EVALUATION OF Z[F(s)] BY DIGITAL COMPUTER

By

T. Kovács

Department of Automation, Technical University Budapest

(Received May 4, 1971)

Presented by Prof. Dr. F. Csáki

Relationship between Laplace and Z-transforms and evaluation of Z[F(s)]

As it is known the sampled function $f^*(t)$ of $f(t)$ can be written in the form [1, 2, 3]

$$f^*(t) = f(t) \sum_{n=0}^{\infty} \delta(t - nT) = 1^*(t) f(t) .$$

Taking the Laplace transform of this equation we obtain

$$\mathfrak{F}(s) \triangleq \mathfrak{L}[f^*(t)] = \int_{-\infty}^{\infty} 1(t) f^*(t) e^{-st} dt = \int_{-\infty}^{\infty} 1^*(t) f(t) e^{-st} dt.$$

Since $\mathfrak{L}\delta[(t - kT)] = e^{-kst}$ we establish

$$F^*(s) \triangleq \mathfrak{L}[f^*(t)] = \sum_{n=0}^{\infty} f(nT) e^{-nsT}$$

that is

$$\mathfrak{F}(z) = F^*(s)\,|_{s=T^{-1} \ln (z)}$$

Thus the connection between the Z-transform and the Laplace transform is

$$\mathfrak{F}(z)\,|_{z=e^{Ts}} = F^*(s) \triangleq \mathfrak{L}[f^*(t)] = \mathfrak{L}[1^*(t) f(t)].$$

With the application of the convolution theorem for the Laplace transform, ($f(t)$ contains no impulses and initially zero), we obtain

$$\mathfrak{F}(z) = Z[F(s)] = \left[\frac{1}{2j\pi} \int_{c-j\infty}^{c+j\infty} F(p) \frac{1}{1 - e^{-T(s-p)}} \, dp \right] \Bigg|_{e^{Ts} = z}$$

where

$$F(p) = F(s)\,|_{s=p} = \mathfrak{L}[f(t)]\,|_{s=p}$$

and

$$1/(1 - e^{-Ts}) = 1 + e^{-Ts} + e^{-2Ts} + \ldots + e^{-nTs} + \ldots = \mathfrak{L}[\delta_T]$$

for

$$|e^{-Ts}| < 1$$

This integral can be evaluated by the residue theorem. That is, if

$$F(s) = \frac{A(s)}{B(s)}$$

then $F^*(s)$ can be written in the following general form:

$$F^*(s) = \sum_{\text{Roots of } B(p)} \text{Residues of } \left[\frac{A(p)}{B(p)} \frac{1}{1 - e^{pT}e^{-sT}}\right]$$

where $B(p) = B(s)\big|_{s=p}$

If $F(s)$ is a rational fractional function with simple poles, then the expansion theorem for the Z-transformation can be expressed in the following form

$$F^*(s) = \sum_{i=1}^{n} \frac{A(p)}{B'(p)} \frac{1}{1 - e^{-T(s-p)}}.$$

If $F(s)$ is a rational fractional function with r multiple poles of multiplicity m then the following formula can be applied for the expansion

$$\mathfrak{F}(z) = F^*(s)\big|_{z=e^{Ts}} = \sum_{i=1}^{r} \lim_{s-p} \frac{1}{(m-1)!} \frac{\mathrm{d}^{(m_i-1)}}{\mathrm{d}s^{(m_i-1)}}$$

$$\left\{\left[(s-p)^{m_i} F(s)\right] \frac{z}{z - e^{sT}}\right\}\Big|_{p=s_i}$$

(It should be mentioned that the substitution $z = e^{-sT}$ is also usual. It entails no special difficulty to rewrite the relations in the suitable form.)

The general description of the program developed for the evaluation of Z[F(s)]

It is supposed that the original transfer function $F(s)$ is given in the form:

$$F(s) = \frac{\sum_{j=0}^{r} a_j s^j}{\prod_{i=1}^{r} (s - v_i)^{m_i}}$$

where $\hat{v}_i$ is a pole of multiplicity m_i and equals $v_{i,1} + jv_{i,2}$.
The Z-transform of this equation can be written as:

$$Z[F(s)] = \frac{\sum_{i=0}^{m} A_i z^i}{\sum_{j=0}^{m} B_j z^j}$$

Our task is to give a general ALGOL program to calculate the coefficients A and B based on the preceding general expansion formula.

To this aim we adapted the simpler notations and derived formulae of [2].

