a and b. Consider the algorithm, see flow chart 2, to determine a and b for the special case where $y = f(x)$ is continuous for $x > 0$ with $f(x) < 0$ for $0 < x \leq N$ where N is some small positive computer number and $f(x) > 0$ for $x \geq M$ where M is some large positive computer number. Then a can be taken to be 1.0 at first. If f(a) is positive, then the correct a can be found by dividing by sixteen until f(a) becomes negative. If f(a) is negative, then the correct b can be found by multiplying by sixteen until f(b) becomes positive. This special case can be extended to the case where $y = f(x)$ is any odd degree polynomial with a nonzero constant coefficient. The modification is ideally suited for finding the best first zero of a cubic polynomial, see computer printout 1.

CONCLUSION: I have found this modification to be extremely valuable in my studies on numerical methods. Once the best solution has been obtained, other methods of arriving at a solution can be more fully studied. Since the modification requires twenty-four iterations, the modification would not be considered a fast method. This time limitation can be reduced by using an assembler program where the binary search technique can be more effectively implemented. Also, the time limitation could be further improved by additional research into algorithms that determine whether the value of a function is zero, positive, or negative.

Given: The continuous function y = f(x) on [a,b] with

$16^n \leq a < b \leq 16^{n+1}$ or $-16^{n+1} \leq a < b \leq -16^n$
and f(a) *f(b) < 0.
This subprogram determines a best computer zero for y = f(x) on [a,b].
Input variables: a, fa, b, fb
Output variables: a, fa, b, fb,xo,fxo



