Vysoká škola strojní a textilní v Liberci

fakulta strojní

Martin Adamec

Generování náhodného signálu při daném tvaru spektrální hustoty

DIPLOMOVÁ PRÁCE

1994

obor 23-40-8

Automatizované systémy řízení výrobních procesů ve strojírenství

Katedra technické kybernetiky.

Generování náhodného signálu při daném tvaru spektrální hustoty

Jméno a příjmení autora: Martin ADAMEC

Vedoucí práce: Ing. Bedřich Janeček, CSc.

Konzultant: Doc.Ing. Vladimír Kracík, CSc.

Rozsah práce a příloh:

Počet stran: 62

Počet příloh: 4

Počet obrázků: 18

VYSOKÁ ŠKOLA STROJNÍ A TEXTILNÍ V LIBERCI

Fakulta strojní

Katedra technické kybernetiky Školní rok: 1993/94

ZADÁNÍ DIPLOMOVÉ PRÁCE

pro Martina ADAMCE

obor 23-40-8 Automatizované systémy řízení výrobních procesů ve strojírenství

Vedoucí katedry Vám ve smyslu zákona č. 172/1990 Sb. o vysokých školách určuje tuto diplomovou práci:

Název tématu:

Generování náhodného signálu při daném tvaru spektrální hustoty

Zásady pro vypracování:

1) Prostudujte teorii náhodných stacionárních procesů
2) Proveďte aproximaci spektrální hustoty racionálních funkcí a vypočtěte přenos tvarovacího filtru
3) Generujte pomocí vypočteného tvarovacího filtru náhodný proces a při použití korelační funkce vypočtěte jeho spektrální hustotu

Rozsah grafických prací:

Rozsah průvodní zprávy: přibližně 50 stran

Seznam odborné litaratury:

1) Hanuš,B. a kol.: Teorie automatického řízení II.
Skripta VŠST, Liberec 1985

2) Kadeřábek,J.; Kracík,V.: Úvod do počtu pravděpodobnosti.
Skripta VŠST, Liberec 1970

Vedoucí diplomové práce: Ing.Bedřich Janeček, CSc.

Konzultant: Doc.Ing.Vladimír Kracík, CSc.

Zadání diplomové práce: 29.10.1993

Termín odevzdání diplomové práce: 27.5.1994

L.S.

Vedoucí katedry

Doc.Ing.Vojtěch Kenopa, CSc.

Děkan

Prof.Ing.Jaroslav Exner, CSc.

V Liberci dne 29.10. 1993

Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury.

V Liberci dne 27. května 1994 .....

Poděkování

Úvodem své diplomové práce bych velice rád poděkoval svým konzultantům ing. Bedřichovi Janečkovi, CSc. a doc. ing. Vladimíru Kracíkovi, CSc. za odbornou pomoc, cenné rady a připomínky k mé diplomové práci.

Dále děkuji svým rodičům za podporu, kterou mi poskytovali po celou dobu studia.

Obsah

Úvod

Společenský vývoj na konci 20tého století klade stále vyšší požadavky na automobilovou dopravu. Člověk chce jezdit nejenom rychle a úsporně, ale i bezpečně a pohodlně. Těmto požadavkům je zcela podřízena výroba automobilů. A i když toto odvětví průmyslu v hospodářsky vyspělých zemích prožívá v současné době krizi, konstruktéři předních automobilových firem neustále navrhují technická vylepšení, hledají nové způsoby pohonu a druhy pohonných hmot, designéři vytvářejí elegantní tvary karoserií. Samozřejmě i zde se projevuje prudký rozvoj elektroniky a výpočetní techniky. Jestliže dříve byl automobil spíše mechanickou záležitostí, dnes dostávají hlavní slovo elektronické řídicí prvky. To vše se děje s cílem co nejvíce uspokojit spotřebitele. V zájmu konkurenceschopnosti na světových trzích se samozřejmě tomuto trendu musí přizpůsobit i náš výrobce automobilů ŠKODA Volksvagen Group Mladá Boleslav a dodavatelé jednotlivých komponentů.

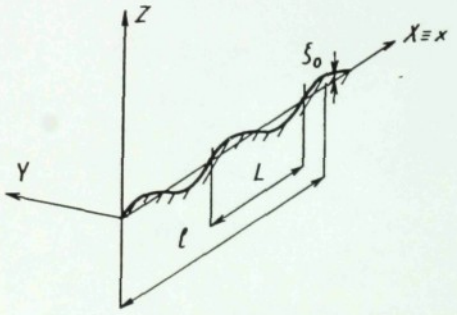
Jedním z dodavatelů je akciová společnost ATESO Jablonec nad Nisou, která již řadu let vyrábí pružicí jednotky a brzdy pro vozy značky ŠKODA. Společnost ATESO vyvíjí nový elektromagneticky řízený tlumič, který by měl zlepšit celkové pérování podvozku. Pro řízení tlumiče již existuje tzv. Karnoppův algoritmus. Tento algoritmus ve své diplomové práci ověřovala ing. Iva Sedláková. Chování Karnoppova algoritmu bylo posuzováno podle reakcí na několik

vstupních poruch. Ukázalo se však, že průběh a kvalitu řízení vstupující porucha podstatně ovlivňuje. Z toho vyplývá, že pro hodnocení kteréhokoliv řídicího algoritmu je nezbytné provést simulace v podmínkách co nejbližších realitě. Pro další fázi ověřování tohoto algoritmu ale i jiných (stavových) regulátorů ing. Sedláková navrhovala vytvoření programu, který by generoval povrch vozovky. A právě vytvořením tohoto programu se zabývá tato diplomová práce.

1. Seznámení s problematikou spektrálních hustot vozovek.

1.1 Nerovnosti vozovky a nebořivého terénu.

Vozidlo při jízdě přejíždí svými koly nerovnosti podkladu. Tyto nerovnosti mají určitý průběh výšky f vzhledem k základní referenční rovině (Z=0) jak ve směru jízdy vozidla X, tak i ve směru Y k němu kolmém.



Obr. 1. Harmonická nerovnost vozovky v ose X.

Předpokládáme-li, že ve směru jízdy jedné stopy vozidla je průběh výšek nerovnosti harmonický (obr. 1), přičemž amplituda je f_0 a délka jedné vlny nerovnosti je L, je průběh v ose X od určitého zvoleného počátku možno popsat v délkové doméně např. jako

$$f = f_0 \sin 2\pi x/L = f_0 \sin 2\pi \Phi x = f_0 \sin \Omega x. \qquad (1.1\text{-}1)$$

Z uvedého vzorce se dá snadno odvodit pojem tzv. délkové frekvence

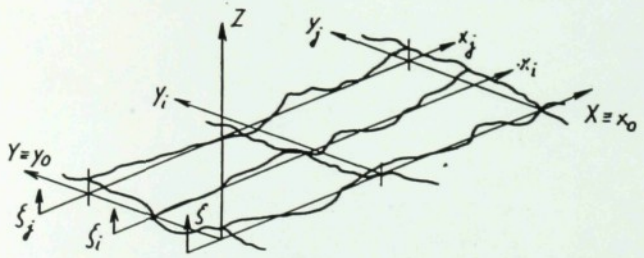
$$\Phi = 1/L \quad (1/m), \tag{1.1-2}$$

resp. délková úhlová frekvence

$$\Omega = 2\pi/L \quad (rad/m). \tag{1.1-3}$$

1.2 Nerovnosti vozovky jako stacionární ergodické funkce odlehlosti.

Výška nerovnosti vozovky ζ je funkcí jak souřadnice X (v ose jízdy vozidla), tak i v souřadnici Y, tj. v ose příčné ke směru jízdy vozidla (obr. 2). Přejíždí-li



Obr. 2. Znázornění nerovností vozovky v rovině (X, Y).

vozovku jednostopé vozidlo, stačí znát statistické vlastnosti závislosti $f(X)$ v jedné stopě. U dvoustopého vozidla není samozřejmě nutno znát statistický popis celého průběhu $f(X, Y)$, avšak pouze popis v obou stopách $f_P(X)$ a $f_L(X)$ a jejich vzájemné statistické vztahy.

Každý centrovaný stacionární ergodický náhodný proces v jedné stopě x_i je definován buď svou autokorelační funkcí $K_i(\tau)$ v závislosti na délkovém posuvu τ, resp. normovanou autokorelační funkcí $\mathring{K}(\tau) = K_i(\tau)/\sigma^2$, kde σ^2 je rozptyl procesu, nebo jednosněrnou spektrální hustotou $S_i(\Omega)$ v závislosti na délkové úhlové frekvenci Ω.

Mezi oběma těmito funkcemi platí známý vztah

$$S_i(\Omega) = 2\int_0^{\infty} K_i(\tau) e^{-i\Omega\tau}\, d\tau. \qquad (1.2\text{-}1)$$

Dva současně probíhající centrované ergodické náhodné procesy $f_i(x)$, $f_j(x)$ jsou pak ještě dále definovány buď vzájemnými korelačními funkceni $K_{f_i f_j}(\tau)$, $K_{f_j f_i}(\tau)$, resp. normovanými vzájemnými korelačními funkcemi

$$\mathring{K}_{f_i f_j}(\tau) = K_{f_i f_j}(\tau)/\sigma_i/\sigma_j, \qquad (1.2\text{-}2)$$

nebo vzájemnými jednostrannými spektrálními hustotami $S_{f_i f_j}(i\Omega)$, $S_{f_j f_i}(i\Omega)$.

Vzhledem k tomu, že vzájemné spektrální hustoty

$S_{fifj}(\Phi)$ a $S_{fjfi}(\Phi)$ jsou komplexně sdružené, je možno tyto údaje nahradit koherenční funkcí $\Gamma_{fifj}(\Phi)$, definovanou jako

$$\Gamma^2_{fifj}(\Phi) = \frac{S_{fifj}(\Phi)^2}{S_{fi}(\Phi).S_{fj}(\Phi)}. \qquad (1.2\text{-}3)$$

1.3 Statistický popis vozovky přejížděné jedním kolem.

Autokorelační funkce a spektrální hustoty rozdílných vozovek v jedné stopě byly měřeny již řadou autorů.

Z měření, uváděných v literatuře, uvádím údaje Parchilovského (AP 1968/8) a Brauna (Deutsche Kraftfahrforschung, Nr.186/1966).

Parchilovskij udává pro nerovnosti vozovek směrodatné odchylky a průběhy normovaných autokorelačních funkcí, uvedené v tabulce 1.

Pro proces s autokorelační funkcí

$$K(\ell) = \sigma^2 e^{-\alpha|\ell|}\cos 2\pi\Phi_0\ell \qquad (1.3\text{-}1)$$

je jednostranná spektrální hustota

$$S(\Omega) = 2\sigma^2\alpha\left[\frac{1}{(\alpha^2+4\pi^2(\Phi+\Phi_0)^2+(\alpha^2+4\pi^2(\Phi-\Phi_0)^2}\right], \qquad (1.3\text{-}2)$$

takže pomocí uvedené tabulky je možno přímo stanovit i spektrální hustoty nerovností jednotlivých vozovek.

Prakticky vždy se uvažuje Gaussovské rozložení hustoty

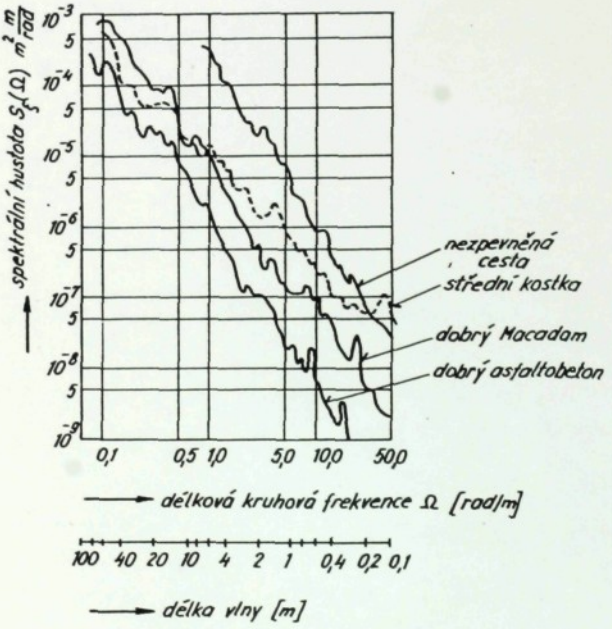
Typ vozovky	Směrodatná odchylka $\sigma(m)$	Normovaná autokorelační funkce $K_f(\tau)$, $\tau(m)$
asfaltová	$(0.38-0.54)10^{-2}$	$0.45e^{-2.5\|\tau\|}+0.55e^{-0.32\|\tau\|}$
dlážděná v dobrém stavu	$1.34 .10^{-2}$	$0.55e^{-4.0\|\tau\|}+0.3e^{-0.2\|\tau\|}+$ $+0.15e^{-0.15\|\tau\|}\cos 0.78$
dlážděná ve středním st.	$(1.45-1.5).10^{-2}$	$0.75e^{-1.5\|\tau\|}+0.25e^{-0.15\|\tau\|}\cos 0.75$
dlážděná ve špatném st.	$1.85 .10^{-2}$	$e^{-0.7\|\tau\|}$
dlážděná s vpadlinami	$(2.0-2.24).10^{-2}$	$0.75e^{-2.0\|\tau\|}+0.25e^{-0.05\|\tau\|}\cos 0.4$
polní cesta na kraji pole	$(0.55-0.82)10^{-2}$	$0.8e^{-0.65\|\tau\|}+0.2e^{-0.15\|\tau\|}\cos 2.0$
polní cesta mezi poli	$(1.3-2.18).10^{-2}$	$e^{-0.3\|\tau\|}$

Tabulka 1.

pravděpodobnosti výskytu výšek nerovností. Pak pravděpodobnost, že absolutní výška nerovnosti $|\tau|$ (vyvýšenina nebo jako vpadlina) přesáhne 3σ je ≈ 0.0027, nebo-li velmi malá. Proto prakticky považujeme $\pm 3\sigma$ za největší rozkmity nerovností vozovek.

Pokud je v autokorelační funkci nerovností obsažena periodická složka s délkovou frekvencí Φ_0, obsahují nerovnosti určitou výraznou složku s touto frekvencí. Pravidelné vlny tohoto typu mohou být způsobeny buď přímo technologií stavby vozovky, nebo jejím opotřebením.

Výsledky měření spektrálních hustot nerovností vozovek v SRN jsou ukázány na obrázku 3. Všeobecně se ukazuje, že v



Obr. 3. Spektrální hustoty čtyř druhů vozovek (Braun).

grafu log(S_f) log(Φ resp. Ω) je průběh S_f(Φ resp. Ω) přibližně přímkový, nebo-li spektrální hustotu je možno přibližně nahradit vztahem

$$S_f(\Omega) = S_f(\Omega_0).\Omega_0^{\,n}/\Omega^n, \qquad (1.3\text{-}3)$$

kde Ω_0 (rad/m) je zvolená referenční délková úhlová frekvence, $S_f(\Omega_0)$ spektrální hustota nerovnosti vozovky jí

odpovídající a n je konstanta. Čím je $S_f(\Omega_0)$ větší, tím je vozovka nerovnější, čím je n vyšší, tím nižší jsou spektrální složky vozovky s vyššími Ω (a tedy kratšími vlnami). V tabulce 2 jsou uvedeny konstanty pro typické vozovky.

Vozovka	stav	n	$S_f(\Omega_0)$ $m^2.m$, $\Omega_0=1$ rad/m
cementobetonová	velmi dobrý	2.29	$0.6 .10^{-6}$
	střední	1.97	$8.7 .10^{-6}$
	špatný	1.72	$56.3 .10^{-6}$
asfaltobetonová	velmi dobrý	2.20	$1.3 .10^{-6}$
	dobrý	2.18	$6.0 .10^{-6}$
	špatný	2.18	$22.3 .10^{-6}$
Macadan	dobrý	2.26	$8.9 .10^{-6}$
	špatný	2.15	$42.9 .10^{-6}$
	velmi špatný	2.15	$158.0 .10^{-6}$
dlážděná	dobrý	1.75	$13.7 .10^{-6}$
	špatný	1.81	$36.4 .10^{-6}$
	velmi špatný	1.81	$323.0 .10^{-6}$
nezpevněná	dobrý	2.25	$31.8 .10^{-6}$
	špatný	2.14	$602.0 .10^{-6}$
	velmi špatný	2.14	$1630.0 .10^{-6}$

Tabulka 2.

Směrodatná odchylka průběhů nerovností na uvedených

vozovkách se obdrží jako

$$\sigma_f = [S_f \int_{\Omega_1}^{\Omega_2} (\Omega) d\Omega]^{1/2} = [\frac{S_f(\Omega_0).\Omega^n}{(n-1)} . (\frac{1}{\Omega_1^{n-1}} - \frac{1}{\Omega_2^{n-1}})]^{1/2} . \qquad (1.3\text{-}4)$$

V citované práci jsou mezemi Ω_1=0.06, Ω_2=60 rad/m.

V [1] je pak dále rozveden statistický popis vozovky přejížděné dvěma koly v jedné stopě, statistický popis vozovky přejížděné dvěma koly jedné nápravy ve dvou stopách a statistický popis nerovností vozovky přejížděné čtyřmi koly (dvěma nápravami) ve dvou stopách. Z těchto popisů lze vycházet při modelování a následné simulaci povrchu vozovky. Simulaci takového signálu lze podle teorie stacionárních procesů (viz Dodatek) provést tímto způsobem.