Let us now consider the following expression

$$(s - \hat{v}_i)F(s) = (s - \hat{v}_i)\left[\frac{N}{\prod_{j=1}^{r}(s - \hat{v}_j)^{m_j}}\right] = \frac{N}{D}.$$

From this the meaning of N and D is clear. Denote by D', D'' and N', N'' the first and the second derivatives of D and N, respectively.
With these notations the argument of the general expansion formula becomes

$$\text{for } m = 1 \quad \frac{N}{D}\frac{z}{z - e^{sT}}$$

$$\text{for } m = 2 \quad \frac{N'D - ND'}{D^2}\frac{z}{z - e^{sT}} + \frac{N}{D}\frac{Te^{sT}}{(z - e^{sT})^2}$$

and for $m = 3$

$$\frac{1}{2}\frac{(N''D - ND'')D - 2(N'D - ND')D'}{D^3}\frac{z}{z - e^{sT}} +$$

$$+ \frac{1}{2}\frac{N}{D}\frac{T^2e^{sT}z(z + e^{sT})}{(z - e^{sT})^3} + \frac{N'D - ND'}{D^2}\frac{Te^{sT}}{(z - e^{sT})^2}.$$

From these equations the restriction $m \leq 3$ is seen for the multiplicity of the poles. This relationship is valid almost in all practical cases. In our program one more restriction exists in the degrees of the polynomials of the numerator and the denominator of $F(s)$: they cannot exceed 30. But to rewrite the parts of the program connected with this limitation requires little effort.

Our program contains procedures for some complex arithmetic (nadd, nmult, div) and procedures for operations with polynomials (mult, add, test). For calculation of a polynomial and its derivative values, a complex Horner scheme is used, developed by us for this special purpose (Horner).

The program was designed for a computer MINSK-22 at the Computer Center of the Automation Research Institute of the Hungarian Academy of Sciences. The used algorithmic language was the representation of the ALGOL-60 relative to this computer.

The program was tested already succesfully in many problems, and it deserves our full satisfaction.

