УПРОЩЕНИЕ ПЕРЕДАТОЧНЫХ ФУНКЦИЙ С ПОМОЩЬЮ НЕПРЕРЫВНОЙ ДРОБИ

Т. КОВАЧ и Л. ЛАКАТОШ
Кафедра Автоматизации Будапештского Технического Университета

(Поступила 18-го марта 1972 г.)
Представлена проф. др. Ф. Чаки

Введение

При анализе и синтезе системы регулирования часто возникает потребность, чтобы передаточная функция приближалась к передаточной функции, степень которой ниже оригинальной, или матрица состояния к матрице состояния, степень которой меньше оригинальной. Задача обычно разрешается так, что из полюсов передаточной функции оставляются только доминантные полюсы, а из собственного значения матриц состояния — только доминантные собственные значения. Имеется такое разрешение, что корни, соответствующие круговой частоте в точке перелома амплитудовой — частотной характеристики пропускаются, если амплитуда в точке перелома находится над высшим или под самым низким лимитом. При этих методах основную роль играют доминантные корни, но передаточные функции системы автоматического регулирования часто не имеют доминантых корней. Поэтому метод доминантных корней не может быть распространенным.

Суть демонстрируемого метода (1) заключается в нижеследующем: Если степень передаточной функции высока, то функцию с помощью непрерывной дроби приведем к выраженной форме и отдаленные члены непрерывной дроби откинем. Если систему автоматического регулирования характеризуем системой уравнений состояния, то упрощение означает, что определенные части матрицы состояния не принимаются во внимание.

Связь передаточной функции, выраженной непрерывной дробью, с блок-схемой

Элементарным математическим методом можем увидеть (2), что

$$R(s) = \frac{a_0 + a_1 s + a_2 s^2 + \dots}{b_0 + b_1 s + b_2 s^2 + \dots}$$

дробно-рациональная функция может быть выражена в следующей форме:

$$R(s) = H_1 + \cfrac{s}{H_2 + \cfrac{s}{H_3 + \dots}} \,. \qquad (1)$$

Альгоритм вычисления элементов непрерывной дроби, например, находится в (2). Если степень числителя $R(s)$ выше, чем знаменателя, тогда число значений Hi есть $2n$ (при степени числителя n), в противном случае $2p + 1$ (при степени числителя p). (1) можно записать и таким образом:

$$R(s) = H_1 + \cfrac{1}{\cfrac{H_2}{s} + \cfrac{1}{H_3 + \cfrac{1}{\cfrac{H_4}{s} + \ldots}}}.$$

В нижеследующем будем заниматья такой передаточной функцией, где степень знаменателя выше степени числителя. В таких случаях $1/W(s)$ целесо-

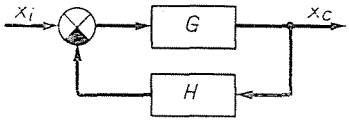

Рис. 1

образно записать в форме непрерывной дроби. Если степень знаменателя $W(s)$ есть n, тогда

$$W(s) = \cfrac{1}{H_1 + \cfrac{1}{\cfrac{H_2}{s} + \ldots + \cfrac{1}{\cfrac{H_{2n}}{s}}}}. \qquad (2)$$

На основании (2) можем записать систематически одну блоксхему. С целью его подготовления проверим систему автоматического регулирования, находящуюся на рис. 1.

G и H передаточные коэффициенты.

Передаточный коэффициент замкнутой системы:

$$W = \frac{G}{1+GH} = \frac{1}{H + \dfrac{1}{G}}.$$

Это самый простой вид непрерывной дроби.

Если «G» большой, то

$$W = \frac{1}{H},$$

то есть поведение системы основательно определяет H. На рис. 2. изображена система немножко сложнее чем выше,
так как

$$W = \frac{1}{H + \dfrac{1}{F_1 + G_1}}. \tag{3}$$

Аппроксимации, упомянутой в прежнем случае соответствует, если $(F_1 + G_1)$ пропускаем. Видимо, во много раз лучшая аппроксимация, если пропустим только одну из них.

