1

Thick Plate Toolbox I.

Ing. Petr HORA, CSc.

prosinec 2002

[1]Ústav termomechaniky AV ČR
Centrum diagnostiky materiálu
Veleslavínova 11
301 14 Plzeň

Obsah

Seznam obrázků

Kapitola 1

Úvod

Obsahem této zprávy je popis matematického modelu odezvy signálu akustické emise, který je zprogramován v matematickém prostředí MATLAB. Dále jej budeme nazývat „Thick Plate Toolbox“. Tento toolbox se skládá z následujících položek:

- výpočet disperzních křivek,
- zpřesňující výpočet křivek pro stanovení posuvů,
- manager disperzních křivek,
- výpočet simulace šíření napěťových vln v desce,
- program pro stanovení disperzních křivek z měření nebo simulace.

Kapitola 2

Výpočet disperzních křivek v tlusté desce

2.1 Rayleigh-Lambova rovnice

Předpokládejme lineární (elastický), isotropní, homogenní, nepiezoelektrický a neabsorbující materiál. Uvažujme harmonickou vlnu šířící se deskou; souřadnicový systém viz obrázek 2.1; výchylka na povrchu, $\mathbf{u}(x,t)$, může být popsána obecným analytickým výrazem (viz Brekhovskikh [Bre60]) jako,

$$\mathbf{u}\left(x,t\right)=\mathbf{A}\left(\omega\right)e^{i(\omega t-kx-\theta)}\ , \tag{2.1}$$

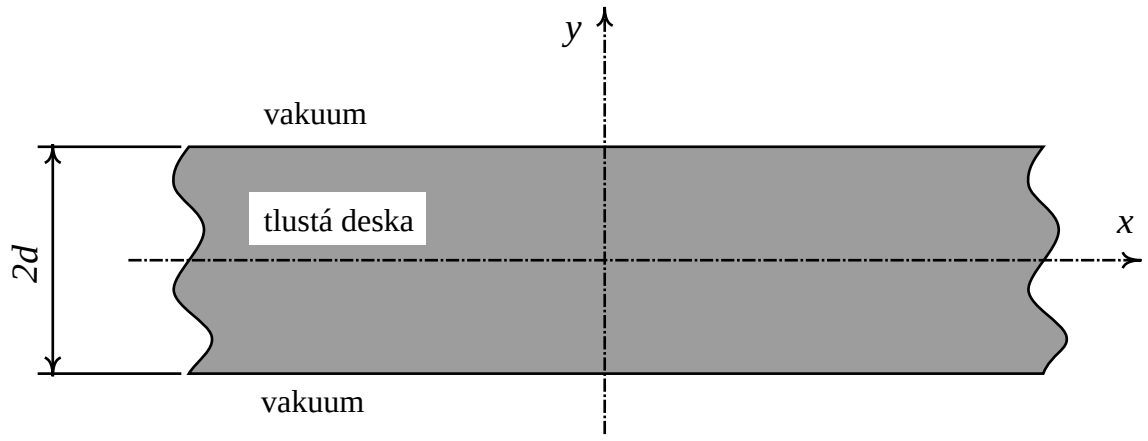
kde $\mathbf{A}(\omega)$ je frekvenčně závislá amplitudová konstanta, $\omega=2\pi f$ je úhlová frekvence, vlnové číslo $k=\omega/c$, c je fázová rychlost a θ označuje fázi.

Lambovy vlny jsou dvojrozměrné šířící se vibrace ve volných deskách. Jejich výchylky mohou být symetrické (symetrické módy) nebo antisymetrické (antisymetrické módy) vzhledem ke střední rovině desky. Rychlosti všech Lambových vln jsou disperzní a v desce tloušťky $2d$ bude pro frekvenci f existovat konečný počet módů šíření, které mohou být určeny z počtu reálných kořenů Rayleigh-Lambovy rovnice:

$$\frac{\tanh kd\sqrt{1-\left(c/c_2\right)^2}}{\tanh kd\sqrt{1-\left(c/c_1\right)^2}}-\left(4\frac{\sqrt{\left[1-\left(c/c_1\right)^2\right]\left[1-\left(c/c_2\right)^2\right]}}{\left[2-\left(c/c_2\right)^2\right]^2}\right)^{\pm1}=0\ . \tag{2.2}$$

kde znaménko + resp. - se vztahuje k symetrickým (dilatačním) resp. antisymetrickým (příčným) Lambovým vlnám a kde c_1 resp. c_2 je rychlost dilatační resp. příčné vlny.