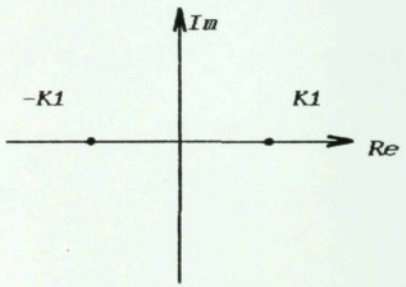
Mějme soustavu s přenosem $F(p)$. Bude-li do takovéto soustavy vstupovat náhodný signál se spektrální hustotou bílého šumu, můžeme při vhodném přenosu této soustavy na jejím výstupu získat náhodný signál se spektrální hustotou vozovky. Takovou soustavu potom můžeme nazvat filtrem. Máme-li generátor bílého šumu, stačí tedy pouze nalézt vhodný filtr, a to následujícím způsobem.

2. Nalezení filtru.

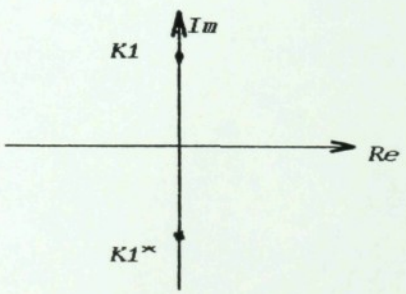
2.1 Pozitivně reálná funkce.

Podle kapitoly 4.4 spektrální hustota stacionárního procesu je racionální funkce proměnné Ω, ve které se vyskytují pouze sudé mocniny Ω. Jelikož spektrální hustota má pouze reálné koeficienty, vyskytují se kořeny jak čitatele, tak i jmenovatele v párech komplexně sdružených. Jelikož dále čitatel i jmenovatel je funkce sudá, vyskytují se kořeny v následujících možných sestavách:

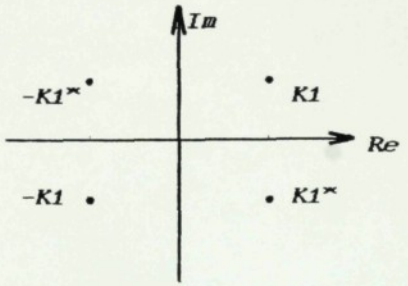
a) dvojice reálných kořenů



b) dvojice kořenů na imaginární ose



c) kořeny s reálnou a imaginární složkou



Jelikož spektrální hustota (lépe řečeno $S(\Omega)d\Omega$) udává střední čtvercovou amplitudu kmitu s frekvencí Ω, vyplývá, že $S(\Omega)$ musí být pro všechna Ω nezáporné. Pro potřebu rozkladu navíc požadujeme, aby $S(\Omega) > 0$ pro všechna Ω. Říkáme, že čitatel i jmenovatel spektrální hustoty musí být tzv. pozitivně reálná funkce, tj. v úvahu připadají varianty *b* a *c*. Píšeme-li $i\Omega = p$, potom pro spektrální hustotu zapsanou v této proměnné p pozitivní reálnost znamená, že $S(p)$ je reálná a pozitivní na imaginární ose. Např.

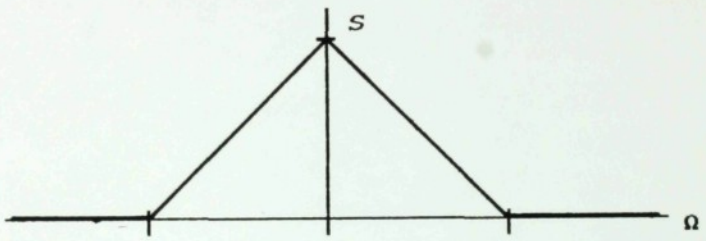
$$\frac{1}{(1-p^2)} \qquad (2.1\text{-}1)$$

je pozitivně reálná funkce.

2.2 Aproximace spektrální hustoty.

V první fázi pokusu o syntézu modelu pro generování

daného procesu byla řešena úloha aproximace spektrální hustoty podle obrázku 4 racionální funkcí $\tilde{S}(\Omega)$ se sudými mocninami. V případě, že při této aproximaci získáme



Obr. 4. Spektrální hustota.

pozitivně reálnou funkci, lze provést faktorizaci

$$\tilde{S}(\Omega) = \frac{\tilde{S}_c(\Omega)}{\tilde{S}_{jm}(\Omega)} = \frac{B(i\Omega).B(-i\Omega)}{A(i\Omega).A(-i\Omega)}, \qquad (2.2\text{-}1)$$

kde

$$F(i\Omega) = \frac{B(i\Omega)}{A(i\Omega)} \qquad (2.2\text{-}2)$$

představuje stabilní filtr (A,B jsou mnohočleny).

Užívaný algoritmus pro tuto faktorizaci spočívá na principu Newtonovy metody, která vychází z linearizace daného vztahu. Metoda spočívá na následujícím postupu.

Předpokládáme, že na j-tém kroku máme jakýsi j-tý odhad, pro který přibližně platí

$$B_j(i\Omega).B_j(-i\Omega) \approx \tilde{S}_c(\Omega) \qquad (2.2\text{-}3)$$

Rozdíl

$$B_{j+1}(i\Omega).B_{j+1}(-i\Omega) - \tilde{S}_c(\Omega) \qquad (2.2\text{-}4)$$

(který není roven nule) nahradíme lineární aproximací v bodě j

$$B_{j+1}(i\Omega).B_{j+1}(-i\Omega) - \tilde{S}_c(\Omega) \approx$$

$$\approx B_j(i\Omega).B_j(-i\Omega) + dB_j(i\Omega).B_j(-i\Omega) + B_j(i\Omega).dB_j(-i\Omega) - \tilde{S}_c(\Omega), \qquad (2.2\text{-}5)$$

ale

$$dB_j(i\Omega) = B_{j+1}(i\Omega) - B_j(i\Omega). \qquad (2.2\text{-}6)$$

Dosadíme-li (2.2-6) do (2.2-5) dostaneme

$$B_j(i\Omega).B_j(-i\Omega) + (B_{j+1}(i\Omega) - B_j(i\Omega)).B_j(-i\Omega) +$$

$$+ B_j(i\Omega).(B_{j+1}(-i\Omega) + B_j(-i\Omega)) - \tilde{S}_c(\Omega). \qquad (2.2\text{-}7)$$

Po úpravě dostáváme

$$B_{j+1}(i\Omega).B_j(-i\Omega) + B_j(i\Omega).B_{j+1}(-i\Omega) =$$

$$= \tilde{S}_c(\Omega) + B_j(i\Omega).B_j(-i\Omega). \qquad (2.2\text{-}8)$$

Tato rovnice představuje diofantickou rovnici pro neznámý polynom

$$B_{j+1}(i\Omega),$$

přičemž polynom $B_j(i\Omega)$ je známý. Řešení této diofantické rovnice se provede metodou porovnání odpovídajících si koeficientů. Je dokázáno, že tento postup konverguje při libovolně zvoleném počátečním stabilním mnohočlenu B_0. Podobný postup použijeme i pro jmenovatel $S_{jm}(\Omega)$.

Úloha aproximace spektrální hustoty racionální funkcí je řešena metodou nejmenších čtverců. Ukažme to na konkrétním příkladě.

Spektální hustota ve tvaru

$$S(\Omega) = 1-\Omega, \qquad (2.2\text{-}9)$$

je aproximována racionální funkcí

$$\tilde{S}(p) = \frac{b_2p^4+b_1p^2+b_0}{a_4p^8+a_3p^6+a_2p^4+a_1p^2+a_0}. \qquad (2.2\text{-}10)$$

Minimalizujeme $\Sigma\varepsilon_i^2$, kde ε_i je

$$\varepsilon_i = b_2\Omega_i^4+b_1\Omega_i^2+b_0 - (1-\Omega_i)(\Omega_i^8+a_3\Omega_i^6+a_2\Omega_i^4+a_1\Omega_i^2+a_0), \qquad (2.2\text{-}11)$$

nebo-li

$$\varepsilon_i = \mathbf{x}_i^T.\mathbf{p} - f_i, \qquad (2.2\text{-}12)$$

kde

$$\mathbf{p} = \begin{vmatrix} b_2 \\ b_1 \\ b_0 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{vmatrix} , \quad f_i = (1-\Omega_i)\Omega_i^8 \; a$$

$$\mathbf{x}^T = [\Omega_i^4, \Omega_i^2, 1, -\Omega_i^6(1-\Omega_i), -\Omega_i^4(1-\Omega_i), \ldots, -(1-\Omega_i)].$$

a tedy celkově

$$\boldsymbol{\varepsilon} = \begin{vmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ . \\ . \\ . \\ \varepsilon_i \end{vmatrix} = \mathbf{X}.\mathbf{p} - \mathbf{f}, \qquad (2.2\text{-}13)$$

kde

$$\mathbf{X} = \begin{vmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ . \\ . \\ . \\ \mathbf{x}_i^T \end{vmatrix} \quad \text{a} \quad \mathbf{f} = \begin{vmatrix} (1-\Omega_1)\Omega_i^8 \\ (1-\Omega_2)\Omega_i^8 \\ . \\ . \\ . \\ (1-\Omega_i)\Omega_i^8 \end{vmatrix}.$$

Podle teorie metody nejmenších čtverců vektor **p**, který minimalizuje

$$\Sigma \varepsilon_i^2 = \varepsilon^T . \varepsilon \qquad (2.2\text{-}14)$$

je dán

$$\mathbf{p} = [\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{f}. \qquad (2.2\text{-}15)$$

Tímto postupem byla aproximována funkce

$$\tilde{S}(p) = \frac{8{,}36.10^{-1}p^4 - 7{,}35.10^{-3}p^2 + 1{,}43.10^{-5}}{p^8 - 1{,}54.10^{-3}p^6 + 1{,}33.10^{-3}p^4 - 3{,}60.10^{-5}p^2 + 3{,}07.10^{-7}}. \qquad (2.2\text{-}16)$$

Ukázalo se ale, že čitatel není pozitivně reálná funkce (čitatel má reálný kořen). Z tohoto důvodu byl zvolen následující přístup.

Aproximaci spektrální hustoty budeme hledat v následujícím tvaru

$$\tilde{S}(\Omega) \approx F(p).F(-p), \qquad (2.2\text{-}17)$$

kde $p = i\Omega$ a $F(p)$ je přenos se stabilním čitatelem i jmenovatelem. Přenos $F(p)$ byl zvolen v následujícím tvaru

$$F(p) = \frac{(p-\Omega)(p-\Omega^*)(p-\alpha_4)\alpha_6}{(p-\omega)(p-\omega^*)(p-\tau)(p-\tau^*)(p-\alpha_5)}, \qquad (2.2\text{-}18)$$

neboli volili jsme přenos s jedním reálným kořenem v čitateli, s jedním reálným kořenem ve jmenovateli, s párem komplexních kořenů v čitateli a se dvěma páry komplexně sdružených kořenů ve jmenovateli.

Píšeme-li

$$\Omega = \alpha_1 + \beta_1 i, \quad \omega = \alpha_2 + \beta_2 i, \quad \tau = \alpha_3 + \beta_3 i,$$

a $p = i\Omega$, je

$$F(i\Omega).F(-i\Omega) = \tilde{S}(\Omega) =$$

$$= \frac{[\Omega_i{}^4+2(\alpha_1{}^2-\beta_1{}^2)\Omega_i{}^2+(\alpha_1{}^2+\beta_1{}^2)^2](\Omega_i{}^2+\alpha_4{}^2)\alpha_6{}^2}{[\Omega_i{}^4+2(\alpha_2{}^2-\beta_2{}^2)\Omega_i{}^2+(\alpha_2{}^2+\beta_2{}^2)^2]} .$$

$$\cdot \frac{1}{[\Omega_i{}^4+2(\alpha_3{}^2-\beta_3{}^2)\Omega_i{}^2+(\alpha_3{}^2+\beta_3{}^2)^2](\Omega_i{}^2+\alpha_5{}^2)} . \qquad (2.2\text{-}19)$$

Vidíme, že s je parametrický vektor

$$s = \begin{vmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \end{vmatrix} = \begin{vmatrix} s_1 \\ s_2 \\ . \\ . \\ . \\ . \\ . \\ . \\ s_n \end{vmatrix} .$$

Tento parametrický vektor hledáme z podmínky minima funkcionálu

$$\int_0^T [\ \tilde{S}(\Omega)-S(\Omega)\ ]^2 d\Omega , \qquad (2.2\text{-}20)$$

kde $S(\Omega)$ je zadaná spektrální hustota (2.2-9).

Pro výpočet parametrů tohoto funkcionálu byl zvolen následující postup.

2.3 Výpočet parametrů funkcionálu gradientní metodou.

Prvním krokem výpočtu parametrů funkcionálu je stanovení jeho velikosti. Pro numerický výpočet použijeme tzv. Simpsonovo pravidlo. Označme si

$$[\ \tilde{S}(\Omega)-S(\Omega)\ ]^2 = H(\Omega). \qquad (2.3\text{-}1)$$

Rozdělíme-li interval, ve kterém chceme počítat velikost integrálu (zde interval $\Omega \in \langle 0,T \rangle$) na n stejně velkých částí, kdy $n > 0$ a je sudé (např. $n = 600$), potom můžeme pro

$$\Omega_i = \frac{T}{n}.i \ , \quad kde\ i = 0,\ 1,\ \ldots\ ,\ n,$$

psát

$$F(s) = \int_0^T H(\Omega)d\Omega = C.[H(\Omega_0) + 4H(\Omega_1) + 2H(\Omega_2) + 4H(\Omega_3) +$$

$$+ \ \ldots\ + 4H(\Omega_{599}) + H(\Omega_{600})] \qquad (2.3\text{-}2)$$

a $F(s)$ nazvěme účelovou funkcí.

Protože pro náš případ není nutné vypočítávat konstantu C, jak bude vidět v následujícím popisu, byla proto její velikost zvolena $C = 0.01$.

Dalším krokem je výpočet gradientu **gradF(s)**. Píšeme-li

gradF(s) = gradF, potom

$$\mathbf{gradF}^{T} = \left(\frac{\partial F(s)}{\partial s_1}; \frac{\partial F(s)}{\partial s_2}; \ldots; \frac{\partial F(s)}{\partial s_n}\right). \qquad (2.3\text{-}3)$$

Gradient účelové funkce udává velikost změny a směr, ve kterém účelová funkce v daném bodu S nejrychleji roste. Vektor obrácený ke gradientu má směr největšího spádu (nejstrmějšího sestupu). Abychom nemuseli " ručně " vypočítávat parciální derivace podle všech n - složek, (v našem případě devět pariálních derivací), nahradíme jednotlivé derivace následujícím výpočtem

$$\frac{\partial F(s)}{\partial s_i} = gradF_i = [F(s_{i+}) - F(s_{i-})]/2/h_0 \qquad (2.3\text{-}4)$$

kde

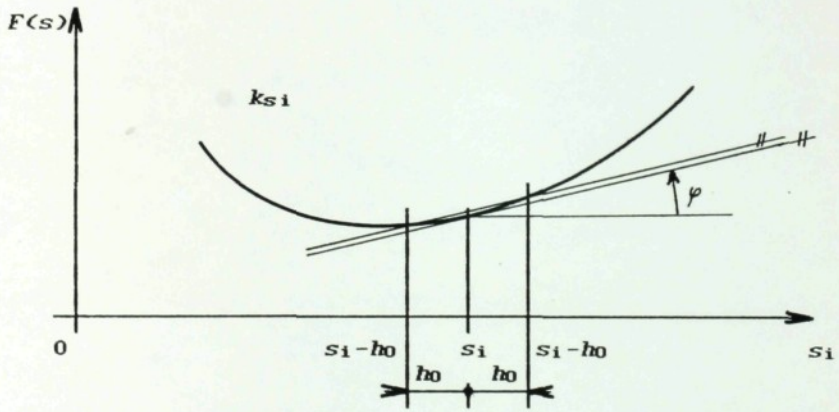
$$s_{i+} = \begin{vmatrix} s_1 \\ s_2 \\ . \\ . \\ . \\ s_{i-1} \\ s_i + h_0 \\ s_{i+1} \\ . \\ . \\ . \\ s_n \end{vmatrix}, \quad s_{i-} = \begin{vmatrix} s_1 \\ s_2 \\ . \\ . \\ . \\ s_{i-1} \\ s_i - h_0 \\ s_{i+1} \\ . \\ . \\ . \\ s_n \end{vmatrix},$$

pro i = 1 až n, h_0 volíme přibližně 10^{-4}. Gradient účelové

funkce potom můžeme psát jako

$$\mathbf{gradF} = \begin{vmatrix} [F(s_1+) - F(s_1-)/2/h_0] \\ [F(s_2+) - F(s_2-)/2/h_0] \\ \vdots \\ [F(s_n+) - F(s_n-)/2/h_0] \end{vmatrix}.$$

Tento postup můžeme graficky znázornit pomocí následujícího obrázku.



k_{si} ... křivka účelové funkce proměnné s_i

Obr. 5. Nahrazení parciálních derivací.

Parciální derivace účelové funkce podle složky s_i je tedy přibližně rovna směrnici tečny křivky účelové funkce proměnné s_i v bodě S_i. Protože hledáme pouze směrnici této tečny, není nutné znát přesnou hodnotu účelové funkce

(proto $C = 0.01$, viz výpočet velikosti funkcionálu (2.3-2)). Novou hodnotu složek s_i získáme pomocí vztahu

$$s_{ni} = s_i - h.gradFn_i, \tag{2.3-5}$$

kde

$$gradFn_i = \frac{gradF_i}{\|gradF\|}, \tag{2.3-6}$$

je tzv. normovaný gradient a

$$\|gradF\| = \sqrt{\sum_i gradF_i^2} \tag{2.3-7}$$

pro $i = 1$ až n. Celkově pak

$$s_n = s - h.gradFn. \tag{2.3-8}$$

Minimalizace funkcionálu (2.2-20) potom probíhá následujícím způsobem.