Поэтому для общей передаточной функции, обозначенной (2) можем нарисовать блоксхему рис. 3.

При сравнении рис. 2 и 3, а также (3) и (2) выражению H в (3) соответствует H_1 в (2), F_1 соответствует H_2/s, а G_1 соответствует непрерывная дробь, последующая H_2/s. (На рис. 3. соответствующая блоксхема отмечена штриховой линией.) Нерперывная дробь, последующая H_2/s по своему характеру сходится с основной непрерывной дробью. Соответственно вышеизложенным, на основании (2) действительно можно нарисовать рис. 3.

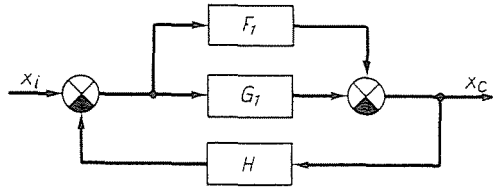

Рис. 2

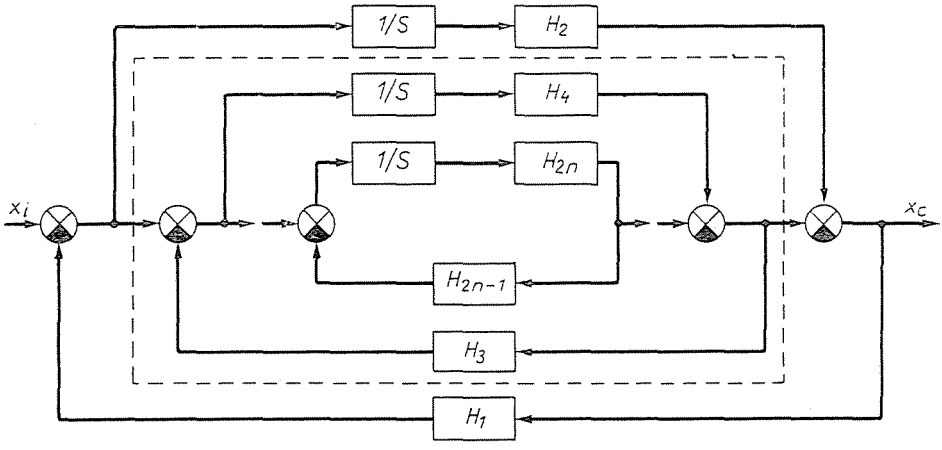

Рис. 3

Определение уравнений состояния на основании передаточных функций, выраженных непрерывной дробью

Если выходы интеграторов блоксхем рисунка 3. считаем переменными состояния, то систему можно характеризовать следующей системой дифференицальных уравнений.

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} H_2 H_1 & H_4 H_1 & \dots & H_{2n} H_1 \\ H_2 H_1 & H_4(H_1+H_3) & \dots & H_{2n}(H_1+H_3) \\ \vdots & & & \\ H_2 H_1 & H_4(H_1+H_3) & \dots & H_{2n}(H_1+H_3+\dots+H_{2n+1}) \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} x_i$$

А выражение выходного сигнала

$$x_c = H_2 z_1 + H_4 z_2 + \dots + H_{2n} z_n .$$

На основании настоящей блоксхемы правильность вышеуказанных двух формул можно непосредственно признать.

На основании этого, элемент, стоящий в строке k и в столбце p матрицы можно записать как умножение $A[k, p]$ на $B[k, p]$, где

$$A[k, p] = H_{2p}$$

$$B[k, p] = B[p, k] \text{ и } B[k, k] = \sum_{j=1}^{k} H_{2j-1}$$

$$B[k, p] = B[p, p] \text{ если } k > p$$

$$B[k, p] = B[k, k[\text{ если } k < p .$$

Программа для упрощения передаточной функции

Входные данные программы: степень знаменателя передаточной функции и коэффициенты передаточной функции, заданной отношением полиномов.

