AN ALGOL PROGRAM FOR GENERATING ROOT LOCUS DIAGRAMS

By

T. Kovács

Department for Automation Technical University, Budapest

(Received December 16. 1968.)

Presented by Prof. Dr. F. Csáki

Introduction

The stability investigation in linear systems with constant lumped parameters is an important method. The characteristic equation of such system has the form

$$F(s) = g(s) + Ke^{-\tau s}\, h(s) = 0$$

where the polynomials $g(s) = s^n + as^{n-1} + \dots$ and $h(s) = s^m + bs^{m-1} + \dots$ have real coefficients, K is a real parameter and τ is the dead time.

For determining the stability the root locus diagram is an excellent tool. With the picture of the position of the roots in the complex plane for all K the designer is enable to vary the gain so that the system will be stable. For detailed description of the method see [5, 7]. It must be mentioned that the Algol-program presented here refers to the basic work of Krall and Fornaro [3, 4]. This publication was suggested by the fact that the Algol language was widely used in Europe and so by the hope that the program will be somewhat helpful for designers.

The basic theorem of the method

Let $F(s) = g(s) + Ke^{-\tau s} h(s)$. We define the root locus of $F(s)$ as a set of points s such, that $h(s) = 0$ or $F(s) = 0$. For $K \geq 0$, s is on the positive branch of the root locus, for $K < 0$, s is on the negative branch of the root locus.

It can be verified [3, 4] that the point $s = x + jy$ is on the root locus of $F(s)$ if and only if

$$\Phi(x,y) = \cos \tau y \sum_{k=0}^{\infty} \frac{(-1)^k y^{2k+1}}{(2k+1)!} \times$$
$$\times \sum_{i=0}^{2k+1} \binom{2k+1}{i} (-1)^{2k+1-i}\, h^{(i)}(x) g^{(2k+1-i)}(x) -$$
$$- \sin \tau y \sum_{k=0}^{\infty} \frac{(-1)^k y^{2k}}{(2k)!} \sum_{i=0}^{2k} \binom{2k}{i} (-1)^{2k-i}\, h^{(i)}(x) g^{(2k-i)}(x) = 0$$

The expressions for K

$$K = e^{\tau x} |g(s)|^2 \cos \tau y / Re(h(s)\overline{g(s)})$$

or

$$K = e^{\tau x} |g(s)|^2 \sin \tau y (\mathrm{Im}(h(s)\overline{g(s)})$$

In the most important case, for $\tau = 0$, the first expression must be applied.

Computational procedure

Let first choose the rectangular region in which the root locus is desired. Denote this region $[x_l, x_r]$, $[y_b, y_t]$. Divide the interval $[x_l, x_r]$ into increments $x_l, x_l + \Delta x, \ldots, x_r$. For each point $x + m \Delta x$ in turn, divide $[y_b, y_t]$ into increments $y_t, y_t - \Delta y, \ldots, y_b$. Then compute $\Phi(x, y)$ at these points, fixing first x, then letting y vary from y_t to y_b. If $\Phi(x, y)$ changes sign at any y, this fact indicates that the root locus of $F(s)$ lies in the interval $y, y + \Delta y$. With the method of subintervals (i.e. by halving Δy etc.) as it can be seen the position of the root locus point can be obtained at the desired accuracy.

The program

For the sake of shortness, the flow chart and the specifications will not be discussed here. Even so, the program is quite understandable. Notice yet that the designer has to give $g(s)$ and $h(s)$ (in factorized or in polynomial form), the dead-time, the desired accuracy, the rectangular region, increments Δx and Δy. If $g(s)$ and $h(s)$ are in a polynomial form, the orders of g and of h are also needed. If $g(s)$ and $h(s)$ have a factorized form, the real roots and the complex roots of $g(s)$ and $h(s)$ are to be given. Still, two variables remained to be introduced: *i*-enter and *i*-stop.