Vypočte se hodnota účelové funkce ve výchozím bodě $F_o = F(s_0)$. Pro navržený tvar přenosu filtru (2.2-18) byl výchozím bodem výpočtu zvolen vektor s_0 se složkami

$$s_0 = \begin{vmatrix} -1 \\ -1/6 \\ -0.5 \\ 0.5 \\ 0.25 \\ 1 \\ -2 \\ -1 \\ 1 \end{vmatrix}. \tag{2.3-9}$$

Následuje výpočet gradientu **gradF**. Po výpočtu normovaného gradientu **gradFn** proběhne výpočet nových složek vektoru s (koeficienty α_1 až α_6 a β_1 až β_3). Pro tyto přepočtené složky je stanovena nová hodnota účelové funkce. Jestliže je tato hodnota nižší než předchozí, potom se cyklicky opakuje výpočet gradientu účelové funkce pro nový vektor s, výpočet nového vektoru s atd. Jestliže je hodnota nové účelové funkce vyšší než předešlá, potom se zmenší krok h na $0.95h$ a program běží od výpočtu gradientu znovu. Výpočet končí, je-li velikost kroku h menší než 10^{-8} (nebo složky gradientu **gradF** menší než 10^{-3}). Tento postup je přehledně zobrazen vývojovým diagramem viz obrázek 6.

V průběhu výpočtu jsou pak ještě pro potřebu pozitivní reálnosti spektrální hustoty $S(\Omega)$ (viz kapitola 2.1) hlídány hodnoty jednotlivých složek vektoru s a to následujícím způsobem.

Pro koeficienty α_1 až α_5 platí

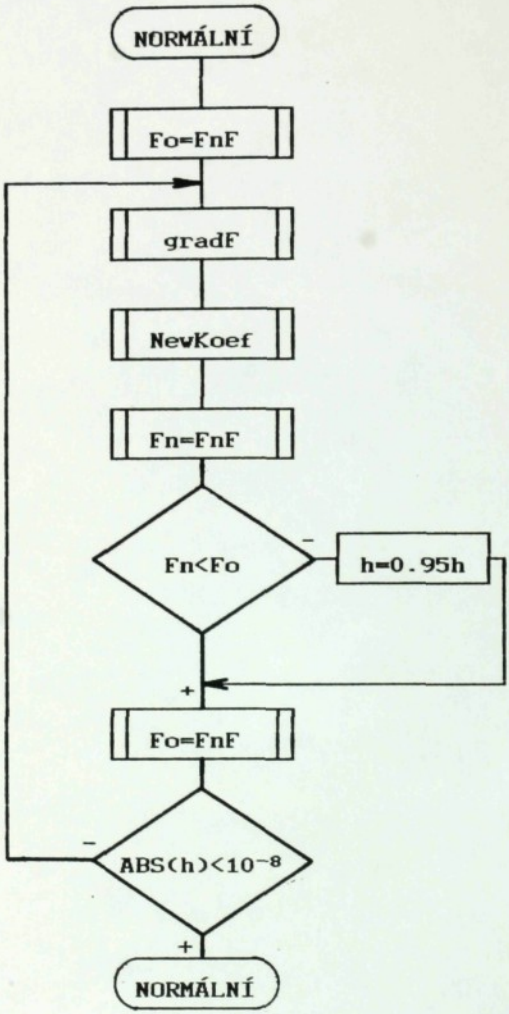
$$\alpha_i < -0.1.$$

Jestliže tuto hranici překročí, dosadí se $\alpha_i = -0.1$. Pro koeficienty β_1 až β_3 platí

$$abs(\beta_i) < 10.$$

Po překročení této hranice se dosadí $\beta_i = \pm 10$. A pro α_6 platí

$$\alpha_6 > 0.1.$$

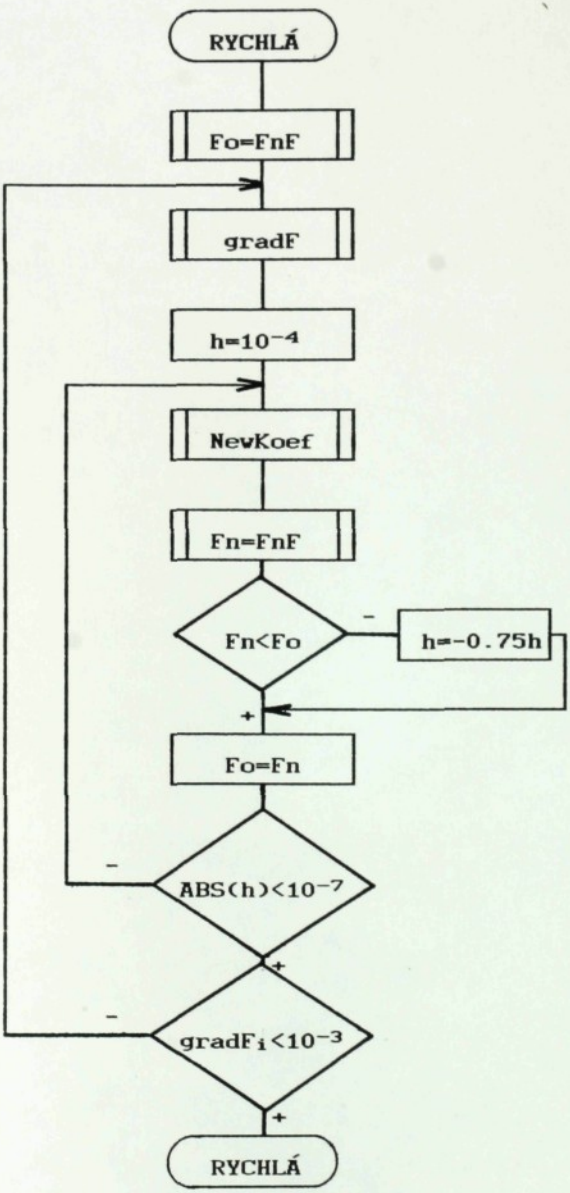


Obr. 6. Vývojový diagram pro "normální" gradientní metodu.

Překročí-li tuto hranici, pak $\alpha_6 = 0.1$.

Pro několik prvních cyklů výpočtu se hodnota účelové funkce snižuje přibližně o jeden řád na dva cykly, ale při přibližování se k minimu se rychlost změny rapidně snižuje. Pro velké n se výpočet gradientů i za použití výkonné výpočetní techniky velice prodlužuje. Pro získání slušných výsledků se čas výpočtu pohybuje kolem 8 hodin. I když se jedná o výpočet víceméně jednorázový, byl napsán ještě jeden program pro tzv. rychlou gradientní metodu. Ta spočívá v následujícím pustupu.

Prvním krokem jako v předešlé metodě je výpočet hodnoty účelové funkce ve výchozím bodě $F_0 = F(s_0)$. Výchozím bodem je opět vektor s_0 se shodnými složkami viz (2.3-9). Nyní následuje výpočet gradientu, výpočet normovaného gradientu a výpočet nových hodnot složek vektoru s. Pro tyto upravené složky je stanovena nová hodnota účelové funkce. Až potud se oba programy shodují. Jestliže je hodnota nové účelové funkce nižší než předchozí, potom se, oproti předešlé metodě, pouze přepočítávají koeficienty vektoru s se stále stejným gradientem. Jestliže je hodnota nové účelové funkce vyšší než předešlá, pak se zmenší krok h na $-0.75h$ a opět se přepočítávají pouze koeficienty vektoru s. Pokud se velikost kroku zmenší pod 10^{-7}, proběhne výpočet nového gradientu, krok h se dosadí $h = 10^{-4}$ a znovu se opakuje výpočet nových hodnot vektoru s. Výpočet končí, jsou-li velikosti všech složek gradientu **gradF** menší než 10^{-3}. Vývojový diagram pro tento postup je na obrázku 7.



Obr. 7. Vývojový diagram pro "rychlou" gradientní metodu.

I při této metodě jsou hlídány velikosti jednotlivých složek vektoru s, viz dříve.

I když pro několik prvních kroků výpočtu není zmenšení hodnoty účelové funkce tak výrazné, bylo přesto pro dosažení stejné hodnoty účelové funkce zapotřebí přibližně poloviny času oproti předešlému postupu.

Porovnání výsledků obou metod je provedeno v následující tabulce. Jsou zde uvedeny hodnoty koeficientů α_i a β_i a odpovídající velikosti účelových funkcí.

	normální	rychlá
α_1	-8.3356915567E-01	-4.2506397706E-01
α_2	-5.8784112852E-01	-1.0533244962E+00
α_3	-1.0338370401E+00	-7.6541164000E-01
β_1	1.0838876739E+00	1.7063728559E+00
β_2	4.2560314072E-01	4.3705636463E-01
β_3	9.6303953113E-01	7.3060288722E-01
α_4	-1.9073323740E+00	-1.9624787745E+00
α_5	-1.0690857765E+00	-1.1199257595E+00
α_6	3.0073745618E-01	2.5619876910E-01
F	1.4004988506E-02	8.1459791103E-03

Tabulka 3.

Přenos (2.2-18) můžeme po roznásobení přepsat na tvar

$$F(p) = \frac{b_3 p^3 + b_2 p^2 + b_1 p + b_0}{a_5 p^5 + a_4 p^4 + a_3 p^3 + a_2 p^2 + a_1 p + a_0}. \qquad (2.3\text{-}10)$$

V následující tabulce jsou pro porovnání uvedeny koeficienty a_i a b_j pro oba postupy.

	normální	rychlá
b_0	1.0725036827E+00	1.5590833334E+00
b_1	1.5187446905E+00	1.2237767991E+00
b_2	1.0751733692E+00	7.2245835926E-01
b_3	3.0079915129E-01	2.5668272929E-01
a_0	1.1241236206E+00	1.6360323266E+00
a_1	4.7250701153E+00	6.3416787611E+00
a_2	8.7326009233E+00	1.0687489913E+01
a_3	8.4216340929E+00	9.7270465058E+00
a_4	4.3125127303E+00	4.7590606856E+00
a_5	1.0000000000E+00	1.0000000000E+00

Tabulka 4.

Spektrální hustota (2.2-9) pak byla ještě dále aproximována ve tvaru (2.2-17), ale přenos $F(p)$ byl zvolen následující

$$F(p) = \frac{\beta^2}{\alpha^2 - p^2}. \qquad (2.3\text{-}11)$$

Stejným postupem byl získán přenos s koeficienty α a β viz následující tabulka (F je odpovídající hodnota účelové funkce).

	normální	rychlá
β	7.1485844159E-01	7.2081961000E-01
α	-7.2177854349E-01	-7.2837981078E-01
F	2.7526503567E-02	2.7639095401E-02

Tabulka 5.

Po roznásobení můžeme přenos (2.3-11) zapsat ve tvaru

$$F(p) = \frac{b_0}{a_2 p^2 + a_1 p + a_0} \qquad (2.3\text{-}12)$$

kde koeficienty a_i a b_j jsou v následující tabulce.

	normální	rychlá
b_0	5.1096328341E-01	5.1958092255E-01
a_0	5.2083229046E-01	5.3053714651E-01
a_1	1.4433742279E+00	1.4567596185E+00
a_2	1.0000000000E+00	1.0000000000E+00

Tabulka 6.

Průběhy aproximovaných spektrálních hustot pro přenosy

(2.2-18) resp. (2.3-10) a (2.3-11) resp. (2.3-12) vypočtené tzv. normální gradientní metodou jsou zobrazeny v grafech viz příloha.

Máme tedy nalezeny přenosy soustav resp. filtrů. Pomocí Laplaceovy transformace je můžeme přepsat do diferenciálních rovnic a ty pak řešit např. metodou Runge-Kutta, viz následující popis.

3. Generování náhodného signálu dané spektrální hustoty.

3.1 Řešení diferencilních rovnic metodou Runge-Kutta.

Vyjděme z diferenciální rovnice prvního řádu

$$y^{(1)} = f(l,y), \qquad (3.1\text{-}1)$$

a nechť její řešení existuje a je jednoznačné.
Celý délkový interval, ve kterém hledáme řešení, rozdělíme na řadu bodů - uzlů (obvykle ekvidistantních) a funkční hodnoty y v každém uzlu počítáme postupně z předtím určených hodnot v uzlech předcházejících

$$y_{i+1} = F(\Delta l, l_i, y_i, y_{i-1}, \ldots, y_{i-k+1}). \qquad (3.1\text{-}2)$$

Metoda se nazývá k uzlovou, jestliže se výpočet provádí z hodnot předcházejících k uzlů. Výpočet se zahajuje v uzlu, kde hodnoty jsou zadány počátečními podmínkami ($y_0 = y(l_0)$, $l = l_0$). Výpočet hodnoty funkce v uzlu $i+1$ se provádí podle vzorce

$$y_{i+1} = y_i + \Delta l.D(l_i, y_i, \Delta l) = y_i + \Delta l.D_i, \qquad (3.1\text{-}3)$$

U čtyřbodové metody je směrová funkce D_i ve tvaru

$$D_i = \frac{1}{6}.(K_1 + 2K_2 + 2K_3 + K_4) \qquad (3.1\text{-}4)$$

kde je

$$K_1 = f(l,y); \quad l = l_i; \quad y = y_i;$$

$$K_2 = f(l,y); \quad l = l_i + \tfrac{1}{2}\Delta l; \quad y = y_i + \tfrac{1}{2}K_1 \Delta l;$$

$$K_3 = f(l,y); \quad l = l_i + \tfrac{1}{2}\Delta l; \quad y = y_i + \tfrac{1}{2}K_2 \Delta l;$$

$$K_4 = f(l,y); \quad l = l_i + \Delta l; \quad y = y_i + K_3 \Delta l. \qquad (3.1\text{-}5)$$

Pro řešení soustavy n diferenciálních rovnic prvního řádu ve vektorovém zápisu

$$\mathbf{y}(l) = \mathbf{f}(l,y), \qquad (3.1\text{-}6)$$

lze čtyřbodovou metodu upravit následovně

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \Delta l.\mathbf{D}_i \qquad (3.1\text{-}7)$$

kde je

$\mathbf{y}$ vektor $[n,1]$ závisle proměnných,

$\mathbf{D}_i$ vektor $[n,1]$ směrové funkce

$$\mathbf{D}_i = \frac{1}{6}.(K_1 + 2K_2 + 2K_3 + K_4), \qquad (3.1\text{-}8)$$

$$K_1 = f(l_i, y_i)$$

$$K_2 = f(l_i + \tfrac{1}{2}\Delta l, y_i + \tfrac{1}{2}\Delta l.K_1)$$

$$K_3 = f(l_i + \tfrac{1}{2}\Delta l, y_i + \tfrac{1}{2}\Delta l.K_2)$$

$$K_4 = f(l_i + \Delta l, y_i + \Delta l.K_3). \qquad (3.1\text{-}9)$$

3.2 Úprava popisu systému pro řešení metodou Runge-Kutta.

Vycházíme z přenosu filtru

$$F(p) = \frac{B(p)}{A(p)}, \qquad (3.2\text{-}1)$$

kde

$$B(p) = b_n p^n + b_{n-1} p^{n-1} + \ldots + b_1 p + b_0 \quad \text{a}$$

$$A(p) = p^m + a_{m-1} p^{m-1} + \ldots + a_1 p + a_0, \qquad (3.2\text{-}2)$$

přičemž platí $m > n$.

