Aplikace matematiky

Pavla Holasová

Algorithms. 26. GARSIDE. Determination of the roots of a polynomial according to Garside, Jarrat and Mack

Aplikace matematiky, Vol. 17 (1972), No. 2, 157–167

Persistent URL: http://dml.cz/dmlcz/103404

ALGORITMY

26. GARSIDE

BESTIMMUNG DER WURZELN EINES POLYNOMS NACH GARSIDE, JARRAT UND MACK

PAVLA HOLASOVÁ, Matematicko-fysikální fakulta KU, Praha

Procedure *GARSIDE* (n, rco, ico);
integer n;
array rco, ico;
Comment

n	Grad des Polynoms $f(z)$
eps	Genauigkeitsschranke
it	Anzahl der Iterationsschritte
$it2$	Anzahl der Starts mit neuen Anfangswerten
w	Grösse zur Bestimmung der drei Anfangswerte
$rco[i]$	Realteil des i-ten Koeffizienten von $f(z)$
$ico[i]$	Imaginärteil des i-ten Koeffizienten von $f(z)$
$drco[i]$	Realteil des i-ten Koeffizienten von $f'(z)$
$dico[i]$	Imaginärteil des i-ten Koeffizienten von $f'(z)$
zrv $(v = 1, 2, 3)$	Realteil des v-ten Wertes zur Bestimmung von $zr4$
ziv $(v = 1, 2, 3)$	Imaginärteil des v-ten Wertes zur Bestimmung von $zi4$
frv $(v = 1, 2, 3)$	Realteil von (1) $f(zv)$, (2) $f'(zv)/f(zv)$
fiv $(v = 1, 2, 3)$	Imaginärteil von (1) $f(zv)$, (2) $f'(zv)/f(zv)$;