Jedná se o transcendentní rovnice a lze je řešit jen numericky. Výpočet je dosti komplikovaný, protože hyperbolické funkce přecházejí pro některé vlnové délky ve funkce goniometrické, které



Obr. 2.1: Schématická reprezentace geometrie desky a použitý souřadnicový systém.

jsou periodické. Je-li dále v textu odkaz na R-L funkci, myslí se tím vyhodnocení levé strany Rayleigh-Lambovy rovnice.

2.2 Disperzní křivky pro dilatační vlny

Přepišme nyní Rayleigh-Lambovu rovnici pro symetrické Lambovy vlny (podélné vlny) do tvaru:

$$\left[2-\xi^2\right]^2 \cosh k_1 kd \sinh k_2 kd - 4k_1 k_2 \sinh k_1 kd \cosh k_2 kd = 0 \ , \tag{2.3}$$

kde $\xi = c/c_2$, $k_L = (c_2/c_1)^2$, $k_1 = \sqrt{1-k_L\xi^2}$ a $k_2 = \sqrt{1-\xi^2}$. Tato rovnice musí být uvedena ve třech tvarech podle hodnoty proměnné ξ. Tyto tvary jsou:

Pro $\xi \leq 1$

$$\begin{aligned} \left[2-\xi^2\right]^2 \cosh kd\sqrt{1-k_L\xi^2} \sinh kd\sqrt{1-\xi^2} \\ - 4\sqrt{1-k_L\xi^2}\sqrt{1-\xi^2} \sinh kd\sqrt{1-k_L\xi^2} \cosh kd\sqrt{1-\xi^2} \ &= \ 0. \end{aligned} \tag{2.4}$$