В результате мы получаем коэффициенты аппроксимационных передаточных функций разных степеней. На основании условий Рауса проводим и проверку устойчивости аппроксимационных передаточных функций.

Краткое описание структуры программы:

В первую очередь нужно вычислить элементы непрерывной дроби. За этим следует вычисление упрощенных передаточных функций в два цикла. Внешний цикл по одному определяет упрощенные передаточные функции, данные k, например, с принятием во внимание элементов $2k$, и делает проверку устойчивости аппроксимационных передаточных функций на основании условий Рауса. На вычисление упрощенных передаточных функций очередь

попадает во внутреннем цикле. Процедура *smr*, фигурирующая в этом цикле принимает обратное значение дробно-рациональной функции и умножает на *s* а процедура *adh* додает ему один констант. Программа находится в прил.1.

Пример

Упрощаемая передаточная функция:

$$W(s) = \frac{s^2+15s+50}{s^4+5s^5+32s^2+79s+50}.$$

Аппроксимация третьего порядка:

$$W(s) = \frac{18.053s^2+89.26s-310}{s^3-175.57s^2-486.06s-310}$$

неустойчивый, значит непринимаемый.

Аппроксимация второго порядка:

$$W_2(s) = \frac{0.0349s+3.7313}{s^2+4.811s+3.7313}$$

а аппроксимация первого порядка:

$$W_1(s) = \frac{0.7812}{s+0.7812}.$$

Программа для определения матрицы состояния

Определение матрицы состояния на основании изложенных о непрерывной дроби является уже очень сложным заданием. Но очень важно, является ли матрица состояния аппроксимации матрицей состояния устойчивости системы. Проверку устойчивости мы проводим по методу Ляпунова за использованием процедуры так называемых неопределенных элементов (3), (4).

Суть этого метода коротко заключается в нижеследующем:

Допустим, что линейную систему характеризует следующая система однородных дифференциальных уравнений:

$$\dot{x} = \mathbf{A}x$$

Функция Ляпунова будет:

$$V(x) = x^T \mathbf{P} x .$$

Её производная:

$$\dot{V}(x) = x^T (\mathbf{A}^T \mathbf{P} + \mathbf{P}\mathbf{A}) x .$$

Для ассимптотической устойчивости важно, чтобы в случае положительно определенного p, $\mathbf{A}^T \mathbf{P} + \mathbf{P}\mathbf{A}$ был минимально отрицательно полуопределенный. Но если мы берем любое $\mathbf{P}$ (но положительно определенным), то $\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A}$ вообщем будет неопределенным.

Используем следующее обозначение:

$$\mathbf{A}^T \mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{D} . \tag{4}$$

На основании вышеизложенных, в случае положительно определенного $\mathbf{P}$, $\mathbf{D}$ должен быть минимум положительно полуопределенный. Можно доказать (3), если на $\mathbf{D}$ принимаем хотя положительно определенную матрицу, то положительная определенность полученного $\mathbf{p}$ будет нужным и довольным условием для ассимптотической устойчивости системы. Нужно, значит, вычислить (4) на «$\mathbf{P}$». Это мы можем сделать следующим образом: (4).

Из элементов $\mathbf{A}$ нужно конструировать такую матрицу α, из элементов $\mathbf{D}$ такую столбцовую матрицу δ, а из элементов «$\mathbf{P}$» такую столбцовую матрицу π, чтобы (4) был эквивалентный с уравнением матриц

$$\alpha\pi = -\frac{1}{2}\delta , \tag{5}$$

которое уже возможно решить по методу Гаусса на «π». После этого из элементов π можно составить $\mathbf{P}$, положительную определенность которого можно проверить на основании теоремы Сильвестра; (5). Структуру программы коротко можно формулировать так:

В первую очередь нужно вычислить элементы непрерывной дроби. За этим следует определение матрицы состояния. Потом программа циклически проверяет условия устойчивости аппроксимационных матриц состояния.