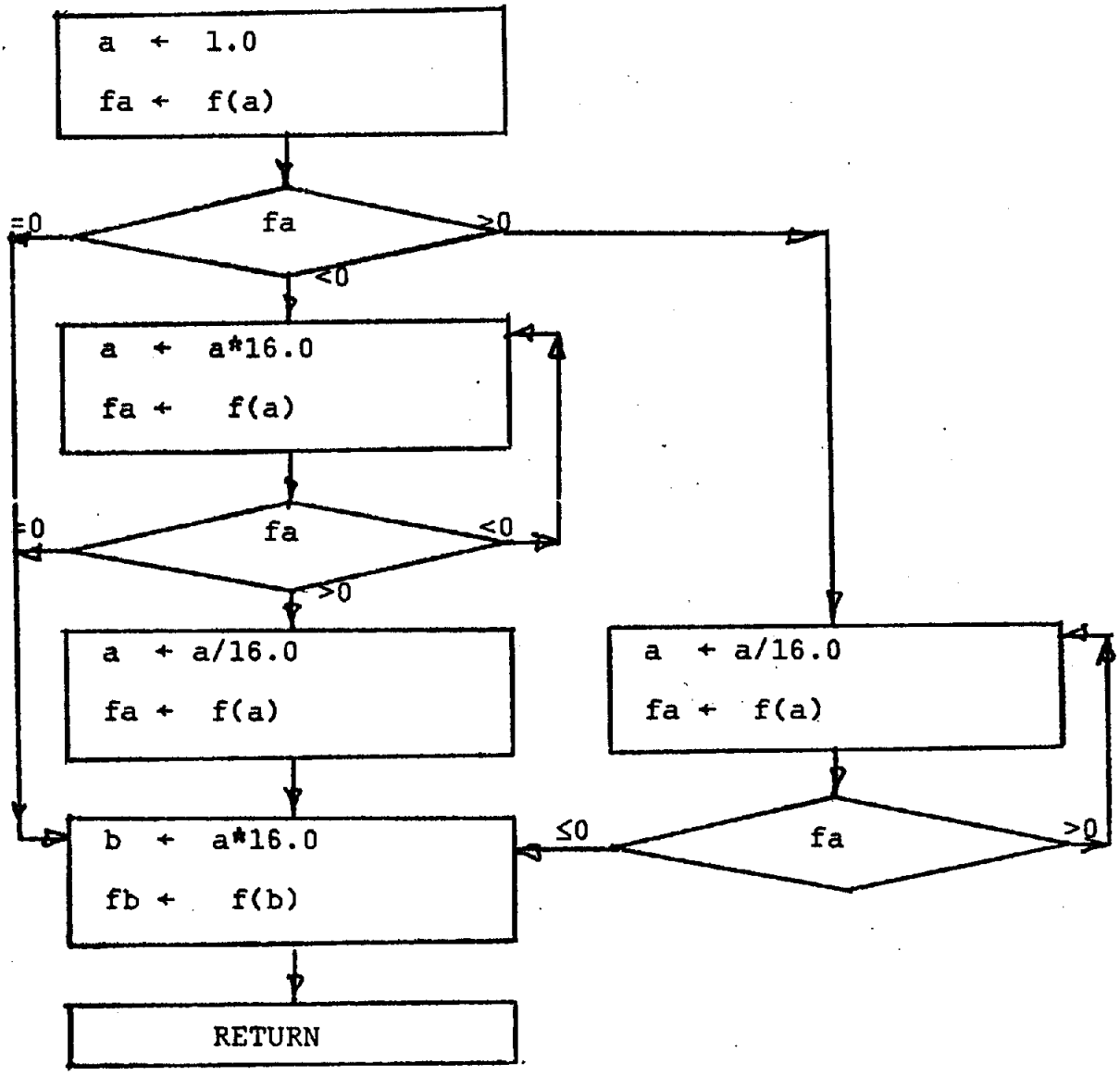

Flow Chart 1

Given: The continuous function $y = f(x)$ for $x > 0$ with $f(x) < 0$ for $0 < x \leq N$ and $f(x) > 0$ for $x \geq M$. This flow chart determines a, f(a), b, and f(b) where

$$16^n = a < b = 16^{n+1} \text{ with } f(a)*f(b) \leq 0.$$

Input variables: none

Output variables: a, fa, b, fb




FLOW CHART 2

```
POLYNOMIAL NUMBER 13   P(X) = ((A3*X+A2)*X+A1)*X+A0

A0 = -7902.00000     A1 = -7819.00000     A2 = -5019.00000     A3 =      0.00005

START
     A  =      0.1677722E 08      47100000      FA  =     -0.1176604E 19      00105423
     B  =      0.2684355E 09      48100000      FB  =      0.6054834E 21      5220D2C5

ITERATIONS IN REHISM
 1   XO =      0.1426063E 09      47880000      FXO =      0.4293683E 20      512530E4
 2   XO =      0.7969178E 08      474C0000      FXO =     -0.6569321E 19      D0582AE8
 3   XO =      0.1111491E 09      476A0000      FXO =      0.6652061E 19      505C50DE
 4   XO =      0.9542042E 08      47580000      FXO =     -0.2257876E 19      D01F5596
 5   XO =      0.1032847E 09      47628000      FXO =      0.1549321E 19      50158048
 6   XO =      0.9935258E 08      475EC000      FXO =     -0.5071196E 18      CF709A6A
 7   XO =      0.1013187E 09      4760A000      FXO =      0.4817541E 18      4F6AF88D
 8   XO =      0.1003356E 09      475F3000      FXO =     -0.2237678E 17      CE4F7F91
 9   XO =      0.1008271E 09      47602800      FXO =      0.2272673E 18      4F3276A7
10   XO =      0.1005814E 09      475FEC00      FXO =      0.1018371E 18      4F169CC5
11   XO =      0.1004585E 09      475FCE00      FXO =      0.3957839E 17      4E8C9C57
12   XO =      0.1003971E 09      475FBF00      FXO =      0.8582594E 16      4E1E7DD3
13   XO =      0.1003663E 09      475FB780      FXO =     -0.6926245E 16      CE189B62
14   XO =      0.1003817E 09      475FBB40      FXO =      0.8258017E 15      4D2EF0FF
15   XO =      0.1003740E 09      475FB960      FXO =     -0.3031136E 16      CDAC4CD6
16   XO =      0.1003779E 09      475FBA50      FXO =     -0.1102816E 16      CD3EB016
17   XO =      0.1003798E 09      475FBAC8      FXO =     -0.1582239E 15      CC8FE760
18   XO =      0.1003807E 09      475FBB04      FXO =      0.3534597E 15      4D141784
19   XO =      0.1003803E 09      475FBAE6      FXO =      0.7793540E 14      4C46E1C0
20   XO =      0.1003800E 09      475FBAD7      FXO =     -0.4014481E 14      CC2482F1
21   XO =      0.1003801E 09      475FBADE      FXO =      0.3857516E 14      4C23157B
22   XO =      0.1003801E 09      475FBADA      FXO =     -0.7848717E 12      CAB68E00
23   XO =      0.1003801E 09      475FBADC      FXO =     -0.7848719E 12      CAB68E03
24   XO =      0.1003801E 09      475FBADD      FXO =      0.3857515E 14      4C23157A

FINISH
     A  =      0.1003801E 09      475FBADC      FA  =     -0.7848719E 12      CAB68E03
     B  =      0.1003801E 09      475FBADD      FB  =      0.3857515E 14      4C23157A
     XO =      0.1003801E 09      475FBADC      FXO =     -0.7848719E 12      CAB68E03
     IC =                 24      00000018
```

Computer Printout 1