begin
integer $i, k, it, it2$;
real $zr0, zi0, zra, zia, zr1, zi1, zr2, zi2, zr3, zi3, fr1, fi1, fr2, fi2, fr3, fi3, w, eps, w1$;
array $drco, dico\ [0 : n]$;
real procedure $re\ (a, b, c, d)$;
real a, b, c, d;
begin $re := a \times c - b \times d$ **end**;
real procedure $im\ (a, b, c, d)$;
real a, b, c, d;
begin $im := a \times d + b \times c$ **end**;
real procedure $re1\ (a, b, c, d, f, g)$;

```
real a, b, c, d, f, g;
begin re1 := a × c × f − b × d × f − a × d × g − b × c × g end;
real procedure im1 (a, b, c, d, f, g);
real a, b, c, d, f, g;
begin im1 := a × d × f + b × c × f + a × c × g − b × d × g end;
real procedure re2 (a, b, c, d);
real a, b, c, d;
begin re2 := a × a × c − b × b × c − 2 × a × b × d end;
real procedure im2 (a, b, c, d);
real a, b, c, d;
begin im2 := 2 × a × b × c + a × a × d − b × b × d end;
procedure itpol (END);
label END;
begin boolean b1, b2, b3, bo1, bo2;
          real h1, h2, h3, h4, j1, j2, j3, j4, r1, r2, r3, r4, r5, r6, r7, i1, i2, i3, i4, i5,
          i6, i7, q, q1, q2, q3, p, ar1, ar2, ar3, ai1, ai2, ai3, zhr, zhi, frh, fih, min;
          b1 := b2 := b3 := bo1 := bo2 := false;

ANEW:     h1 := zr1 − zr2; j1 := zi1 − zi2;
          h2 := fr2 − fr1; j2 := fi2 − fi1;
          h3 := re2(zr1, zi1, fr1, fi1);
          j3 := im2(zr1, zi1, fr1, fi1);
          h4 := re2(zr2, zi2, fr2, fi2);
          j4 := im2(zr2, zi2, fr2, fi2);

NEWGO:    r1 := zr2 − zr3; i1 := zi2 − zi3; r2 := zr3 − zr1;
          i2 := zi3 − zi1; r3 := fr3 − fr2; i3 := fi3 − fi2;
          r4 := re2(zr3, zi3, fr3, fi3);
          i4 := im2(zr3, zi3, fr3, fi3);
          r5 := re(r1, i1, h2, j2); i5 := im(r1, i1, h2, j2);
          r6 := −r5 + re(h1, j1, r3, i3);
          i6 := −i5 + im(h1, j1, r3, i3);
          q  := r6 × r6 + i6 × i6;
          if q <₁₀−20 then b1 := true else

          begin ar1 := zr3 + (re1(r2, i2, r5, i5, r6, −i6))/q;
          ai1 := zi3 + (im1(r2, i2, r5, i5, r6, −i6))/q

          end;

          r5 := re(r1, i1, h3 − h4, j3 − j4) + re(h1, j1, r4 − h4, i4 − j4);
          i5 := im(r1, i1, h3 − h4, j3 − j4) + im(h1, j1, r4 − h4, i4 − j4);
          q  := r5 × r5 + i5 × i5;
          if q <₁₀ − 20 then b2 := true else
```

```
begin  r6 := re(-r2, -i2, r1, i1); i6 := im(-r2, -i2, r1, i1);
       r7 := -k × h1 + h3 - h4; i7 := -k × j1 + j3 - j4;
      ar2 := zr3 + (re1(r6, i6, r7, i7, r5, -i5))/q;
      ai2 := zi3 + (im1(r6, i6, r7, i7, r5, -i5))/q
end;
q := fr3 × fr3 + fi3 × fi3;
if q <₁₀ - 20 then b3 := true else
begin ar3 := zr3 - fr3/q;
      ai3 := zi3 + fi3/q
end;
if b1 ∧ b2 ∧ b3 then
begin exit; goto END end;
q1 := if b1 then ₁₀5 else ((zr3 - ar1)↑2 + (zi3 - ai1)↑2);
q2 := if b2 then ₁₀5 else ((zr3 - ar2)↑2 + (zi3 - ai2)↑2);
q3 := if b3 then ₁₀5 else ((zr3 - ar3)↑2 + (zi3 - ai3)↑2);
p  := q1 - q2;
if p < 0 then
begin p := q1 - q3;
      if p < 0 then
      begin zhr := ar1; zhi := ai1 end else
      begin zhr := ar3; zhi := ai3 end
end else
begin p := q2 - q3;
      if p < 0 then
      begin zhr := ar2;
            zhi := ai2 end else
      begin zhr := ar3;
            zhi := ai3 end
end;
fpol(k, zhr, zhi, frh, fih, rco, ico);
it := it + 1;
q := frh × frh + fih × fih;
if bo1 then
begin if q < min then
      begin bo2 := false; min := q;
            zr0 := zhr; zi0 := zhi
      end else
      begin if bo2 then go to OMEGA
                   else bo2 := true
      end
end else
```

```
begin if sgrt (q) < eps then
      begin bol1 := true; min := q;
            zr0 := zhr; zi0 := zhi
      end else
      begin if it > 50 then
            begin it2 := it2 + 1; it := 0;
                  data(END); go to ANEW
            end
      end
end;
          ffpol(k, zhr, zhi, frh, fih, OK);
          zr1 := zr2; zi1 := zi2; zr2 := zr3; zi2 := zi3;
          fr1 := fr2; fi1 := fi2; fr2 := fr3; fi2 := fi3;
          zr3 := zhr; zi3 := zhi; fr3 := frh; fi3 := fih;
          h1  := r1; j1 := i1; h2 := r3; j2 := i3;
          h3  := h4; j3 := j4; h4 := r4; j4 := i4; go to NEWGO;

 OMEGA: end itpol;
          procedure fpol(g, zr, zi, fr, fi. ar, ai);
