1969

A Polyalgorithm for the Automatic Solution of Nonlinear Equations

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
69-032

A POLYALGORITHM FOR THE AUTOMATIC SOLUTION OF NONLINEAR EQUATIONS

John R. Rice

February 1969
CSD TR 32

ABSTRACT

This paper discusses the design of a polyalgorithm for the automatic solution of a nonlinear equation $F(x) = 0$ for one variable. The polyalgorithm is part of NAPSS. The function $F(x)$ is described by a computer program and is examined only by evaluation. The three main parts of the paper are: brief discussion of the objectives, description of the polyalgorithm and the testing made of it.

A POLYALGORITHM FOR THE AUTOMATIC SOLUTION OF NONLINEAR EQUATIONS

John R. Rice*

1. INTRODUCTION AND THE PROBLEM. We consider the mathematical problem of given a function F(x), find values XROOT so that F(XROOT) = 0. We assume that F(x) is described by a computer program; in particular, we cannot examine F(x) in any way except by evaluation. F(x) is a function of one real variable.

   This paper has three main parts: a brief discussion of the objectives of a polyalgorithm for the automatic solution of this problem, a longer discussion of the polyalgorithm developed and some remarks on the testing made of the polyalgorithm. This polyalgorithm has been developed primarily for the NAPSS system. A general description of NAPSS is given in [5], and various aspects of the system are described in [1], [2] and [6]. A detailed philosophy and discussion of the development of polyalgorithms for automatic numerical analysis is given in [3].

2. POLYALGORITHM OBJECTIVES. There are a number of possible uses of this polyalgorithm. The objectives for most of these are indicated by the following:

   A) To solve this problem with no additional information. That is to say implement the statement

      SOLVE F(X)=0   FOR X

   B) To allow some guidance by the user via qualifying phrases. Typical qualifying phrases are
      a) NUMBER 3 (of roots desired)
      b) GUESS 13.1 (for root)
      c) INTERVAL [-12, 104] (roots must be in here)
      d) WORK 15 SECONDS (time limit on computation)
      e) OUTPUT LEVEL 3 (specifies amount of output desired)

*This polyalgorithm has been developed for the NAPSS system which is partially supported by a grant from the National Science Foundation, GP-05850.

c) To provide the user with considerable information, if desired, about the solutions of the problem, and the effort made to solve the problem.

3. POLYALGORITHM COMPONENTS AND STATUS. The current version of the polyalgorithm is a set of Fortran subroutines. The code is about 2500 statements. Three almost identical versions exist, one each for ordinary batch processing, the NAPSS system and remote batch processing from a console.

The basic components of the polyalgorithm are:

a) Initialization, user interface	e) Order of Roots
b) Overall Search Strategy	f) F(X) Deflation
c) Numerical Methods for F(X) = 0	g) Logical Control
d) Root Acceptance Tests	h) Historical Information

These components are discussed in varying detail.

4. INITIALIZATION AND USER INTERFACE. The polyalgorithm is controlled by a basic subroutine with about 20 arguments. This subroutine initializes a large number of variables and sets default options as required. There are about 100 variables to be initialized, including 50 print control switches.

The user interface for batch processing consists of a few small subroutines with 2 to 6 arguments which allow the user some flexibility in his use of the polyalgorithm. These subroutines may also be used from consoles. There is a Fortran preprocessor (written in SNOBOL4) which allows more natural statements, but still results in batch processing. Statements such as

```
EQ1.3    $    COS(X)**2 + X*ABS(X-3.1)*EXP(2.*X) = 0
SOLVE    EQ1.3    FOR X    NUMBER 1 INTERVAL -3.,2.
```

are translated in Fortran and then the polyalgorithm is accessed in the normal way.

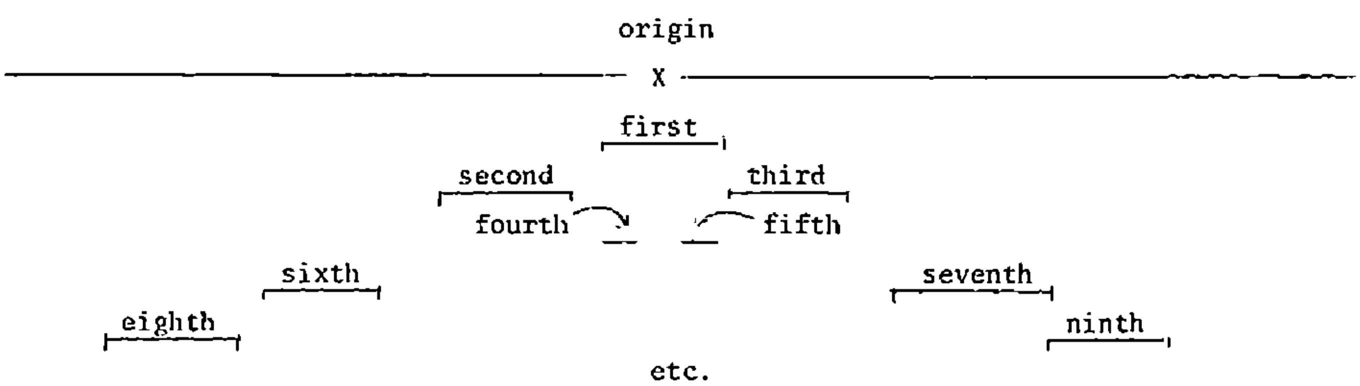
The NAPSS system provides a more natural and flexible interface as well as allowing interactive use. At certain points the polyalgorithm may receive instructions from the user and the user may request additional information or effort from the polyalgorithm.

