

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 mezikas ε vypočítává koeficienty diskrétního přenosu, tj. přenosu v transformaci z, ε . Jako výchozí popis soustavy slouží přenos ve spojitě 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 nejhodnější způsobem popisu přenosová funkce v transformaci z, ε :

$$(1) \quad G(z, \varepsilon) = \frac{p_0(\varepsilon) + p_1(\varepsilon)z^{-1} + \dots + p_r(\varepsilon)z^{-r}}{1 + q_1z^{-1} + \dots + q_rz^{-r}},$$

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

Pro racionální přenosy

$$(2) \quad F(p) = \frac{b_1p^{r-1} + \dots + b_{r-1}p + b_r}{p^r + a_1p^{r-1} + \dots + a_{r-1}p + a_r}$$

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 nevládnutelnou. 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 diskretních bodech.

Níže uvedená metoda využívá skutečnosti, že je možné numericky řešit pochody v lineární soustavě, aniž by se počítaly kořeny charakteristické rovnice. Zapišme-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 diskretnímu přenosu.

2. ODVOZENÍ METODY

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

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

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

$$(4) \quad \begin{aligned} & (\dots((y' + a_1 y - b_1 u)' + a_2 y - b_2 u)' + \dots + a_{r-1} y - b_{r-1} u)' + \\ & + a_r y - b_r u = 0 \end{aligned}$$

a zavedeme r stavových proměnných x_i :

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

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

$$(6) \quad 0 = x_{r-1}' + a_r x_{r-1} - b_r u,$$

248 která vznikla z dané diferenciální rovnice, dávají vektorovou diferenciální rovnici

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

čili

$$(8) \quad \mathbf{x}' = \mathbf{Ax} + \mathbf{Bu}.$$

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

$$(9) \quad \mathbf{x}(t) = e^{\mathbf{A}t} \mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)} \mathbf{B}u(\tau) d\tau.$$

Vyjádříme čas v diskretním tvaru

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

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

$$(11) \quad u(t) = u(n).$$

Potom bude mít řešení tvar

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

Po úpravě a jednoduché substituci v integrálu dostaneme

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

Pro výpočet naznačených integrálů lze použít vzorce

$$(14) \quad \int_0^T e^{A\eta} d\eta = A^{-1} (e^{AT} - 1).$$

Ten však selhává v případě, když 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:

$$(15) \quad e^{At} = \mathbf{1} + \frac{At}{1!} + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \dots$$

Odtud se odvodí

$$(16) \quad \int_0^T e^{A\eta} d\eta = T \left[\mathbf{1} + \frac{AT}{2!} + \frac{(AT)^2}{3!} + \frac{(AT)^3}{4!} + \dots \right].$$

Při označení

$$(17) \quad e^{AT} = \mathbf{R}, \quad e^{AT\varepsilon} = \mathbf{R}^\varepsilon, \quad \left(\int_0^{eT} e^{A\eta} d\eta \right) \mathbf{B} = \mathbf{S}, \quad \left(\int_0^{eT} e^{A\eta} d\eta \right) \mathbf{B} = \mathbf{D}$$

dostáváme z (13) diskretní stavovou rovnici:

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

pro $\varepsilon = 0$:

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

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

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

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

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

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

$$(22) \quad \mathbf{y} = \mathbf{T} \mathbf{x} + \mathbf{W} \mathbf{u},$$

250 čili

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

kde

$$(24) \quad T_{nj} = [\mathbf{R}^n]_{0j},$$

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

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

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 \mathbf{T} s vektorem \mathbf{S} :

$$(26) \quad \mathbf{W} = \begin{bmatrix} 0 & & & & \\ \mathbf{T}_0 \mathbf{S} & 0 & & & \\ \mathbf{T}_1 \mathbf{S} & \mathbf{T}_0 \mathbf{S} & 0 & & \\ \vdots & & & \ddots & \\ & & & & \end{bmatrix}.$$

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

$$(27) \quad \mathbf{x} = \mathbf{T}^{-1} \mathbf{y} + \mathbf{T}^{-1} \mathbf{W} \mathbf{u},$$

$$(28) \quad x_j(0) = \sum_{i=0}^{r-1} \{ [\mathbf{T}^{-1}]_{ji} y_i + [\mathbf{T}^{-1} \mathbf{W}]_{ji} u(i) \}.$$

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

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

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

$$(30) \quad y(r) - \sum_{i=0}^{r-1} \{ \sum_{j=0}^{r-1} T_{rj} [\mathbf{T}^{-1}]_{ji} \} y(i) = \sum_{i=0}^{r-1} \{ \sum_{j=0}^{r-1} T_{rj} [\mathbf{T}^{-1} \mathbf{W}]_{ji} + W_{ri} \} u(i),$$

který je tvaru

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

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

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

Získali jsme rekurentní rovnici soustavy, která odpovídá diskrétnímu přenosu.