The program

The complete program is given in the following.

```
begin
integer r, mmax, m, n;
input (r, mmax, m, n);
begin
integer jj, ii, kk, zl, z2, i, j, k, l, hi;
real pl, ql, p2, q2, p3, q3, p4, q4, t, h;
real array pr, pi[1:r], szaml, neve [0:n, 1:2], v[1:r, 1:2], aa[0:m, 1:2], denom,
     segit, ab[0:n, 1:2], a[1:60, 0:30, 1:2], c[1:r 1:mmax, 1:2];
integer array mu[1:r], d[1:60];
procedure nadd (al, a2, bl, b2, cl, c2);
value al, a2, bl, b2; real al, a2, bl, b2, cl, c2;
begin
cl: = al + bl; c2: = a2 + b2;
end nadd;
procedure nmult (al, a2, bl, b2, cl, c2);
value al, a2, bl, b2; real al, a2, bl, b2, cl, c2;
begin
cl: = al×bl — a2×b2;
c2: = a2×bl + al×b2;
end nmult;
procedure div(al, a2, bl, b2, cl, c2);
value al, a2, bl, b2;
real al, a2, bl, b2, cl, c2;
begin real nev;
nev: = bl×bl + b2×b2;
cl: = (al×bl + a2×b2)/nev;
c2: = (a2×bl — al×b2)/nev;
end div;
procedure mult(numl, num2, num3);
value numl, num2; integer numl, num2, num3;
begin integer j, k, l, s, x, y;
real xl, x2;
```

```
x: = d[num1]; y: = d[num2]; s: = d[num3]: = x + y;
begin array b[0:s, 1:2];
for k: = 0 step 1 until s do
begin
b[k, 1]: = b[k, 2]: = 0;
if x < k then l: = x else l: = k;
j: = k — y; if j < 0 then j: = 0;
for j: = j step 1 until l do
begin
nmult(a[num1, j, 1], a[num1, j, 2], a[num2, k — j, 1], a[num2, k — j, 2],
      x1, x2);
nadd(b[k, 1], b[k, 2], x1, x2, b[k, 1], b[k, 2]);
end end;
for k: = 0 step 1 until s do
for j: = 1 step 1 until 2 do
a[num3, k, j]: = b[k, j];
end;
end mult;
procedure test(num1, num2, z1, z2);
value num1, num2; integer num1, num2, z1, z2;
begin if d[num1] ≥ d[num2] then
begin z1: = num1; z2: = num2; end else
begin z1: = num2; z2: = num1;
end end test;
procedure add(num1, num2, num3);
value num1, num2; integer num1, num2, num3;
begin integer na, nb, nc, nd;
na: = d[num3]: = d[num1];
nb: = na — d[num2]; nd: = nb — 1;
for nc: = na step —1 until nb do
nadd(a[num1, nc, 1], a[num1, nc, 2], a[num2, nc — nb, 1],
a[num2, nc — nb, 2], a[num3, nc, 1], a[num3, nc, 2]);
for nc: = 0 step 1 until nd do
for na: = 1 step 1 until 2 do
a[num3, nc, na]: = a[num1, nc, na];
end add;
procedure horner(n, a1, k, r, x1, x2);
value n, k, x1, x2;
integer n, k; real x1, x2; array a1, r;
begin integer i, j, l;
array rr[1:2];
rr[1]: = a1[0, 1]; rr[2]: = a1[0, 2];
```

```
for i: = 0 step 1 until k do
for j: = 1 step 1 until 2 do
r[i, j]: = rr[j];
if k > n then begin
for i: = n + 1 step 1 until k do
for j: = 1 step 1 until 2 do
r[i, j]: = 0; end;
for j: = 1 step 1 until n do
begin
nmult(r[0, 1], r[0, 2], x1, x2, r[0, 1], r[0, 2]);
nadd(r[0, 1], r[0, 2], a1[j, 1], a1[j, 2], r[0, 1], r[0, 2]);
l: = if n — j > k then k else n — j;
for i: = 1 step 1 until l do
begin
nmult(r[i, 1], r[i, 2], x1, x2, r[i, 1], r[i, 2]);
nadd(r[i, 1], r[i, 2], r[i — 1, 1], r[i — 1, 2], r[i, 1], r[i, 2]);
end end;
end horner;
input(aa, mu, v, t);
d[r + 1]: = 0; a[r + 1, 0, 1]: = 1; a[r + 1, 0, 2]: = 0;
for i: = 1 step 1 until r do
begin
d[i]: = 1; a[i, 0, 1]: = 1; a[i, 0, 2]: = 0;
a[i, 1, 1]: = —v[i, 1]; a[i, 1, 2]: = —v[i, 2];
hi: = mu[i];
for j: = 1 step 1 until hi do
mult(r + 1, i, r + 1);
end;
for l: = 1 step 1 until r do
begin
k: = mu[l] — 1;
horner(m, aa, k, szaml, v[l, 1], v[l, 2]);
hi: = d[r + 1];
for i: = 0 step 1 until hi do
for j: = 1 step 1 until 2 do
ab[i, j]: = a[r + 1, i, j];
hi: = mu[l];
k: = 2×k + 1;
horner(n, ab, k, neve, v[l, 1], v[l, 2]);
div(szaml[0, 1], szaml[0, 2], neve[hi, 1],
neve[hi, 2], c[l, 1, 1], c[l, 1, 2]);
k: = 0;
```

```
if hi > 1 then
begin
kezd: k: = k + 1;
div(szaml[k, 1], szaml[k, 2], szaml[0, 1],
szaml[0, 2], p1, q1);
div(—neve[hi + k, 1], —neve[hi + k, 2], neve[hi, 1], neve[hi, 2], p2, q2);
nadd(p1, q1, p2, q2, p3, q3);
nmult(c[l, 1, 1], c[l, 1, 2], p3, q3, c[1, k + 1, 1], c[l, k + 1, 2]);