The first one makes possible to branch the program depending on the form of $g(s)$ and of $h(s)$. The second one makes it possible to generate the root locus points of several systems, in one operation at the same time.

```
  begin real delta,bgn,endc,dec,y0,y1,top,tau,cl,xl,e,aroot,kval,xx;
  integer i,j,m,ienter,ng,nh,ng1,nh1,n1g,n1h,nn,limit,kk,ki,kik,limt,n1,n,
     im,k,isig,istop;
  array x[1:100],cg[1:11,1:11],ch[1:11,1:11],g[1:22],h[1:22],a[1:21],
  c[1:15], y[1:10], s[1:10];
procedure compco(c,v);
  integer v;
  array c;
```

```
  begin
    real t,t1;
    integer i,ii,j,k,lim,lowlim,m,n,nn;
    array a[1:6,1:13],crts[1:6,1:2],d[1:15],rrts[1:12];
    for i:=1 step 1 until 15 do
      begin
        d[i]:=0;
        c[i]:=0;
      end:
    for j:=1 step 1 until 6 do
    for i:=1 step 1 until 2 do
    crts[j,i]:=0;
      for j:=1 step 1 until 12 do
    rrts[j]:=0;
    for j:=1 step 1 until 6 do
    for i:=1 step 1 until 13 do
    a[j,i]:=0;
    lowlim:=1;
    ii:=2;
      begin comment: input (m.n);
      end;
    if m=0 then go to e26;
    for i:=1 step 1 until m do
    for j:=1 step 1 until 2 do
      begin
        comment: input (crts[1:m, 1:2]);
      end;
    go to e28;
e26: ii:=1;
e28: if n=0 then go to e32;
    for i:=1 step 1 until n do
      begin
        comment: input (rrts[1:n]);
      end;
e32: if ii=1 then go to e30;
    for i:=1 step 1 until m do
      begin
        a[i,1]:=1;
        a[i,2]:=-2×crts[i, 1];
        a[i,3]:=crts[i,1]↑2+crts[i,2]↑2;
      end;
    for i:=1 step 1 until 3 do
```

```
        c[i]:=a[1,i];
        if (m-1)=0 then go to e16;
        j:=2;
        lim:=3;
e17:    for i:=1 step 1 until lim do
        d[i]:=c[i];
        lim:=lim+2;
        for i:=1 step 1 until lim do
          begin
            c[i]:=0;
            for k:=1 step 1 until i do
            c[i]:=c[i]+a[j,k]×d[i+1-k];
          end;
        j:=j+1;
        if (m-j) < 0 then go to e16 else go to e17;
e30:    c[1]:=1;
c[2]:=-rrts[1];
        lowlim:=2;
        nn:=2;
        go to e18;
e16:    nn:=2×m+1;
        if n=0 then go to e19;
e18:    if (lowlim-n)> 0 then go to e19;
        for j:=lowlim step 1 until n do
          begin
            t:=-rrts[j];
            nn:=nn+1;
            for k:=1 step 1 until nn do
              begin
                t1:=-rrts[j]×c[k+1];
                c[k+1]:=t+c[k+1];
                t:=t1;
              end;
          end;
e19:    v:=nn;
 end compco;
 real procedure horner(z,k,a);
   value z;
   real z;
   integer k;
   array a;
   begin
```

```
        integer i,nt;
        real y;
        if k<0 then go to d1 else if k>0 then go to d3;
        horner:=a[1]; go to d4;
d1:     horner:=0; go to d4;
d3:     y:=a[1]; nt:=k+1;
        for i:=2 step 1 until nt do
        y:=y×z+a[i]; horner:=y;
d4:end horner;
 procedure derev(u);
      value u;
      real u;
      begin
        integer i,j,jj,kg,kh;
        array coef1[1:11],coef2[1:11];
        for j:=1 step 1 until 22 do
          begin
            g[j]:=0; h[j]:=0
          end;

          for j:=0 step 1 until 10 do
            begin
              jj:=j+1;
              for i:=1 step 1 until 11 do
                begin
                  coef1[i]:=cg[jj,i];
                  coef2[i]:=ch[jj,i];
                end;
