MULTI-STEP NUMERICAL LAPLACE INVERSION

Dalibor BIOLEK
Department of Electrical Engineering
Military Academy of Brno
Kounicova 65, PS13, 612 00 Brno
ČSFR

Abstract

An accurate numerical Laplace inversion algorithm is presented. In comparison with original multi-step inversion method, some improvements are made. The concrete Turbo Pascal routines are introduced.

Keywords:

numerical Laplace inversion, computer program, algorythm, multi-step Laplace inversion, state, equiation, matrix, vector, LU decomposition

Introduction

Various numerical Laplace inversion algorithms have been published so far [1],[2]. However, loss of accuracy with the growing time is their common disadvantage.

An interesting multi-step algorithm is described in [3]. This method is very accurate and numerically stable and also provides accurate periodical solutions. The problem of polynomial Laplace inversion

$$K(s) = \frac{\sum_{i=1}^{m} a_i \cdot s^{i-1}}{s^n + \sum_{i=1}^{n} b_i \cdot s^{i-1}} \qquad (1)$$

is discussed with restrictions

$$m \le n, \quad n > 1 \qquad (2)$$

List of computer program in Fortran is available. However, some misprint probably occurred because the program does not provide right results. The authors set out in [3] that the special form of below mentioned matrix A enables to simplify their algorithm, but without closed explanation.

In view of that facts we decided to form own algorithm. In comparison with [3], our algorithm has less strict conditions:

$$m \le n + 1, \quad n >= >1 \qquad (3)$$

Multi-step Laplace inversion [3]

Equation (1) can be rewritten as follows:

$$K(s) = \frac{Y}{W} = \frac{\sum_{i=1}^{m} a_i \cdot s^{-(n+1-i)}}{1 + \sum_{i=1}^{n} b_i \cdot s^{-(n+1-i)}} \qquad (4)$$

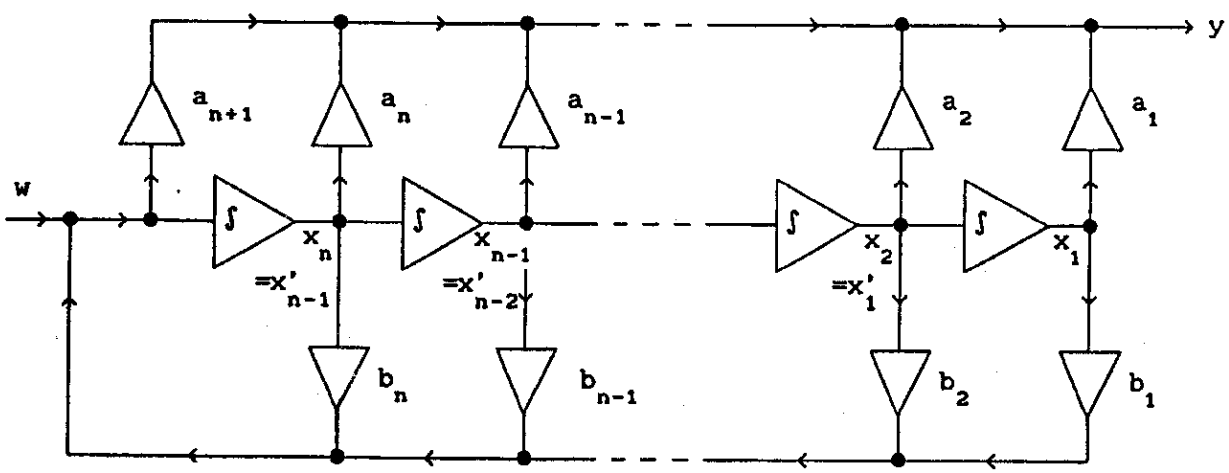

Fig.1
System with transfer function (1), m = n+1.