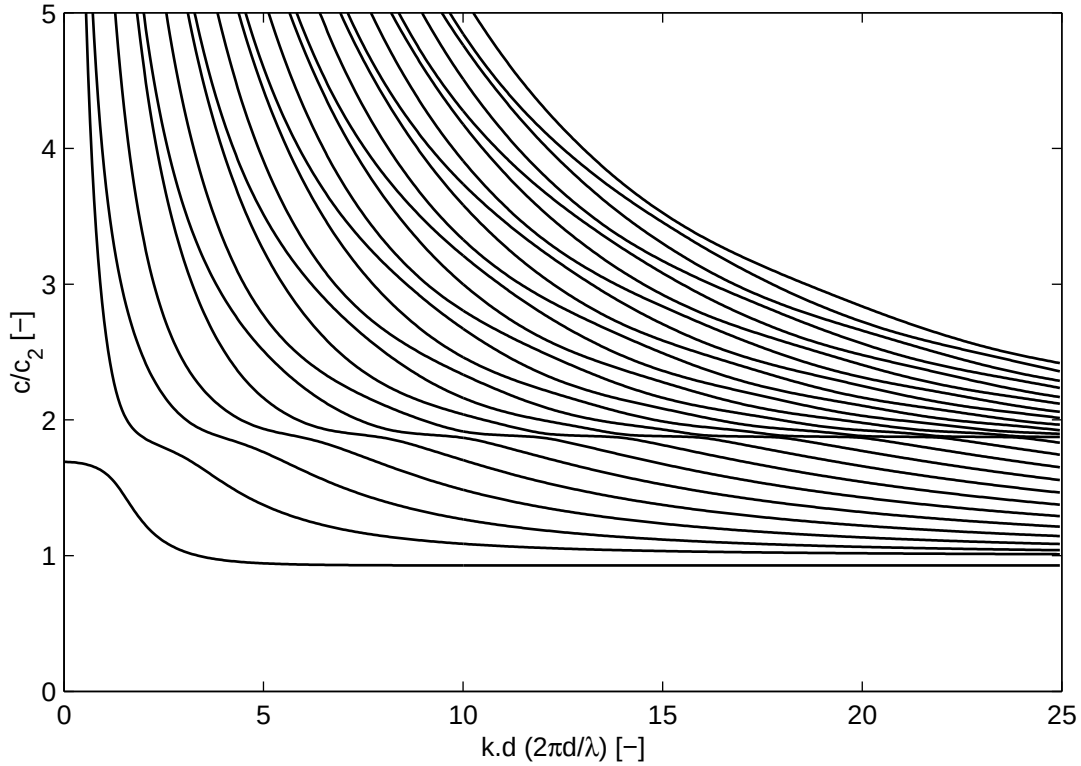
Pro $1 < \xi \leq \sqrt{1/k_L}$

$$\begin{aligned} \left[2-\xi^2\right]^2 \cosh kd\sqrt{1-k_L\xi^2} \sin kd\sqrt{\xi^2-1} \\ - 4\sqrt{1-k_L\xi^2}\sqrt{\xi^2-1} \sinh kd\sqrt{1-k_L\xi^2} \cos kd\sqrt{\xi^2-1} \ &= \ 0. \end{aligned} \tag{2.5}$$

Pro $\sqrt{1/k_L} < \xi$

$$\begin{aligned} \left[2-\xi^2\right]^2 \cos kd\sqrt{k_L\xi^2-1} \sin kd\sqrt{\xi^2-1} \\ + 4\sqrt{k_L\xi^2-1}\sqrt{\xi^2-1} \sin kd\sqrt{k_L\xi^2-1} \cos kd\sqrt{\xi^2-1} \ &= \ 0. \end{aligned} \tag{2.6}$$

Grafickou podobu prvních 25 křivek zobrazuje obr. 2.2.



Obr. 2.2: Prvních 25 disperzních křivek pro dilatační vlny pro $\mu = 0.3$, fázové rychlosti.

Analyzujme dále rovnice (2.4) až (2.6) pro hodnoty $kd \to 0$ (velmi dlouhé vlny ve srovnání s tloušťkou desky), kde budeme začínat s výpočtem.

Musíme odlišit první křivku od ostatních. Když v rovnici (2.2) nahradíme funkce tanh prvními dvěma členy Taylorova rozvoje ($\tanh z = z - \frac{1}{3}z^3$), dostaneme

$$\frac{c}{c_2} \doteq 4\left(1 - \frac{c_2}{c_1}\right)\left[1 - \frac{1}{3}\left(1 - 2\frac{c_2^2}{c_1^2}\right)\left(2\pi\frac{d}{\lambda}\right)\right].$$

To je parabolická závislost; je to vlastně Rayleighova korekce pro desku. Pro zvlášť malé hodnoty d/λ lze psát, že

$$c = c_3 = \sqrt{\frac{E}{(1-\mu^2)\,\rho}} \Rightarrow \xi = \sqrt{\frac{2}{1-\mu}},$$

což je rychlost dilatačních vln ve dvojrozměrném kontinuu (nekonečně tenká deska).

Vyšší disperzní křivky (druhou počínaje) vykazují pro klesající hodnoty kd prudký neomezený vzrůst rychlostí c/c_2. Můžeme proto v rovnici (2.2) zanedbat ve srovnání s hodnotami $(c/c_2)^2$ resp. $(c/c_1)^2$ jedničky a dvojku. Dostaneme pak

$$\frac{\tan\left[\frac{c_2}{c_1}\frac{c}{c_2}kd\right]}{\tan\left[\frac{c}{c_2}kd\right]} = \frac{-\left(\frac{c}{c_2}\right)^2}{4\frac{c_2}{c_1}}.$$

Vzhledem ke stále rostoucímu poměru c/c_2, musí buď jedna množina kořenů ležet v blízkém okolí nulových bodů funkce tangens $\tan\left[\frac{c}{c_2}kd\right]$, tedy

$$\xi = \frac{c}{c_2} = i\pi\frac{1}{kd},\ i = 1,\ 2,\ 3,\ \ldots \tag{2.7}$$

anebo musí druhá množina kořenů ležet v okolí míst nespojitosti funkce tangens $\tan\left[\frac{c_2}{c_1}\frac{c}{c_2}kd\right]$, tedy

$$\xi = \frac{c}{c_2} = (2j-1)\,\frac{\pi}{2}\frac{1}{kd}\frac{c_1}{c_2},\ j = 1,\ 2,\ 3,\ \ldots \tag{2.8}$$