В этом цикле находится и процедура *supmtx*, которая на основании матрицы состояния определяет матрицу α; процедура *gaussj*, которая на основании (5) определяет π, и процедура *arrnge*, которая из π определяет $\mathbf{P}$. Положительную определенность $\mathbf{P}$ проверяем в одном внутреннем цикле. Проверка проводится процедурой *levsyl*, процедура расчитывает детерминант, и если этот меньше нуля, докажет неустойчивость.

Передаточной функции, фигурирующей в предыдущем примере соответствует следующая матрица состояния:

$$\mathbf{A} = \begin{bmatrix} -0.78125 & 0.7462686 & -18.01875 & 18.05373 \\ -0.78125 & -4.02955 & 97.30125 & -97.49015 \\ -0.78125 & -4.02985 & 180.3812 & -180.7314 \\ -0.78125 & -4.02985 & 180.3812 & -180.5701 \end{bmatrix}$$

при условии $\mathbf{D} = 2\mathbf{J}$ из решения (4)

$$\mathbf{P} = \begin{bmatrix} 1.152108 & 0.162576 & -8.879069 & 3.844363 \\ 0.162576 & 1.041095 & -21.84313 & 21.08028 \\ -3.879069 & -21.84313 & 1378.177 & -1366.787 \\ 3.844363 & 21.08028 & -1366.787 & 1357.017 \end{bmatrix}.$$

Детерминанты:

$$d_4 = 2879.244$$
$$d_3 = 1077.971$$
$$d_2 = 1.173024$$
$$d_1 = 1.152108$$

Если (4) решим на основании аппроксимационной матрицы состояния третьего порядка:

$$\mathbf{P} = \begin{bmatrix} 1.134323 & 0.708393 & 0.07483673 \\ 0.0708393 & 0.5591244 & 0.2978578 \\ 0.07483673 & 0.2978578 & 0.1626023 \end{bmatrix}.$$

Детерминанты:

$$d_3 = -0.004614958$$
$$d_2 = 0.62921$$
$$d_1 = 1.134323$$

Эта аппроксимация аналогично результатам, полученным предыдущим методом, является тоже неустойчивой.
При аппроксимации второго порядка:

$$\mathbf{P} = \begin{bmatrix} 1.146476 & 0.1335236 \\ 0.1335236 & 0.2728747 \end{bmatrix},$$

так как

$$d_2 = 0.2950158$$
$$d_1 = 1.146476$$

Эта аппроксимация устойчивая.

При аппроксимации первого порядка:

$$\mathbf{P} = [1.28]$$

то есть и эта аппроксимация устойчивая.

