Raabo.

A/S REGNECENTRALEN

SCANDINAVIAN INFORMATION PROCESSING SYSTEMS

RCSL no: 53-M4

Edition: May 1970

Author: J. Runge-Erichsen

Title: pzero

Keywords: RC 4000, Software, pzero, Polynomials, Algol Procedure, ISO Tape

Abstract: The boolean procedure pzero(order, coef, root) calculates all roots (complex or real) of the polynomial p(z) = SUM(coef(i)×z××i) i = 0, 1, ... , order, order = 2, 3, 4. 14 pages.

pzero(order, coef, root)

1. Function and parameters

pzero calculates all roots (complex or real) of 2nd, 3rd and 4th order polynomials with real coefficients.

Call: pzero(order, coef, root)

pzero
: is a boolean procedure which is false if order > 4 or order < 2 or coef(order) = 0. In this case no computations are made, otherwise pzero is true.

order
: (call value, integer)
  specifies the order of the polynomial.

coef
: (call value, array) minimum bounds(o:order)
  specifies the coefficients of the polynomial
  p(z):= SUM(coef(i)×z××i), i:= 0, 1, ... , order.

root
: (return value, array) minimum bounds(1:order, 1:2)
  If pzero is true then root specifies the calculated roots
  zi, i:= 1, 2, ... , order so that
  Re zi = root(i, 1) and
  Im zi = root(i, 2).

2. Method

2.1. General

Extremely large or small roots are detected at the first stage of computation, and further computations are executed on the quotient polymial, where quotient polynomial everywhere in this description means p(z) / PRODUCT(z - zi)
where zi are the roots allready found by pzero.
The (quotient) polynomial is now normalized so that coef(order) equals 1.

2.2. Second order polynomials: $p(z) = z\times\times 2 + b\times z + c$

The quantity $d := b\times\times 2 - 4\times c$ determines whether the roots are complex ($d < 0$) or real ($d > 0$).

If the roots are real then the numerically largest root is calculated from

$$z1 := (-b - sqn(b)\times sqrt(d))/2$$

and the remaining root from

$$z2 := \text{if } z1 = 0 \text{ then } 0 \text{ else } c/z1$$

otherwise the roots are calculated from

$$z1 := (-b + i\times sqrt(-d))/2$$

and

$$z2 := (-b - i\times sqrt(-d))/2$$

where i is the imaginary unit.

2.3. Third order polynomials: $p(z) = z\times\times 3+a\times z\times\times 2+b\times z+c$

If there are multiple roots then all of the roots are calculated directly from the coefficients a,b and c, otherwise one real root is determined by a Newton Iteration whereafter the remaining two roots are calculated from the quotient second order polynomial (see 2.2).

Analysis and iteration starting point:

The transformation $w = z + a/3$ yields

$$p(z) = 0 \iff q(w) = w\times\times 3 + d\times w + e = 0.$$

Define

$$r := 27\times e\times\times 2 + 4\times d\times\times 3.$$

a) 1 real and 2 complex conjugate roots:

are called $2\times k$, $-k + i\times m$ and $-k - i\times m$
where i is the imaginary unit.

$$q(w) = w\times\times 3 + (m\times\times 2 - 3\times k\times\times 2)\times w - 2\times k\times (k\times\times 2 + m\times\times 2)$$

implies

$$r = [2\times m\times (m\times\times 2 + 9\times k\times\times 2)]\times\times 2 \geq 0$$

and defining

$$f(m) = 4 \times |e| = 8\times |k\times (k\times\times 2 + m\times\times 2)| \geq 8\times |k|\times\times 3$$

yields

$$(4\times |e|)\times\times (1/3) \geq 2\times |k|$$

b) 3 real roots:

are called k, m and -k - m where k is the numerically largest root.

$$q(w) = w\times\times 3 - (k\times\times 2 + m\times\times 2 + k\times m)\times w + k\times m\times(k + m)$$

implies

$$r = -[(k - m)\times(2\times k + m)\times(k + 2\times m)]\times\times 2 \leq 0 \qquad (\times\times)$$

and defining

$$f(m) = 4\times|d|/3 = 4\times|k\times\times 2 + m\times\times 2 + k\times m|/3$$

yields

$$\min f(m) = f(-d/2) = k\times\times 2$$

so

$$2 \times \mathrm{sqrt}(|d|/3) \geq |k|$$

From $(\times)$ and $(\times\times)$ we see that

$r < 0 \Rightarrow$ real roots

$r = 0 \Rightarrow$ multiple roots

$r > 0 \Rightarrow$ complex roots

and iteration starting point s is chosen to be

$$s := 2\times \mathrm{sqrt}(|d|/3) \text{ if } r < 0$$
$$(4\times|e|)\times\times(1/3) \text{ if } r > 0.$$

The quotient second order polynomial:

is calculated from

$$(z\times\times 2 + p\times z + q)\times(z - z1) = z\times\times 3 + a\times z\times\times 2 + b\times z + c$$

where z1 is the real root obtained by iteration.