The state system model with the input w and output y in Fig.1 corresponds with this equation. The state equations are true:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdot\cdot & 0 \\ 0 & 0 & 1 & 0 & \cdot\cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot\cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot\cdot & \cdot \\ 0 & 0 & 0 & 0 & \cdot\cdot & 1 \\ -b_1 & -b_2 & -b_3 & -b_4 & \cdot\cdot & -b_n \end{bmatrix} \cdot$$

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \\ 1 \end{bmatrix} \cdot w \qquad (5)$$

$$\frac{d}{dt}\mathbf{x} = \mathbf{A} \,.\, \mathbf{x} + \mathbf{B} \,.\, w \qquad (6)$$

Applying the Laplace transform yields

$$(s \,.\, \mathbf{E} - \mathbf{A}) \,.\, \mathbf{X} = \mathbf{R}$$

$$\mathbf{R} = \mathbf{B} \,.\, W + \mathbf{x}^0 \qquad (7)$$

where

$\mathbf{R} = \left[R_1, R_2, .., R_n\right]$ – right side vector;

$\mathbf{x}^0 = \left[x_1^0, x_2^0, .., x_n^0\right]$ – state vector of initial conditions;

$\mathbf{E}$ – identity matrix.

In harmony with [3], the numerical Laplace inversion of (1) will be solved as follows:

1- We set $\mathbf{x}^0 = \mathbf{0}$, $W = 1$, the time step $t = h$ and final time *tmax*.

2- Solution of matrix equation (7) for $s = z_i/h$, $i = 1, ..., M/2$.

The complex numbers z_i are the approximating coefficients of rational function $R_{M,N}(z) \approx e^z$ [3]. The accuracy of Laplace inversion depends on numbers M and N. The precise values of z_i for $M = 2$, 4 and 10 are published in [3].

Solving (7), we obtain $M/2$ state vectors $\mathbf{X}^i$, $i = 1, ..., M/2$ in complex domain.

3- Compute the state vector in time t:

$$\mathbf{x} = -\frac{1}{h}\sum_{i=1}^{M/2} \mathrm{Re}\left\{K P^i \mathbf{X}^i\right\}, \qquad (8)$$

where $K P^i$, $i = 1, ..., M/2$ is set of complex constant published in [3].

4- The output y in time t can be computed from Eq. (6).

5- Let $t = t + h$. If $t > tmax$, the program is terminated.

6- We set $\mathbf{x}^0 = \mathbf{x}$, $W = 0$, and go to the step 2.

In the first step we set $W = 1$ - the Laplace transform of Dirac impulse. The impulse response $g(t=h) = L^{-1}\left\{K(s)\right\}$, $t = h$, is computed in following

```
Program INVLAP;
{$N + }
uses crt;
const
nmax = 20;
type
vec = array[1..nmax] of double;
mat = array[1..nmax] of vec;
complex = record
Re: double;
Im: double;
end;
vecc = array[1..nmax] of complex;
matc = array[1..nmax] of vecc;
var
i,order: integer;
a,b,x,w: vec;
t,respon:real;
{-------------------------------------------------------------
Procedures addc,subc,mulc and divc for complex addition,substraction,
multiplication and division must be included
{-------------------------------------------------------------}
procedure VECTORZERO(var b:vec);
begin fillchar(b,sizeof(b),#0); end;
{-------------------------------------------------------------}
procedure VECTORZEROC(var b:vecc);
begin fillchar(b,sizeof(b),#0); end;
{-------------------------------------------------------------}
procedure Ht(a,b:vec;n:integer;tmax:real;nump:word);
{Laplace inversion of polynomial transfer function K(s)
in time instants (1,2,3,..,nump)*tmax/nump }

label BEG,FIN;
var i,j,kr:integer;
x:vec;
kp,w,res,xpom:vecc;
step:double;
z,pomc:complex;
begin
step:= tmax/nump;
fillchar(res,sizeof(res),#0); {-------------------}
z.re:= 1/step;z.im:= z.re;pomc.re:= 0;pomc.im:= 0;
for j:= 1 to n-1 do
begin
res[j]:= pomc;res[j].re:= res[j].re + b[j]; { LU decomposition }
DIVC(res[j],z,res[j]);pomc:= res[j];
end;
res[n].re:= z.re + pomc.re + b[n];res[n].im:= z.im + pomc.im;{------
-------------}
VECTORZEROC(w);w[n].re:= 1;kr:= 1;respon:= a[n + 1];
if a[n + 1] <  > 0 then writeln('Dirac impulse in
t = 0,strength = ',a[n + 1]);
BEG:VECTORZERO(x);VECTORZEROC(xpom);xpom[n]:= w[n];
for j:= 1 to n-1 do
begin
MULC(res[j],w[j],pomc);SUBC(xpom[n],pomc,xpom[n]);
end;
DIVC(xpom[n],res[n],xpom[n]);
for j:= n-1 downto 1 do
begin
ADDC(xpom[j + 1],w[j],xpom[j]);DIVC(xpom[j],z,xpom[j]);
end;
for j:= 1 to n do x[j]:= -2*xpom[j].im/step;
for i:= 1 to n do respon:= respon + (a[i]-a[n + 1]*b[i])*x[i];
{-------------------------------------------------------------}
{response plotting, for example}
writeln(t*kr/nump,sin(t*kr/nump),respon);
{-------------------------------------------------------------}
respon:= 0;kr:= kr + 1;if kr > nump then goto FIN;
VECTORZEROC(w);
for i:= 1 to n do w[i].re:= x[i];
goto BEG;
FIN:end;
{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 = = = MAIN PRO-
GRAM - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 = }
begin
{ K(s) = (a1 + a2s + .. + a(n + 1)s ^ n)/(b1 + b2s + .. + 1*s ^ n) }
order:= 2;VECTORZERO(a);VECTORZERO(b);a[1]:= 1;b[1]:= 1;
t:= 20;
Ht(a,b,order,t,200);
repeat until keypressed;
end.
```