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

$$(33) \quad \mathbf{R}^t \mathbf{x}(0) = \hat{\mathbf{x}}(0), \quad \mathbf{R}^t \mathbf{S} = \hat{\mathbf{S}},$$

čímž dostáváme rovnici podobnou jako (23):

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

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

$$(35) \quad \hat{\mathbf{W}} = \begin{bmatrix} D_0 & & & & \\ \mathbf{T}_0 \mathbf{S} & D_0 & & & \\ \mathbf{T}_1 \mathbf{S} & \mathbf{T}_0 \mathbf{S} & D_0 & & \\ \vdots & & & \ddots & \\ & & & & D_0 \end{bmatrix}.$$

Ve výsledné rekurentní rovnici pak bude o jeden člen více:

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

3. SINGULÁRNÍ PŘÍPAD

Vyšetřme nyní, co se stane, nebude-li matice \mathbf{T} regulární. Nechť je prvních k řádků $0, 1, \dots, k-1$ 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. hodnota matice \mathbf{T} je k . To lze jednoduše dokázat matematickou indukcí ze způsobu, jakým byla matice \mathbf{T} utvořena. Napišme tuto lineární kombinaci takto:

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

přítom $q_1, \dots, q_k; p_1, \dots, p_k$ jsou nějaká čísla. Vidíme tedy, že proces v soustavě splňuje 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 inverzi, 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 zabráňuje 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

$$(38) \quad \omega_s T = k \cdot \pi, \quad k > 0 \text{ celé}$$

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

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

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

$$(40) \quad y(n, \varepsilon) = K_1 e^{\eta_s(n+\varepsilon)T} \cos \omega_s(n+\varepsilon)T + K_2 e^{\eta_s(n+\varepsilon)T} \sin \omega_s(n+\varepsilon)T = \\ = (K_1 e^{\eta_s \varepsilon T} \cos \pi \varepsilon + K_2 e^{\eta_s \varepsilon T} \sin \pi \varepsilon) (-e^{\eta_s T})^n = K(\varepsilon) z_s^n.$$

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

$$(41) \quad e^{At} = \mathbf{1} + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \dots,$$

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

$$(42) \quad \|\mathbf{A}\| = r \max_{i,j} |A_{ij}|$$

a jejich vlastností

$$(43) \quad \|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|; \quad \|\mathbf{AB}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|.$$

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

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

V našem případě

$$(45) \quad e^{At} = \sum_{k=0}^{n-1} \frac{\mathbf{A}^k t^k}{k!} + \frac{\mathbf{A}^n e^{A\vartheta}}{n!} t^n.$$

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

$$(46) \quad e^{\mathbf{A}} = \sum_{k=0}^{n-1} \frac{\mathbf{A}^k}{k!} + \mathbf{R}_n, \quad \mathbf{R}_n = \frac{\mathbf{A}^n e^{\mathbf{A}\theta}}{n!}.$$

Odhadněme velikost zbytku za předpokladu $\|\mathbf{A}\| \leq 1$:

$$(47) \quad \|\mathbf{R}_n\| \leq \frac{\|\mathbf{A}\|^n \|e^{\mathbf{A}\theta}\|}{n!} \leq \frac{\|e^{\mathbf{A}\theta}\|}{n!}.$$

Poměrná chyba přibližně

$$(48) \quad \varrho = \frac{\|\mathbf{R}_n\|}{\|e^{\mathbf{A}}\|} \doteq \frac{\|\mathbf{R}_n\|}{\|e^{\mathbf{A}\theta}\|} \leq \frac{1}{n!}.$$

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

$$(49) \quad \varrho \leq \frac{(10^h)^n}{n!},$$

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 $\|\mathbf{A}\|$ byla také zjištěna experimenty na počítači.

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

$$(50) \quad e^{\mathbf{A}} = e^{\mathbf{A}_m 10^h} = (e^{\mathbf{A}_m})^{10^h},$$

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 dvojnásobek matice v paměti. Umocněním se ovšem původní chyba zvětší; poměrná chyba vzroste 10^h -krát:

$$(51) \quad \varrho = 10^h \frac{1}{n!},$$

což je podstatně lepší než dříve.

Podobného obratu je zapotřebí použít i u mocninné řady pro funkci $\mathbf{F}(\mathbf{A}) = \int_0^1 e^{\mathbf{A}t} dt$. Zde je však přechod od funkce normalizované matice na funkci matice původní trochu složitější; je zapotřebí h -krát po sobě počítat polynom 9. stupně.

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 těžé 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 $T \rightarrow 0$ se blíží všechny koeficienty čitatele k nule, zatímco koeficienty jmenovatele zůstávají na těžé 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 nevhodnější použít hotových programů, které jsou ve výpočtově 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]) >  $10^{-6} \vee \textit{eps} < -10^{-6} \vee \textit{eps} < 1.000001 \vee$ 
    abs(smet) <  $10^{-6} \vee \textit{period} < -10^{-6}$  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];

```

```

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;
      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;
      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
    end
  end

```

```

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 vyst 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.