Kybernetika

Jan Ježek

Algoritmus pro výpočet diskrétního přenosu lineární dynamické soustavy

Kybernetika, Vol. 4 (1968), No. 3, (246)--259

Persistent URL: http://dml.cz/dmlcz/124631

KYBERNETIKA ČÍSLO 3, ROČNÍK 4/1968

Algoritmus pro výpočet diskrétního přenosu lineární dynamické soustavy

JAN JEŽEK

Popisuje se numerická metoda, která pro danou lineární dynamickou soustavu, danou periodu vzorkování a daný mezičas ε vypočítává koeficienty diskrétního přenosu, tj. přenosu v transformaci z, ε. Jako výchozí popis soustavy slouží přenos ve spojité Laplaceově transformaci. Algoritmu lze použít při návrhu číslicové regulace nebo při číslicovém modelování.

1. ÚVOD

Při syntéze regulačních obvodů s číslicovým počítačem, zastávajícím funkci regulátoru ve zpětnovazební smyčce, se vychází z matematického modelu regulované soustavy. Pro tyto účely je nejvhodnějším způsobem popisu přenosová funkce v transformaci z, ε:

$$G(z, \varepsilon) = \frac{p_0(\varepsilon) + p_1(\varepsilon)\, z^{-1} + \ldots + p_r(\varepsilon)\, z^{-r}}{1 + q_1 z^{-1} + \ldots + q_r z^{-r}}, \tag{1}$$

viz [1], [2], [3]. Tuto funkci lze získat buď statistickým zpracováním vstupního a výstupního signálu soustavy (identifikací), nebo matematickým rozborem dynamických vlastností soustavy. Ten obvykle vede na diferenciální rovnice a přenosy ve spojité Laplaceově transformaci. Tyto přenosy je potom zapotřebí převést na diskrétní tvar.

Pro racionální přenosy

$$F(p) = \frac{b_1 p^{r-1} + \ldots + b_{r-1} p + b_r}{p^r + a_1 p^{r-1} + \ldots + a_{r-1} p + a_r} \tag{2}$$

je běžně známou metodou převodu rozklad v parciální zlomky, převod těchto zlomků podle slovníku transformace z, ε a opětné sloučení. Tento způsob poskytuje pro

výsledky výrazy v uzavřeném tvaru, ale není příliš vhodný pro strojové výpočty. Je tomu tak proto, že je zapotřebí řešit algebraickou rovnici a rozlišovat více případů podle násobnosti kořenů a podle toho, jsou-li reálné či komplexní sdružené. Při větším počtu pólů se stává složitost výpočtu, hlavně při $\varepsilon \neq 0$, téměř nezvládnutelnou. Jednodušší a přehlednější metodu navrhl Weiss [1], [4]; ta odstraňuje rozklad na zlomky a mechanizuje výpočet čitatele. Je však zapotřebí znát kořeny jmenovatele a časový průběh odezvy v diskrétních bodech.

Níže uvedená metoda využívá skutečnosti, že je možné numericky řešit pochody v lineární soustavě, aniž by se počítaly kořeny charakteristické rovnice. Zapíšeme-li diferenciální rovnici soustavy, odpovídající přenosu (2), v maticovém tvaru, můžeme řešení vyjádřit ve tvaru exponenciály matice, viz např. [5]; tu lze potom vypočítat pomocí mocninné řady. Tak lze z daných počátečních podmínek napočítat řešení pro okamžiky T, $2T$, $3T$ atd. Z toho potom postupným vylučováním počátečních podmínek získáme pro vztah mezi vstupním a výstupním signálem rekurentní rovnici, která odpovídá diskrétnímu přenosu.

2. ODVOZENÍ METODY

Proveďme nyní podrobně všechny naznačené úvahy. Nejprve převedeme diferenciální rovnici r-tého řádu

$$y^{(r)} + a_1 y^{(r-1)} + \ldots + a_r y = b_1 u^{(r-1)} + \ldots + b_r u \,, \tag{3}$$

která odpovídá přenosové funkci (2), zavedením stavových proměnných na maticový tvar. Upravíme:

$$(\ldots((y' + a_1 y - b_1 u)' + a_2 y - b_2 u)' + \ldots + a_{r-1} y - b_{r-1} u)' + a_r y - b_r u = 0 \tag{4}$$

a zavedeme r stavových proměnných x_i:

$$\begin{aligned} x_0 &= y \,, \\ x_1 &= x_0' + a_1 x_0 - b_1 u \,, \\ &\vdots \\ x_i &= x_{i-1}' + a_i x_0 - b_i u \,, \\ &\vdots \\ x_{r-1} &= x_{r-2}' + a_{r-1} x_0 - b_{r-1} u \,. \end{aligned} \tag{5}$$