Z podmínek (2.7) a (2.8) je patrné, že disperzní křivky mají pro oblast velkých ξ hyperbolický průběh v závislosti na kd a tvoří zde dva systémy křivek. Hodnoty v okolí $kd \to 0$ jsou dokumentovány v tabulce 2.1 pro $kd = 0.005$ a v tabulce 2.2 pro $kd = 0.1$. V obou případech se uvažovalo $\mu = 0.3$. Ve sloupci *Původní odhad* jsou hodnoty vypočtené dle vztahů (2.7) a (2.8), ve sloupci *Modifikovaný odhad* jsou hodnoty modifikované dle vztahů v kapitole 2.5.1; modifikovány byly pouze tučně vysázené hodnoty. Pozoruhodná je blízkost odhadů skutečným hodnotám.

2.3 Disperzní křivky pro příčné vlny

Přepišme nyní Rayleigh-Lambovu rovnici pro antisymetrické Lambovy vlny (příčné vlny) do tvaru:

$$\left[2 - k_T\eta^2\right]^2 \sinh k_3 kd \cosh k_4 kd - 4k_3k_4 \cosh k_3 kd \sinh k_4 kd = 0\ , \tag{2.9}$$

kde $\eta = c/c_1$, $k_T = (c_1/c_2)^2$, $k_3 = \sqrt{1-\eta^2}$ a $k_4 = \sqrt{1-k_T\eta^2}$. Tato rovnice musí být uvedena ve třech tvarech podle hodnoty proměnné η. Tyto tvary jsou:

Pro $\eta \le \sqrt{1/k_T}$

$$\begin{aligned}\left[2 - k_T\eta^2\right]^2 \sinh kd\sqrt{1-\eta^2} \cosh kd\sqrt{1-k_T\eta^2} \\ - 4\sqrt{1-\eta^2}\sqrt{1-k_T\eta^2} \cosh kd\sqrt{1-\eta^2} \sinh kd\sqrt{1-k_T\eta^2} \ &= \ 0.\end{aligned} \tag{2.10}$$

Pro $\sqrt{1/k_T} < \eta \le 1$

$$\begin{aligned}\left[2 - k_T\eta^2\right]^2 \sinh kd\sqrt{1-\eta^2} \cos kd\sqrt{k_T\eta^2-1} \\ + 4\sqrt{1-\eta^2}\sqrt{k_T\eta^2-1} \cosh kd\sqrt{1-\eta^2} \sin kd\sqrt{k_T\eta^2-1} \ &= \ 0.\end{aligned} \tag{2.11}$$