If $|a + z1| > |a|/8$ then p is calculated from

$$p := a + z1$$

and

$$q := b + p\times z1 \quad \text{if} \quad |b + p\times z1| \geq |b|/8$$
$$- c/z1 \quad \text{if} \quad |b + p\times z1| < |b|/8$$

otherwise

$$q := -c/z1$$

and

$$p := (q - b)/z1.$$

2.4. Fourth order polynomials: $p(z) = z^4 + a z^3 + b z^2 + c z + d$

I: A linear transformation $w = z + a/4$ yields

$$p(z) = 0 \Leftrightarrow q(w) = w^4 + l w^2 + m w + n = 0.$$

now the sum of the transformed roots equals zero.

II: The transformed roots are calculated using the method of Descartes. This method involves the solution of a third order equation, and this is performed as described in 2.3.

III: The roots are now accepted if $|\mathrm{Re}\ zi| > |a|/32$ so at least one root must be accepted unless they all equal zero.

IV a) <u>if one root is accepted by III</u>

then the reciprocal roots are calculated from

$$d z^4 + c z^3 + b z^2 + a z + 1 = 0$$

as described in 2.4-I, II and III.

If one of the reciprocal roots is accepted then we have two accepted roots, so the remaining two roots are calculated as described in IV b, otherwise the former accepted root is not used. The number of accepted reciprocal roots then determines whether further calculations are performed as described in IV, b, c or d.

b) <u>if two roots are accepted by III (or IV)</u>

then the quotient second order polynomial is calculated and solved as described in 2.2.

c) <u>if three roots are accepted by III (or IV)</u>

then the remaining real root is calculated using the fact that the product of the roots equals d.

d) <u>if four roots are accepted by III (or IV)</u>

then no further calculations are performed by pzero.

3. Accuracy, time- and storage requirements

3.1. Accuracy

If an actual equation is ill-conditioned and you want the roots to a specified degree of accuracy a much greater accuracy may be necessary in the intermediate calculations. On the other hand a user is not supposed to know anything about the conditioning of the actual equation, so standard input to RC4000 of 48-bits reals is used.

3.2. Time- and storage requirements

Approximate cpu-time used by pzero:	(order - 1)×0.02 sec.
Codelength:	12 segments
Typographical length:	223 lines incl. last comment.

4. Test and discussion:

pzero has been tested on the RC4000 computer with a testprogram which performs

1) generation of order and coefficients
2) call of pzero
3) calculation of root generated coefficients of the polynomial
   p(z) := PRODUCT(z - zi), i:= 1, 2, ... , order
4) calculation of relative differences between the original and the root generated coefficients.

Now the smallness of the differences is chosen as a measure of the goodness of pzero.
pzero has been tested with a large number of both prepared ill-conditioned coefficients and random coefficients input and in both cases with satisfying results.

Some test examples (the check column describes the relative differences):

example number 1

given equation check

coef(4)=	1.0000000000$_{10}$ 0		
coef(3)=	1.0000000000$_{10}$ 0	-5.82$_{10}$-11	
coef(2)=	1.0000000000$_{10}$ 0	-5.82$_{10}$-11	
coef(1)=	1.0000000000$_{10}$ -7	-5.90$_{10}$ -4	
coef(0)=	1.0000000000$_{10}$ 0	0.00$_{10}$ 0	

calculated roots

3.5180794434$_{10}$ -1+7.2034175772$_{10}$	-1	×i
3.5180794434$_{10}$ -1-7.2034175772$_{10}$	-1	×i
-8.5180794432$_{10}$ -1+9.1129213536$_{10}$	-1	×i
-8.5180794432$_{10}$ -1-9.1129213536$_{10}$	-1	×i

example number 2

given equation check

coef(4)=	1.0000000000$_{10}$ 0		
coef(3)=	-6.8619274672$_{10}$ -1	0.00$_{10}$ 0	
coef(2)=	-8.8228487860$_{10}$ -1	-6.60$_{10}$-11	
coef(1)=	6.8619274672$_{10}$ -1	0.00$_{10}$ 0	
coef(0)=	-1.1771512141$_{10}$ -1	0.00$_{10}$ 0	

calculated roots

3.4309637339$_{10}$	-1
1.0000000000$_{10}$	0
3.4309637335$_{10}$	-1
-1.0000000000$_{10}$	0