Rovnice pro x_1 až x_{r-1} spolu s rovnicí

$$0 = x_{r-1}' + a_r x_0 - b_r u \,, \tag{6}$$

která vznikla z dané diferenciální rovnice, dávají vektorovou diferenciální rovnici

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_r \end{bmatrix}' = \begin{bmatrix} -a_1 & 1 & & \\ -a_2 & & 1 & \\ & & & 1 \\ -a_r & & & \end{bmatrix} . \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{r-1} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_r \end{bmatrix} u \,, \tag{7}$$

čili

$$\boldsymbol{x}' = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u \,. \tag{8}$$

Převod na vektorový tvar je ovšem možno provést více různými způsoby. Zde uvedená cesta má tu výhodu, že výstupní veličina $y(t)$ je přímo jednou ze stavových veličin a že v rovnici vystupuje jen vstupní veličina $u(t)$, ne její derivace. Tento způsob je znám v programování analogových počítačů pod názvem metoda postupné integrace.

Obecné řešení rovnice (8) má tvar

$$\boldsymbol{x}(t) = \mathrm{e}^{\boldsymbol{A}t}\boldsymbol{x}(0) + \int_0^t \mathrm{e}^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}u(\tau)\,\mathrm{d}\tau \,. \tag{9}$$

Vyjádříme čas v diskrétním tvaru

$$\begin{aligned} t &= (n + \varepsilon)T, \qquad 0 \leqq \varepsilon, \sigma < 1 \,, \\ \tau &= (k + \sigma)T, \end{aligned} \tag{10}$$

kde n, k jsou celá čísla, T je perioda vzorkování. Vstupní signál předpokládáme ve tvaru stupňovité funkce, která má vždy konstantní hodnotu mezi dvěma okamžiky vzorkování, tedy

$$u(t) = u(n) \,. \tag{11}$$

Potom bude mít řešení tvar

$$\begin{aligned} \boldsymbol{x}(n, \varepsilon) = \mathrm{e}^{\boldsymbol{A}(n+\varepsilon)T}\boldsymbol{x}(0) + \sum_{k=0}^{n-1} \int_0^1 \mathrm{e}^{\boldsymbol{A}(n+\varepsilon-k-\sigma)T}\boldsymbol{B}u(k)\,T\,\mathrm{d}\sigma + \\ + \int_0^{\varepsilon} \mathrm{e}^{\boldsymbol{A}(\varepsilon-\sigma)T}\boldsymbol{B}u(n)\,T\,\mathrm{d}\sigma \,. \end{aligned} \tag{12}$$

Po úpravě a jednoduché substituci v integrálu dostaneme

$$\begin{aligned} \boldsymbol{x}(n, \varepsilon) = \mathrm{e}^{\boldsymbol{A}Tn}\,\mathrm{e}^{\boldsymbol{A}T\varepsilon}\boldsymbol{x}(0) + \sum_{k=0}^{n-1} \mathrm{e}^{\boldsymbol{A}T(n-k-1)}\,\mathrm{e}^{\boldsymbol{A}T\varepsilon}\left(\int_0^T \mathrm{e}^{\boldsymbol{A}\eta}\,\mathrm{d}\eta\right)\boldsymbol{B}u(k) + \\ + \left(\int_0^T \mathrm{e}^{\boldsymbol{A}\eta}\,\mathrm{d}\eta\right)\boldsymbol{B}u(n) \,. \end{aligned} \tag{13}$$

Pro výpočet naznačených integrálů lze použít vzorce

$$\int_0^T \mathrm{e}^{\boldsymbol{A}\eta}\,\mathrm{d}\eta = \boldsymbol{A}^{-1}\left(\mathrm{e}^{\boldsymbol{A}T} - \boldsymbol{1}\right). \tag{14}$$

Ten však selhává v případě, když $\boldsymbol{A}$ není regulární. To nastane tehdy, bude-li aspoň jeden kořen charakteristické rovnice roven nule (astatická soustava). Proto je vhodnější použít k výpočtu integrálu, podobně jako k výpočtu exponenciály, mocninné řady:

$$\mathrm{e}^{\boldsymbol{A}t} = \boldsymbol{1} + \frac{\boldsymbol{A}t}{1!} + \frac{(\boldsymbol{A}t)^2}{2!} + \frac{(\boldsymbol{A}t)^3}{3!} + \ldots . \tag{15}$$

Odtud se odvodí

$$\int_0^T \mathrm{e}^{\boldsymbol{A}\eta}\,\mathrm{d}\eta = T\left[\boldsymbol{1} + \frac{\boldsymbol{A}T}{2!} + \frac{(\boldsymbol{A}T)^2}{3!} + \frac{(\boldsymbol{A}T)^3}{4!} + \ldots\right]. \tag{16}$$