              kg:=ng−j;
              kh:=nh−j;
              g[jj]:=horner(u,kg,coef1);
              h[jj]:=horner(u,kh,coef2);
            end;
      end derev;
 real procedure triho(z,k,a);
      value z;
      real z;
      integer k;
      array a;
        begin
          integer i,ii,nt,k1;
          real w;
```

```
          array trig[1:2];
          if (tau)≠0 then go to t2;
          triho:=horner(z,k,a);
          go to t4;
t2:       k1:=(k−2×(k÷2))+1;
          if (k1−1)=0 then go to t3;
          trig[1]:=sin(tau×z);
          trig[2]:=−cos(tau×z);
          go to t5;
t3:       trig[1]:=−cos(tau×z);
          trig[2]:=sin(tau×z);
t5:       w:=a[1]×trig[2];
          nt:=k+1;
          for i:=2 step 1 until nt do
        begin
          ii:=(i−2×(i÷2))+1;
          w:=w×z+a[i]×trig[ii];
        end;
        triho:=w;
t4:     end triho;
procedure search(lo,hi,dec,s,j,a,n);
     value lo,hi,dec,a,n;
     integer j,n;
     real lo,hi,dec;
     array s,a;
     begin
        real temphi, y1,y2,y;
        temphi:=hi;
        j:=0;
        y1:=triho(temphi,n,a);
f4:  y2:=triho(temphi−dec,n,a);
        y:=y1×y2;
        if y<0 then go to f1 else if y>0 then go to f2;
        j:=j+1;
        s[j]:=temphi-dec;
        y1:=triho(temphi-dec-dec/10, n,a);
        go to f3;
f1:  j:=j+1;
        s[j]:=temphi-dec;
f2:     y1:=y2;
f3:     temphi:=temphi-dec;
```

```
        if (temphi-lo)>0 then go to f4;
        end;
  procedure root(xl,lex,eps,aroot,a,n);
     value xl,lex,eps,a,n;
     real xl,lex,eps,aroot;
     array a;
     integer n;
     begin
       real h,xr,yl,yr,y;
          h:=lex;
  r1:     xr:=xl+h/2;
  r2:     yl:=triho(xl,n,a);
          yr:=triho(xr,n,a);
          y:=yl×yr;
        if y=0 then go to r3 else  if y > 0 then go to r4;
       if (abs(xr—xl)-eps)<0 then go to r5;
          h:=h/2;
          go to r1;
  r4:     xl:=xr;
          xr:=xr+h/2;
          go to r2;
  r3:     if yl=0 then go to r6;
         aroot:=xr;
          go to  r7;
  r6:     aroot:=xl;
          go to r7;
  r5: aroot:=xl—abs(xr—xl)/2;
  r7: end root;
real procedure fact(k);
      value k; integer k;
      begin integer i,ik;
        ik:=1;
        for i:=1 step 1 until k do
        ik:=ik×i;
        fact:=ik;
      end fact;
real procedure biger(p,q);
      value p,q;
      integer p,q;
      begin
      if p>q then biger:=p else biger:=q;
      end biger;
```

```
real procedure comb(k,ik);
      value k,ik; integer k,ik;
      begin integer i,c;
        if ik=1 then begin c:=k; go to c2 end;
        else if ik≤0 then
        begin c:=1; go to c2 end;
        c:=k;
        for i:=2 step 1 until ik do
        c:=c×(k−i+1)/i;
  c2:   comb:=c;
      end comb;
real procedure resolk(y,xi);
      value y,xi; real y,xi;
      begin real rg,rh,ig,ih,s,denom,u;
        integer n2,m2,top,i,j,io,ie;
        array yp[1:14];
        top:=biger(ng,nh)+2;
        yp[1]:=1; yp[2]:=y;
        for i:=3 step 1 until top do
        yp[i]:=yp[i−1]×y;
        rg:=0; ig:=0; n2:=ng÷2; s:=−1;
        for j:=0 step 1 until n2 do
        begin
        io:=2×j+1; ie:=2×j; s:=s×(−1);
        rg:=rg+s×g[ie+1]/fact(ie)×yp[ie+1];
        ig:=ig+s×g[io+1]/fact(io)×yp[io+1];
        end;
        m2:=nh÷2; rh:=0; ih:=0; s:=−1;
        for j:=0 step 1 until m2 do