Křivka	Původní odhad	Modifikovaný odhad	Přesná hodnota	Rel. chyba odhadu
1.	1.690305491046	1.690305491046	1.690307215843	1.02e-06
2.	587.738167926950	*558.435774805155*	587.729890829032	-1.41e-05
3.	628.318530717959	*659.734457253857*	628.329272242020	1.71e-05
4.	1256.637061435917	1256.637061435917	1256.637399647246	2.69e-07
5.	1763.214503780850	1763.214503780850	1763.215127094155	3.54e-07
6.	1884.955592153876	1884.955592153876	1884.956214163816	3.30e-07
7.	2513.274122871835	2513.274122871835	2513.274290454122	6.67e-08
8.	2938.690839634749	2938.690839634749	2938.691377537339	1.83e-07
9.	3141.592653589793	3141.592653589793	3141.592884652897	7.35e-08
10.	3769.911184307752	3769.911184307752	3769.911294029018	2.91e-08
11.	4114.167175488649	4114.167175488649	4114.167593681469	1.02e-07
12.	4398.229715025710	4398.229715025710	4398.229852003530	3.11e-08
13.	5026.548245743669	5026.548245743669	5026.548325250091	1.58e-08
14.	5289.643511342549	5289.643511342549	5289.643849640508	6.40e-08
15.	5654.866776461628	5654.866776461628	5654.866873925895	1.72e-08
16.	6283.185307179586	6283.185307179586	6283.185366264594	9.40e-09
17.	6465.119847196449	6465.119847196449	6465.120132768677	4.42e-08
18.	6911.503837897545	6911.503837897545	6911.503913793199	1.10e-08
19.	7539.822368615503	7539.822368615503	7539.822407680047	5.18e-09
20.	7640.596183050348	7640.596183050348	7640.596436945271	3.32e-08
21.	8168.140899333462	8168.140899333462	8168.140961586977	7.62e-09
22.	8796.459430051420	*8765.043503515522*	8796.459381561843	-5.51e-09
23.	8816.072518904248	*8846.507790997504*	8816.072822027294	3.44e-08
24.	9424.777960769379	9424.777960769379	9424.778013550977	5.60e-09
25.	9991.548854758148	9991.548854758148	9991.549004693266	1.50e-08
26.	10053.096491487338	10053.096491487338	10053.096566713130	7.48e-09
27.	10681.415022205296	10681.415022205296	10681.415067966382	4.28e-09
28.	11167.025190612047	11167.025190612047	11167.025339912976	1.34e-08
29.	11309.733552923255	11309.733552923255	11309.733605469042	4.65e-09
30.	11938.052083641214	11938.052083641214	11938.052123924223	3.37e-09

Tab. 2.1: Hodnoty asymptotických odhadů, jejich modifikace a přesné hodnoty kořenů disperzních křivek pro šíření dilatačních vln v tlusté desce pro $kd = 0.005$.

Pro $1 < \eta$

$$\begin{aligned} \left[2 - k_T\eta^2\right]^2 \sin kd\sqrt{\eta^2-1}\cos kd\sqrt{k_T\eta^2-1} \\ + 4\sqrt{\eta^2-1}\sqrt{k_T\eta^2-1}\cos kd\sqrt{\eta^2-1}\sin kd\sqrt{k_T\eta^2-1} \;&=\; 0. \end{aligned} \tag{2.12}$$

Grafickou podobu prvních 25 křivek zobrazuje obr. 2.3.

Analyzujme dále rovnice (2.10) až (2.12) pro hodnoty $kd \to 0$ (velmi dlouhé vlny ve srovnání s tloušťkou desky), kde budeme začínat s výpočtem.

Musíme opět odlišit první křivku od ostatních. Když v rovnici (2.2) nahradíme funkce tanh prvními dvěma členy Taylorova rozvoje ($\tanh z = z - \frac{1}{3}z^3$), dostaneme

$$\eta = \frac{c}{c_1} \doteq kd\sqrt{\frac{2}{3}\frac{1}{1-\mu}}\frac{c_2}{c_1},$$

což je rovnice tečny v počátku; je to vlastně Kirchhoffova deska.