Při označení

$$\mathrm{e}^{\boldsymbol{A}T} = \boldsymbol{R}\,,\quad \mathrm{e}^{\boldsymbol{A}T\varepsilon} = \boldsymbol{R}^{\varepsilon}\,,\quad \left(\int_0^T \mathrm{e}^{\boldsymbol{A}\eta}\,\mathrm{d}\eta\right)\boldsymbol{B} = \boldsymbol{S}\,,\quad \left(\int_0^{\varepsilon T} \mathrm{e}^{\boldsymbol{A}\eta}\,\mathrm{d}\eta\right)\boldsymbol{B} = \boldsymbol{D} \tag{17}$$

dostáváme z (13) diskrétní stavovou rovnici:

$$\boldsymbol{x}(n,\varepsilon) = \boldsymbol{R}^n\boldsymbol{R}^{\varepsilon}\boldsymbol{x}(0) + \sum_{k=0}^{n-1} \boldsymbol{R}^{n-k-1}\boldsymbol{R}^{\varepsilon}\boldsymbol{S}u(k) + \boldsymbol{D}u(n)\,, \tag{18}$$

pro $\varepsilon = 0$:

$$\boldsymbol{x}(n) = \boldsymbol{R}^n\boldsymbol{x}(0) + \sum_{k=0}^{n-1} \boldsymbol{R}^{n-k-1}\boldsymbol{S}u(k)\,. \tag{19}$$

Ta umožnuje k danému stavovému vektoru v čase $n = 0$ a k dané posloupnosti hodnot vstupního signálu $u(k)$, $k = 0, 1, 2, \ldots$ postupně napočítávat nový stavový vektor pro všechny další okamžiky n. Zabývejme se nejprve případem $\varepsilon = 0$. Zapišme rovnici (19) indexově (pro tyto úvahy je vhodné číslování řádků a sloupců od 0 do $r - 1$):

$$x_i(n) = \sum_{j=0}^{r-1} [\boldsymbol{R}^n]_{ij}\, x_j(0) + \sum_{k=0}^{n-1}\sum_{j=0}^{r-1} [\boldsymbol{R}^{n-k-1}]_{ij}\, S_j u(k)\,. \tag{20}$$

Rovnice nás bude zajímat jen pro $i = 0$, protože udává přímo výstupní veličinu:

$$y(n) = x_0(n) = \sum_{j=0}^{r-1} [\boldsymbol{R}^n]_{0j}\, x_j(0) + \sum_{k=0}^{n-1}\sum_{j=0}^{r-1} [\boldsymbol{R}^{n-k-1}]_{0j}\, S_j u(k)\,. \tag{21}$$

Vezměme nyní $n = 0, 1, \ldots, r-1$ a považujme $y(n)$ za vektor, podobně $u(k)$. Pak lze chápat rovnici (21) maticově:

$$\boldsymbol{y} = \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{u}\,, \tag{22}$$

čili

$$y(n) = \sum_{k=0}^{n-1} T_{nj} x_j(0) + \sum_{k=0}^{n-1} W_{nk} u(k)\,, \tag{23}$$

kde

$$T_{nj} = [\boldsymbol{R}^n]_{0j}\,, \tag{24}$$

tj. n-tý řádek matice $\boldsymbol{T}$ je tvořen nultým řádkem matice $\boldsymbol{R}^n$; dále

$$W_{nk} = \begin{cases} \sum_{j=0}^{r-1} [\boldsymbol{R}^{n-k-1}]_{0j}\, S_j = \sum_{j=0}^{n-1} T_{n-k-1,j} S_j & \text{pro} \quad k < n\,, \\ 0 \quad \text{pro} \quad k \geqq n \end{cases} \tag{25}$$

je trojúhelníková matice, jejíž prvky závisí na jen rozdílu $n - k$ (vzdálenosti od diagonály) a mají tvar skalárního součinu $(n - k - 1)$-ho řádku matice $\boldsymbol{T}$ s vektorem $\boldsymbol{S}$:

$$\boldsymbol{W} = \begin{bmatrix} 0 & & \\ \boldsymbol{T}_0\boldsymbol{S} & 0 & \\ \boldsymbol{T}_1\boldsymbol{S} & \boldsymbol{T}_0\boldsymbol{S} & 0 \\ \vdots & & \ddots \end{bmatrix}. \tag{26}$$