Protože

$$F(p) = \frac{Y}{U}, \qquad (3.2\text{-}3)$$

můžeme psát

$$A(p).Y = B(p).U. \qquad (3.2\text{-}4)$$

Podle Laplaceovy transformace můžeme psát $y^{(n)} = p^n$ a dále, dosadíme-li vztahy (3.2-2) do (3.2-4), dostáváme diferenciální rovnici ve tvaru

$$y^{(m)} + a_{m-1} y^{(m-1)} + \ldots + a_1 y^{(1)} + a_0 y =$$

$$= b_n u^{(n)} + b_{n-1} u^{(n-1)} + \ldots + b_1 u^{(1)} + b_0 u, \qquad (3.2\text{-}5)$$

tedy diferenciální rovnici m - tého stupně s derivacemi na

pravé straně. Abychom se při numerickém řešení této rovnice vyhnuli derivování u, tedy derivacím vstupního signálu, upravíme vztah (3.2-3) na

$$F(p) = \frac{Y}{L} \cdot \frac{L}{U}, \qquad (3.2\text{-}6)$$

kde

$$\frac{Y}{L} = b_n p^n + b_{n-1} p^{n-1} + \ldots + b_1 p + b_0 \quad \text{a}$$

$$\frac{L}{U} = \frac{1}{p^m + a_{m-1} p^{m-1} + \ldots + a_1 p + a_0}. \qquad (3.2\text{-}7)$$

Po úpravě lze rovnice (3.2-7) přepsat na tvar

$$y = b_n l^{(n)} + b_{n-1} l^{(n-1)} + \ldots + b_1 l^{(1)} + b_0 l \quad \text{a}$$

$$l^{(m)} = u - a_{m-1} l^{(m-1)} - \ldots - a_1 l^{(1)} - a_0 l. \qquad (3.2\text{-}8)$$

Pro numerické řešení tohoto systému metodou Runge-Kutta musíme matematický model systému vhodnou volbou stavových veličin převést na zápis ve tvaru m diferenciálních rovnic prvního řádu. Proto byly zvoleny následující stavové veličiny

$$\begin{aligned} l_0 &= l, \\ l_{i+1} &= l_i^{(1)} \text{ pro } i = 0 \text{ až } (m-2), \\ l^{(1)}_{m-1} &= u - a_0 l_0 - a_1 l_1 - \ldots - a_{m-1} l_{m-1}. \end{aligned} \qquad (3.2\text{-}9)$$

Dynamický systém popsaný rovnicemi (3.2-8) potom můžeme popsat stavovými rovnicemi

$$\mathbf{l}^{(1)} = \mathbf{A}.\mathbf{l} + \mathbf{B}.u$$
$$y = \mathbf{C}.\mathbf{l} + \mathbf{D}.u \ , \qquad (3.2\text{-}10)$$

kde matice **A** je matice systému

$$\mathbf{A} = \begin{vmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{m-2} & -a_{m-1} \end{vmatrix},$$

matice **B** je matice buzení

$$\mathbf{B} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ . \\ . \\ . \\ 1 \end{vmatrix},$$

matice **C** je matice výstupu

$$\mathbf{C} = [b_0, b_1, \dots, b_n, 0, \dots, 0, 0]$$

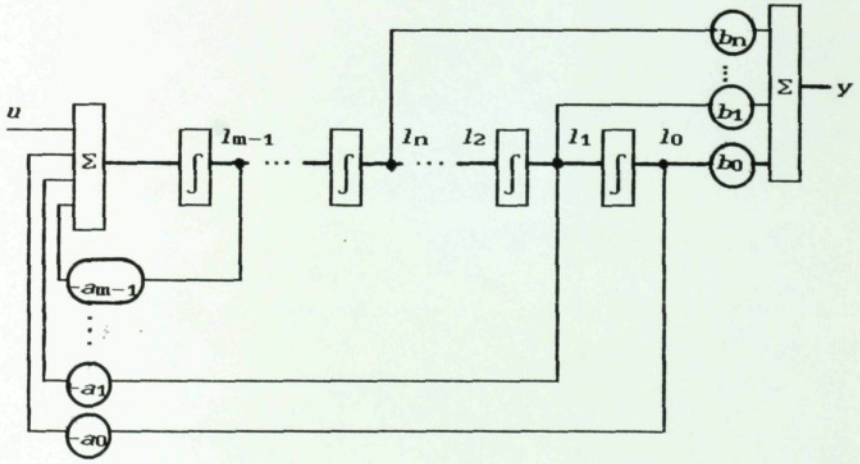
a matice **D** je matice převodu

$$\mathbf{D} = [0].$$

Takto popsaný systém lze již řešit metodou Runge-Kutta bez jakýchkoliv potíží. Simulační schema pro tento popis v podstatě odpovídá metodě snižování řádu derivace (viz obrázek 8.).

Podle výše uvedeného popisu byla pro řešení diferenciální rovnice (3.2-5) napsána procedura Generuj (viz Příloha).

Pro ověření správnosti této procedury byla napsána



Obr. 8. Simulační schema.

ještě jiná procedura Generuj1, ale pro systém popsaný následující rovnicí

$$\mathbf{y}^{(1)} = \mathbf{A}.\mathbf{y} + \mathbf{g}.u, \qquad (3.2\text{-}11)$$

kde matice **A** je

$$\mathbf{A} = \begin{vmatrix} -a_0 & -a_1 & \dots & -a_{m-3} & -a_{m-2} & -a_{m-1} \\ 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 \end{vmatrix},$$

a matice **g**

$$\mathbf{g} = \begin{vmatrix} g_0 \\ g_1 \\ . \\ . \\ . \\ g_{m-2} \\ g_{m-1} \end{vmatrix} = \begin{vmatrix} b_{m-1} \\ b_{m-2} - g_0 a_{m-1} \\ b_{m-3} - g_0 a_{m-2} - g_1 a_{m-1} \\ \\ \\ \\ b_0 - g_0 a_1 - \dots - g_{m-2} a_{m-1} \end{vmatrix}.$$

Metoda Runge-Kutta se potom upraví následovně:

$$y_{i+1} = y_i + D_i, \qquad (3.2\text{-}12)$$

kde je D_i vektor směrové funkce

$$D_i = \frac{1}{6}.(K_1 + 2K_2 + 2K_3 + K_4), \qquad (3.2\text{-}13)$$

$$K_1 = \Delta l.(\mathbf{A}.y + \mathbf{g}.u), \quad y = y_i,$$

$$K_2 = \Delta l.(\mathbf{A}.y + \mathbf{g}.u), \quad y = y_i + \tfrac{1}{2}.K_1,$$

$$K_3 = \Delta l.(\mathbf{A}.y + \mathbf{g}.u), \quad y = y_i + \tfrac{1}{2}.K_2,$$

$$K_4 = \Delta l.(\mathbf{A}.y + \mathbf{g}.u), \quad y = y_i + K_3. \qquad (3.2\text{-}14)$$

Porovnáním výsledků simulací těchto dvou procedur byla

zjištěna naprostá shodnost průběhů generovaného signálu a tedy potvrzena správnost obou simulačních procesů.

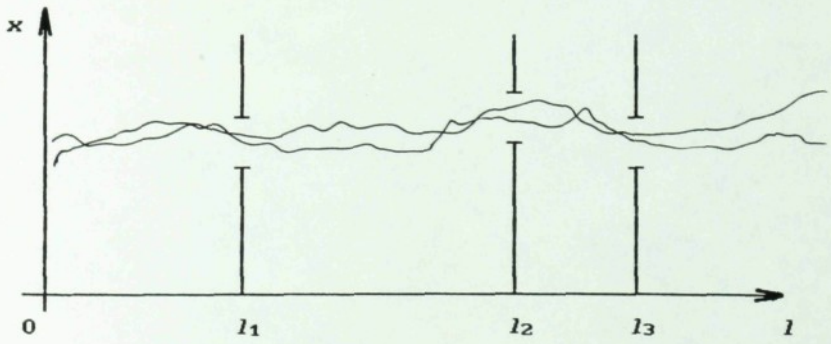
Dodatek

4. Statistická dynamika.

4.1 Pravděpodobnostní popis.

Náhodná funkce $x(\alpha,l)$ je funkce argumentu l vázaná na elementární jev α nějakého experimentu a tedy je vlastně funkcí dvou argumentů α, l. Argument l má význam délky. Většinou $x(\alpha,l)$ je číselná veličina, obecně ale lze uvažovat veličiny libovolné povahy. Jednotlivé možné hodnoty veličiny $x(\alpha,l)$ nazýváme potom stavy.

Náhodná reálná veličina je popsána distribuční funkcí $G(X)$, podobně dvojrozměrná veličina distribuční funkcí $G(X,Y)$. Náhodná funkce je popsána, je-li pro každou posloupnost bodů l_1, ..., l_n dáno rozdělení náhodného vektoru $(x(l_1), \ldots, x(l_n))$ např. pomocí distribuční funkce. Je-li např. takové rozdělení pro každou n - tici l_1,



Obr. 9.

.... l_n normální, mluvíme o normálním nebo Gaussovském procesu. Pravděpodobnostní popis definovaný uvedeným způsobem si můžeme názorně představit viz obrázek 9. Je-li dáno např. rozdělení vektoru $(x(l_1), x(l_2), x(l_3))$, je dána např. pravděpodobnost jevu, že funkce "projde" zvolenými otvory a obdobně pro jiné jevy tohoto typu.

4.2 Charakteristiky náhodných funkcí.

Podobně jako u "obyčejné" náhodné veličiny podávají charakteristiky přehlednou informaci o povaze rozdělení pravděpodobností, tak i v případě náhodných funkcí hrají charakteristiky důležitou roli.

U náhodných funkcí užíváme zpravidla dvou základních charakteristik.

a) Střední hodnota $\bar{x}(l) = E\, x(l)$. (4.2-1)

b) Korelační funkce $K_x(l_1, l_2) = E\,(x(l_1).x(l_2))$. (4.2-2)

Význam $\bar{x}(l)$ je zřejmý. Označme

$$\mathring{x}(l) = x(l) - \bar{x}(l), \tag{4.2-3}$$

potom $E\,\mathring{x}(l) = 0$ pro všechna l. A tedy $K_{\mathring{x}}(l_1, l_2)$ představuje vlastně kovarianci veličin $x(l_1)$, $x(l_2)$, což je míra lineární závislosti veličin $x(l_1)$, $x(l_2)$. Speciálně je

$$K_x(l_1, l_1) = E\, x^2(l_1) = \bar{x}^2(l_1), \tag{4.2-4}$$

$$K_{\mathring{x}}(l_1, l_1) = \sigma^2(x(l_1)). \tag{4.2-5}$$

Obdobně pro dvě náhodné funkce $x(l)$ a $y(l)$ se zavádí tzv. vzájemná korelační funkce

$$K_{xy}(l_1, l_2) = E\, x(l_1).y(l_2). \qquad (4.2\text{-}6)$$

$K_{xy}(l_1, l_2)$ je potom kovariance veličin $x(l_1)$, $y(l_2)$. $K_x(l_1, l_2)$ se také nazývá autokorelační funkce.

Zmíněné charakteristiky je možno exaktně určit, je-li dáno rozdělení vektoru $(x(l_1), x(l_2))$ resp. $(x(l_1), y(l_2))$ pro každé l_1, l_2.

4.3 Stacionární procesy.

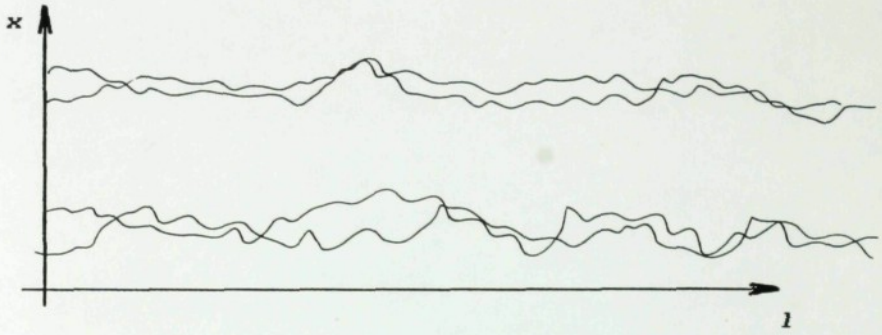
Důležitost teorie stacionárních procesů spočívá v tom, že tyto procesy představují vhodné matematické modely pro řadu praktických fyzikálních, biologických a jiných procesů.

Proces $x(l,\alpha)$ nazýváme (silně) stacionární, je-li pro každou konečnou posloupnost $l_1, \ldots, l_n$ rozdělení vektoru $(x(l_1 + L), x(l_2 + L), \ldots, x(l_n + L))$ nezávislé na posunu L.

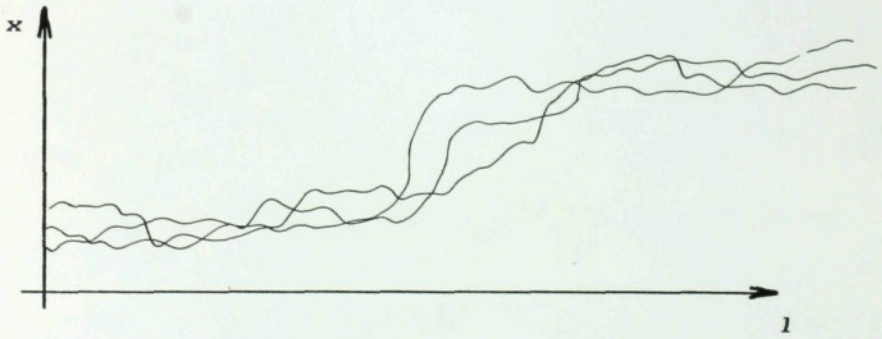
Představu o stacionárním procesu si můžeme utvořit podle obrázku 10 a 11. Na obrázku 10 je několik realizací stacionárního procesu a na obrázku 11 je několik realizací nestacionárního procesu. Charakteristiky stacionárního procesu mají specifické vlastnosti.

$\bar{x}(l) = E\, x(l) = \bar{x}$ střední hodnota je tedy konstanta (nezávislá na délce).

Dále $K_x(l_1, l_2) = E\ x(l_1).x(l_2)$ je podle definice



Obr. 10. Realizace stacionárního procesu.



Obr. 11. Realizace nestacionárního procesu.

stacionarity nezávislá na posunu L, tj.

$$K_x(l_1, l_2) = K_x(l_1+L, l_2+L) \qquad (4.3\text{-}1)$$

pro libovolné L a tedy pro $L = -l_2$ je

$$K_x(l_1, l_2) = K_x(l_1 - l_2, 0) = K_x(\tau), \qquad (4.3\text{-}2)$$

kde je $\tau = l_1 - l_2$. Korelační funkce stacionárního procesu je tedy závislá pouze na rozdílu $l_1 - l_2 = \tau$.

Korelační funkce $K_x(\tau)$ stacionárního procesu má tedy následující vlastnosti:

$$a)\ K_x(0) = E\,x^2(0) = E\,x^2(l) = \overline{x^2}, \qquad (4.3\text{-}3)$$

tj. $K_x(0)$ je střední kvadratická hodnota procesu $x(l)$, která přitom nezávisí na l. Dále

$$b)\ K_x(0) = \sigma^2(x(l)) = \sigma^2(x). \qquad (4.3\text{-}4)$$

$$c)\ K_x(\tau) = K_x(-\tau), \qquad (4.3\text{-}5)$$

tj. $K(\tau)$ je sudá funkce, což plyne z následujícího

$$K_x(\tau) = K_x(l_1, l_2) = K_x(l_2, l_1) = K_x(l_2 - l_1) = K_x(-\tau).$$

$d)$ Pro každé $\tau \neq 0$ platí

$$|K_x(\tau)| \leq K_x(0). \qquad (4.3\text{-}6)$$

Je totiž

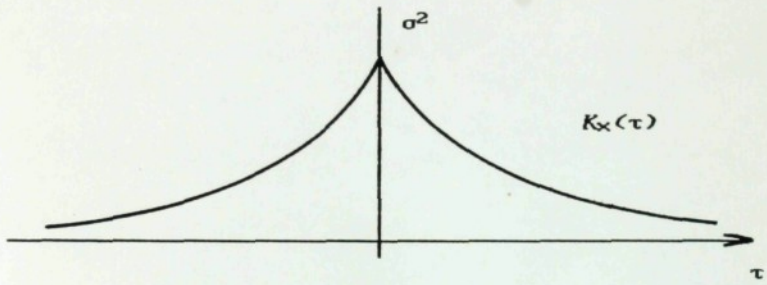
$$E\,(x(0) \pm x(\tau))^2 = E\,x^2(0) + E\,x^2(\tau) \pm 2E\,x(0).x(\tau) =$$

$$= 2K_x(0) \pm 2K(\tau) \geq 0.$$

Pro stacionární proces je tedy $\overline{x}(l)$ konstantní a korelační funkce je závislá pouze na $\tau = l_1 - l_2$. Obráceně,

jsou-li splněny tyto dvě vlastnosti, nemusí být proces ještě (silně) stacionární.

Uveďme dva příklady korelačních funkcí.



Obr. 12.

Tento typ korelační funkce

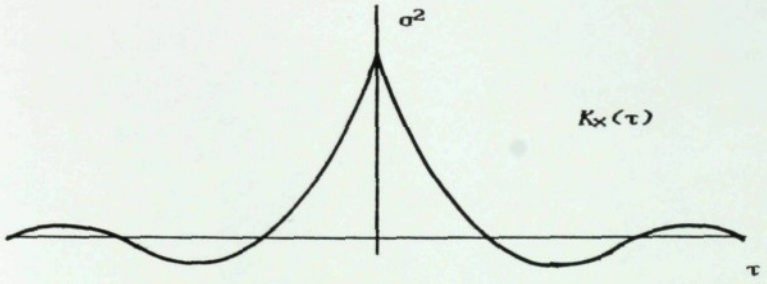
$$K_X(\tau) = \sigma^2 e^{-\alpha|\tau|} \qquad (4.3\text{-}7)$$

odpovídá mnoha procesům v technické praxi. Jeho charakteristiky, tj. normalita a závislost příštího vývoje pouze na výchozím stavu, jsou typické pro mnoho praktických procesů.

Jiný (obecnější) běžný typ korelační funkce stacionárního procesu je

$$K_X(\tau) = \sigma^2 e^{-\alpha|\tau|}.\cos \beta\tau \qquad (4.3\text{-}8)$$

(viz obrázek 13).



Obr. 13.

Ergodické stacionární procesy.

U většiny praktických stacionárních procesů lze pozorovat tzv. ergodicitu, což je vlastnost zjednodušující zkoumání takového procesu. Stacionární proces je ergodický, platí-li

$$E\,x(l) = \bar{x} = \lim \frac{1}{2L} \int_{-T}^{T} x_1(l).dl, \qquad (4.3\text{-}9)$$

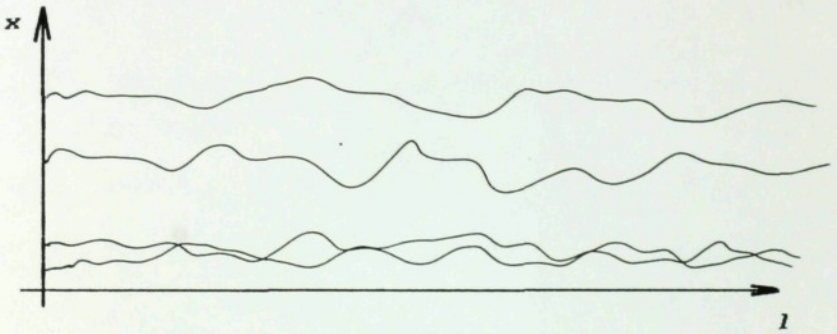
$$K(\tau) = \lim \frac{1}{2L} \int_{-T}^{T} x_1(l).x_1(l+\tau).dl, \qquad (4.3\text{-}10)$$

kde $x_1(l)$ je libovolná realizace procesu.