example number 3

given equation check

coef(4)=	1.0000000000$_{10}$ 0		
coef(3)=	-1.4215286873$_{10}$ 1	0.00$_{10}$ 0	
coef(2)=	7.1429889252$_{10}$ 1	1.04$_{10}$ -10	
coef(1)=	-1.4369489911$_{10}$ 2	3.63$_{10}$ -10	
coef(0)=	8.5480296736$_{10}$ 1	6.97$_{10}$ -10	

calculated roots

4.4050104852$_{10}$	0
4.4051469640$_{10}$	0
4.4051294242$_{10}$	0
9.9999999956$_{10}$	-1

example number 4

given equation check

coef(4)= 1.0000000000₁₀	0		
coef(3)=-2.4628394422₁₀	0	4.32₁₀	-11
coef(2)= 2.7192690981₁₀	0	0.00₁₀	0
coef(1)=-1.2204258468₁₀	0	4.77₁₀	-11
coef(0)= 2.0540067855₁₀	-1	1.42₁₀	-10

calculated roots

6.7204851918₁₀	-1+1.1559340115₁₀	-2	x1
6.7204851918₁₀	-1-1.1559340115₁₀	-3	x1
6.7437120188₁₀	-1+1.1667236046₁₀	-3	x1
6.7437120188₁₀	-1-1.1667236046₁₀	-3	x1

example number 5

given equation check

coef(4)= 1.0000000000₁₀	0		
coef(3)=-5.9418329952₁₀	0	0.00₁₀	0
coef(2)= 1.3239517255₁₀	1	0.00₁₀	0
coef(1)=-1.3111166744₁₀	1	7.10₁₀	-11
coef(0)= 4.8690226978₁₀	0	2.39₁₀	-10

calculated roots

1.4854582489₁₀	0		
1.4844816864₁₀	0		
1.4859465301₁₀	0+8.4572793336₁₀	-4	x1
1.4859465301₁₀	0-8.4572793336₁₀	-4	x1