5. OVERALL SEARCH STRATEGY. There are two distinct cases. The simplest one is when an interval is specified.

Interval Search: The secant method is started at a sequence of points in the interval. For the interval [0,1] these points are 1/2, 0, 1, 1/4, 3/4, 1/8, 3/8, 5/8, 7/8, 1/16...,15/16,1/32,... While these points are used for secant method starts, a check for sign changes is also made. If one is found, the half interval method is used in combination with the secant method.

At appropriate times the sweeps through the intervals are halted and some auxiliary computations are made. These are No. 2 and No. 4 described below for the general search.

General Search. The main part of the search strategy is to generate a sequence of intervals to be searched. One may visualize the sequence graphically (NOT shown to scale).



Each of these intervals is searched using 3 to 5 initial points for the secant method (depending on the circumstances).

There are 5 auxiliary computations and tests made during the search. They are:

No. 1: Origin Shift. The points where the secant method terminates are examined and some retained. If F(X) is sufficiently small there, the origin is shifted to this point. Once the origin is not zero, the small intervals near the origin are no longer examined.

No. 2: Root Neighborhood Check. After a set (normally 8) of "larger" intervals are searched, the polyalgorithm stops and searches an interval about the roots already found.

No. 3: U-Shape Adjustment. As the expansion of the search proceeds away from the origin, one can easily move completely out of the realm of possible zeros. This is usually accompanied by the curve y=F(X) becoming U-shaped. A set of variables is maintained to measure this, and from time to time the origin is perturbed and the general search is restarted.

No. 4: Check of termination points of the secant method. Those points saved in No. 1 might well not result in an origin shift. From time to time, all points saved in this array are used as secant starting points. If these points are not found often and if nothing happens, they are then deleted from the array.

No. 5: Asymptote Checks. Asymptote limits are established and maintained at three places in the polyalgorithm. Once these are exceeded, the search in that phase is aborted. Too many violations of these limits terminates the polyalgorithm.

6. NUMERICAL METHODS FOR F(X)=0. Three basic methods are used: Secant, Half-Interval, and Descent. The Secant Method is fairly standard, the termination criteria used are

a) Iterates Converge
b) F(X) becomes small
c) Too far outside of requested interval
d) Asymptote to zero found
e) Too many iterations

The multiplicity of a root is estimated after 10 iterations and the method modified to take this into account. Provisions are made to force additional iterations in the presence of multiple roots.

The Half-Interval Method operates in conjunction with the secant method; i.e., at each new halfway point, the secant method is initiated for a short run. If none of these secant method attempts work, the point of sign change is classified as a discontinuity.

The Descent Method used is a simple descent on the function ABS(F(X)). It is useful (even essential) to have such a method to "refine" the location of a root whenever round-off effects become noticeable. It is used only after a root is "found" by the secant method.

7. ROOT ACCEPTANCE TESTS. The convergence of the secant method is not sufficient evidence to accept a number as a zero of F(X). Four other tests are used (XROOT = tentative root to be tested).

   Test 1: Is there a sign change very close to XROOT?

   Test 2: Is F(XROOT) much smaller than nearby values?

   Test 3: Is F(XROOT) = 0?

   Test 4: Is F(XROOT) somewhat smaller than nearby values and also absolutely small?

   If a tentative root fails all of these acceptance tests, the descent method is used to refine the root and the new value is retested.

8. F(X) DEFLATION. Let XROOT(I), I = 1,2,..., NROOT be the roots found with orders (multiplicities) ORD(I) and sign change indicators IND(I). We operate on the function

$$\mathrm{FCN(X)} = \frac{\mathrm{F(X)} * \left[\prod_{\mathrm{I=1}}^{\mathrm{NROOT}} |\mathrm{X-XROOT(I)}|^{\mathrm{ORD(I)}}\right]}{\prod_{\mathrm{I=1}}^{\mathrm{NROOT}} \left[|\mathrm{X-XROOT(I)}|^{\mathrm{ORD(I)}} * \left(\mathrm{SIGN}\left(\mathrm{X-XROOT(I)}\right)\right)^{\mathrm{IND(I)}}\right]}$$

   Our experience indicates that successful deflation depends upon the multiplicities of the roots being computed reasonably accurately. This is less critical if roots are of integer multiplicity and one may specify that all roots are simple if this is known a priori.