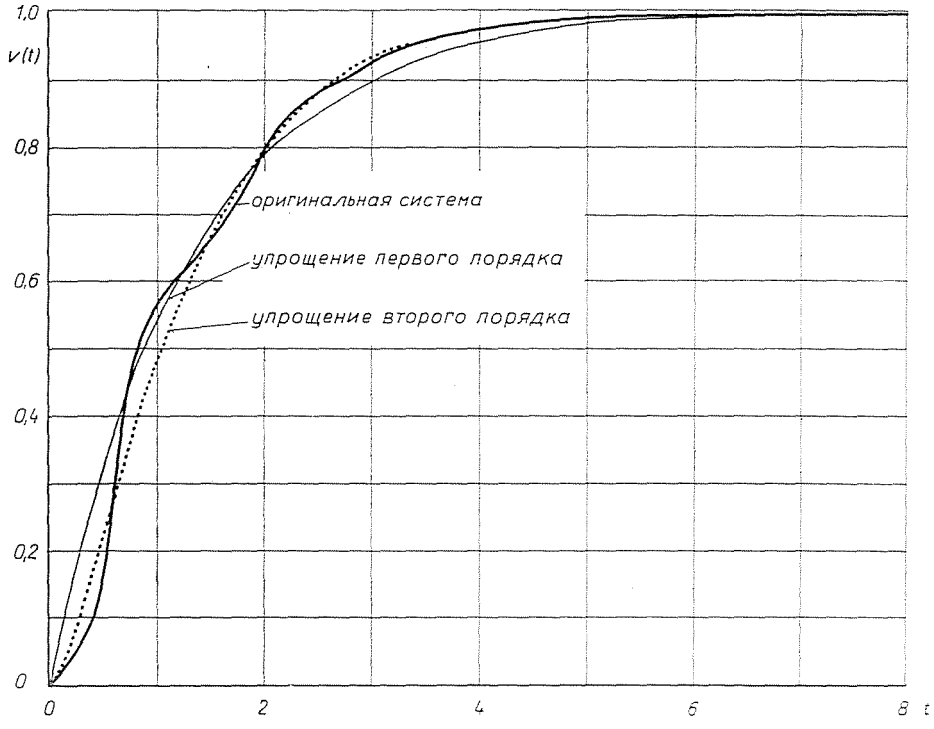

Рис. 4

Переходные функции оригинальных и аппроксимационных систем мы вычислили с помощью программы для моделирования системы. Переходные функции можем увидеть на рис. 4. Как видно, даже аппроксимация первого порядка имеет хорошее качество.

Программы были выполнены для электронно-цифровой вычислительной машины типа МИНСК-22 Исследовательского Института Автоматизации Академии Наук ВНР.

Резюме

Представлен метод для упрощения передаточных функций с помощью непрерывной дроби. Если степень передаточной функции высока, то функцию с помощью непрерывной дроби приведем к выраженной форме и отдаленные члены непрерывной дроби откинем. Если систему характеризуем системой уравнений состояния, то упрощение означает, что определенные части матрицы откинем. Представлены программы, решающие задачи.

Литература

1. Chen, C. F.—Shieh, L. S.: A novel approach for linear model simplification. Int. J. Control **8**, (1968).
2. Vlach, J.: Computerized analysis of linear networks. New York, 1969.
3. Schultz, D.—Melsa, J.: State functions and linear control systems. New York, 1967.
4. Chen, C. F.—Haas, I. J.: Elements of control systems analysis. New York, 1968.
5. Csáki, F.: Korszerű szabályozáselmélet. Akadémiai Kiadó. Budapest, 1970.
5/a. Csáki, F.: Modern Control Theories. Nonlinear, Optimal and Adaptive Systems. Akadémiai Kiadó. Budapest, 1972.
6. Lőcs, Gy.—Vigassy, J.: FORTRAN programozási nyelv. Budapest, 1970.