6. Complete algol text:

```
pzero=set 12
pzero=algol
external
message pzero,version 22/5-70,RCSL 53-M4;
boolean procedure pzero(order,coef,root);
value                 order;
integer               order;
array                        coef,root;
```

```
begin
array arr(0:4);
integer accept,i;
real x,push,a,b,c,d;
boolean ok;

procedure order4;
begin
real a,b,c,d,x,push;
integer i;
   push:=arr(3)/4;
   c:=((-3×push××2+arr(2))×push-arr(1))×push+arr(0);
   b:=(push×arr(3)-arr(2))×2×push+arr(1);
   a:=-3×arr(3)××2/8+arr(2);
   if b<>0 then
   begin
      order3(2×a,a××2-4×c,-b××2);
      for i:=0,1+i while root(i,2)<>0 or root(i,1)<0 do;
      x:=root(i,1);
      d:=b;
      b:=a+x;
      a:=sqrt(x);
      x:=d/a;
      if abs(b-x)>abs(b+x) then b:=b-x else
      begin
         b:=b+x;
         a:=-a
      end;
      b:=b/2
   end     else
   if a××2<4×c then
   begin
      b:=sqrt(c);
      a:=sqrt(2×b-a)
   end          else
   begin
      b:=a+sgn(a)×sqrt(a××2-4×c)/2;
      a:=0
   end;
```

```
    order2(a,b,1);
    order2(-a,if b=0 then 0 else c/b,3);
    x:=abs push/8;
    for i:=1,2,3,4 do
    if abs(root(i,1)-push)>x then
    begin
       accept:=1+accept;
       root(accept,1):=root(i,1)-push;
       root(accept,2):=root(i,2)
    end;
exit:
end order4;



procedure order3(a,b,c);
value            a,b,c;
real             a,b,c;
begin
real push,p,q,r;
   push:=-a/3;
   p:=a××2-3×b;
   q:=(-2×push××2+b)×push+c;
   r:=(27×c-a×(18×b-4×a××2))×c+b××2×(4×b-a××2);
   if abs r<=((27×abs c+abs a×(18×abs b+4×a××2))×abs c
             +b××2×(4×abs b+a××2))×3₁₀-11
   then
   begin
      d:=(a××2+3×abs b)×3₁₀-11;
      if p+d<0 then goto newton;
      q:=if p-d<0 then 0 else sgn(q)×sqrt(p)/3;
      root(1,1):=root(2,1):=push+q;
      root(3,1):=push-2×q;
      root(1,2):=root(2,2):=root(3,2):=0;
      goto exit
   end;
```

```
newton:
   r:=push-sgn(q)×(if r<0  and p>=0 then 2×sqrt(p)/3 else(4×abs q)××(1/3));
   for p:=((2×r+a)×r××2-c)/((3×r+2×a)×r+b),
          ((2×r+a)×r××2-c)/((3×r+2×a)×r+b)
          while abs(p-push)<abs(r-push) do r:=p;
   root(1,1):=r;
   root(1,2):=0;
   p:=a+r;
   q:=b+p×r;
   q:=if abs p<abs a/8 or abs q<abs b/8 then  -c/r else q;
   p:=if abs p<abs a/8 then (q-b)/r else p;
   order2(p,q,2);
exit:
end order3;

procedure order2(b,c,first);
value            b,c,first;
real             b,c;
integer              first;

begin
real d;
   d:=b××2-4×c;
   d:=sgn(d)×sqrt(abs d);
   if d<0 then
   begin
      root(first,1):=root(1+first,1):=-b/2;
      root(first,2):=d/2;
      root(1+first,2):=-d/2
   end  else
   begin
      d:=root(first,1):=(-b-sgn(b)×d)/2;
      root(1+first,1):=if d=0 then 0 else c/d;
      root(first,2):=root(1+first,2):=0
   end
end order2;

   accept:=0;
   ok:=pzero:=order>1 and order<5 and coef(order)<>0;
```

```
   if ¬,ok then goto finis;
   for i:=order step -1 until 0 do arr(i):=coef(i);
low:
   x:=if arr(1)=0 then arr(0) else -arr(0)/arr(1);
   for i:=0,1+i while arr(i)-arr(1+i)×x=arr(i) do
   if i=order-1 then
   begin
     for i:=0 step 1 until order-1 do arr(i):=arr(1+i);
     goto comb
   end;
   x:=-arr(order-1)/arr(order);
   for i:=0,1+i while arr(i)×x-arr(i-1)=arr(i)×x do
   if i=order-1 then goto comb;
   goto normal;
comb:
   root(order,1):=x;
   root(order,2):=0;
   order:=order-1;
   if order>1 then goto low;
   root(1,1):=-arr(0)/arr(1);
   root(1,2):=0;
   goto finis;
normal:
   x:=arr(order);
   for i:=order step -1 until 0 do arr(i):=arr(i)/x;
   case order-1 of
   begin
      order2(arr(1),arr(0),1);
      order3(arr(2),arr(1),arr(0));
      begin
         order4;
select:  case  accept of
         begin
            begin
               arr(4):=root(1,1);
               x:=coef(0);
               for i:=0,1,2,3 do arr(i):=coef(4-i)/x;
               accept:=0;
```

```
   order4;
   if accept>1 then
   begin
      for i:=1,1+i while i<=accept and i<5 do
      if root(i,2)=0 then root(i,1):=1/root(i,1)
                          else
      begin
         x:=root(i,1)××2+root(i,2)××2;
         root(i,1):=root(1+i,1):=root(i,1)/x;
         root(i,2):=root(i,2)/x;
         root(1+i,2):=-root(i,2);
         i:=1+i
      end;
   end                    else
   begin
      root(2,1):=1/root(1,1);
      root(1,1):=arr(4);
      accept:=2
   end;
   x:=coef(4);
   for i:=0,1,2,3 do arr(i):=coef(i)/x;
   goto select
end;


begin
   d:=-root(1,1)-root(2,1);
   c:=root(1,1)×root(2,1)-root(1,2)×root(2,2);
   b:=arr(0)/c;
   a:=if abs (arr(1)/b-d)<abs (arr(3)-d)
      then arr(3)-d
      else (arr(1)-b×d)/c;
   order2(a,b,3)
end;
```

```
            begin
               a:=if root(1,2)=0
                  then root(1,1)×(root(2,1)×root(3,1)-root(2,2)×root(3,2))
                  else root(3,1)×(root(1,1)××2+root(1,2)××2);
               root(4,1):=arr(0)/a;
               root(4,2):=0
            end;;;
         end
      end
   end;
finis:
end pzero;

comment:
   pzero(order,coef,root) calculates real and complex roots
   of 2nd, 3rd and 4th order polynomials with real coefficients:
      p(z)= coef(order)×z××order+...+coef(1)×z+coef(0).

   pzero is false if order>4 or order<2 or coef(order)=0,
   otherwise pzero is true.

   order   (call value,integer) specifies the order of the
                                polynomial.
   coef    (call value,array) specifies the coefficients of
                              the polynomial.
   root    (return value,array).
           If pzero is true then root specifies the roots of
           the polynomial: zi, i=1,2,...,order so that
           Re zi = root(i,1)
           Im zi = root(i,2);


end;
```