The REDUCE Root Finding Package
Mod 1.94, 28 May 1993

Stanley L. Kameny
E-mail: valley!stan@rand.org

1 Introduction

The root finding package is designed so that it can be used as an independent package, or it can be integrated with and called by `SOLVE`. This document describes the package in its independent use. It can be used to find some or all of the roots of univariate polynomials with real or complex coefficients, to the accuracy specified by the user.

2 Root Finding Strategies

For all polynomials handled by the root finding package, strategies of factoring are employed where possible to reduce the amount of required work. These include square-free factoring and separation of complex polynomials into a product of a polynomial with real coefficients and one with complex coefficients. Whenever these succeed, the resulting smaller polynomials are solved separately, except that the root accuracy takes into account the possibility of close roots on different branches. One other strategy used where applicable is the powergcd method of reducing the powers of the initial polynomial by a common factor, and deriving the roots in two stages, as roots of the reduced power polynomial. Again here, the possibility of close roots on different branches is taken into account.

3 Top Level Functions

The top level functions can be called either as symbolic operators from algebraic mode, or they can be called directly from symbolic mode with symbolic mode arguments. Outputs are expressed in forms that print out correctly in algebraic mode.

3.1 Functions that refer to real roots only

Three top level functions refer only to real roots. Each of these functions can receive 1, 2 or 3 arguments.

The first argument is the polynomial p, that can be complex and can have multiple or zero roots. If arg2 and arg3 are not present, all real roots are found. If the additional arguments are present, they restrict the region of consideration.

- If arguments are (p,arg2) then Arg2 must be POSITIVE or NEGATIVE. If arg2=NEGATIVE then only negative roots of p are included; if arg2=POSITIVE then only positive roots of p are included. Zero roots are excluded.
- If arguments are (p,arg2,arg3) then Arg2 and Arg3 must be r (a real number) or EXCLUDE r, or a member of the list POSITIVE, NEGATIVE, INFINITY, -INFINITY. EXCLUDE r causes the value r to be excluded from the region. The order of the sequence arg2, arg3 is unimportant. Assuming that arg2 $\leq$ arg3 when both are numeric, then

{-INFINITY,INFINITY}	is equivalent to	{} represents all roots;
{arg2,NEGATIVE}	represents	$-\infty < r < arg2$;
{arg2,POSITIVE}	represents	$arg2 < r < \infty$;

  In each of the following, replacing an *arg* with EXCLUDE *arg* converts the corresponding inclusive $\leq$ to the exclusive $<$

{arg2,-INFINITY}	represents	$-\infty < r \leq arg2$;
{arg2,INFINITY}	represents	$arg2 \leq r < \infty$;
{arg2,arg3}	represents	$arg2 \leq r \leq arg3$;

- If zero is in the interval the zero root is included.

REALROOTS This function finds the real roots of the polynomial p, using the REALROOT package to isolate real roots by the method of Sturm

sequences, then polishing the root to the desired accuracy. Precision of computation is guaranteed to be sufficient to separate all real roots in the specified region. (cf. MULTIROOT for treatment of multiple roots.)

ISOLATER This function produces a list of rational intervals, each containing a single real root of the polynomial p, within the specified region, but does not find the roots.

RLROOTNO This function computes the number of real roots of p in the specified region, but does not find the roots.

3.2 Functions that return both real and complex roots

ROOTS p; This is the main top level function of the roots package. It will find all roots, real and complex, of the polynomial p to an accuracy that is sufficient to separate them and which is a minimum of 6 decimal places. The value returned by ROOTS is a list of equations for all roots. In addition, ROOTS stores separate lists of real roots and complex roots in the global variables ROOTSREAL and ROOTSCOMPLEX.

The order of root discovery by ROOTS is highly variable from system to system, depending upon very subtle arithmetic differences during the computation. In order to make it easier to compare results obtained on different computers, the output of ROOTS is sorted into a standard order: a root with smaller real part precedes a root with larger real part; roots with identical real parts are sorted so that larger imaginary part precedes smaller imaginary part. (This is done so that for complex pairs, the positive imaginary part is seen first.)