Прил. 1.

```
begin==
integer n,z,k,i,dn,dd;==
real x;==
standard('40',n): z:=2×n;==
begin==
array a,b[0:n+1],s[0:n],r[1:z],t[-1:2,0:n];==
procedure smr(z); integer z:==
begin integer d,j;==
for j:=0 step 1 until dd do==
t[2,j+1]:=t[-1,j];==
t[2,0]:0;==
for j:=0 step 1 until dn do==
t[-1,j]:=t[1,j];==
for j:=0 step 1 until dd+1 do==
t[1,j]:=t[2,j];==
d:=dn; dn:=dd+1; dd:=d==
end smr:==
procedure adh(k); integer k;==
begin integer j:==
for j:=0 step 1 until dn do==
t[2,j]:=r[k]×t[-1,j];==
for j:=0 step 1 until dd do==
t[1,j]:=t[1,j]+t[2,j]==
end adh;==
procedure routh(dn); integer dn:==
begin integer j,k; real q:==
q:=sign(t[1,0]):==
for j:=1 step 1 until dn do==
if sign(t[1,j])≠q then==
begin j:0; goto adr end;==
if q<0 then==
for j:=0 step 1 until dn do t[1,j]:=-t[1,j];==
```

```
if t[1,dn]=0 or t[1,dn−1]=0 then==
begin j:=0; goto adr end;==
for k:=dn−1 step −1 until 1 do==
begin q:=−t[1,k+1]/t[1,k];==
for j:=k−1 step −2 until 1 do==
t[1,j]:=t[1,j]+q×t[1,j−1];==
if t[1,k−1]<0 then begin j:=0; goto adr end==
end;==
j:1;==
adr: standard('2',j)==
end routh;==
for i:=0 step 1 until n do==
begin standard ('40',x); a[i]:=x;==
standard('40',x); b[i]:=x; s[i]:=0;==
end;==
a[n+1]:=b[n+1]:=0; k:=1;==
if b[0]=0 then goto a1;==
r[1]:=a[0]/b[0];==
for k:=2 step 1 until z do==
begin x:=a[0]/b[0];==
for i:=0 step 1 until n do==
s[i]:=a[i+1]−x×b[i+1];==
if s[0]=0 then goto a1;==
r[k]:=b[0]/s[0];==
for i:=0 step 1 until n do==
begin a[i]:=b[i]; b[i]:=s[i]==
end==
end;==
for k:=1 step 1 until z do==
standard('2',r[k]);==
for z:=z step −2 until 2 do==
begin t[1,0]:=r[z]; t[−1,0]:=1;==
dn:=dd:=0;==
for i:=z−1 step −1 until 1 do==
begin smr(z); adh(i)==
end;==
standard('2',dn);==
for i:=0 step 1 until dn do==
standard('2',t[1,i]);==
for i:=0 step 1 until dd do==
standard('2',t[−1,i]);==
routh(dn);==
```

```
end; goto a2;==
a1: i:=111; standard('2',i,k);==
a2: end==
end;==
```