          integer g; real zr, zi, fr, fi;
          array ar, ai;
          begin boolean bool; integer i; real zh, zhr, zhi, ari, aii;
               fr := ar[0]; fi := ai[0]; zh := 1;
               if abs(zi) <₁₀ - 15 then
                 begin for i := 1 step 1 until g do
                        begin zh := zh × zr; fr := fr + zh × ar[i];
                                  fi := fi + zh × ai[i]
                        end;
                        go to FIN
                 end;
                 if abs(zr) <₁₀ - 15 then
                 begin bool := true;
                         for i := 1 step 1 until g do
                         begin zh := zh × zi;
                                 if bool then
                                 begin bool := false; fr := fr - zh × ai[i];
                                         fi := fi + zh × ar[i]; zh := -zh
                                 end else
                                 begin bool := true; fr := fr + zh × ar[i];
                                         fi := fi + zh × ai[i]
                                 end
                         end;
```

```
                go to FIN
                end;
                zhr := zr; zhi := zi;
                for i := 1 step 1 until g do
                begin zh := zhr; ari := ar[i]; aii := ai[i];
                      fr  := fr + ari × zhr − aii × zhi;
                      fi  := fi + ari × zhi + aii × zhr;
                      zhr := zhr × zr − zhi × zi;
                      zhi := zhi × zr + zh  × zi
                end;
FIN: end fpol;
     procedure ffpol(g, zr, zi, fr, fi, OK);
     integer g; real zr, zi, fr, fi; label OK;
     begin real rf, if, h, hr, hi;
          fpol(g − 1, zr, zi, rf, if, drco, dico);
          h := fr × fr + fi × fi;
          if sgrt(h) <₁₀ − 20 then
          begin zr0 := zr; zi0 := zi; go to OK end else
          begin hr  := (re(rf, if, fr, −fi))/h;
                hi  := (im(rf, if, fr, −fi))/h;
                fr  := hr; fi := hi
          end
     end ffpol;
     procedure diff(g, ar, ai, dr, di);
     integer g; array ar, ai, dr, di;
     begin integer i, ii;
          for i := g step − 1 until 1 do
          begin dr[i] := i × ar[i];
                di[i] := i × ai[i]
          end;
          for i := 1 step 1 until g do
          begin ii := i − 1;
                dr[ii] := dr[i];
                di[ii] := di[i]
          end
     end diff;
     procedure reduct(g, zr, zi, ar, ai);
     integer g; real zr, zi; array ar, ai;
     begin integer i, ii; real hr, hi;
          hr := ar[g];
          hi := ai[g];
          for i := g − 1 step − 1 until 1 do
```

```
        begin ar[i] := ar[i] + hr × zr − hi × zi;
              ai[i] := ai[i] + hr × zi + hi × zr;
              hr := ar[i]; hi := ai[i]
        end;
        for i := 1 step 1 until g do
        begin ii := i − 1;
              ar[ii] := ar[i];
              ai[ii] := ai[i]
        end
end reduct;
procedure first;
begin real rh1, rh2, ih1, ih2, fh1, fh2, fh3, h;
      rh1 := rco[0]; ih1 := ico[0];
      rh2 := rco[k]; ih2 := ico[k];
      h   := sqrt(rh1 × rh1 + ih1 × ih1); eps := (₁₀−9) × h;
      w   := ((h/(sqrt(rh2 × rh2 + ih2 × ih2)))↑(1.0/k))/5;
      zr1 := 0; zi1 := w; zr2 := −w;
      zi2 := w; zr3 := 0; zi3 := 2 × w;
      fpol(k, zr1, zi1, fr1, fi1, rco, ico);
      fpol(k, zr2, zi2, fr2, fi2, rco, ico);
      fpol(k, zr3, zi3, fr3, fi3, rco, ico); correct;
      diff(k, rco, ico, drco, dico);
      ffpol(k, zr1, zi1, fr1, fi1, OK);
      ffpol(k, zr2, zi2, fr2, fi2, OK);
      ffpol(k, zr3, zi3, fr3, fi3, OK)
end first;
procedure newz;
begin real rh1, ih1, rh2, ih2, h;
      rh1 := rco[0]; ih1 := ico[0];
      rh2 := rco[k]; ih2 := ico[k];
      h   := sqrt(rh1↑2 + ih1↑2);
      eps := (₁₀ − 9) × h;
      w   := ((h/(sqrt(rh2↑2 + ih2↑2)))↑(1.0/k))/5;
      zr1 := zr2 := −w;
      zr3 := zra := zr0;
      zi3 := zia := −zi0;
      if zia > 0 then
      begin zi1 := w; zi2 := 2 × w end else
      begin zi1 := −w; zi2 := −2 × w end;
      fpol(k, zr1, zi1, fr1, fi1, rco, ico);