        begin
        io:=2×j+1; ie:=2×j; s:=s×(−1);
        rh:=rh+s×h[ie+1]/fact(ie)×yp[ie+1];
        ih:=ih+s×h[io+1]/fact(io)×yp[io+1];
        end;
        denom:=rh×rg+ih×ig;
        if abs(denom)<10↑(−10) then go to r1;
        resolk:=−exp(tau×xi)×cos(tau×y)×(rg↑2+ig↑2)/denom;
  r1: end resolk;
  s1: begin
      for i:=1 step 1 until 100 do
      x[i]:=0
```

```
  begin comment: input (delta, bgn,endc);
  end;
    m: =abs(bgn-endc)/delta+1;
    x[1]:=bgn;
    if (m-100)≤0 then go to p1;
    m:=100;
 p1: for i:=2 step 1 until m do
    x[i]:=x[i-1]+delta;
    begin comment input (dec,y0,y1,top,isig);
    end;
    e:=10↑(-isig);
    for i:=1 step 1 until 11 do
    for j:=1 step 1 until 11 do
    begin
    ch[i,j]:=0;
    cg[i,j]:=0;
    end;
    begin comment: input (ienter);
    end;
    if (ienter-1)=0 then go to p2 else go to p3;
p2: compco(c,ng1);
    ng:=ng1-1;
    for i:=1 step 1 until ng1 do
    cg[1,i]:=c[i];
    compco(c,nh1);
    nh:=nh1-1;
    for i:=1 step 1 until nh1 do
    ch[1,i]:=c[i];
    go to p4;
p3: begin comment: input (ng,nh);
    end;
    ng1:=ng+1;
    nh1:=nh+1;
    for i:=1 step 1 until ng1 do
    begin comment: input cg[1,1:ng1];
    end;
    for i:=1 step 1 until nh1 do
    begin comment: input ch[1,1:nh1];
    end;
 p4: n1g:=ng1;
    for i:=2 step 1 until ng1 do
```

```
    begin
      nlg:=nlg−1;
      for j:=1 step 1 until nlg do
      cg[i,j]:=(nlg+1−j)×cg[i−1,j];
    end;
  nlh:=nhl;
  for i:=2 step 1 until nhl do
    begin
      nlh:=nlh−1;
      for j:=1 step 1 until nlh do
      ch[i,j]:=(nlh+1−j)×ch[i−1,j];
    end:
  begin comment: output(cg,ch);
  end;
  nn:=ng+nh;
  limit:=(nn−1)÷2;
  if (limit−9)≤0 then go to p5;
  begin comment: output (nn);
  end;
  stop;
p5: begin comment: input (tau);
  end;
  for im:=1 step 1 until m do
  begin
    derev(x[im]);
    for i:=1 step 1 until 10 do
    begin
      s[i]:=0; y[i]:=0;
    end;
  for i:=1 step 1 until 21 do
  a[i]:=0;
  if (nn−2)>0 then go to p6;
  kk:=1; ki:=nn;
  for i:=0 step 1 until kk do
  begin
  kik:=kk −i;
  a[ki]:=a[ki]+comb(kk,i)×(−1)↑kik×h[i+1]/fact(kk)×g[kik+1];
  end;
  go to p7;
p6: for k:=0 step 1 until limit do
  begin
  kk:=2×k+1; cl:=(−1)↑k; ki:=nn+1−kk;
```

```
     for i:=0 step 1 until kk do
     begin
     kik:=kk−i;
     a[ki]:=a[ki]+comb(kk,i)×(−1)↑kik×h[i+1]/fact(kk)×g[kik+1];
     end;
     a[ki]:=a[ki]×c1;
     end;
p7:  if (tau)=0 then go to p8;
     limt:=nn÷2;
     a[nn+1]:=h[1]×g[1];
     if (limt)≤0 then go to p8;
     for k:=1 step 1 until limt do
     begin
     c1:=(−1)↑k; kk:=2×k; ki:=nn+1−kk;
     for i=0 step 1 until kk do
     begin
     kik:=kk−i;
     a[ki]:=a[ki]+comb(kk,i)×(−1)↑kik×h[i+1]/fact(kk)×g[kik+1];
     end;
     a[ki]:=a[ki]×c1;
     end;
p8:  n1:=nn+1;
     xx:=x[im];
     begin comment: output (xx);
     end;
     n:=nn;
     if (y0+y1)≠0 then go to p9;
     y1:=top;
     y0:=0;
p9:  search(y0,y1,dec,s,j,a,n);
     begin comment: output (j);
     end;
     if j≤0 then go to p11;
     for i:=1 step 1 until j do
     begin
     xl:=s[i];
     root(xl,dec,e,aroot,a,n);
     kval:=resolk(aroot,xi);
     begin comment: output(aroot,kval);
     end;
     end;
p11: end;
```

```
end;
begin comment: input (istop) and output (istop);
end;
if istop=1 then go to s1;
end;
```