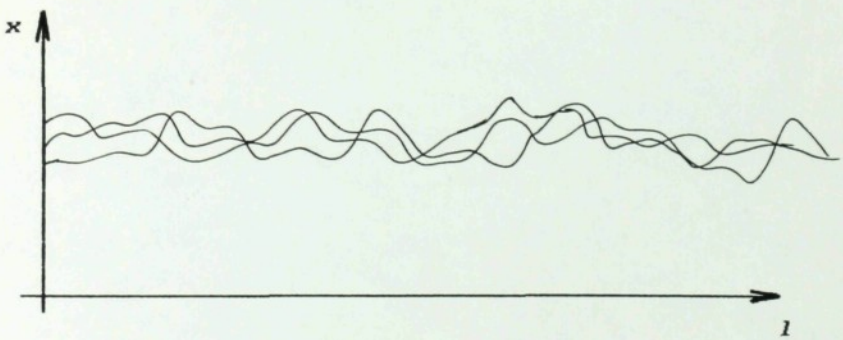
Stacionární proces je tedy ergodický, lze-li charakteristiky stanovit z jediné realizace podle uvedených vzorců, které představují vlastně aritmetické průměry na jedné realizaci. Lze ukázat, že Gaussovský proces s korelační funkcí zmíněného exponenciálního typu je

ergodický. Ukáže se totiž, že rozptyly integrálů, pomocí nichž jsou $E\ x(l)$ a $K_x(L)$ vyjádřeny, konvergují k nule pro $L \to \infty$.

Pro názornost uveďme grafy několika relizací procesu stacionárního neergodického (obrázek 14) a procesu stacionárního ergodického (obrázek 15).



Obr. 14. Realizace stacionárního neergodického procesu.



Obr. 15. Realizace stacionárního ergodického procesu.

Spektrální rozklad stacionárního procesu.

Při zkoumání těchto otázek je formálně výhodné uvažovat komplexní náhodné veličiny (komplexní náhodné funkce). Komplexní náhodná veličina $z(\alpha)$ je náhodná veličina nabývající komplexních hodnot, tj.

$$z(\alpha) = x(\alpha) + iy(\alpha), \qquad (4.3\text{-}11)$$

kde $x(\alpha)$, $y(\alpha)$ jsou reálné náhodné veličiny. Náhodná veličina $z(\alpha)$ je tedy popsána rozdělením vektoru $(x(\alpha), y(\alpha))$. Užíváme dále charakteristik

$$\bar{z} = E\, z = E\, x + iE\, y, \qquad (4.3\text{-}12)$$

$$|z|^2 = E\, z.z^{*} = E\,(x^2 + y^2) = \bar{x^2} + \bar{y^2}, \qquad (4.3\text{-}13)$$

tj. střední čtvercová vzdálenost od počátku (* značí komplexně sdružené číslo).

$$\sigma^2(z) = E\,(z-\bar{z})(z-\bar{z})^{*} = E\, z.z^{*} - \bar{z}.\bar{z}^{*}, \qquad (4.3\text{-}14)$$

tj. střední čtvercová vzdálenost od střední hodnoty.

Kovariance dvou komplexních veličin z, u je $E\,(z-\bar{z})(u-\bar{u})^{*}$.

Podobně uvažujeme komplexní náhodnou funkci

$$z(l) = x(l) + iy(l). \qquad (4.3\text{-}15)$$

Definujeme

$$E\, z(t) = E\, x(t) + iE\, y(t) = \bar{x}(t) + i\bar{y}(t), \qquad (4.3\text{-}16)$$

$$K_z(t_1, t_2) = E\, z(t_1).z^*(t_2). \qquad (4.3\text{-}17)$$

Proces $z(t)$ je stacionární, je-li rozdělení vektoru

$$[x(t_1+L), y(t_1+L),\ x(t_2+L), y(t_2+L),\ \ldots,\ x(t_n+L), y(t_n+L)] \qquad (4.3\text{-}18)$$

nezávislé na L.

Korelační funkce potom bude

$$K_z(\tau) = E\, z(\tau).z^*(0), \qquad (4.3\text{-}19)$$

$K_z(\tau)$ je ovšem komplexní funkce (nabývá komplexních hodnot). Uvažujme obecný případ

$$z(t) = \delta_1 e^{i\Omega_1 t} + \delta_2 e^{i\Omega_2 t} + \ldots + \delta_n e^{i\Omega_n t}, \qquad (4.3\text{-}20)$$

kde $\Omega_1 < \Omega_2 < \ldots < \Omega_n$ jsou reálná čísla a $\delta_1, \ldots, \delta_n$ jsou náhodné komplexní veličiny, kde $E\, \delta_i = 0$ pro $i = 1, \ldots, n$, $E\, \delta_i \delta_j^* = 0$ pro $i \neq j$, $E\, \delta_i \delta_i^* = b_i$. Od součtového tvaru (4.3-20) můžeme přejít k integrálu. Píšeme

$$\delta_1 + \delta_2 + \ldots + \delta_k = Z(\Omega_k), \qquad (4.3\text{-}21)$$

$$\delta_k = \Delta Z(\Omega_k) = Z(\Omega_k) - Z(\Omega_{k-1}), \qquad (4.3\text{-}22)$$

takže

$$z(l) = \sum_j e^{i\Omega_j l}\, Z(\Omega_j), \qquad (4.3\text{-}23)$$

podobně píšme

$$H(\Omega_k) = b_1 + \ldots + b_k, \qquad (4.3\text{-}24)$$

takže

$$K_z(\tau) = \sum_1^n e^{i\Omega_j \tau} \Delta H(\Omega_j). \qquad (4.3\text{-}25)$$

Je-li dále $\ldots, \Omega_{-2}, \Omega_{-1}, \Omega_0, \Omega_1, \ldots, \lim_{n\to\infty} \Omega_n = \infty$, $\lim_{n\to\infty} \Omega_{-n} = -\infty$, dále $\max|\Omega_{i+1} - \Omega_i| \to 0$, přechází součet (4.3-23) v integrál

$$z(l) = \int_{-\infty}^{\infty} e^{i\Omega l}\, dL(\Omega) \qquad (4.3\text{-}26)$$

a podobně

$$K_z(\tau) = \int_{-\infty}^{\infty} e^{i\Omega\tau}\, dH(\Omega). \qquad (4.3\text{-}27)$$

Vyjádření procesu $z(l)$ ve tvaru (4.3-26) nazýváme jeho spektrálním rozkladem, podobně mluvíme o spektrálním rozkladu korelační funkce (4.3-27).

V praktických příkladech zpravidla existuje spojitá derivace

$$H'(\Omega) = S_z(\Omega), \qquad (4.3\text{-}28)$$

a tedy

$$dH(\Omega) = S_z(\Omega)d\Omega, \qquad (4.3\text{-}29)$$

a potom můžeme psát

$$K_z(\tau) = \int_{-\infty}^{\infty} e^{i\Omega\tau} S_z(\Omega)d\Omega, \qquad (4.3\text{-}30)$$

neboli veličiny $K_z(\tau)$ a $S_z(\Omega)$ jsou navzájem přiřazeny Fourierovou transformací a tedy naopak platí

$$S_z(\Omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} K_z(\tau)e^{-i\Omega\tau}\, d\tau. \qquad (4.3\text{-}31)$$

Funkci $S_z(\Omega)$ nazýváme spektrální hustota procesu $z(t)$. $S(\Omega)d\Omega$ je tedy střední čtvercová amplituda komplexního kmitu s úhlovou rychlostí Ω. Ze vztahu (4.3-31) vyplývají tyto vlastnosti $S_z(\Omega)$:

a) $S_z(\Omega)$ je nezáporná

b) je-li $z(t)$ reálný proces, je $K_z(\tau)$ sudá a tedy

$$S_z(\Omega) = \frac{1}{\pi} \int_{0}^{\infty} K_z(\tau)\, cos\Omega\tau\, d\tau. \qquad (4.3\text{-}32)$$

a tedy $S_z(\Omega)$ je sudá.

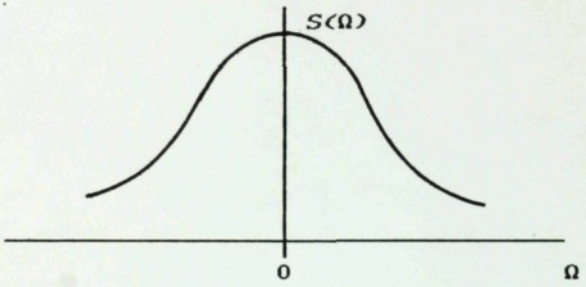
Ze vztahu (4.3-30) dále čteme (pro reálný proces)

$$K_z(0) = |z|^2 = \sigma^2(z) = \int_{-\infty}^{\infty} S_z(\Omega)d\Omega = 2\int_{0}^{\infty} S_z(\Omega)d\Omega. \qquad (4.3\text{-}33)$$

(4.3-33) představuje důležitý vztah mezi spekrální hustotou a důležitou charakteristikou $\sigma^2(z)$.

Korelační funkci ve tvaru $K_Z(\tau) = \sigma^2 e^{-\alpha|\tau|}$ odpovídá spektrální hustota

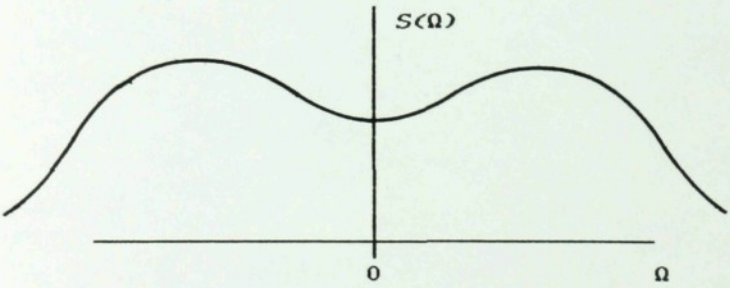
$$S_Z(\Omega) = \frac{1}{\pi} \frac{\alpha\sigma^2}{\alpha^2+\Omega^2}, \qquad (4.3\text{-}34)$$



Obr. 16. Spektrální hustota (4.3-34).

obdobně korelační funkci $K_Z(\tau) = \sigma^2 e^{-\alpha|\tau|} \cos \beta\tau$ odpovídá spektrální hustota

$$S_Z(\Omega) = \frac{\sigma^2}{2\pi} \left(\frac{1}{\alpha^2+(\Omega+\beta)^2} + \frac{1}{\alpha^2+(\Omega-\beta)^2} \right). \qquad (4.3\text{-}35)$$



Obr. 17. Spektrální hustota (4.3-35).

Bílý šum.

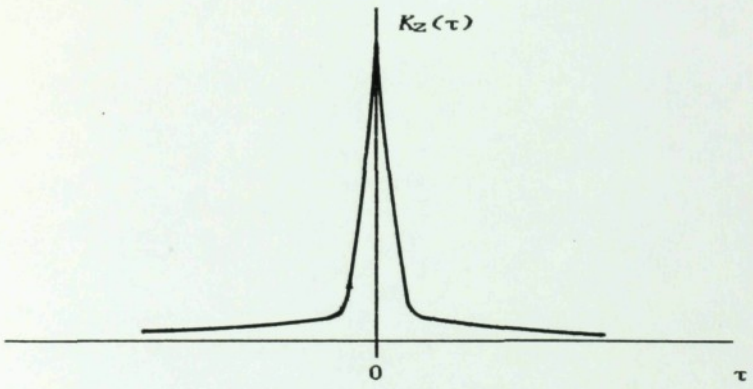
Mějme danou korelační funkci

$$K_Z(\tau) = \sigma^2 e^{-\alpha|\tau|},$$

a příslušnou spektrální hustotu

$$S_Z(\Omega) = \frac{1}{\pi} \cdot \frac{\alpha\sigma^2}{\alpha^2+\Omega^2}.$$

Uvažujme nyní $\sigma^2 = K\alpha$, kde K je nějaká kladná konstanta a nechť $\alpha \to \infty$, potom zřejmě $S_Z(\Omega) \to K/\pi$ pro každé Ω, dále $K_Z(\tau) \to 0$ pro $\tau \neq 0$, $K_Z(0) \to \infty$. Pro velké α je průběh $K_Z(\tau)$ na obrázku 18. Při popsaném limitním přechodu docházíme



Obr. 18. Korelační funkce.

k procesu, který je charakterizován konstantní spektrální

hustotou. Takto popsaný proces se nazývá bílý šum. Tento bílý šum nelze přesně fyzikálně realizovat, ale je možno se mu dosti přiblížit. Vzhledem k vlastnosti jeho korelační funkce vidíme, že pro dva libovolně blízké okamžiky t_1, t_2 jsou veličiny $z(t_1)$, $z(t_2)$ nekorelované. V případě, že proces je navíc Gaussovský, jsou veličiny $z(t_1)$, $z(t_2)$ nezávislé. Gaussovský bílý šum si lze tedy představit jako sled velmi krátkých nezávislých impulzů, přičemž střední čtvercová hodnota výšky impulzů je nekonečná.

4.4 Tvarovací filtr.

Běžný typ korelační funkce stacionárního procesu lze zapsat ve tvaru součtu

$$\sum_i \sigma_i^2 e^{-a_i|\tau|}, \qquad (4.4\text{-}1)$$

kde $Re\ a_i > 0$. Pro reálný proces se komplexní a_i vyskytují v párech komplexně sdružených. Spektrální hustota $S_x(\Omega)$ takového reálného procesu potom je součet funkcí tvaru (4.3-35), čili spektrální hustota je racionální funkcí Ω se specifickými vlastnostmi:

a) čitatel i jmenovatel obsahuje pouze sudé mocniny Ω s reálnými koeficienty,

b) stupeň jmenovatele je aspoň o 2 větší než stupeň čitatele.

$S_x(\Omega)$ je tedy sudá (což platí obecně pro spektrální hustotu reálného procesu) a tedy kořeny čitatele i jmenovatele jsou symetricky rozloženy kolem počátku. Vzhledem k tomu, že koeficienty jsou pouze reálné, vyskytují se pouze kořeny v párech komplexně sdružených viz kapitola 2.1 ad *c*. Dosaďme dále $p = i\Omega$, čili $\Omega = -ip$, potom $S_x(\Omega) = \tilde{S}_x(p) = \tilde{S}_x(i\Omega)$. Funkce $\tilde{S}_x(p)$ má zřejmě také vlastnosti *a*, *b* a tudíž také platí uvedený důsledek. Znamená to, že kořeny čitatele a jmenovatele racionální funkce $\tilde{S}_x(p)$ jsou dány ve čtveřicích, jak je naznačeno na obrázku v kapitole 2.1 ad *c*.

Vidíme tedy dále, že lze provést rozklad

$$\tilde{S}_x(p) = S_x^1(p).S_x^1(-p), \qquad (4.4\text{-}2)$$

kde $S_x^1(p)$ je zlomek, kde v čitateli (resp. ve jmenovateli) je součin všech kořenových činitelů čitatele (resp. jmenovatele) s kořeny vlevo od imaginární osy. Lze také psát

$$S_x(\Omega) = \tilde{S}_x(i\Omega) = S_x^1(i\Omega).S_x^{1*}(i\Omega). \qquad (4.4\text{-}3)$$

Výraz $S_x^1(i\Omega)$ představuje ale jistý stabilní přenos. Odtud tedy vyplývá: je-li na vstupu filtru, jehož přenos je $S_x^1(i\Omega)$, bílý šum se spektrální hustotou $S_0(\Omega) = 1$, je na výstupu proces se spektrální hustotou $S_x^1(i\Omega).S_x^{1*}(i\Omega).S_0(\Omega) = S_x(\Omega)$. Takovýto filtr tedy nazýváme tvarovací filtr procesu $x(t)$. Tento filtr tedy umožňuje produkovat (generovat) ze standartního bílého šumu proces s

předem danou korelační funkcí (resp. spektrální hustotou).

Závěr

Jak bylo uvedeno v úvodu, tato práce navazuje na diplomovou práci vypracovanou ing. Ivou Sedlákovou, která ověřovala algoritmy pro řízení tlumiče automobilu. Z výsledků této diplomové práce vyplynulo, že průběh a kvalitu řízení vstupující porucha podstatně ovlivňuje. Poruchou v tomto případě rozumíme nerovnosti vozovky.

Výsledkem této práce je tedy program, který generuje náhodný signál s danou spektrální hustotou, tj. se spektrální hustotou vozovky, viz obrázek 4. Tvar této spektrální hustoty odpovídá změřeným spektrálním hustotám skutečných vozovek, ale v jiném souřadném systému, viz obrázek 3.

Po aproximaci této spektrální hustoty byl na ukázku pro tuto aproximaci generován náhodný signál viz Příloha.

Pro praktické využití tohoto programu bych navrhoval aproximovat spektrální hustoty skutečných vozovek podle vztahu (1.3-3) pro konstanty uvedené v tabulce 2 a potom generovat jednotlivé náhodné signály.