Fig.2

List of program for numerical Laplace inversion; M = 2, N = 0.

Procedures ADDC, SUBC, MULC and DIVC for complex addition, subtraction, multiplication and division must be completed

steps 2 to 4. At the end of first run, the state vector becomes the vector of initial conditions for the transient analysis in second run. The input signal W must already be zero.

Taking entire response solution to pieces, the numerical mistake of classical "single-step" inversion algorithm is essentially reduced.

Solution of complex matrix state equation

The most computationally expensive part of algorithm is concentrated in step 2. In accordance with [3], the LU decomposition was used for equation (7). We describe our procedure how to do it fast and economically.

The LU decomposition of matrix $s\mathbf{E} - \mathbf{A}$ can be performed as follows:

```
procedure Ht(a,b:vec;n:integer;tmax:real;nump:word);
{Laplace inversion of polynomial transfer function K(s)
in time instants (1,2,3,...,nump)*tmax/nump }

label BEG,FIN;
var i,j,kr:integer;
x:vec;
z,kp,w,xpom:vecc;
res:array[1..2,1..nmax + 1] of complex;
step:double;
pomc:complex;
begin
z[1].re: = 3.77901986701093;z[1].im: = 1.380176524272843;
z[2].re: = 2.220980032988807;z[2].im: = 4.160391445506932;

kp[1].re: = 2.256958744418140;kp[1].im: = -39.63308700050173;
kp[2].re: = -2.256958744418140;kp[2].im: = 11.10883163787590;
step: = tmax/nump;
fillchar(res,sizeof(res),#0);
for i: = 1 to 2 do
begin
z[i].re: = z[i].re/step;z[i].im: = z[i].im/step;
pomc.re: = 0;pomc.im: = 0;
for j: = 1 to n-1 do
begin
res[i,j]: = pomc;res[i,j].re: = res[i,j].re + b[j];
DIVC(res[i,j],z[i],res[i,j]);pomc: = res[i,j];
end;
res[i,n].re: = z[i].re + pomc.re + b[n];
res[i,n].im: = z[i].im + pomc.im;
end;
VECTORZEROC(w);w[n].re: = 1;kr: = 1;respon: = a[n + 1];
if a[n + 1] < > 0 then writeln('Dirac impulse in
t = 0,strength = ',a[n + 1]);
BEG:VECTORZERO(x);VECTORZEROC(xpom);
for i: = 1 to 2 do
begin
xpom[n]: = w[n];
for j: = 1 to n-1 do
begin
MULC(res[i,j],w[j],pomc);SUBC(xpom[n],pomc,xpom[n]);
end;
DIVC(xpom[n],res[i,n],xpom[n]);
for j: = n-1 downto 1 do
begin
ADDC(xpom[j + 1],w[j],xpom[j]);DIVC(xpom[j],z[i],xpom[j]);
end;
for j: = 1 to n do x[j]: = x[j] + (kp[i].im*xpom[j].im-
kp[i].re*xpom[j].re)/step;
end;
for i: = 1 to n do respon: = respon + (a[i]-a[n + 1]*b[i])*x[i];
{-------------------------------------------------------------------}
{response plotting, for example}
writeln(t*kr/nump,sin(t*kr/nump),respon);
{-------------------------------------------------------------------}
respon: = 0;kr: = kr + 1;if kr > nump then goto FIN;
VECTORZEROC(w);
for i: = 1 to n do w[i].re: = x[i];
goto BEG;
FIN:end;
```