Illustrative examples

The program was already tested on many problems. To demonstrate some results the root loci of two systems are attached on Figs 1 to 4. The open loop transfer functions of this systems were

$$G(s) = K \frac{s + 6}{s^2 + 6s + 25}$$

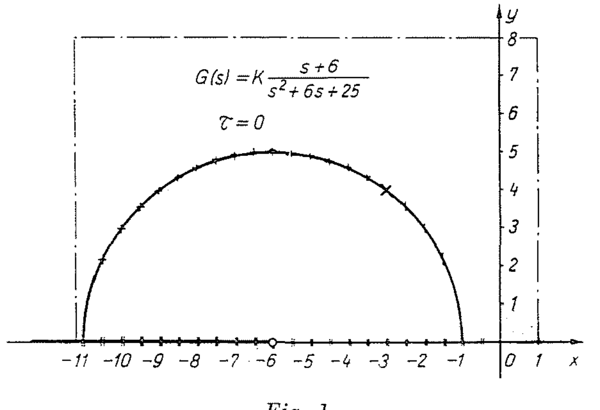

Fig. 1

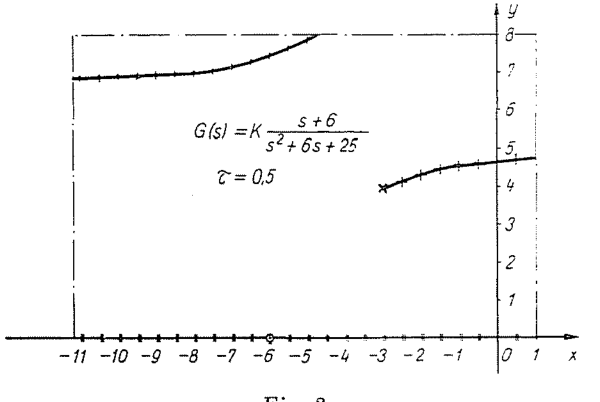

Fig. 2

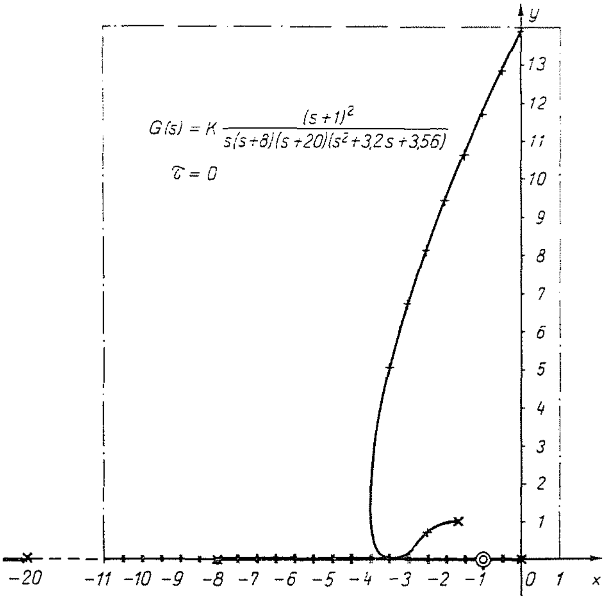

Fig. 3

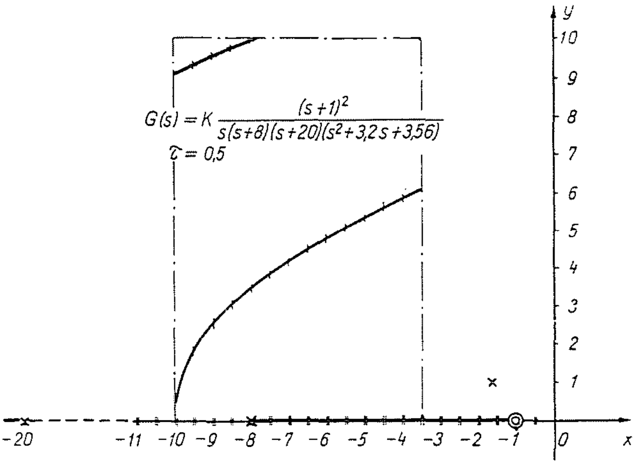

Fig. 4

and

$$G(s) = K \frac{(s+1)^2}{(s^2 + 3.2s + 3.56)\, s(s+8)(s+20)} \,.$$

In Fig. 2 and Fig. 4 the case $\tau = 0.5$ is shown.

Parameters for execution

To the above disscussed examples we supply here the parameters for execution in right sequence. (Table 1)

Table 1

Parameters for execution

Δx	x_l	x_r	Δy	y_b	y_t	top	isig	i-enter	m	n
0.5	−11	1	0.2	0	8	8	7	1	1	0
0.5	−11	1	0.2	0	8	8	7	1	1	0
0.5	−11	1	0.1	0	14	14	7	1	1	3
0.5	−10	−3.5	0.2	0	10	10	7	1	1	3
R. part of c. roots of g	I. part of c. roots of g	Real roots of g			m	n	Real roots of h		tau	i-stop
−3	4	—	—	—	0	1	—	−6	0	1
−3	4	—	—	—	0	1	—	−6	0.5	1
−1.6	1	−20	−8	0	0	2	−1	−1	0	1