Křivka	Původní odhad	Modifikovaný odhad	Přesná hodnota	Rel. chyba odhadu
1.	1.689100714727	1.689100714727	1.689789315325	4.08e-04
2.	29.386908396347	*28.002018012266*	29.236774166788	-5.14e-03
3.	31.415926535898	*32.986722862693*	31.615375308963	6.31e-03
4.	62.831853071796	62.831853071796	62.838623058861	1.08e-04
5.	88.160725189042	88.160725189042	88.173180647975	1.41e-04
6.	94.247779607694	94.247779607694	94.260226472054	1.32e-04
7.	125.663706143592	125.663706143592	125.667058637322	2.67e-05
8.	146.934541981737	146.934541981737	146.945297835386	7.32e-05
9.	157.079632679490	157.079632679490	157.084255029032	2.94e-05
10.	188.495559215388	188.495559215388	188.497753975896	1.16e-05
11.	205.708358774432	205.708358774432	205.716721894743	4.07e-05
12.	219.911485751286	219.911485751286	219.914225556067	1.25e-05
13.	251.327412287183	251.327412287183	251.329002636751	6.33e-06
14.	264.482175567127	264.482175567127	264.488941139167	2.56e-05
15.	282.743338823081	282.743338823081	282.745288191354	6.89e-06
16.	314.159265358979	314.159265358979	314.160447285036	3.76e-06
17.	323.255992359822	323.255992359822	323.261703493527	1.77e-05
18.	345.575191894877	345.575191894877	345.576709844268	4.39e-06
19.	376.991118430775	376.991118430775	376.991900153822	2.07e-06
20.	382.029809152517	382.029809152517	382.034886568751	1.33e-05
21.	408.407044966673	408.407044966673	408.408290056512	3.05e-06
22.	439.822971502571	*438.2521751175776*	439.822012234719	-2.18e-06
23.	440.803625945212	*442.3253895498 75*	440.809677849715	1.37e-05
24.	471.238898038469	471.238898038469	471.239953682915	2.24e-06
25.	499.577442737907	499.577442737907	499.580441085778	6.00e-06
26.	502.654824574367	502.654824574367	502.656329421496	2.99e-06
27.	534.070751110265	534.070751110265	534.071666341294	1.71e-06
28.	558.351259530602	558.351259530602	558.354245479493	5.35e-06
29.	565.486677646163	565.486677646163	565.487728613858	1.86e-06
30.	596.902604182061	596.902604182061	596.903409850329	1.35e-06

Tab. 2.2: Hodnoty asymptotických odhadů, jejich modifikace a přesné hodnoty kořenů disperzních křivek pro šíření dilatačních vln v tlusté desce pro $kd = 0.1$.

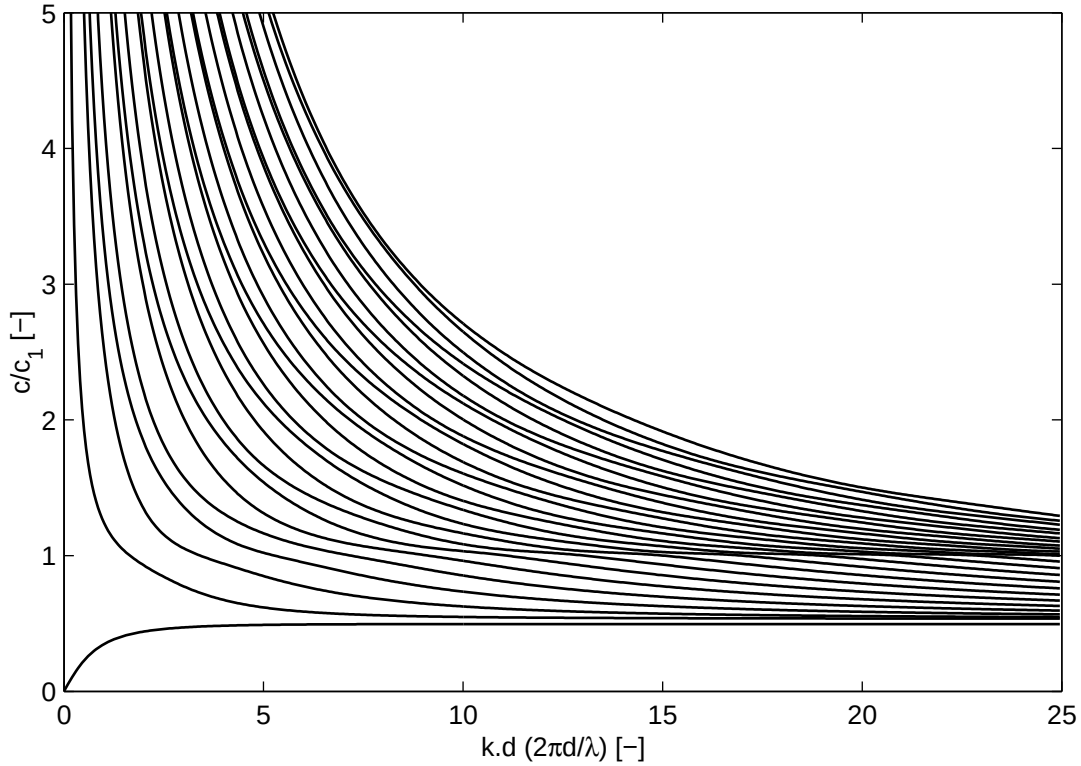
Vyšší disperzní křivky (druhou počínaje) opět vykazují pro klesající hodnoty kd prudký neomezený vzrůst rychlostí c/c_1. Můžeme proto v rovnici (2.2) zanedbat ve srovnání s hodnotami $(c/c_2)^2$ resp. $(c/c_1)^2$ jedničky a dvojku. Dostaneme pak