Literatura

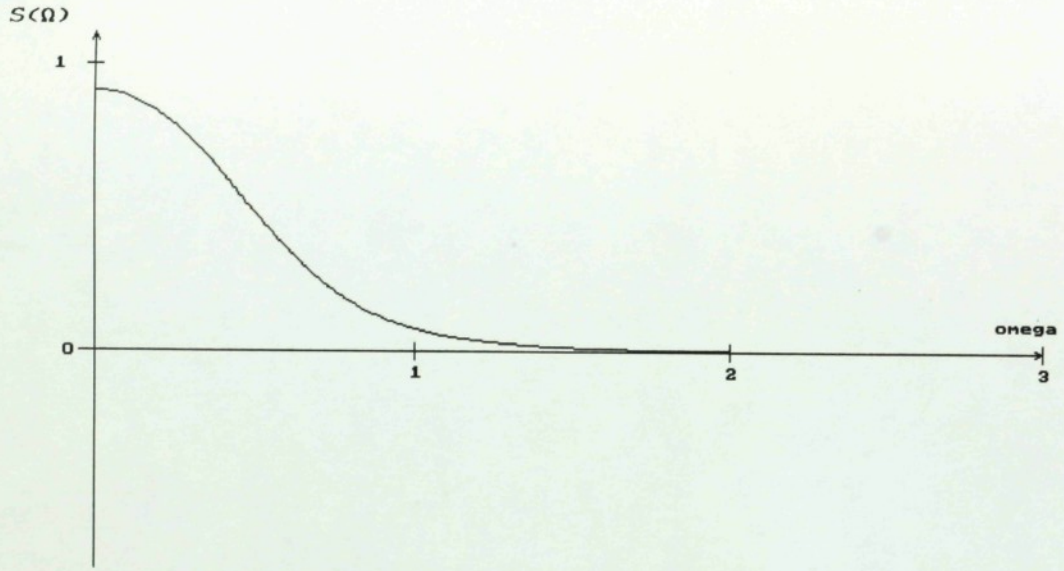
[1] Milan Apetaur : Motorová vozidla 5., ČVUT Praha, 1982

[2] Bořivoj Hanuš, Milan Balda a kol.: Základy technické kybernetiky 1., VŠST Liberec, 1989

[3] Bořivoj Hanuš, Jaroslav Balátě, Ivan Švarc, František Zikeš : Teorie automatického řízení 1., 1.část, VŠST Liberec, 1982

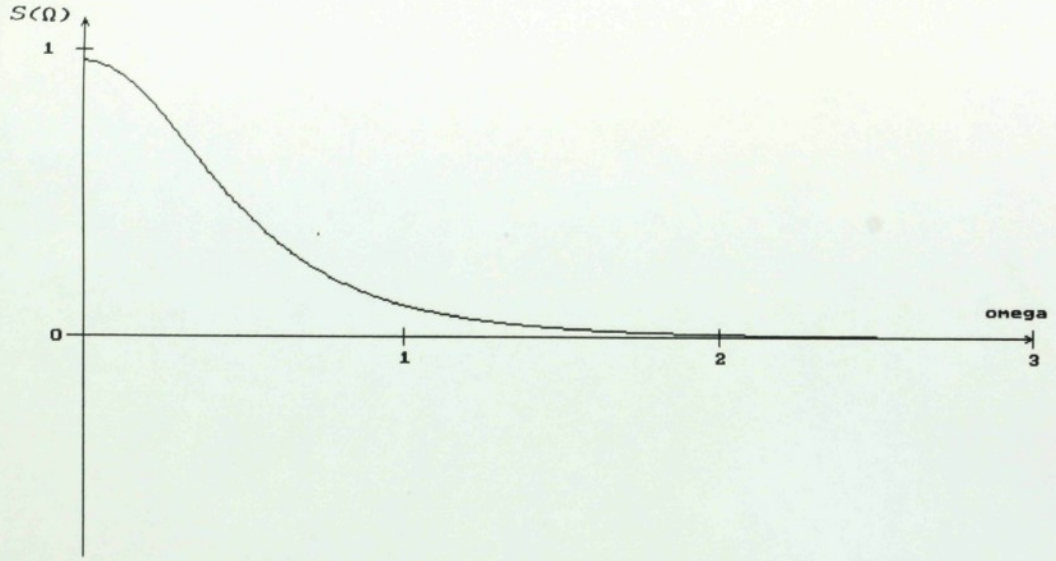
[4] Jiří Kadeřábek, Vladimír Kracík : Úvod do teorie pravděpodobnosti, matematické statistiky a příbuzných oblastí, VŠST Liberec, 1970

Seznam příloh



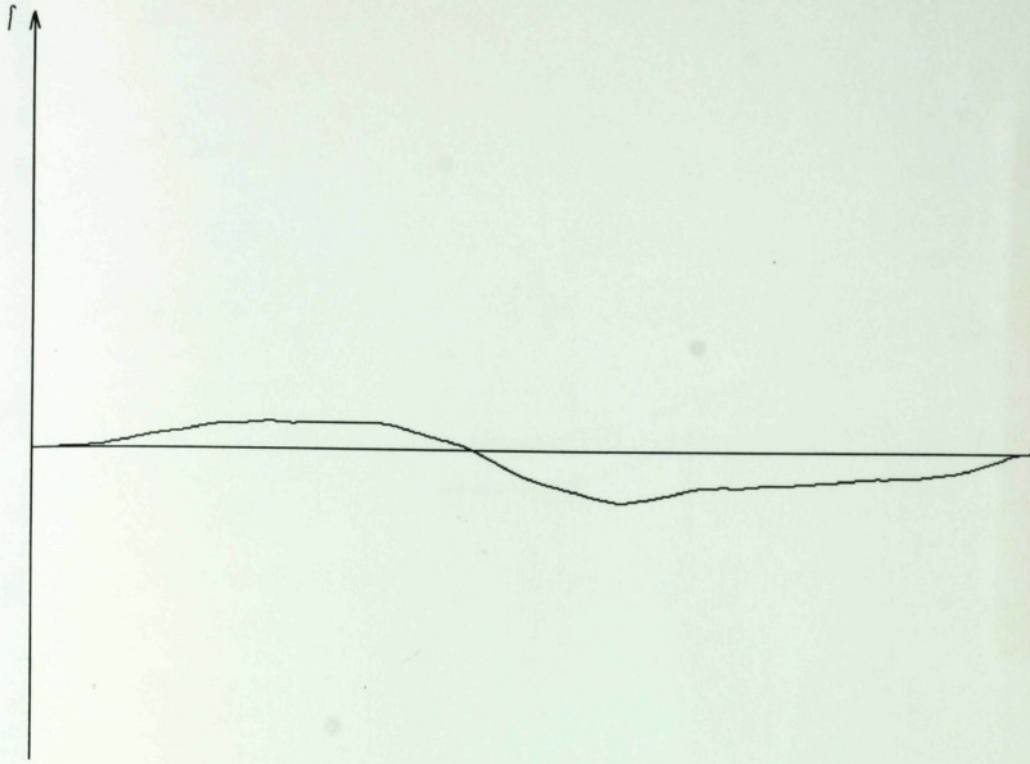
Graf 1. Aproximovaná spektrální hustota pro přenos (2.2-18)

resp. (2.3-10)



Graf 2. Aproximovaná spektrální hustota pro přenos (2.3-11)

resp. (2.3-12)