However, when a polynomial has been factored (by square-free factoring or by separation into real and complex factors) then the root sorting is applied to each factor separately. This makes the final resulting order less obvious. However it is consistent from system to system.

ROOTS_AT_PREC p; Same as ROOTS except that roots values are returned to a minimum of the number of decimal places equal to the current system precision.

ROOT_VAL p; Same as ROOTS_AT_PREC, except that instead of re-

turning a list of equations for the roots, a list of the root value is returned. This is the function that SOLVE calls.

NEARESTROOT(p,s); This top level function uses an iterative method to find the root to which the method converges given the initial starting origin s, which can be complex. If there are several roots in the vicinity of s and s is not significantly closer to one root than it is to all others, the convergence could arrive at a root that is not truly the nearest root. This function should therefore be used only when the user is certain that there is only one root in the immediate vicinity of the starting point s.

FIRSTROOT p; ROOTS is called, but only the first root determined by ROOTS is computed. Note that this is not in general the first root that would be listed in ROOTS output, since the ROOTS outputs are sorted into a canonical order. Also, in some difficult root finding cases, the first root computed might be incorrect.

3.3 Other top level functions

GETROOT(n,rr); If rr has the form of the output of ROOTS, REALROOTS, or NEARESTROOTS; GETROOT returns the rational, real, or complex value of the root equation. An error occurs if $n < 1$ or $n >$ the number of roots in rr.

MKPOLY rr; This function can be used to reconstruct a polynomial whose root equation list is rr and whose denominator is 1. Thus one can verify that if $rr := ROOTS\ p$, and $rr1 := ROOTS\ MKPOLY\ rr$, then $rr1 = rr$. (This will be true if `MULTIROOT` and `RATROOT` are ON, and `ROUNDED` is off.) However, $MKPOLY\ rr - NUM\ p = 0$ will be true if and only if all roots of p have been computed exactly.

3.4 Functions available for diagnostic or instructional use only

GFNEWT(p,r,cpx); This function will do a single pass through the function GFNEWTON for polynomial p and root r. If cpx=T, then any complex part of the root will be kept, no matter how small.

GFROOT(p,r,cpx); This function will do a single pass through the func-

tion GFROOTFIND for polynomial p and root r. If cpx=T, then any complex part of the root will be kept, no matter how small.

4 Switches Used in Input

The input of polynomials in algebraic mode is sensitive to the switches `COMPLEX`, `ROUNDED`, and `ADJPREC`. The correct choice of input method is important since incorrect choices will result in undesirable truncation or rounding of the input coefficients.

Truncation or rounding may occur if `ROUNDED` is on and one of the following is true:

1. a coefficient is entered in floating point form or rational form.
2. `COMPLEX` is on and a coefficient is imaginary or complex.

Therefore, to avoid undesirable truncation or rounding, then:

1. `ROUNDED` should be off and input should be in integer or rational form; or
2. `ROUNDED` can be on if it is acceptable to truncate or round input to the current value of system precision; or both `ROUNDED` and `ADJPREC` can be on, in which case system precision will be adjusted to accommodate the largest coefficient which is input; or
3. if the input contains complex coefficients with very different magnitude for the real and imaginary parts, then all three switches `ROUNDED`, `ADJPREC` and `COMPLEX` must be on.

integer and complex modes (off `ROUNDED`) any real polynomial can be input using integer coefficients of any size; integer or rational coefficients can be used to input any real or complex polynomial, independent of the setting of the switch `COMPLEX`. These are the most versatile input modes, since any real or complex polynomial can be input exactly.

modes rounded and complex-rounded (on `ROUNDED`) polynomials can be input using integer coefficients of any size. Floating point coefficients will be truncated or rounded, to a size dependent upon the system. If complex is on, real coefficients can be input to any precision using integer form, but coefficients of imaginary parts of complex

coefficients will be rounded or truncated.

5 Internal and Output Use of Switches

The REDUCE arithmetic mode switches `ROUNDED` and `COMPLEX` control the behavior of the root finding package. These switches are returned in the same state in which they were set initially, (barring catastrophic error).