      fpol(k, zr2, zi2, fr2, fi2, rco, ico);
      fpol(k, zr3, zi3, fr3, fi3, rco, ico);
```

```
        correct;
        ffpol(k. zr1, zi1, fr1, fi1, OK);
        ffpol(k, zr2, zi2, fr2, fi2, OK);
        ffpol(k, zr3, zi3, fr3, fi3, OK)
    end newz;
    procedure data (END);
    label END;
    begin integer s;
        if it2 = 3 then
        begin newline (3);
            writetext (' Verfahren konvergiert nicht ');
            go to END
        end;
        zr1 := 0;
        zr2 := -w;
        if it2 = 1 then
        begin zi1 := zi2 := 2 × w;
            zr3 := -w; zi3 := 3 × w
        end else
        begin zi1 := zi2 := 3 × w;
            zr3 := -2 × w; zi3 := 4 × w
        end;
        fpol(k, zr1, zi1, fr1, fi1, rco, ico);
        fpol(k, zr2, zi2, fr2, fi2, rco, ico);
        fpol(k, zr3, zi3, fr3, fi3, rco, ico);
        correct;
        ffpol(k, zr1, zi1, fr1, fi1, OK);
        ffpol(k, zr2, zi2, fr2, fi2, OK);
        ffpol(k, zr3, zi3, fr3, fi3, OK)
    end data;
    procedure proof (LAST);
    label LAST;
    begin integer i, ii; real hr, hi;
    AGAIN: hr := rco[0]; hi := ico[0];
        if (hr↑2 + hi↑2) < ₁₀-20 then
        begin for i := 1 step 1 until k do
            begin ii := i - 1;
                rco[ii] := rco[i]; ico[ii] := ico[i]
            end;
            k := k - 1; it := 0; zr0 := zi0 := 0; out;
            if k = 1 then go to LAST;
```

```
                go to AGAIN
            end;
end proof;
procedure correct;
begin real fh1, fh2, fh3, h;
      fh1 := fr1↑2 + fi1↑2;
      fh2 := fr2↑2 + fi2↑2;
      fh3 := fr3↑2 + fi3↑2;
      if (fh2 - fh1) > 0 then
      begin h := zr1; zr1 := zr2; zr2 := h;
            h := zi1; zi1 := zi2; zi2 := h;
            h := fr1; fr1 := fr2; fr2 := h;
            h := fi1; fi1 := fi2; fi2 := h;
            h := fh1; fh1 := fh2; fh2 := h
      end;
      if (fh3 - fh1) > 0 then
      begin h := zr1; zr1 := zr3; zr3 := h;
            h := zi1; zi1 := zi3; zi3 := h;
            h := fr1; fr1 := fr3; fr3 := h;
            h := fi1; fi1 := fi3; fi3 := h;
            h := fh1; fh1 := fh3; fh3 := h
      end;
      if (fh3 - fh2) > 0 then
      begin h := zr2; zr2 := zr3; zr3 := h;
            h := zi2; zi2 := zi3; zi3 := h;
            h := fr2; fr2 := fr3; fr3 := h;
            h := fi2; fi2 := fi3; fi3 := h
      end
end correct;
procedure out;
begin newline (1); writetext ('z'); print (n - k, 2, 0);
      writetext ('='); print (zr0, 0, 11); space (4);
      print (zi0, 0, 11); writetext ('i (');
      it := it + it2 × 50; print (it, 3, 0); writetext (')');
end out;
procedure exit;
begin newline (3);
      writetext (' Verfahren muss abgebrochen werden:
                 Bei a1, a2, a3 ist Nenner = 0 ')
end exit;
procedure newline (x);
```

```
integer x;
comment Diese Prozedur ist im Maschinenkod und bewirkt die Verschiebung des
Papiers um x Zeilen weiter.;
procedure writetext (x);
string x;
comment Diese Prozedur schreibet die Zeichenkette x aus.;
MP: newline (6); writetext ('Loesung:'); newline (1);
    it := it2 := 0; k := n;
    for i := n step - 1 until 0 do
    begin rco[i] := readb; ico[i] := readb end;
    if k < 1 then
    begin newline (1); writetext (' Fehler in den Daten ');
         go to END
    end;
    if k = 1 then go to LAST;
    proof (LAST);
    first;
MORE: itpol (END);
   OK: k := k - 1; out; it := 0; it2 := 0;
       reduct(k + 1, zr0, zi0, rco, ico);
       if k = 1 then
LAST: begin k := 0; it := it2 := 0;
           zr0 := -rco[0]; zi0 := -ico[0];
           zra := rco[1]; zia := ico[1];
           if abs (zra - 1) > ₁₀-10 then
           begin zr0 := zr0/zra; zi0 := zi0/zra;
                 zia := zia/zra
           end;
           if abs (zia) > ₁₀-10 then
           begin w := 1 + zia × zia;
                 w1 := zr0;
                 zr0 := (zr0 + zia × zi0)/w;
                 zi0 := (zi0 - zia × w1)/w
           end;
           out;
           go to END
      end;
      diff (k, rco, ico, drco, dico);
      newz;
      go to MORE;
END: end;
```