Graf 3. Generovaný náhodný signál pro přenos (2.2-18) resp. (2.3-10)

```
program ndip;
uses Crt,DIPGRAF3,VOZOVKA3;
const
     NNum=1;     NDenom=2;      NNDWhole=NNum+NDenom;
type
     vector1=array[1..NNDWhole]of real;
Const
     h0=1E-2;  OmegaK=1;  OmegaK0=2;  q=1;
     NSamp=100;      EPSILONa=-0.1;  EPSILONb=10;   EPSILONg=1E-2;
     k=-1*q/OmegaK;                  c:real=2E-2;
     NSampWhole=Round(OmegaK0*NSamp);
     KONEC:boolean=false;
{     h:real=0.1;
     Ap:vector1=(-1, -1/6, -0.5);
     Bp:vector1=(0.5, 0.25, 1);
     rp:array[1..3]of real=(-2, -1, 1);}
type
     vectora=array[0..2*NDenom+1]of real;
     vectorb=array[0..2*NNum+1]of real;
     vector2=array[1..2*NNDWhole]of real;
     vector4=array[1..4*NNDWhole]of real;
     vector8=array[1..8*NNDWhole]of real;
     pole=array[1..3,0..2]of real;
var
     I,J,L:word;     pc,stupen:byte;  odst,dost:shortint;
     Hom,Omega,Omega2,Omega4,A2,B2,Fo,Fn,fom,Z,Zp,gradFD,h:real;
     F:vector8; gradF:vector4;  A,B,CiJm:vector1;
     AB:pole;    FY,FF,FK:text;     kl:char;
     am,an:vectora; bm,bn:vectorb;    r:array[1..6]of real;
     Ap,Bp:vector1; rp:array[1..6]of real;

function FnZ(J:word):real;
var
    Z:real;
begin
    Omega:=OmegaK*J/NSamp;
    Omega2:=Sqr(Omega); Omega4:=Sqr(Omega2);
    for L:=1 to NNDWhole do begin
          A2:=Sqr(A[L]); B2:=Sqr(B[L]);
          CiJm[L]:=Omega4+2*Omega2*(A2-B2)+Sqr(A2+B2);
    end;
    Z:=(Omega2+Sqr(r[1]))*Sqr(r[3])/(Omega2+Sqr(r[2]));
    for L:=1 to NNum do Z:=Z*CiJm[L];
    for L:=Succ(NNum) to NNDWhole do Z:=Z/CiJm[L];
    FnZ:=Z;
end;

function FnF:real;
var
    Z,F:real;
begin
    for J:=0 to NSampWhole do begin
         Z:=FnZ(J);  if J<NSamp then fom:=k*Omega+q else fom:=0;
         Hom:=Sqr(Z-fom);
        { gotoxy(0,4);writeln('Z=',z);}
         if Odd(J)        then F:=F+4*Hom
```

```
                        else F:=F+2*Hom;
        if J=NSampWhole then F:=F-Hom;
        if J=0          then F:=Hom;
    end;
    FnF:=F*c;
end;

procedure Spektrum;
begin
    ASSIGN(FY,'DATAY.dat');  Rewrite(FY);
    writeln(FY,4*NNum); writeln(FY,4*NDenom);
    for J:=0 to 250(NSampWhole) do begin
    Zp:=FnZ(J);
    writeln (FY,Omega,' ',Zp);
    end;  Close(FY);
end;

procedure gradientF;
begin
    for I:=1 to 18(4*(NNDWhole+3)) do begin
        for J:=1 to NNDWhole do begin A[J]:=Ap[J]; B[J]:=Bp[J]; end;
        for J:=1 to 3 do r[J]:=rp[J];
        case I of
             1: A[1]:=A[1]+h0;    2: A[1]:=A[1]-h0;
             3: A[2]:=A[2]+h0;    4: A[2]:=A[2]-h0;
             5: A[3]:=A[3]+h0;    6: A[3]:=A[3]-h0;
             7: B[1]:=B[1]+h0;    8: B[1]:=B[1]-h0;
             9: B[2]:=B[2]+h0;   10: B[2]:=B[2]-h0;
            11: B[3]:=B[3]+h0;   12: B[3]:=B[3]-h0;
            13: r[1]:=r[1]+h0;   14: r[1]:=r[1]-h0;
            15: r[2]:=r[2]+h0;   16: r[2]:=r[2]-h0;
            17: r[3]:=r[3]+h0;   18: r[3]:=r[3]-h0;
            19: r[4]:=r[4]+h0;   20: r[4]:=r[4]-h0;
            21: r[5]:=r[5]+h0;   22: r[5]:=r[5]-h0;
            23: r[6]:=r[6]+h0;   24: r[6]:=r[6]-h0;
        end;
        F[I]:=FnF;
    end;
    for j:=1 to NNDWhole do writeln('alfa=',ap[j],'    beta=',bp[j]);
    for j:=1 to 3 do writeln('r=',rp[j]);
   (xxxxxxxxxxxxx vypocet gradF xxxxxxxxxxxxx)
    for J:=1 to 9(2*(NNDWhole+3)) do gradF[J]:=(F[2*J-1]-F[2*J])/2/h0;
    gradFD:=0;
    for J:=1 to 9 do gradFD:=gradFD+Sqr(gradF[J]); gradFD:=Sqrt(gradFD);
    for J:=1 to 9 do gradF[J]:=gradF[J]/gradFD;
end;

procedure NewKoef;
begin
    for J:=1 to 2*NNDWhole do begin
        if J<=NNDWhole then begin
            Ap[J]:=Ap[J]-h*gradF[J];
            if Ap[J]>EPSILONa then Ap[J]:=EPSILONa;
        end;
        if J>NNDWhole then begin
            Bp[J-NNDWhole]:=Bp[J-NNDWhole]-h*gradF[J];
```

```
            if ABS(Bp[J-NNDWhole])>EPSILONb then Bp[J-NNDWhole]:=EPSILONb;
        end;
    end;
    for J:=7 to 9 do begin
        rp[J-6]:=rp[J-6]-h*gradF[J];
{        if J<9 then begin
            if rp[J-6]>EPSILONa then rp[J-6]:=EPSILONa;
        end;}
        if J=9 then begin
            if rp[J-6]<0.1 then rp[J]:=0.1;
        end;
    end;
end;

procedure Uloz;
begin
    ASSIGN(FK,'DATAK.dat');   Rewrite(FK);
    for J:=1 to 3 do writeln(FK,Ap[J]);
    for J:=1 to 3 do writeln(FK,Bp[J]);
    for J:=1 to 3 do writeln(FK,rp[J]);
    writeln(FK,h);
    Close(FK);
end;

procedure Nacti;
begin
    ASSIGN(FK,'DATAK.dat');   Reset(FK);
    for J:=1 to 3 do readln(FK,Ap[J]);
    for J:=1 to 3 do readln(FK,Bp[J]);
    for J:=1 to 3 do readln(FK,rp[J]);
    readln(FK,h);
    Close(FK);
end;
(xxxxxxxxxxxx main xxxxxxxxxxxx)
begin     ClrScr;
    writeln('Pocitam spektralni hustotu.');
    writeln('Prosim, cekej tise.');
    Nacti;
    for J:=1 to NNDWhole do begin A[J]:=Ap[J]; B[J]:=Bp[J]; end;
    for J:=1 to 3 do r[J]:=rp[J];
    Fo:=FnF;

    repeat
    gradientF;
    NewKoef;
    (xxxxxxxx vypocet nove funkcni hodnoty xxxxxxxx)
    for J:=1 to NNDWhole do begin A[J]:=Ap[J]; B[J]:=Bp[J]; end;
    for J:=1 to 3 do r[J]:=rp[J];
    Fn:=FnF; writeln('Fn=',fn,'   h=',h); writeln;
    if Fn>Fo then h:=-0.9*h; Fo:=Fn;
    if ABS(h)<1E-7 then KONEC:=true;   I:=0;
    if KeyPressed then kl:=readkey;
    if kl='g' then begin
        Uloz;
        Spektrum; Graf;
        kl:='p';
```

```
    end;
    if kl='k' then KONEC:=true;
    if kl='v' then KONEC:=true;
    until KONEC;
{xxxxxxxxxxxx uschovani koeficientu xxxxxxxxxxxxx}
    Uloz;
{    ASSIGN(FK,'a:\DIP2\DATAK.dat');    Rewrite(FK);
    for J:=1 to 3 do writeln(FK,Ap[J]);
    for J:=1 to 3 do writeln(FK,Bp[J]);
    for J:=1 to 3 do writeln(FK,rp[J]);
    writeln(FK,h);
    Close(FK);}
{xxxxxxxxxxxx vypocet spektralni hustoty xxxxxxxxxxxx}
{    Spektrum;
    Graf;}
   if kl='v' then begin
{xxxxxxxxxxxx vypocet koeficientu filtru xxxxxxxxxxxx}
    for I:=1 to NNDWhole do begin
         AB[I,0]:=A[I]*A[I]+B[I]*B[I];
         AB[I,1]:=-2*A[I];
         AB[I,2]:=1;
    end;
{    xxxxxxxxx koeficienty am[i] xxxxxxxxx}
    for J:=0 to 2 do am[J]:=AB[NNum+1,J];
    if NDenom>1 then begin
         for pc:=(NNum+2) to NNDWhole do begin
              stupen:=(pc-NNum)*2;
              for J:=stupen downto 0 do begin
                   an[J]:=0;
                   odst:=J; if odst>(stupen-2) then odst:=stupen-2;
                   dost:=J-2; if dost<0 then dost:=0;
                   for I:=odst downto dost do an[J]:=an[J]+am[I]*AB[pc,J-I]
              end;
              for J:=0 to stupen do am[J]:=an[J];
         end;
    end;
    am[5]:=am[4];
    for j:=4 downto 1 do am[J]:=am[J-1]-am[J]*r[2];
    am[0]:=-1*am[0]*r[2];
{    xxxxxxxxx koeficienty bm[j] xxxxxxxxx}
    for J:=0 to 2 do bm[J]:=AB[1,J];
    if NNum>1 then begin
         for pc:=2 to NNum do begin
              stupen:=pc*2;
              for J:=stupen downto 0 do begin
                   bn[J]:=0;
                   odst:=J; if odst>(stupen-2) then odst:=stupen-2;
                   dost:=J-2; if dost<0 then dost:=0;
                   for I:=odst downto dost do bn[J]:=bn[J]+bm[J]*AB[pc,J-I]
              end;
              for J:=0 to stupen do bm[J]:=bn[J];
         end;
    end;
    bm[3]:=bm[2];
    for j:=2 downto 1 do bm[J]:=bm[J-1]-bm[J]*r[1];
    bm[0]:=-1*bm[0]*r[1];
```

```
    for J:=0 to 3 do bm[J]:=bm[J]*r[3];

    ASSIGN(FF,'DATAFK.dat');    Rewrite(FF);
    writeln(FF,2*NNum+2,' ',2*NDenom+2);
{    writeln('Koeficienty filtru:');}
    for J:=0 to 2*NDenom+1 do begin writeln(FF,am[J]);writeln('am',j,'=',am
        writeln(FF); writeln;
    for J:=0 to 2*NNum+1    do begin writeln(FF,bm[J]);writeln('bm',j,'=',bm
    Close(FF); writeln;
{    for J:=1 to NNDWhole do begin
        writeln('alfa',J,'=',A[J],'  beta',J,'=',B[J]);
    end;
    repeat until KeyPressed; ReadKey;}
    Generuj('DATAFK.dat');
   end;
end.
```

```
program ndip;
uses Crt,DIPGRAF3,VOZOVKA3;
const
     NNum=1;     NDenom=2;      NNDWhole=NNum+NDenom;
type
     vector1=array[1..NNDWhole]of real;
Const
     h0=1E-3;  OmegaK=1;  OmegaK0=2;  q=1;
     NSamp=100;      EPSILONa=-0.1;  EPSILONb=10;    EPSILONg=1E-2;
     k=-1*q/OmegaK;                  c:real=2E-2;
     NSampWhole=Round(OmegaK0*NSamp);
     KONEC:boolean=false;
{     Ap:vector1=(-1, -1/6, -0.5);
     Bp:vector1=(0.5, 0.25, 1);
     rp:array[1..3]of real=(-2, -1, 1);}
type
     vectora=array[0..2*NDenom+1]of real;
     vectorb=array[0..2*NNum+1]of real;
     vector2=array[1..2*NNDWhole]of real;
     vector4=array[1..4*NNDWhole]of real;
     vector8=array[1..8*NNDWhole]of real;
     pole=array[1..3,0..2]of real;
var
     I,J,L:word;     pc,stupen:byte;  odst,dost:shortint;
     Hom,Omega,Omega2,Omega4,A2,B2,Fo,Fn,fom,Z,Zp,gradFD,h:real;
     F:vector8; gradF:vector4;  A,B,CiJm:vector1;
     AB:pole;    FY,FF,FK:text;     kl:char;
     am,an:vectora; bm,bn:vectorb;    r:array[1..6]of real;
     Ap,Bp:vector1; rp:array[1..3]of real;

function FnZ(J:word):real;
var
    Z:real;
begin
    Omega:=OmegaK*J/NSamp;
    Omega2:=Sqr(Omega); Omega4:=Sqr(Omega2);
    for L:=1 to NNDWhole do begin
          A2:=Sqr(A[L]); B2:=Sqr(B[L]);
          CiJm[L]:=Omega4+2*Omega2*(A2-B2)+Sqr(A2+B2);
    end;
    Z:=(Omega2+Sqr(r[1]))*Sqr(r[3])/(Omega2+Sqr(r[2]));
    for L:=1 to NNum do Z:=Z*CiJm[L];
    for L:=Succ(NNum) to NNDWhole do Z:=Z/CiJm[L];
    FnZ:=Z;
end;

function FnF:real;
var
    Z,F:real;
begin
    for J:=0 to NSampWhole do begin
         Z:=FnZ(J);  if J<NSamp then fom:=k*Omega+q else fom:=0;
         Hom:=Sqr(Z-fom);
        { gotoxy(0,4);writeln('Z=',z);}
         if Odd(J)       then F:=F+4*Hom
                         else F:=F+2*Hom;
```

```
        if J=NSampWhole then F:=F-Hom;
        if J=0          then F:=Hom;
    end;
    FnF:=F*c;
end;

procedure Spektrum;
begin
    ASSIGN(FY,'DATAY.dat');  Rewrite(FY);
    writeln(FY,4*NNum); writeln(FY,4*NDenom);
    for J:=0 to 250(NSampWhole) do begin
    Zp:=FnZ(J);
    writeln (FY,Omega,' ',Zp);
    end;  Close(FY);
end;

procedure gradientF;
begin
    for I:=1 to 18(4*(NNDWhole+3)) do begin
        for J:=1 to NNDWhole do begin A[J]:=Ap[J]; B[J]:=Bp[J]; end;
        for J:=1 to 3 do r[J]:=rp[J];
        case I of
            1: A[1]:=A[1]+h0;    2: A[1]:=A[1]-h0;
            3: A[2]:=A[2]+h0;    4: A[2]:=A[2]-h0;
            5: A[3]:=A[3]+h0;    6: A[3]:=A[3]-h0;
            7: B[1]:=B[1]+h0;    8: B[1]:=B[1]-h0;
            9: B[2]:=B[2]+h0;   10: B[2]:=B[2]-h0;
           11: B[3]:=B[3]+h0;   12: B[3]:=B[3]-h0;
           13: r[1]:=r[1]+h0;   14: r[1]:=r[1]-h0;
           15: r[2]:=r[2]+h0;   16: r[2]:=r[2]-h0;
           17: r[3]:=r[3]+h0;   18: r[3]:=r[3]-h0;
           19: r[4]:=r[4]+h0;   20: r[4]:=r[4]-h0;
           21: r[5]:=r[5]+h0;   22: r[5]:=r[5]-h0;
           23: r[6]:=r[6]+h0;   24: r[6]:=r[6]-h0;
        end;
        F[I]:=FnF;
    end;
    for j:=1 to NNDWhole do writeln('alfa=',ap[j],'    beta=',bp[j]);
    for j:=1 to 3 do writeln('r=',rp[j]);
   (xxxxxxxxxxxxx vypocet gradF xxxxxxxxxxxxx)
    for J:=1 to 9(2*(NNDWhole+3)) do gradF[J]:=(F[2*J-1]-F[2*J])/2/h0;
    gradFD:=0;
    for J:=1 to 9 do gradFD:=gradFD+Sqr(gradF[J]); gradFD:=Sqrt(gradFD);
    for J:=1 to 9 do gradF[J]:=gradF[J]/gradFD;
end;

procedure NewKoef;
begin
    for J:=1 to 2*NNDWhole do begin
        if J<=NNDWhole then begin
            Ap[J]:=Ap[J]-h*gradF[J];
            if Ap[J]>EPSILONa then Ap[J]:=EPSILONa;
        end;
        if J>NNDWhole then begin
            Bp[J-NNDWhole]:=Bp[J-NNDWhole]-h*gradF[J];
            if ABS(Bp[J-NNDWhole])>EPSILONb then Bp[J-NNDWhole]:=EPSILON
```

```
        end;
    end;
    for J:=7 to 9 do begin
        rp[J-6]:=rp[J-6]-h*gradF[J];
        if J<9 then begin
            if rp[J-6]>EPSILONa then rp[J-6]:=EPSILONa;
        end;
        if J=9 then begin
            if rp[J-6]<0.1 then rp[J]:=0.1;
        end;
    end;
end;

procedure Uloz;
begin
    ASSIGN(FK,'DATAKr.dat');   Rewrite(FK);
    for J:=1 to 3 do writeln(FK,Ap[J]);
    for J:=1 to 3 do writeln(FK,Bp[J]);
    for J:=1 to 3 do writeln(FK,rp[J]);
    Close(FK);
end;

procedure Nacti;
begin
    ASSIGN(FK,'DATAKr.dat');   Reset(FK);
    for J:=1 to 3 do readln(FK,Ap[J]);
    for J:=1 to 3 do readln(FK,Bp[J]);
    for J:=1 to 3 do readln(FK,rp[J]);
    Close(FK);
end;
(xxxxxxxxxxxx main xxxxxxxxxxxx)
begin     ClrScr;
    writeln('Pocitam spektralni hustotu.');
    writeln('Prosim, cekej tise.');
    Nacti;
    for J:=1 to NNDWhole do begin A[J]:=Ap[J]; B[J]:=Bp[J]; end;
    for J:=1 to 3 do r[J]:=rp[J];
    Fo:=FnF;

    repeat
    gradientF; h:=0.001;
   repeat
    NewKoef;
    (xxxxxxxx vypocet nove funkcni hodnoty xxxxxxxx)
    for J:=1 to NNDWhole do begin A[J]:=Ap[J]; B[J]:=Bp[J]; end;
    for J:=1 to 3 do r[J]:=rp[J];
    Fn:=FnF; writeln('Fn=',fn,'   h=',h);
    if Fn>Fo then h:=-0.75*h;
    Fo:=Fn;
   until ABS(h)<1E-7;
    if KeyPressed then kl:=readkey;
    if kl='g' then begin
        Uloz;
        Spektrum; Graf;
        kl:='p';
    end;
```

```
    if kl='k' then KONEC:=true;
    if kl='v' then KONEC:=true;
    until KONEC;
{xxxxxxxxxxxx uschovani koeficientu xxxxxxxxxxxxx}
    Uloz;
{    ASSIGN(FK,'a:\DIP2\DATAKr.dat');   Rewrite(FK);
    for J:=1 to 3 do writeln(FK,Ap[J]);
    for J:=1 to 3 do writeln(FK,Bp[J]);
    for J:=1 to 3 do writeln(FK,rp[J]);
    Close(FK);}
{xxxxxxxxxxxx vypocet spektralni hustoty xxxxxxxxxxxx}
{    Spektrum;
    Graf;}
   if kl='v' then begin
{xxxxxxxxxxxx vypocet koeficientu filtru xxxxxxxxxxxx}
    for I:=1 to NNDWhole do begin
        AB[I,0]:=A[I]*A[I]+B[I]*B[I];
        AB[I,1]:=-2*A[I];
        AB[I,2]:=1;
    end;
{    xxxxxxxxx koeficienty am[i] xxxxxxxxx}
    for J:=0 to 2 do am[J]:=AB[NNum+1,J];
    if NDenom>1 then begin
        for pc:=(NNum+2) to NNDWhole do begin
            stupen:=(pc-NNum)*2;
            for J:=stupen downto 0 do begin
                an[J]:=0;
                odst:=J; if odst>(stupen-2) then odst:=stupen-2;
                dost:=J-2; if dost<0 then dost:=0;
                for I:=odst downto dost do an[J]:=an[J]+am[I]*AB[pc,J-I]
            end;
            for J:=0 to stupen do am[J]:=an[J];
        end;
    end;
    am[5]:=am[4];
    for J:=4 downto 1 do am[J]:=am[J-1]-am[J]*r[2];
    am[0]:=-1*am[0]*r[2];
{    xxxxxxxxx koeficienty bm[j] xxxxxxxxx}
    for J:=0 to 2 do bm[J]:=AB[1,J];
    if NNum>1 then begin
        for pc:=2 to NNum do begin
            stupen:=pc*2;
            for J:=stupen downto 0 do begin
                bn[J]:=0;
                odst:=J; if odst>(stupen-2) then odst:=stupen-2;
                dost:=J-2; if dost<0 then dost:=0;
                for I:=odst downto dost do bn[J]:=bn[J]+bm[J]*AB[pc,J-I]
            end;
            for J:=0 to stupen do bm[J]:=bn[J];
        end;
    end;
    bm[3]:=bm[2];
    for J:=2 downto 1 do bm[J]:=bm[J-1]-bm[J]*r[1];
    bm[0]:=-1*bm[0]*r[1];
    for J:=0 to 3 do bm[J]:=bm[J]*r[3];
```

```
    ASSIGN(FF,'DATAFKr.dat');   Rewrite(FF);
    writeln(FF,2*NNum+2,' ',2*NDenom+2);
{    writeln('Koeficienty filtru:');}
    for J:=0 to 2*NDenom+1 do begin writeln(FF,am[J]);writeln('am',j,'=',am
        writeln(FF); writeln;
    for J:=0 to 2*NNum+1   do begin writeln(FF,bm[J]);writeln('bm',j,'=',bm
    Close(FF); writeln;
{    for J:=1 to NNDWhole do begin
        writeln('alfa',J,'=',A[J],'  beta',J,'=',B[J]);
    end;
    repeat until KeyPressed; ReadKey;}
    Generuj('DATAFKr.dat');
  end;
end.
```

```
unit DIPGRAF3;
interface
procedure graf;
implementation
uses
  dos,graph,Crt;
type POLE=array [1..250] of real;
     PPOLE=array [1..10,1..10] of integer;
     POL=array [1..20] of integer;
     PPOL=array [1..10] of real;
var N,I,J,K,Gd,Gm,MER,MER1,MERX,MERY,MX,MY:integer;
    m,Barva,P,kk,k2,MM:integer;
    A:longint;