Fig.3a
Procedure Ht for more precise calculation, M = 4, N = 2

```
In procedure Ht, below mentioned coefficients must be used:

z[1].re: = 11.830093739168819;z[1].im: = 1.583753005885813;
z[2].re: = 11.220853779395119;z[2].im: = 4.792984167585689;
z[3].re: = 9.833383722175002;z[3].im: = 8.033106334266296;
z[4].re: = 7.781146264464616;z[4].im: = 11.388891684904993;
z[5].re: = 4.234522494797000;z[5].im: = 14.957043781281156;

kp[1].re: = 16286.62368050479;kp[1].im: = -139074.711551 6051;
kp[2].re: = -28178.111171305162;kp[2].im: = 74357.582372741 76;
kp[3].re: = 14829.740252331 42;kp[3].im: = -19181.808185018 36;
kp[4].re: = -2870.418161032078;kp[4].im: = 1674.109484084304;
kp[5].re: = 132.165941247 4876;kp[5].im: = 17.476747988771 84;

Cycles in variable i are now from 1 to 5.
Variable res is now
res: array [1..5,1..nmax + 1] of complex;
```

Fig.3b
Procedure Ht for more precise calculation, M = 10, N = 8

$$s\mathbf{E} - \mathbf{A} = \begin{bmatrix} s & -1 & 0 & 0 & .. & 0 & 0 \\ 0 & s & -1 & 0 & .. & 0 & 0 \\ : & & & & & & : \\ 0 & 0 & 0 & 0 & .. & s & -1 \\ b_1 & b_2 & b_3 & b_4 & .. & 0 & b_n \end{bmatrix} =$$

$$= \underbrace{\begin{bmatrix} 1 & 0 & .. & 0 & 0 \\ 0 & 1 & .. & 0 & 0 \\ : & & & & : \\ 0 & 0 & .. & 1 & 0 \\ \alpha_1 & \alpha_2 & .. & \alpha_{n-1} & 1 \end{bmatrix}}_{\mathbf{L}} \cdot \underbrace{\begin{bmatrix} s & -1 & 0 & .. & 0 & 0 \\ 0 & s & -1 & .. & 0 & 0 \\ : & & & & & : \\ 0 & 0 & 0 & .. & s & -1 \\ 0 & 0 & 0 & .. & 0 & \beta \end{bmatrix}}_{\mathbf{U}} \quad (9)$$

The $\alpha_1, .., \alpha_n$ and β coefficients are computed by following procedure:

$$\begin{aligned} \alpha_1 &= \frac{b_1}{s} \\ \alpha_2 &= \frac{b_2 + \alpha_1}{s} \\ &: \\ \alpha_{n-1} &= \frac{b_{n-1} + \alpha_{n-2}}{s} \\ \beta &= s + b_n + \alpha_{n-1} \end{aligned} \quad (10)$$

Investigating Eq.(9), the single vector is sufficient for storing all needed elements of matrices L and U:

$$vec = \left[\alpha_1, .., \alpha_{n-1}, \beta\right] . \quad (11)$$

After calculating (11), set of equations (7) can be solved in classical manner. Taking account of L and U matrices structure, the resulting vector $\mathbf{X} = \left[X_1, X_2, .., X_n\right]$ components are

$$X_n = \frac{R_n - \alpha_1 R_1 - \alpha_2 R_2 - .. - \alpha_{n-1} R_{n-1}}{\beta}$$

$$X_{n-1} = \frac{X_n + R_{n-1}}{s}$$

$$X_{n-2} = \frac{X_{n-1} + R_{n-2}}{s} \qquad (12)$$

$$X_1 = \frac{X_2 + R_1}{s}$$

Results

The list of Turbo Pascal program for numerical Laplace inversion is in Fig.2. For simplicity, only approximation $R_{2,0}(z)$ - $M = 2, N = 0$ has been used. For more precise calculation, a better approximation can be chosen, but with longer computing time (see procedures Ht in Fig.3:

a) $M = 4, N = 2$,
b) $M = 10, N = 8$.