Je-li matice $\boldsymbol{T}$ regulární, lze rovnici (22) řešit vzhledem k počátečnímu stavu:

$$\boldsymbol{x} = \boldsymbol{T}^{-1}\boldsymbol{y} + \boldsymbol{T}^{-1}\boldsymbol{W}\boldsymbol{u}\,, \tag{27}$$

$$x_j(0) = \sum_{l=0}^{r-1} \{[\boldsymbol{T}^{-1}]_{jl}\, y_l + [\boldsymbol{T}^{-1}\boldsymbol{W}]_{jl}\, u(l)\}\,. \tag{28}$$

Toto řešení dosadíme do rovnice (21) pro $n = r$:

$$y(r) = \sum_{j=0}^{r-1} T_{rj} x_j(0) + \sum_{k=0}^{r-1} W_{nk} u(k)\,. \tag{29}$$

Tím dostáváme vztah mezi $r + 1$ po sobě jdoucími hodnotami výstupní veličiny:

$$y(r) - \sum_{l=0}^{r-1} \{\sum_{j=0}^{r-1} T_{rj}[\boldsymbol{T}^{-1}]_{jl}\}\, y(l) = \sum_{l=0}^{r-1} \{\sum_{j=0}^{r-1} T_{rj}[\boldsymbol{T}^{-1}\boldsymbol{W}]_{jl} + W_{rl}\}\, u(l)\,, \tag{30}$$

který je tvaru

$$y(r) + \sum_{l=0}^{r-1} q_{r-l} y(l) = \sum_{l=0}^{r-1} p_{r-l} u(l)\,. \tag{31}$$

Protože jde o soustavu s konstantními parametry, nezmění se tento vztah volbou jiného počátku času:

$$y(n + r) + \sum_{l=0}^{r-1} q_{r-l} y(n + l) = \sum_{l=0}^{r-1} p_{r-l} u(n + l)\,. \tag{32}$$

Získali jsme rekurentní rovnici soustavy, která odpovídá diskrétnímu přenosu.

Přejděme nyní k případu $\varepsilon \neq 0$. Ve stavové rovnici (18) označíme

$$\boldsymbol{R}^{\varepsilon}\boldsymbol{x}(0) = \hat{\boldsymbol{x}}(0)\,, \quad \boldsymbol{R}^{\varepsilon}\boldsymbol{S} = \hat{\boldsymbol{S}}\,, \tag{33}$$

čímž dostáváme rovnici podobnou jako (23):

$$y(n) = \sum_{j=0}^{r-1} T_{nj} x_j(0) + \sum_{k=0}^{n} \hat{W}_{nk} u(k). \tag{34}$$

Jediný rozdíl je v horní mezi druhého součtu, což znamená, že matice $\hat{\boldsymbol{W}}$ má i diagonální prvek od nuly různý:

$$\hat{\boldsymbol{W}} = \begin{bmatrix} D_0 & & \\ \boldsymbol{T}_0\boldsymbol{S} & D_0 & \\ \boldsymbol{T}_1\boldsymbol{S} & \boldsymbol{T}_0\boldsymbol{S} & D_0 \\ \vdots & & \ddots \end{bmatrix}. \tag{35}$$

Ve výsledné rekurentní rovnici pak bude o jeden člen více:

$$y(n + r) + \sum_{l=0}^{r-1} q_{r-l} y(n + l) = \sum_{l=0}^{r} p_{r-l} u(n + l)\,. \tag{36}$$

3. SINGULÁRNÍ PŘÍPAD

Vyšetřme nyní, co se stane, nebude-li matice $\boldsymbol{T}$ regulární. Nechť je prvních k řádků $0, 1, \ldots, k - 1$ lineárně nezávislých a k-tý řádek jejich lineární kombinací ($k < r$, regulární případ nastává pro $k = r$). Pak každý řádek další než k-tý je také jejich lineární kombinací, tj. hodnost matice $\boldsymbol{T}$ je k. To lze jednoduše dokázat matematickou indukcí ze způsobu, jakým byla matice $\boldsymbol{T}$ utvořena. Napišme tuto lineární kombinaci takto:

$$y(k) = -\sum_{l=0}^{k-1} q_{k-l} y(l) + \sum_{l=0}^{k-1} p_{k-l} u(l)\,. \tag{37}$$

přitom $q_1, \ldots, q_k; p_1, \ldots, p_k$ jsou nějaká čísla. Vidíme tedy, že proces v soustavě splnuje rekurentní rovnici nižšího, k-tého řádu, která bude pro nás v singulárním případě výsledkem. Koeficienty p_j, q_j obdržíme stejným postupem jako v regulárním případě: postupným vylučováním stavových proměnných x_i ze soustavy rovnic. V regulárním případě tak vlastně provádíme inversi, naznačenou v (27).