COMPLEX The root finding package controls the switch `COMPLEX` internally, turning the switch on if it is processing a complex polynomial. For a polynomial with real coefficients, the starting point argument for NEARESTROOT can be given in algebraic mode in complex form as rl + im * I and will be handled correctly, independent of the setting of the switch `COMPLEX`. Complex roots will be computed and printed correctly regardless of the setting of the switch `COMPLEX`. However, if `COMPLEX` is off, the imaginary part will print out ahead of the real part, while the reverse order will be obtained if COMPLEX is on.

ROUNDED The root finding package performs computations using the arithmetic mode that is required at the time, which may be integer, Gaussian integer, rounded, or complex rounded. The switch `BFTAG` is used internally to govern the mode of computation and precision is adjusted whenever necessary. The initial position of switches `ROUNDED` and `COMPLEX` are ignored. At output, these switches will emerge in their initial positions.

6 Root Package Switches

Note: switches `AUTOMODE`, `ISOROOT` and `ACCROOT`, present in earlier versions, have been eliminated.

RATROOT (Default OFF) If `RATROOT` is on all root equations are output in rational form. Assuming that the mode is `COMPLEX` (i.e. `ROUNDED` is off,) the root equations are guaranteed to be able to be input into REDUCE without truncation or rounding errors. (Cf. the function MKPOLY described above.)

MULTIROOT (Default ON) Whenever the polynomial has complex coefficients or has real coefficients and has multiple roots, as determined

by the Sturm function, the function `SQFRF` is called automatically to factor the polynomial into square-free factors. If `MULTIROOT` is on, the multiplicity of the roots will be indicated in the output of ROOTS or REALROOTS by printing the root output repeatedly, according to its multiplicity. If `MULTIROOT` is off, each root will be printed once, and all roots should be normally be distinct. (Two identical roots should not appear. If the initial precision of the computation or the accuracy of the output was insufficient to separate two closely-spaced roots, the program attempts to increase accuracy and/or precision if it detects equal roots. If, however, the initial accuracy specified was too low, and it was not possible to separate the roots, the program will abort.)

TRROOT (Default OFF) If switch `TRROOT` is on, trace messages are printed out during the course of root determination, to show the progress of solution.

ROOTMSG (Default OFF) If switch `ROOTMSG` is on in addition to switch `TRROOT,` additional messages are printed out to aid in following the progress of Laguerre and Newton complex iteration. These messages are intended for debugging use primarily.

7 Operational Parameters and Parameter Setting.

ROOTACC# (Default 6) This parameter can be set using the function ROOTACC n; which causes `ROOTACC#` to be set to MAX(n,6). If `ACCROOT` is on, roots will be determined to a minimum of `ROOTACC#` significant places. (If roots are closely spaced, a higher number of significant places is computed where needed.)

system precision The roots package, during its operation, will change the value of system precision but will restore the original value of system precision at termination except that the value of system precision is increased if necessary to allow the full roots output to be printed.

PRECISION n; If the user sets system precision, using the command PRECISION n; then the effect is to increase the system precision to n, and to have the same effect on ROOTS as ROOTACC n; ie. roots will now be printed with minimum accuracy n. The original conditions can then be restored by using the command PRECISION RESET; or PRECISION NIL;.

ROOTPREC n; The roots package normally sets the computation mode and precision automatically. However, if ROOTPREC n; is called and n is greater than the initial system precision then all root computation will be done initially using a minimum system precision n. Automatic operation can be restored by input of ROOTPREC 0;.

8 Avoiding truncation of polynomials on input

The roots package will not internally truncate polynomials. However, it is possible that a polynomial can be truncated by input reading functions of the embedding lisp system, particularly when input is given in floating point (rounded) format.

To avoid any difficulties, input can be done in integer or Gaussian integer format, or mixed, with integers or rationals used to represent quantities of high precision. There are many examples of this in the test package. It is usually best to let the roots package determine the precision needed to compute roots.

The number of digits that can be safely represented in floating point in the lisp system are contained in the global variable `!!NFPD`. Similarly, the maximum number of significant figures in floating point output are contained in the global variable `!!FLIM`. The roots package computes these values, which are needed to control the logic of the program.

The values of intermediate root iterations (that are printed when `TRROOT` is on) are given in bigfloat format even when the actual values are computed in floating point. This avoids intrusive rounding of root printout.