Comparison of precision in numerical inversion for the three aforementioned approximations is in Fig.4. The curves were derived by inversion of $K(s) = 1/(s^2 + 1)$ with periodical solution $\sin(t)$, $tmax = 20$ sec.

References

[1] ZAKIAN,V.: Rational Approximation of Transfer Function Matrix of Distributed Systems. Electronics Letters, 6, No.15, 1970.

[2] MANN,H.: Využití počítače při elektrotechnických návrzích. SNTL/ALFA, Praha 1984. In Czech.

[3] VLACH,J.-SINGHAL,K.: Mašinnyje metody analiza i projektirovanija elektronnych schem. Moskva 1988. Russian translation from "Computer Methods for Circuit Analysis and Design", New York, Van Nostrand - Reinhold, 1983.

About author, ...

Dalibor Biolek was born in Ostrava, Czechoslovakia, in 1959. He received the Ing. degree from the Technical University of Brno, Czechoslovakia, in 1983, and the CSc. (Ph.D.) degree in radioelectronics in 1989. He is currently the lecturer at the department of Electrical Engineering of the Military Academy Brno. His research interests include switched capacitor networks, computer-aided analysis and numerical methods in circuit theory.

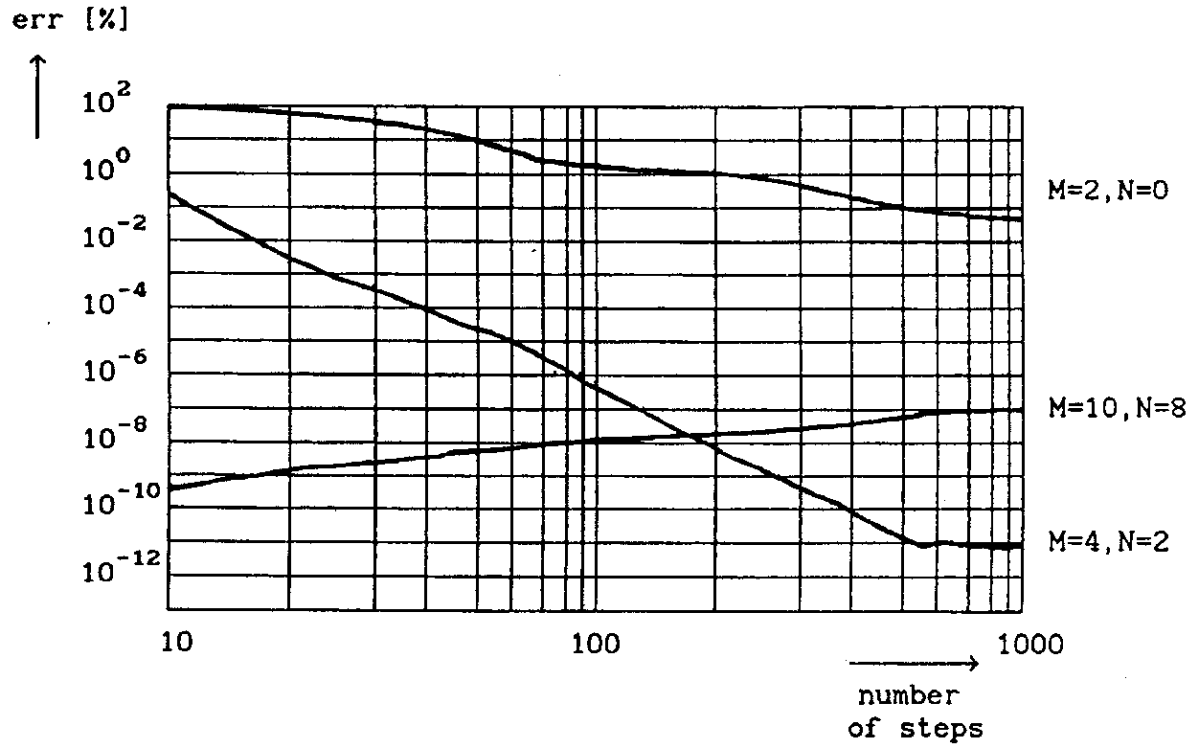

Fig.4

Precision of Laplace inversion for various approximating coefficients, $K(s) = \frac{1}{s^2 + 1}$