Aby se zamezilo růstu numerických chyb, nevylučují se stavové proměnné při výpočtu podle pořadí indexů, ale v každém kroku se volí ta z nich, při níž je koeficient v absolutní hodnotě největší (Gaussova metoda s hlavními prvky). Tím se zabraňuje dělení příliš malými čísly (roste absolutní chyba), které by mělo za následek řádový vzrůst koeficientů v odčítaném řádku a sčítání čísel různých řádových velikostí (roste relativní chyba).

Zmíněný případ snížení řádu při převodu přenosu na diskrétní tvar nastává, platí-li pro některý pól spojitého přenosu $p_s = \eta_s + j\omega_s$ rovnost

$$\omega_s T = k \,.\, \pi\,, \quad k > 0 \text{ celé} \tag{38}$$

(stroboskopický efekt). Jde o skutečné snížení řádu (dimense stavového prostoru). Dvouparametrová část obecného řešení spojité soustavy

$$y(t) = K_1 \, \mathrm{e}^{\eta_s t} \cos \omega_s t + K_2 \, \mathrm{e}^{\eta_s t} \sin \omega_s t \tag{39}$$

přejde v diskrétní oblasti na jednoparametrovou část (např. $k = 1$):

$$\begin{aligned} y(n, \varepsilon) &= K_1 \, \mathrm{e}^{\eta_s (n+\varepsilon) T} \cos \omega_s (n + \varepsilon)\, T + K_2 \, \mathrm{e}^{\eta_s (n+\varepsilon) T} \sin \omega_s (n + \varepsilon)\, T = \\ &= \left(K_1 \, \mathrm{e}^{\eta_s \varepsilon T} \cos \pi\varepsilon + K_2 \, \mathrm{e}^{\eta_s \varepsilon T} \sin \pi\varepsilon\right) \left(-\mathrm{e}^{\eta_s T}\right)^n = K(\varepsilon)\, z_s^n \,. \end{aligned} \tag{40}$$

V komplexní oblasti se tento příklad projeví tak, že dva komplexně sdružené póly v rovině p přejdou na jeden pól v rovině z. Tento pól, zdánlivě dvojnásobný, je však jednoduchý, protože jeden kořenový činitel se krátí proti čitateli (identicky při každém ε).

4. ODHAD PŘESNOSTI VÝPOČTU EXPONENCIÁLY MATICE

Pro výpočet této maticové funkce byla použita mocninná řada

$$\mathrm{e}^{\boldsymbol{A}t} = \boldsymbol{1} + \boldsymbol{A}t + \frac{(\boldsymbol{A}t)^2}{2!} + \frac{(\boldsymbol{A}t)^3}{3!} + \ldots\,, \tag{41}$$

o které je známo, že v okolí bodu $t = 0$ velmi rychle konverguje. Při odhadu chyby použijeme normy čtvercové matice řádu r, viz [6]:

$$\|\boldsymbol{A}\| = r \max_{i,j} |A_{ij}| \tag{42}$$

a jejích vlastností

$$\|\boldsymbol{A} + \boldsymbol{B}\| \leqq \|\boldsymbol{A}\| + \|\boldsymbol{B}\|\,; \quad \|\boldsymbol{A}\boldsymbol{B}\| \leqq \|\boldsymbol{A}\| \,.\, \|\boldsymbol{B}\|\,. \tag{43}$$

Pro maticovou funkci $\boldsymbol{F}(t)$ skalární proměnné t platí Taylorova věta

$$\boldsymbol{F}(t) = \sum_{k=0}^{n-1} \frac{\boldsymbol{F}^{(k)}(0)}{k!}\, t^k + \frac{\boldsymbol{F}^{(n)}(\vartheta)}{n!}\, t^n\,, \quad 0 \leqq \vartheta \leqq t\,. \tag{44}$$

V našem případě

$$\mathrm{e}^{\boldsymbol{A}t} = \sum_{k=0}^{n-1} \frac{\boldsymbol{A}^k t^k}{k!} + \frac{\boldsymbol{A}^n \, \mathrm{e}^{\boldsymbol{A}\vartheta}}{n!}\, t^n\,. \tag{45}$$

Nám jde o výpočet exponenciály číselné matice, proto položíme $t = 1$:

$$e^{\boldsymbol{A}} = \sum_{k=0}^{n-1} \frac{\boldsymbol{A}^k}{k!} + \boldsymbol{R}_n , \quad \boldsymbol{R}_n = \frac{\boldsymbol{A}^n e^{\boldsymbol{A}\vartheta}}{n!} . \tag{46}$$

Odhadněme velikost zbytku za předpokladu $\|\boldsymbol{A}\| \leqq 1$:

$$\|\boldsymbol{R}_n\| \leqq \frac{\|\boldsymbol{A}\|^n \|e^{\boldsymbol{A}\vartheta}\|}{n!} \leqq \frac{\|e^{\boldsymbol{A}\vartheta}\|}{n!} . \tag{47}$$