9. LOGICAL CONTROL. There is a multitude of small, local logical control decisions. The "overall" control depends on the relationships

   (i) between elapsed time and progress through search

   (ii) between number of roots found and progress through search

   (iii) between various quantities in the secant, half-interval, and descent methods and their correlation with the root acceptance tests and root order computations.

   The first two of these are used for termination and the third for deciding what tactic to use next.

10. HISTORICAL INFORMATION. There are a number of questions which the polyalgorithm is to be able to answer.

a) What roots were found?
b) What is the nature of the roots found?
c) How were they found and how "hard" were they to find?
d) What did the polyalgorithm do?
e) What is the polyalgorithm doing?
f) Debug dump (not intelligible to the average user)

The NAPSS system allows the user to

g) Request more information on certain points
h) Request additional effort - perhaps with changed specifications

It requires a rather large number of variables, print controls, information collecting statements, and output statements to answer these questions in a half reasonable way. It is more difficult to implement this phase of the polyalgorithm than one would think beforehand.

11. TESTING THE POLYALGORITHM. Reliability is the most critical attribute of a polyalgorithm for the automatic solution of a mathematical problem. However, oomplete reliability is unattainable and one can construct problems without difficulty which lead to erroneous results. Most such constructions are pathological in nature and thus irrelevant to the actual effectiveness of the polyalgorithm.

Testing is made more difficult because of the following two facts: a) one has a very limited number of "real life" problems to solve, (b) the bulk of these problems are routine and hence provide little contribution to measuring reliability. The result is that most of the functions used to test the polyalgorithm are artificial. These functions are described in more detail below.

Efficiency is the attribute with second priority. In fact, the development process consisted of first finding a way to handle a difficulty correctly and then improving the efficiency of the computation. The considerations of user convenience and flexibility came after a reasonably reliable and efficient polyalgorithm was available.

Note that the polyalgorithm is of such a nature that it cannot really compete (on the basis of efficiency) with simple minded schemes for simple problems. This polyalgorithm requires 15 to 20 function evaluations to find any zero (2-4 for initialization, 5 for secant method, 4 for root acceptance tests, 5 for order of the root).

12. THE TEST FUNCTIONS. The set of test functions is given explicitly in [4]. These functions (about 80 in all) have one or more of the properties listed below. We give a sample of each property and the number of the functions with this property.

a) Simple (25) $F(x)=\cos(x) - xe^x$

b) Clustered roots (7) $F(x) = (y+16)(\log_{10}(1+y^2))\sqrt{|y-8|}$ where $y = x-1312$

c) Multiple roots (14) $F(x) = (x-17)^2|x-17.1|^{1.8}(x-20)$

d) Fractional order roots (7) $F(x) = |x+157.2|^{1.5}|x-361.2|^{.7}(x-10^{-6})/|x-10^{-11}|$

e) Discontinuities (4) $F(x) = (1+x^2)\text{sgn}(\sin(x))$ $|x| \leq 3\pi$

f) Assymptotic to zero (6) $F(x) = 1/(1+|x|^3)$

g) Round off effects (5) $F(x) = 81-y(108-y(54-y(12-y)))$ where $y=x-1.11111$

h) Non-Functions (3) x = x+1. , F=0. (in Fortran)

i) Pathological (11) $s = \{1,.1,.01,.001,.0001,\ldots\}$

$F(x) = \text{distance}(x,s)$

j) Badly scaled (6)

$$F(x) = \begin{cases} 1+x^4 & |x| \leq 10^8 \\ 1.222*10^9-|x| & |x| > 10^8 \end{cases}$$

k) Real world problems (5) $F(x) = 20/y^{15} + 36/y^{25} + 40/y^{35} + 475/y^{40}$
$-1.12(y^{40}-1)/(xy^{40})-4.5 - 6/y^4 - 3/y^8$ where $y = 1+x$

The polyalgorithm gives results on these functions which are satisfactory to the author. Of course, it does not find all the roots of all of these functions and in some cases (not in the samples above) it finds roots which are debatable.

The amount of computation for some of these functions can be of the order of 10 or 20 seconds (IBM 7094), though it is less than 2 seconds for most of them. The efficiency for most of the cases can be dramatically increased if some a priori knowledge is available. Thus for the example for b) above, the work is cut by a factor of about 100 if the poly-algorithm is told that all three roots lie in the interval [1250,1350].