−1.6	1	−20	−8	0	0	2	−1	−1	0.5	0

Summary

This paper is dealing with an Algol-program which makes it possible to generate the ordinary root locus diagram much faster than before. The time lag diagram may be generated at the same speed and accuracy as an ordinary diagram.

References

1. KRALL, A. M.: An extension and proof of the root locus method. J. SIAM **9**, 644–653, (1961).
2. KRALL, A. M.: Stability criteria for feedback systems with time lag. J. SIAM Ser. A, Control, **2**, 160–170, (1965).
3. KRALL, A. M.—FORNARO, R.: An algorithm for generating root locus diagrams. CACM **10** 186–188, (1967).
4. KRALL, A. M.—FORNARO, R.: Root locus diagrams by digital computer. Report of Pennsylvania State University (1967).
5. SHINNERS, S. M.: Techniques of System Engineering. McGraw-Hill, Inc. 1967.
6. PONTRJAGIN, L. S.: On the zeros of some elementary transcendental functions. Amer. Math. Soc. Transl. Ser. **2**, 95–110, (1955).
7. TRUXAL, J. G.: Automatic Feedback Control System Synthesis. McGraw-Hill, Inc. 1955, 298–338.

Tivadar KOVÁCS, Budapest XI., Egri József u. 18–20, Hungary