Poměrná chyba přibližně

$$\varrho = \frac{\|\boldsymbol{R}_n\|}{\|e^{\boldsymbol{A}}\|} \doteq \frac{\|\boldsymbol{R}_n\|}{\|e^{\boldsymbol{A}\vartheta}\|} \leqq \frac{1}{n!} . \tag{48}$$

Tento odhad dává při rozumně proveditelném $n = 11$ přesnost asi 10^{-7}. Je-li však norma původní matice větší, řádově $\|\boldsymbol{A}\| = 10^h$, máme

$$\varrho \leqq \frac{(10^h)^n}{n!} , \tag{49}$$

což je podstatně horší. Použitím tabulek $\log n!$ zjistíme, že při $h = 2$ je i při 200 členech řady tento odhad pro chybu astronomicky velký: 10^{26}. S rostoucím n totiž ϱ nejprve exponenciálně vzrůstá a teprve později se uplatní pokles vlivem $1/n!$. Špatná konvergence řady při větším $\|\boldsymbol{A}\|$ byla také zjištěna experimenty na počítači.

Proto postupujeme jinak: danou matici normalizujeme podobně jako při výpočtech v pohyblivé čárce, tj. vyjádříme ve tvaru $\boldsymbol{A} = \boldsymbol{A}_m . 10^h$ kde $\|\boldsymbol{A}_m\| < 1$, $h \geqq 0$ je celé číslo. Potom

$$e^{\boldsymbol{A}} = e^{\boldsymbol{A}_m \cdot 10^h} = (e^{\boldsymbol{A}_m})^{10^h} , \tag{50}$$

což znamená, že získanou exponenciálu musíme h-krát po sobě umocnit na desátou. To lze celkem rozumně provést: k umocnění matice na desátou potřebujeme čtyři maticová násobení a dvojí přepis matice v paměti. Umocněním se ovšem původní chyba zvětší; poměrná chyba vzroste 10^h-krát:

$$\varrho = 10^h \frac{1}{n!} , \tag{51}$$

což je podstatně lepší než dříve.

Podobného obratu je zapotřebí použít i u mocninné řady pro funkci $\boldsymbol{F}(\boldsymbol{A}) = \int_0^T e^{\boldsymbol{A}\eta} \, d\eta$. Zde je však přechod od funkce normalizované matice na funkci matice původní trochu složitější; je zapotřebí h-krát po sobě počítat polynom 9. stupně.

5. ZHODNOCENÍ VÝSLEDKŮ

Popsaný algoritmus byl naprogramován v jazyce ACT-V pro počítač LGP-21 a v jazyce Algol pro počítač ELLIOTT 4100 a přezkoušen na těchto strojích. Byla vypočtena řada příkladů pro nejrůznější soustavy a pro velký rozsah period vzorkování. Takto získané koeficienty čitatele a jmenovatele souhlasily s výsledky získanými pomocí kořenů charakteristické rovnice na 7 až 8 desetinných míst. Přechodová charakteristika získaná z diskrétního přenosu dělením polynomů, souhlasila u téže soustavy pro různé periody vzorkování navzájem asi na 3 až 4 desetinná místa. U period vzorkování malých vůči časovým konstantám soustavy se projevovaly větší chyby, hlavně ke konci odezvy. To je způsobeno tím, že při $T \to 0$ se blíží všechny koeficienty čitatele k nule, zatímco koeficienty jmenovatele zůstávají na téže řádové výši, blíží se binomickým koeficientům. Tím se v rekurentní rovnici dostávají členy na pravé straně na úroveň zaokrouhlovacích chyb členů na levé straně a přesnost výpočtu odezvy z této rovnice klesá. Tento jev je však vlastní diskrétnímu přenosu a ne metodě, kterou byl vypočten.