13. SAMPLE RESULT. Finally we give a sample of the results from the poly-algorithm for the example of d) above. The output level shown is 3 (of levels 0,1,2,3,4). The following is slightly rearranged from the actual computer output due to the difference in page size.

```
    WE FOUND 3 ROOTS IN 2. SECONDS
I  ROOT I          ORDER                   REMARKS

1  0.10000000E-01   1.00
    THIS ROOT WAS FOUND ON PASS NO. 1 THRU THE SEARCH AFTER 22 FUNCTION EVALUATIONS
    THE SECANT METHOD WENT FROM -0. TO 0.1000E-01 IN 4 ITERATIONS
      AND STOPPED WITH NORMAL CONVERGENCE

2  -0.15720000E 03  1.50
           THERE IS NO SIGN CHANGE AT THIS ROOT
    THIS ROOT WAS FOUND ON PASS NO.2 THRU THE SEARCH AFTER 84 FUNCTION EVALUATIONS
    THE SECANT METHOD WENT FROM -0.1572E 03 TO -0.1572E 03 IN 8 ITERATIONS
           AND STOPPED WITH NORMAL CONVERGENCE

3  0.36120000E 03   0.70
           THERE IS NO SIGN CHANGE AT THIS ROOT
    THIS ROOT WAS FOUND ON PASS NO. 8 THRU THE SEARCH AFTER 716 FUNCTION EVALUATIONS
    THE SECANT METHOD WENT FROM 0.3612E 03 TO 0.3612E 03 IN 18 ITERATIONS
           AND STOPPED WITH NORMAL CONVERGENCE

 WE EVALUATED F(X) 716 TIMES WITH SMALLEST AND LARGEST X-VALUES OF -0.18639E 04
             0.15495E 04
 THE SECANT METHOD WAS STARTED AT 48 POINTS

 WE SEARCHED THE FOLLOWING 18 INTERVALS IN THE ORDER GIVEN
           READ LEFT TO RIGHT, THEN DOWN
 (-0.98765E 00, 0.98765E 00) (-0.15819E 03,-0.15621E 03) (-0.16905E 03,-0.15878E 03)
             (-0.15562E 03,--0.14535E 03)
 (-0.29942E 03,-0.17616E 03) (-0.13824E 03,-0.14977E 02) (-0.18639E 04,-0.38475E 03)
             ( 0.70356E 02, 0.15495E 04)
 (-0.13321E 03,-0.13123E 03) (-0.14111E 03,-0.13340E 03) (-0.13103E 03,-0.12333E 03)
             (-0.21222E 03,-0.14289E 03)
 (-0.12155E 03,-0.52219E 02) (-0.85222E 03,-0.22822E 03) (-0.36219E 02, 0.58778E 03)
             (-0.16034E 03,-0.15406E 03)
 (-0.16656E 03,-0.16002E 03) ( 0.36842E 03, 0.35398E 03) (

 THE NUMBER OF ORIGINS USED IN THE SEARCH IS 2, IN THE ORDER GIVEN
           0.            -0.157199E 03

 DATA IS GIVEN WHEN THE SEARCH WAS RESTARTED NEAR THE ORIGIN BECAUSE OF LARGE FUNCTION
           VALUES FOUND FOR LARGE X-VALUES
      EXPANSION STOPPED ON PASS 5 AVERAGE F-VALUE= 0.670E 06, LEFT, RIGHT EXTREME
           VALUES= 0.155E 08, 0.100E 08
```

REFERENCES

1. Symes, L.R., Evaluation of NAPSS Expressions Involving Polyalgorithms, Functions, Recusion, and Untyped Variables, CSD TR 33, February 1969.
2. Symes, L.R. and Roman, R.V., Structure of a language for a numerical analysis problem solving system in "Interactive Systems for Experimental Applied Mathematics", ed. M. Klerer and J. Reinfelds, Academic Press, 1968, pp. 67-78.
3. Rice, J.R., On the construction of polyalgorithms for automatic numerical analysis, in "Interactive Systems for Experimental Applied Mathematics" ed. M. Klerer and J. Reinfelds, Academic Press, 1968, pp. 301-313.
4. Rice, J.R., A set of 80 test functions for nonlinear equation solvers, CSD TR 34, March 1969.
5. Rice, J.R. and Rosen, S., NAPSS, Numerical Analysis and Problem Solving System, Proc. ACM 21st Natl. Conf., Los Angeles, 1966, ACM Publ. P-66, pp. 51-56.
6. Roman, R. V. and Symes, L.R., Implementation considerations in a numerical analysis problem solving system, in "Interactive Systems for Experimental Applied Mathematics" ed. M. Klerer and J. Reinfelds, Academic Press, 1968, pp. 400-410.