    lx,MAXX,MAXY,LY,POM,MAX,MIN,L,SS:real;
    ZAD,Z:char;
    X,Y:POLE;
    FN:POL;
    UU:PPOLE;
    FS:PPOL;
    FY,F1:text;
    s,sa,sss,ST,OrdNum,OrdDenom:string;
    ui:array [1..10] of string;

procedure HC(XMin, YMin, XMax, YMax,
                        Color1,Color2,Color3,Color4:word;
                        Inverse : boolean; ModeE : byte);
const
  Esc    = 27;
  XScreenMaxGlb = 639;
  YMaxGlb = 479;

var
  ScanLine : integer;
  n1, n2   : byte;
  ChKey    : char;
  Lst      : text;

procedure SendByte(B : byte);
  const
    LPTPortNum = 1;
  var
    Regs : Registers;
  begin
    Regs.AH := 0;
    Regs.AL := B;
    Regs.DX := Pred(LPTPortNum);
    Intr($17, Regs);
  end;

function ConstructByte(X, Y : integer) : byte;
  const
    Bits : array[0..7] of byte = (128,64,32,16,8,4,2,1);
  var
    CByte, Bit : byte;
    PosX       : integer;
    Color      : word;
```

```
  begin
    CByte:=0;
    PosX:=Xmin + (X * 8);
    for Bit := 0 to 7 do
      if ((PosX+Bit)>=XMin) and ((PosX+Bit)<=XMax) and (Y<=YMax)
           and (Y>=YMin) then
      begin
        Color:=GetPixel(PosX+Bit,YMin+YMax-Y);
        if ((Color)<>Color1) and ((Color)<>Color2) and
           ((Color)<>Color3) and ((Color)<>Color4) then CByte:=CByte+Bits[I
      end;
    ConstructByte := CByte;
  end;

procedure DoLine;
  var
    XPixel     : integer;
    PrintByte : byte;
  begin
    if ModeE = 1 then
      begin
        SendByte(Esc);                    { Vyber dvojnasobneho graf. tiskoveho mo
        SendByte(Ord('L'));
      end
    else
      begin                               { Vyber 8-Pin graf. tiskoveho modu }
        SendByte(Esc);
        SendByte(Ord('x'));
        SendByte(ModeE);
      end;
    SendByte(n1);                         { Posle 2 byty kontrolniho kodu }
    SendByte(n2);
    for XPixel := YMin to YMax do
      begin
        PrintByte := ConstructByte(ScanLine,XPixel);
        if Inverse then
          PrintByte := not PrintByte;
        SendByte(PrintByte);
        if ModeE=1 then SendByte(PrintByte);    { Second byte for ESC L ! }
      end;
    SendByte(10);                         { Send line feed }
  end;

begin                                     { HardCopy }
  Assign(Lst,'PRN');
  ReWrite(Lst);
  ModeE := ModeE mod 7;
  if (ModeE = 0) or (ModeE = 5) then ModeE := 4;

  SendByte(Esc);                          { Select 24/216-inch line spacing }
  SendByte(Ord('3'));
  SendByte(24);

  n1 := Lo(Succ(2*(YMax - YMin)));
  n2 := Hi(Succ(2*(YMax - YMin)));
```

```
  for ScanLine := 0 to ((XMax-Xmin) div 8) do
  begin
    if KeyPressed then
    begin
      ChKey:=ReadKey;
      if ChKey=#0 then ChKey:=char(byte(ReadKey)+128);
      if ChKey=#27 then
      begin
        SendByte(Esc); SendByte(2);
        exit;
      end;
    end;
    DoLine;
  end;
  SendByte(Esc); SendByte(2);        { Select 1/6-inch line spacing }
  Close(Lst);
end;                                 { HardCopy }


 procedure osy (i:integer);
    begin
     case i of 1: sa:='1';
               2: sa:='2';
               3: sa:='3';
               4: sa:='4';
               5: sa:='5';
               6: sa:='6';
               7: sa:='7';
               8: sa:='8';
               9: sa:='9';
               10: sa:='10';
               11: sa:='11';
               15: sa:='15';
               20: sa:='20';
               25: sa:='25';
               30: sa:='30';
               35: sa:='35';
               40: sa:='40';
               45: sa:='45';
               50: sa:='50';
               55: sa:='55';
               60: sa:='60';
               end;
               end;
procedure graf;
begin
  {xxxxxxxxxxxxxxxxxxx ZADAVANI HODNOT xxxxxxxxxxxxxxxxxxx}
  assign(FY,'c:\DATAY.dat');
  reset(FY);
  MAXX:=0; MAXY:=0;
  J:=0;
  ClrScr;
  writeln('Nacitam hodnoty.');
  readln(FY,OrdNum);readln(FY,OrdDenom);
  while not EOF(FY) do
  begin
```

```
   readln(FY,X[J],Y[J]);
{   writeln(x[j],y[j]);}
   if (MAXY<abs(Y[J])) then MAXY:=abs(Y[J]);
   if (MAXX<abs(X[J])) then MAXX:=abs(X[J]);
   j:=J+1;
  end;
   close(FY);
   m:=J-1;
   writeln('Mam to nacteno');
   {readln;}

  {xxxxxxxxxxxxxxxxxxx NASTAVENI GRAFIKY xxxxxxxxxxxxxxxxxxxxxxxx}
  Gd:=Detect;
  InitGraph(Gd,Gm,'C:\TP\BGI');
  if Graphresult <> grOk then Halt(1);


  {xxxxxxxxxxxxxx KRESLENI GRAFU ZAVISLOSTI Fs-S xxxxxxxxxxxxxxxxx}

  SetColor(15);                                {osy}
  Line(30,60,30,460);              {Y}
  Line(20,260,630,260);            {X}
  Line(30,60,28,65);                           {sipky}
  Line(30,60,32,65);
  Line(630,260,625,258);
  Line(630,260,625,262);
  if (MAXX<=10) then begin MAXX:=trunc(MAXX)+1;          {vlast.popis os-c
                           MX:=trunc(MAXX);
                           sss:='';kk:=0;
                     end
                else if MAXX<=100 then begin MAXX:=trunc(MAXX/10)+1;
                                             MX:=trunc(MAXX*10);
                                             sss:='0';kk:=5;
                                       end
                                  else begin MAXX:=trunc(MAXX/100)+1;
                                             MX:=trunc(MAXX*100);
                                             sss:='00';kk:=10;
                                       end;
  MERX:=trunc(600/MAXX);
  MER:=MERX+30;
  for I:=1 to trunc(MAXX) do
     begin
      osy(i);
      if i<10 then insert(sss,sa,2);
      if i>=10 then insert(sss,sa,3);
      Line(MER,255,MER,265);
      outtextxy(mer-kk-2,270,sa);
            MER:=MER+MERX;
      end;
  if MAXY<=10 then begin MAXY:=trunc(MAXY)+1;
                         MY:=trunc(MAXY);
                         sss:='';
                   end
              else if MAXY<=100 then begin MAXY:=trunc(MAXY/10)+1;
                                           MY:=trunc(MAXY*10);
                                           sss:='0';
```

```
                                    end
                                else begin MAXY:=trunc(MAXY/100)+1;
                                           MY:=trunc(MAXY*100);
                                           sss:='00';
                                     end;
MERY:=trunc(180/MAXY);
MER:=260;
MER1:=260;
for I:=1 to trunc(MAXY) do
    begin
          MER:=MER-MERY;
          osy (i);
          if i<10 then insert(sss,sa,2);
          if i>=10 then insert(sss,sa,3);
          outtextxy(5,mer-3,sa);
          Line(25,MER,35,MER);
          MER1:=MER1+MERY;
          insert('-',sa,1);
          outtextxy(0,mer1-3,sa);
          Line(25,MER1,35,MER1);
    end;
OutTextXY(10,50,'Y');
OutTextXY(10,257,'0');
OutTextXY(600,240,'omega');
OutTextXY(608,250,'');
OutTextXY(300,320,'Stupen  citatele   je');OutTextXY(480,320,OrdNum);
OutTextXY(300,335,'Stupen jmenovatele je');OutTextXY(480,335,OrdDenom);
MAX:=0;
MERX:=trunc(600/MX);
MERY:=trunc(180/MY);                          {kresleni grafu-Lagrange}
for P:=1 to 2 do
 begin
{BAR(I);}Barva:=14;
  { LX:=0;
   while(LX<=MX) do
     begin
       SS:=0;
       for J:=1 to M do
        begin
          LY:=1;
          for K:=1 to M do
            begin
              if J<>K then begin POM:=(LX-X[K])/(X[J]-X[K]);
                                 LY:=LY*POM;
                           end
            end;
          SS:=SS+LY*Y[J];
        end;}
     SetColor(15);
     if ((LX=0) and(P=2) ) then {if (30+trunc(X[J,I]/MX*600))<500 then}
        for J:=1 to M do Circle(30+trunc(X[J]/MX*600),260-trunc(Y[J]*MEI
{    SetColor(Barva);
      if P=1 then if MAX<abs(SS) then MAX:=abs(SS);
     k2:=k2+1;
     if ((LX<(MX)) and (P=2)) then}
        begin
```

```
            k2:=1;
            for j:=0 to m do begin
             A:=trunc(Y[j]*MERY);
               if j=0 then moveto(30+trunc(X[j]/MX*600),260-A)
                      else lineto(30+trunc(X[j]/mx*600),260-A);
          end;


        LX:=LX+(mx/600);
        end;
    end;


   SetColor(12);
   SetTextStyle(TriplexFont,HorizDir,3);
   OutTextXY(200,50,'Spektralni hustota - vypoctena');
   SetTextStyle(DefaultFont,HorizDir,1);
{   outtextxy(220,90,s);}

  repeat until Keypressed;
ReadKey;  CloseGraph;
{      HC(0,0,GetMaxX,GetMaxY,14,15,13,12,true,1);}
end;end.
```

```
unit VOZOVKA3;
interface
procedure GENERUJ(filtr:string);
procedure GENERUJ1(filtr:string);
procedure GENERUJ2(filtr:string);
implementation
uses Crt,Graph;
Const
  MaxPole=10;      JS=1/6;          PPG:boolean=false;       PocetIter=1;
  DELTAT=0.01/PocetIter;                              {u:word=50;}
  ZHUSTENI=1;
type
  Pole=array[0..MaxPole] of real;
var
  J,I:byte;              FV,FF:text;            uo,u:real{word};
  Y,YO:real;
  AS,BS,K1,K2,K3,K4,L,LO,Di,g:Pole;    Color,ColorBk:integer;
  GraphDriver,GraphMode:integer;                  s:string;
  Size:word;          p:pointer;          NNum,NDenom:byte;

procedure KRESLI1(YP,YPO:real);
Const
  MERITKO=100;             Color  =Green;
  PosunutiY=200;           ColorBk=Black;
  X:integer=1;
var
  Y,YO,Xp,MaxX,MaxY:integer;
begin
  MaxX:=GetMaxX;  MaxY:=GetMaxY;  Xp:=Round(X/(ZHUSTENI*PocetIter));
  YO:=MaxY-Round(YPO*MERITKO)-PosunutiY; Y:=MaxY-Round(YP*MERITKO)-Posunuti
  SetColor(ColorBk);                Line(Xp,0,Xp,MaxY);
  SetColor(Color);                  Line(Pred(Xp),YO,Xp,Y);{PutPixel(x,y,green)
  PutPixel(Xp,MaxY-PosunutiY,3);
  if Xp=MaxX then begin
    X:=0;  SetColor(ColorBk);    Line(0,0,0,MaxY);
  end;
  X:=Succ(X);
end;

procedure KRESLI2(YP,YPO:real;p:pointer);
Const
  MERITKO=100;             Color  =LightMagenta;
  PosunutiY=0;             ColorBk=Black;
var
  MaxX,MaxY,Y,YO:integer;
begin
  MaxX:=GetMaxX; MaxY:=GetMaxY;
  YO:=MaxY-Round(YPO*MERITKO)-PosunutiY; Y:=MaxY-Round(YP*MERITKO)-Posunuti
  GetImage(MaxX-99,MaxY-100,MaxX,MaxY,p^);
{  SetActivePage(1);}
  PutImage(MaxX-100,MaxY-100,p^,NormalPut);
  SetColor(ColorBk);              Line(MaxX,0,MaxX,MaxY);
  SetColor(Color);                Line(Pred(MaxX),YO,MaxX,Y);
  PutPixel(MaxX,MaxY,3);
  {  SetVisualPage(1);}
end;
```

```
procedure GENERUJ1(filtr:string);
begin
  for J:=0 to 10 do begin as[J]:=0; bs[J]:=0; end;
  {  ASSIGN(FV,'DATAV.dat');          ReWrite(FV);}
  ASSIGN(FF,filtr);          reset(FF);
  ReadLn(FF,NNum,NDenom);  writeln('NND',NNum,NDenom);
  for J:=0 to Pred(NDenom) do read(FF,AS[J]);
  for J:=0 to Pred(NNum)   do read(FF,BS[J]);
  Close(FF);
{  for J:=0 to Pred(NDenom) do writeln(as[J]);
  for J:=0 to Pred(NNum)   do writeln(bs[J]);}
  for J:=0 to Pred(Nnum)   do BS[J]:=BS[J]/AS[Pred(NDenom)];
  for J:=0 to Pred(NDenom) do AS[J]:=AS[J]/AS[Pred(NDenom)];
  {xxxxxxxxxx inicializace RK4 xxxxxxxxxxx}
  Randomize;   for J:=0 to Pred(NDenom)-1 do LO[J]:=0(Random);
  {xxxxxxxxxx inicializace Graph xxxxxxxxx}
  GraphDriver:=Detect;
  InitGraph(GraphDriver,GraphMode,'c:\TP\BGI');
  if GraphResult <> grOK then Halt(2);
  SetWriteMode(CopyPut); SetBkColor(ColorBk);
  Size:=ImageSize(GetMaxX-99,GetMaxY-100,GetMaxX,GetMaxY);
  GetMem(p,Size);

  g[5]:=bs[4]; g[4]:=bs[3]-g[5]*as[4]; g[3]:=bs[2]-g[5]*as[3]-g[4]*as[4];
  g[2]:=bs[1]-g[5]*as[2]-g[4]*as[3]-g[3]*as[4];
  g[1]:=bs[0]-g[5]*as[1]-g[4]*as[2]-g[3]*as[3]-g[2]*as[4];

  repeat
  u:=0; for J:=1 to 12 do u:=u+Random; u:=(u-6)/Sqrt(DELTAT);

  for I:=1 to PocetIter do begin
    {xxxxxxxxxxxxxxx K1 xxxxxxxxxxxxxxxx}
    for J:=0 to 4 do L[J]:=LO[J];
    for J:=0 to 3 do K1[J]:=(L[J+1]+g[5-J]*u)*DELTAT;
    K1[4]:=g[1]*u;
    for J:=0 to 4 do K1[4]:=K1[4]-as[J]*L[J]; K1[4]:=K1[4]*DELTAT;
    {xxxxxxxxxxxxxxx K2 xxxxxxxxxxxxxxx}
    for J:=0 to 4 do L[J]:=LO[J]+0.5*K1[J];
    for J:=0 to 3 do K2[J]:=(L[J+1]+g[5-J]*u)*DELTAT;
    K2[4]:=g[1]*u;
    for J:=0 to 4 do K2[4]:=K2[4]-as[J]*L[J]; K2[4]:=K2[4]*DELTAT;
    {xxxxxxxxxxxxxxx K3 xxxxxxxxxxxxxxx}
    for J:=0 to 4 do L[J]:=LO[J]+0.5*K2[J];
    for J:=0 to 3 do K3[J]:=(L[J+1]+g[5-J]*u)*DELTAT;
    K3[4]:=g[1]*u;
    for J:=0 to 4 do K3[4]:=K3[4]-as[J]*L[J]; K3[4]:=K3[4]*DELTAT;
    {xxxxxxxxxxxxxxx K4 xxxxxxxxxxxxxxx}
    for J:=0 to 4 do L[J]:=LO[J]+K3[J];
    for J:=0 to 3 do K4[J]:=(L[J+1]+g[5-J]*u)*DELTAT;
    K4[4]:=g[1]*u;
    for J:=0 to 4 do K4[4]:=K4[4]-as[J]*L[J]; K4[4]:=K4[4]*DELTAT;
    {xxxxxxxxxxxxx Di,LO xxxxxxxxxxxxxx}
    for J:=0 to 4 do begin
      Di[J]:=(K1[J]+2*(K2[J]+K3[J])+K4[J])*JS;
      LO[J]:=LO[J]+Di[J];
```

```
    end;
    L0[Pred(NDenom)]:=u;
    for J:=0 to Pred(NDenom)-1 do
      L0[Pred(NDenom)]:=L0[Pred(NDenom)]-L0[J]*AS[J];
    Y:=L0[0];
{    WriteLn(FV,Y);}
    if PPG then KRESLI1(y,yo{,p}) else PPG:=true; yo:=y;
  end;
  until  KeyPressed;  ReadKey;
{ Close(FV);}
end;
procedure GENERUJ2(filtr:string);
begin   u:=100;
  GraphDriver:=Detect;
  InitGraph(GraphDriver,GraphMode,'c:\TP\BGI');
  if GraphResult <> grOK then Halt(2);
  SetWriteMode(CopyPut); SetBkColor(ColorBk);
      repeat
      u:=u+Random(3)-1;
      if PPG then KRESLI1(u,uo{,p}) else PPG:=true; uo:=u;
      until  KeyPressed;  ReadKey;   end;

procedure GENERUJ(filtr:string);
begin
{ ASSIGN(FV,'DATAV.dat');          ReWrite(FV);}
  ASSIGN(FF,filtr);          reset(FF);
  ReadLn(FF,NNum,NDenom);  writeln('NND',NNum,NDenom);
  for J:=0 to Pred(NDenom) do read(FF,AS[J]);
  for J:=0 to Pred(NNum)   do read(FF,BS[J]);
  Close(FF);
  for J:=0 to Pred(NDenom) do writeln(as[J]);
  for J:=0 to Pred(NNum)   do writeln(bs[J]);
  for J:=0 to Pred(Nnum)   do BS[J]:=BS[J]/AS[Pred(NDenom)];
  for J:=0 to Pred(NDenom) do AS[J]:=AS[J]/AS[Pred(NDenom)];
  {xxxxxxxxxxx inicializace RK4 xxxxxxxxxxxx}
{ Randomize;}   for J:=0 to Pred(NDenom)-1 do L0[J]:=0;{Random;}
  {xxxxxxxxxxx inicializace Graph xxxxxxxxxx}
  GraphDriver:=Detect;
  InitGraph(GraphDriver,GraphMode,'c:\TP\BGI');
  if GraphResult <> grOK then Halt(2);
  SetWriteMode(CopyPut); SetBkColor(ColorBk);
  Size:=ImageSize(GetMaxX-99,GetMaxY-100,GetMaxX,GetMaxY);
  GetMem(p,Size);

  repeat
  u:=0; for J:=1 to 12 do u:=u+Random; u:=(u-6)/Sqrt(DELTAT);
{ u:=1;Random;}
  for I:=1 to PocetIter do begin
    {xxxxxxxxxxxxxxx K1 xxxxxxxxxxxxxxxx}
    for J:=0 to Pred(NDenom)-1 do L[J]:=L0[J];
    if Pred(NDenom)>1 then for J:=0 to Pred(NDenom)-2 do K1[J]:=L[Succ(J)];
    K1[Pred(NDenom)-1]:=u;
    for J:=0 to Pred(NDenom)-1 do
      K1[Pred(NDenom)-1]:=K1[Pred(NDenom)-1]-L[J]*AS[J];
    {xxxxxxxxxxxxxxx K2 xxxxxxxxxxxxxxxx}
    for J:=0 to Pred(NDenom)-1 do L[J]:=L0[J]+0.5*K1[J]*DELTAT;
```

```
    if Pred(NDenom)>1 then for J:=0 to Pred(NDenom)-2 do K2[J]:=L[Succ(J)];
    K2[Pred(NDenom)-1]:=u;
    for J:=0 to Pred(NDenom)-1 do
      K2[Pred(NDenom)-1]:=K2[Pred(NDenom)-1]-L[J]*AS[J];
    {xxxxxxxxxxxxxxx K3 xxxxxxxxxxxxxxx}
    for J:=0 to Pred(NDenom)-1 do L[J]:=L0[J]+0.5*K2[J]*DELTAT;
    if Pred(NDenom)>1 then for J:=0 to Pred(NDenom)-2 do K3[J]:=L[Succ(J)];
    K3[Pred(NDenom)-1]:=u;
    for J:=0 to Pred(NDenom)-1 do
      K3[Pred(NDenom)-1]:=K3[Pred(NDenom)-1]-L[J]*AS[J];
    {xxxxxxxxxxxxxxx K4 xxxxxxxxxxxxxxxx}
    for J:=0 to Pred(NDenom)-1 do L[J]:=L0[J]+K3[J]*DELTAT;
    if Pred(NDenom)>1 then for J:=0 to Pred(NDenom)-2 do K4[J]:=L[Succ(J)];
    K4[Pred(NDenom)-1]:=u;
    for J:=0 to Pred(NDenom)-1 do
      K4[Pred(NDenom)-1]:=K4[Pred(NDenom)-1]-L[J]*AS[J];
    {xxxxxxxxxxxxxx Di,L0 xxxxxxxxxxxxxxx}
    for J:=0 to Pred(NDenom)-1 do begin
      Di[J]:=(K1[J]+2*(K2[J]+K3[J])+K4[J])*JS;
      L0[J]:=L0[J]+Di[J]*DELTAT;
    end;
    if NDenom=NNum then begin
      L0[Pred(NDenom)]:=u;
      for J:=0 to Pred(NDenom)-1 do
        L0[Pred(NDenom)]:=L0[Pred(NDenom)]-L0[J]*AS[J];
    end;
    Y:=L0[0]*BS[0];
    if Pred(NNum)>0 then for J:=1 to Pred(NNum) do Y:=Y+L0[J]*BS[J];
{    WriteLn(FV,Y);}
    if PPG then KRESLI1(y,yo{,p}) else PPG:=true; yo:=y;
  end;
  until  KeyPressed;  ReadKey;
{  Close(FV);}
end;
end.
```