Прил. 2.

```
begin==
integer n, ni,nj;==
standard('40',n); ni:=n×(n+1)/2; nj:=ni+1;==
begin==
integer i,j,k,l,m,lm,f,fi,fj;==
real s;==
array b[1:n,0:n],a[1:ni,0:nj],p[1:n,1:n];==
procedure supmtx(f,b,a); integer f; array a,b;==
begin==
m:=0;==
for i:=1 step 1 until f do==
for j:=i step 1 until f do==
begin m:=m+1; a[m,fj]:=0; lm:=0;==
if i=j then a[m,fj]:=-1.0;==
for k:=1 step 1 until f do==
for l:=k step 1 until f do==
begin lm:=lm+1;==
if k≠i∧k≠j∧l≠i∧l≠j then==
begin a[m,lm]:=0; goto a7 end;==
if k=1∧l≠j then==
begin a[m,lm]:=b[l,j]; goto a7 end;==
if k≠i∧l=j then==
begin a[m,lm]:=b[k,i]; goto a7 end;==
if k≠i∧l≠j∧k=j∧l≠i then==
begin a[m,lm]:=b[l,i]; goto a7 end;==
if k≠i∧l≠j∧k≠j∧l=i then==
begin a[m,lm]:=b[k,j]; goto a7 end;==
if k=i∧l=j∧k=l then==
begin a[m,lm]:=b[k,k]; goto a7 end;==
if k=i∧l=j∧k≠l then==
a[m,lm]:=b[k,k]+b[l,l];==
a7: end==
end;==
standard('2',fi,fj);==
end supmtx;==
procedure gauss(fi,a); integer fi; array a;==
```

```
begin==
for i:=1 step 1 until fi do==
begin if a[i,i]=0 then goto a1 else s:=1/a[i,i];==
for k:=i+1 step 1 until fj do==
a[i,k]:=s×a[i,k];==
for j:=1 step 1 until fi do if j≠i then==
begin s:=a[j,i];==
for k:=i+1 step 1 until fj do==
a[j,k]:=a[j,k]-s×a[i,k];==
end;==
end; goto a2;==
a1:i:=222; standard('2',i); goto a3;==
a2: for i:=1 step 1 until fi do==
standard('2',a[i,fj]);==
end gaussj;==
procedure arrnge(f,p); integer f; array p;==
begin==
k:=0;==
for i:=1 step 1 until f do==
for j:=i step 1 until f do==
begin k:=k+1;==
p[i,j]:=p[j,i]:=a[k,fj];==
end;==
standard('2',f);==
for i:=1 step 1 until f do==
for j:=1 step 1 until f do==
standard('2',p[i,j]);==
end arrnge;==
procedure levsyl( ,lm); integer 1,lm;==
begin integer i,j,k,m;==
real z;==
array u,v[1:n,1:n],alfa[1:n];==
for j:=1 step 1 until 1 do==
for k:=1 step 1 until 1 do==
v[j,k]:=0;==
for j:=1 step 1 until 1 do==
v[j,j]:=1.0;==
for i:=1 step 1 until 1 do==
begin for j:=1 step 1 until 1 do==
for k:=1 step 1 until 1 do==
begin z:=0;==
for m:=1 step 1 until 1 do==
```

```
z:=z+p[j,m]×v[m,k];==
u[j,k]:=z==
end;==
z:=0;==
for m:=1 step 1 until l do==
z:=z+u[m,m]; z:=-z/i;==
if entier(l/2)≠entier( (l+1)/2)==
then alfa[i]:=-z==
else alfa[i]:=z;==
for j:=1 step 1 until l do==
for k:=1 step 1 until l do==
v[j,k]:=u[j,k];==
for j:=1 step 1 until l do==
v[j,j]:=v[j,j]+z;==
end;==
if alfa[l]<0 then lm:=0 else
lm:=1;==
standard('2',alfa[l])==
end levsyl;==
begin==
comment continued fraction;==
for i:=0 step 1 until n do==
begin standard('40',s); a[1,i]:=s;==
standard('40',s); a[2,i]:=s; b[1,i]:=0;==
end;==
a[1,n+1]:=a[2,n+1]:=0; k:=1;==
if a[2,0]=0 then goto a4:==
a[3,1]:=a[1,0]/a[2,0];==
for k:=2 step 1 until 2×n do==
begin s:=a[1,0]/a[2,0];==
for i:=0 step 1 until n do==
b[1,i]:=a[1,i+1]-s×a[2,i+1];==
if b[1,0]=0 then goto a4:==
a[3,k]:=a[2,0]/b[1,0];==
for i:=0 step 1 until n do==
begin a[1,i]:=a[2,i]; a[2,i]:=b[1,i];==
end==
end;==
for k:=1 step 1 until 2×n do==
standard('2',a[3,k]);==
goto a5;==
a4:i:=111; standard('2',i,k); goto a3;==
```

```
a5: end continued fraction;==
begin==
comment state-matrix;==
for i:=1 step 1 until n do==
begin b[i,i]: =0;==
for j:=1 step 1 until i do==
b[i,i]:=b[i,i]+a[3,2×j−1]==
end;==
for i:=2 step 1 until n do==
for j:=1 step 1 until i−1 do==
b[i,j]:=b[j,i]:=b[j,j];==
standard('2',n);==
for i:=1 step 1 until n do==
for i:=1 step 1 until n
do==
begin k:=2×j; b[i,j]:=−a[3,k]×b[i,j];==
standard('2',b[i,j])==
end==
end state-matrix;==
begin==
comment positive definiteness;==
for f;=n step −1 until 1 do==
standard('2',f);==
begin==
fi:=f×(f+1)/2; fj:=fi+1;==
supmtx(f,b,a);==
gaussj(fi,a);
arrnge(f,p);==
standard('2',f);==
j:=1;==
for k:=f step −1 until 1 do==
begin levsyl(k,lm);==
if lm=0 then j:=0;==
end;==
standard('2',j);==
end;==
end positive definiteness;==
a3: end;==
end;==
```

Tivadar Kovács, Lorant Lakatos
1111 Budapest XI., Stoczek u. 17/b
Vengrija