if hi = 3 ∧ k = 1 then begin
nmult(c[l, 2, 1], c[l, 2, 2,], p2, q2, p4, q4);
goto kezd;
end;
if hi = 3 then begin
c[l, 3, 1]: = c[l, 3, 1] + p4;
c[l, 3, 2]: = c[l, 3, 2] + q4;
end;
end hil;
end l;
output(c);
ii: = n + 1;
for i: = 0 step 1 until r do
begin
d[r + 1]: = 0; a[r + 1, 0, 1]: = 1; a[r + 1, 0, 2]: = 0;
for l: = 1 step 1 until r do
begin
d[l]: = 1; a[l, 0, 1]: = 1; a[l, 0, 2]: = 0;
q1: = v[l, 1]×t; q2: = v[l, 2]×t;
p1: = exp(q1);
a[l, 1, 1]: = pr[l]: = —p1×cos(q2);
a[l, 1, 2]: = pi[l]: = —p1×sin(q2);
hi: = mu[l];
if i = 1 then goto vege;
for j: = 1 step 1 until hi do
mult(r + 1, l, r + 1);
if i = 0 ∧ l = r then begin
for kk: = 0 step 1 until n do
for j: = 1 step 1 until 2 do
denom[kk, j]: = a[r + 1, kk, j];
output(denom);
end;
vege: end l;
if i = 0 then goto kt;
```

```
hi: = mu[i];
z2: = ii + 1;
for j: = 1 step 1 until hi do
begin
ii: = ii + 1; l: = d[z2 + hi — j]: = d[r + 1];
for jj: = 0 step 1 until 1 do
for kk: = 1 step 1 until 2 do
a[z2 + hi — j, jj, kk]: = a[r + 1, jj, kk];
if j = hi then goto kt;
mult(r + 1, i, r + 1);
end j;
kt: end i;
z1: = n + 1;
for i: = 1 step 1 until n do
for jj: = 0 step 1 until n do
for kk: = 1 step 1 until 2 do
begin
d[i]: = d[i + z1];
a[i, jj, kk]: = a[z1 + i, jj, kk];
end;
ii: = 0;
for l: = 1 step 1 until r do
begin
ii: = ii + 1; hi: = mu[l];
if hi = 1 then begin
d[n + ii]: = 0; a[n + ii, 0, 1]: = c[l, 1, 1]; a[n + ii, 0, 2]: = c[l, 1, 2];
goto ki;
end hi1;
if hi = 2 then begin
d[n + ii]: = 0;
a[n + ii, 0, 1]: = c[l, 2, 1];
a[n + ii, 0, 2]: = c[l, 2, 2];
ii: = ii + 1; d[n + ii]: = 0;
nmult(c[l, 1, 1], c[l, 1, 2], —t×pr[l], —t×pi[l], a[n + ii, 0, 1], a[n + ii, 0, 2]);
goto ki;
end hi2;
if hi = 3 then
begin
d[n + ii]: = 0; a[n + ii, 0, 1]: = c[l, 3, 1]; a[n + ii, 0, 2]: = c[l, 3, 2];
ii: = ii + 1;
nmult(c[l, 2, 1], c[l, 2, 2], — t×pr[l],
—t×pi[l], a[n + ii, 0, 1], a[n + ii, 0, 2]);
```

```
hi: = ii + 2;
d[n + hi]: = 1;
a[n + hi, 0, 1]: = 1; a[n + hi, 1, 1]: = —pr[l];
a[n + hi, 0, 2]: = 0; a[n + hi, 1, 2]: = —pi[l];
ii: = ii + 1; d[n + ii]: = 0;
p1: = 0.5×t↓2; p2: = p1×c[l, 1, 1]; q2: = p1×c[l, 1, 2];
p3: = —pr[l]; q3: = —pi[l];
nmult[p2, q2, p3, q3, a[n + ii, 0, 1], a[n + ii, 0, 2]);
mult(n + ii, n+ hi, n+ ii);
end hi3;
ki: end l;
ii: = 0;
for l: = 1 step 1 until r do
begin hi: = mu[l];
for j: = 1 step 1 until hi do
begin
ii: = ii + 1;
mult(ii, n + ii, n + ii);
end j;
end l;
a[60, 0, 1]: = 0; a[60, 0, 2]: = 0; d[60]: = 0;
for i: = n + 1 step 1 until n + n do
begin
test(i, 60, z1, z2); add(z1, z2, 60);
end;
d[59]: = 1; a[59, 0, 1]: = 1; a[59, 0, 2]: = 0;
a[59, 1, 1]: = 0; a[59, 2, 2]: = 0;
mult(59, 60, 60);
hi: = d[60];
output(hi);
for i: = 0 step 1 until hi do
for j: = 1 step 1 until 2 do
output(a[60, i, j]);
end end;
```

For the application of this program we attach the parameter list of the program. That is, the parameters are to be read in this order:

r	the number of distinct roots of the denominator
mmax	maximal multiplicity of roots of the denominator
m	degree of the numerator
n	degree of the denominator
aa[0:m, 1:2]	coefficients of the numerator
mu[i]	multiplicity of the root

v[i, 1] real part of the root
v[i, 2] imaginary part of the root
.
.
. $(i = 1, 2, \ldots r)$
t sampling period in secs

Summary

An ALGOL-program is presented for the evaluation of Z[F(s)] for the case of multiple poles of 3 maximum multiplicity.

References

1. CSÁKI, F.: Dynamics of control systems. (in Hungarian). Akadémiai Kiadó, 1966.
2. KÁLMÁN, R. et al.: A system of programs for the real-time digital simulation. Proceedings of MTA AKI, **6** (1968).
3. JURY, E. I.: Theory and application of the Z-transform method. J. Wiley, New-York, 1964.

Tivadar KOVÁCS, Budapest XI., Garami E. tér 3., Hungary