Das Programm bestimmt die Wurzeln des Polynoms $f(z) = a_n z^n + a_{n-1} z^{n-1} + \ldots$ $\ldots + a_1 z + a_0$, mit komplexen Koeffizienten a_i $(i = 0, 1, \ldots, n)$.

In der Prozedur *proof* werden von $f(z)$ etwaige triviale Nullstellen $z = 0$ abgespaltet. In der Prozedur *first* werden die Anfangswerte z_1, z_2, z_3 so bestimmt, dass die Bedingungen $|z_i| < |a_0/a_n|^{1/n}$, $|f(z_3)| \leqq |f(z_2)| \leqq |f(z_1)|$ erfüllt werden. Das eigentliche Iterationsverfahren wird in der Prozedur *itpol* durchgeführt und zwar folgenderweise:

Bei jedem Schritt werden folgende drei Werte berechnet:

$$a = z_3 + \frac{(z_2 - z_3)(z_3 - z_1)(F_2 - F_1)}{(z_3 - z_2)(F_2 - F_1) + (z_1 - z_2)(F_3 - F_2)}$$

$$a' = z_3 + \frac{(z_1 - z_3)(z_2 - z_3)(n(z_2 - z_1) + z_1^2 F_1 - z_2^2 F_2)}{(z_2 - z_3)(z_1^2 F_1 - z_2^2 F_2) + (z_1 - z_2)(z_3^2 F_3 - z_2^2 F_2)}$$

$$a'' = z_3 - 1/F_3$$

$$\left(F_j = \frac{f'(z_j)}{f(z_j)} \quad \text{für} \quad j = 1, 2, 3 \right.$$

$$a = ar1 + i \times ai1$$

$$a' = ar2 + i \times ai2$$

$$a'' = ar3 + i \times ai3$$

$$\left. \text{und} \quad z_4 = zhr + i \times zhi \right).$$

Als z_4 wird derjenige Wert der Ausdrücke a, a' und a'' gewählt, der dem z_3 am nächsten liegt. Dann stellen wir z_2 anstatt z_1, z_3 anstatt z_2, z_4 anstatt z_3 und wiederholen den Iterationsschritt. Ist nach 50 Iterationen die erwünschte Genauigkeit $|f(z_i)| < 10^{-9}|a_0|$ nicht erreicht, so wird das Iterationsverfahren mit neuen Anfangswerten z_i – $|z_i| < |a_0/a_n|^{1/n}$ – gestartet, die in data bereit gestellt werden.

Nach der Beendigung des Iterationsverfahrens führen wir die Division $f(z)/(z - z_0)$ in der Prozedur *reduct* durch. Dann bestimmen wir in der Prozedur *newz* die neuen Anfangswerte z_1, z_2, z_3 für das reduzierte Polynom. Dabei ersetzen wir z_3 durch z_0 und wiederholen das Iterationsverfahren.

Diese Methode konvergiert für einfache, und was bemerkenswert ist, auch für mehrfache Wurzeln sehr schnell. Der in Fortran programmierte Algorithmus wurde auf der Rechenmaschine ICT 1905 geprüft. Dabei wurden die Prozeduren *fpol*, *ffpol*, *reduct* mit doppelter Genauigkeit programmiert. Es wurden 32 nichttriviale Beispiele berechnet. Der höchste Grad des Polynoms war 19. Die Berechnungsgenauigkeit hängt von der Grösse der Wurzeldistanz ab.

Das Fortran-Programm steht bei dem Verfasser zur Verfügung.

Das Testbeispiel:

$$f(x) = x^3 - 5x^2 + 4x + 10 .$$

Die Nullstellen mit der Iterationsanzahl in Klammern:

$$\begin{array}{rl} -1 & (6) \\ 3 + i & (9) \\ 3 - i & (0) \end{array}$$

Das Algol-Programm für die obige Methode wurde auch von V. Klotz [2] vorgelegt. Trotzdem veröffentliche ich mein Programm in der Hoffnung, dass die Leser einige wesentliche Abweichungen, die sich in ihm gegenüber dem soeben erwähnten Programm befinden, würdigen mögen und dass sie seine Veröffentlichung als zweckmässig erfinden.

Literatur

[1] *G. R. Garside, P. Jarratt, C. Mack:* A new method for solving polynomial equations, Computer Journal, 11 (1968) 87—90.

[2] *V. Klotz:* Ein Algol-Programm zur Bestimmung der Wurzeln eines Polynoms nach Garside, Jarratt und Mack, Elektronische Datenverarbeitung, 3 (1969) 115—119.