6. ZÁPIS ALGORITMU

Pro přesný popis algoritmu a eventuální použití je níže uveden zápis ve tvaru procedury v jazyce Algol 60. Tu je možno přímo použít na počítači, doplní-li se příslušné příkazy vstupu a výstupu dat. Formální parametry *rad, cit, jmen* jsou vstupní a týkají se spojitého přenosu, *rad2, cit2, jmen2* jsou výstupní a týkají se diskrétního přenosu. Příslušné deklarace skutečných parametrů (označíme-li je týmiž jmény), musí být **array** *cit, jmen, cit2, jmen2* [0 : *rad*]. Parametr *chyba* je návěští, na které se předává řízení, objeví-li program chybu ve vstupních datech. Jestliže se zajímám jen o výsledky, je nejvhodnější použít hotových programů, které jsou ve výpočtové laboratoři ÚTIA-ČSAV.

```
procedure DISKR (rad, rad2, cit, jmen, cit2, jmen2, period, eps, chyba);
  value rad, period, eps; array cit, jmen, cit2, jmen2;
  integer rad, rad2; real period, eps; label chyba;
  begin integer i, j, k, jmax, e, mez; real smet, skalar, max, pom, diag;
    smet := jmen [0];
    if abs(cit [0]) > 10-6 ∨ eps < - 10-6 ∨ eps < 1.000001 ∨
        abs(smet) < 10-6 ∨ period < - 10-6 then go to chyba;
    if abs(smet-1) > 10-6 then
      for i := 0 step 1 until rad do
        begin cit [i] := cit [i]/smet;
          jmen [i] := jmen [i]/smet
        end;
    mez := rad - 1;
    begin array r, f [0 : mez, 0 : mez], t[0 : rad, 0 : mez],
                w[0 : rad, 0 : rad], c, s, es [0 : mez];
```

```
procedure expon (cas); value cas; real cas;
  begin
    begin array vektor [0 : mez], koef [1 : 10];
      for i := 0 step 1 until mez do
        vektor [i] := -jmen [i + 1] × cas;
      if rad > 1 then max := cas else max := 0;
      for i := 0 step 1 until mez do
        begin smet := abs(vektor [i]);
          if smet > max then max := smet
        end;
      if max < 0.1 then e := 0 else e := entier(0.4343 × ln(max)) + 2;
      smet := 10 ↑ (-e);
      if rad > 1 then skalar := cas × smet;
      for i := 0 step 1 until mez do
        vektor [i] := vektor [i] × smet;
      koef[1] := 1.0; koef[2] := 0.5; koef[3] := 0.16666667; koef[4] := 0.04166667;
      koef[5] := 0.00833333; koef[6] := 0.00138887; koef[7] := 0.00019841;
      koef[8] := 0.00002480; koef[9] := 0.00000275; koef[10] := 0.00000027;
      for i := 0 step 1 until mez do
        for j := 0 step 1 until mez do
          f[i, j] := 0;
      for k := 10 step -1 until 2 do
        begin
        for i := 0 step 1 until mez do
          f[i, i] := f[i, i] + koef[k];
         for i := 0 step 1 until mez do
          begin smet := 0;
              for j := 0 step 1 until mez do
                smet := smet + f[i, j] × vektor[j];
              for j := mez step -1 until 1 do
               f[i, j] := f[i, j - 1] × skalar;
             f[i, 0] := smet
          end
        end;
      for i := 0 step 1 until mez do
        f[i, i] := f[i, i] + 1;
      for i := 0 step 1 until mez do
        begin smet := 0;
        for j := 0 step 1 until mez do
          smet := smet + f[i, j] × vektor[j];
        for j := mez step -1 until 1 do
          r[i, j] := f[i, j - 1] × skalar;
          r[i, 0] := smet
        end;
      for i := 0 step 1 until mez do
        r[i, i] := r[i, i] + 1;
      for i := 0 step 1 until mez do
        for j := 0 step 1 until mez do
          f[i, j] := f[i, j] × cas
```

```
    end;
    begin array g, h, ch[0 : mez, 0 : mez], polyn[1 : 9]; integer q;
      procedure prep(u, v); array u, v;
        begin integer i, j;
          for i := 0 step 1 until mez do
            for j := 0 step 1 until mez do
              v[i, j] := u[i, j]
        end prep;
      procedure nas(u, v, w); array u, v, w;
        begin integer i, j, k; real sum;
          for i := 0 step 1 until mez do
            for j := 0 step 1 until mez do
              begin sum := 0;
                for k := 0 step 1 until mez do
                  sum := sum + u[i,k] × v[k,j];
                w[i, j] := sum
              end
        end nas;
      polyn[1] := 4.5; polyn[2] := 12.0; polyn[3] := 21.0;
      polyn[4] := 25.2; polyn[5] := 21.0; polyn[6] := 12.0;
      polyn[7] := 4.5; polyn[8] := 1.0; polyn[9] := 0.1;
      for k := e step −1 until 1 do
        begin
          for i := 0 step 1 until mez do
            for j := 0 step 1 until mez do
              begin
                g[i, j] := if i = j then r[i, j] −1 else r[i, j];
                h[i, j] := 0
              end;
          for q := 9 step −1 until 1 do
            begin
              for i := 0 step 1 until mez do
                for j := 0 step 1 until mez do
                  ch[i, j] := if i = j then h[i, j] + polyn[q] else h[i, j];
              nas(ch, g, h)
            end;
          for i := 0 step 1 until mez do
            h[i, i] := h[i, i] + 1;
          nas(h, f, g); prep(g, f);
          prep(r, g); nas(r, g, h); prep(h, r); nas(r, h, g); nas(r, g, h); nas(g, h, r)
        end
    end
  end expon;
expon (period);
comment konstrukce matice t;
t[0, 0] := 1;
for j := 1 step 1 until mez do
  t[0, j] := 0;
for i := 1 step 1 until rad do
```

```
    for j := 0 step 1 until mez do
      begin smet := 0;
        for k := 0 step 1 until mez do
          smet := smet + t[i - 1, k] × r[k, j];
        t[i, j] := smet
      end;
  comment konstrukce vektoru es a čísla diag;
  for i := 0 step 1 until mez do
    begin smet := 0;
      for j := 0 step 1 until mez do
        smet := smet + f[i, j] × cit[j ÷ 1];
      s[i] := es[i] := smet
    end;
  diag := 0;
  if eps > 10-6 then
    begin expon(period × eps);
      for i := 0 step 1 until mez do
        begin smet := 0;
          for j := 0 step 1 until mez do
            smet := smet ÷ r[i, j] × s([j];
          es[i] := smet
        end;
      for j := 0 step 1 until mez do
        diag := diag + f[0, j] × cit[j + 1]
    end;
  comment konstrukce matice w;
  for k := 0 step 1 until rad do
    w[k, k] := diag;
  for i := 1 step 1 until rad do
    begin smet := 0;
      for j := 0 step 1 until mez do
        smet := smet + t[i - 1, j] × es[j];
      for k := 0 step 1 until rad - i do
        w[i + k, k] := smet
    end;
  comment inverse;
  for i := 1 step 1 until rad do
    w[i, 0] := w[i, 0] - diag × t[i, 0];
  for k := 1 step 1 until mez do
    begin
      max := 0; jmax := k;
      for j := k step 1 until mez do
        begin
          smet := abs(t[k, j]);
          if smet > max then begin max := smet; jmax := j end
        end;
      if max < 10-6 then begin rad2 := k; go to vysl end;
      if jmax ≠ k then
        for i := k step 1 until rad do
```

```
        begin
          smet := t[i, k]; t[i, k] := t[i, jmax]; t[i, jmax] := smet
        end;
    smet := t[k, k];
    for j := 0 step 1 until mez do
      if j ≠ k then t[k, j] := t[k, j]/smet;
    for j := 0 step 1 until k do
      w[k, j] := w[k, j]/smet;
    for i := k + 1 step 1 until rad do
      begin pom := t[i, k];
        for j := 0 step 1 until mez do
          if j ≠ k then t[i, j] := t[i,j] − pom × t[k,j];
        t[i, k] := pom/smet;
        for j := 0 step 1 until k do
          w[i, j] := w[i, j] − pom × w[k, j]
      end
    end;
      rad2 := rad;
vysl : jmen2[0] := 1;
      for j := 1 step 1 until rad2 do
       jmen2[j] := −t[rad2, rad2 − j];
      for j := 0 step 1 until rad2 do
       cit2[j] := w[rad2, rad2 − j]
    end
  end DISKR;
```

(Došlo dne 23. října 1967.)

LITERATURA

[1] Strejc V. a kolektiv: Syntéza regulačních obvodů s číslicovým počítačem. NČSAV, Praha 1965.

[2] Цыпкин Я. З.: Теория линейных импульсных систем. Физматгиз, Москва 1963.

[3] Ragazzini J. R., Franklin G. F.: Sampled Data Control Systems. McGraw-Hill Book Company, New York 1958.

[4] Weiss J.: Výpočet obrazu v transformaci Z z průběhu originálu. Automatizace *VI* (1963), 8, 58—59.

[5] Zadeh L. A., Desoer C. A.: Linear System Theory — The State Space Approach. New York 1963.

[6] Faddějev D. K., Faddějevová V. N.: Numerické metody lineární algebry. Praha 1964.

Algorithm for Computing the Discrete Transfer Function of a Linear Dynamic System

JAN JEŽEK

In the present paper the numerical method is described, by means of which the discrete transfer function (i. e. the transfer function in z,ε transform for given linear dynamic system, given sampling period and given ε can be computed. As the original description of the system, the transfer function in terms of the continuous Laplace transform is used, being of the form of the rational fractional function. During the computation there is no need to know or to compute the roots of the denominator. The method is based on the solution of the system of linear differential equations in a matrix form. The paper contains the analysis of the accuracy of computations as well as the discussion of the singular case, when the resulting discrete transfer function is of lower order than the original continuous one. The description of the algorithm in Algol 60-language is given.

Ing. Jan Ježek, Ústav teorie informace a automatizace ČSAV, Vyšehradská 49, Praha 2.