$$\frac{\tan\left[\frac{c}{c_2}kd\right]}{\tan\left[\frac{c_2}{c_1}\frac{c}{c_2}kd\right]} = \frac{-\left(\frac{c}{c_2}\right)^2}{4\frac{c_2}{c_1}}.$$

Vzhledem ke stále rostoucímu poměru c/c_2, musí buď jedna množina kořenů ležet v blízkém okolí nulových bodů funkce tangens $\tan\left[\frac{c_2}{c_1}\frac{c}{c_2}kd\right]$, tedy

$$\eta = \frac{c}{c_1} = i\pi\frac{1}{kd},\ i = 1,\ 2,\ 3,\ \ldots \tag{2.13}$$

anebo musí druhá množina kořenů ležet v okolí míst nespojitosti funkce tangens $\tan\left[\frac{c}{c_2}kd\right]$, tedy



Obr. 2.3: Prvních 25 disperzních křivek pro příčné vlny pro $\mu = 0.3$, fázové rychlosti.

$$\eta = \frac{c}{c_1} = (2j-1)\,\frac{\pi}{2}\frac{1}{kd}\frac{c_2}{c_1},\ j = 1,\, 2,\, 3,\, \ldots \tag{2.14}$$

Z podmínek (2.13) a (2.14) je patrné, že disperzní křivky mají pro oblast velkých η opět hyperbolický průběh v závislosti na kd a tvoří zde dva systémy křivek. Hodnoty v okolí $kd \to 0$ jsou dokumentovány v tabulce 2.3 pro $kd = 0.005$ a v tabulce 2.4 pro $kd = 0.1$. V obou případech se uvažovalo $\mu = 0.3$. Ve sloupci *Původní odhad* jsou hodnoty vypočtené dle vztahů (2.7) a (2.8), ve sloupci *Modifikovaný odhad* jsou hodnoty modifikované dle vztahů v kapitole 2.5.1; modifikovány byly pouze tučně vysázené hodnoty.

2.4 Strategie výpočtu

Výpočet disperzních křivek pro podélné resp. příčné vlny je proveden funkcí `tp_dl_roots.m` (výpis na str. 33) resp. `tp_ds_roots.m` (výpis na str. 43). Syntaxe je následující

```
tp_dl_roots(mi, kd_max, no, frequency, fname_dc, [fname_log]);
```

kde `mi` je Poissonovo číslo, `kd_max` je horní hranice kd použitá při výpočtu, `no` je počet disperzních křivek, `frequency` je parametr udávající frekvenci výpisů kd, `fname_dc` je název souboru pro uložení disperzních křivek, `fname_log` je název souboru pro uložení dosažené přesnosti a počtu iterací při výpočtu.

Program začíná u $kd = 0.005$ a pokračuje s krokem 0.005 až do `kd_max`.

Křivka	Původní odhad	Modifikovaný odhad	Přesná hodnota	Rel. chyba odhadu
1.	0.002608202655	0.002608202655	0.002608156367	-1.77e-05
2.	167.925190836271	167.925190836271	167.928118945263	1.74e-05
3.	503.775572508814	503.775572508814	503.775497703362	-1.48e-07
4.	628.318530717959	628.318530717959	628.319459456469	1.48e-06
5.	839.625954181357	839.625954181357	839.626176661646	2.65e-07
6.	1175.476335853900	1175.476335853900	1175.476347381359	9.81e-09
7.	1256.637061435917	1256.637061435917	1256.637540857493	3.82e-07
8.	1511.326717526443	1511.326717526443	1511.326820812185	6.83e-08
9.	1847.177099198985	1847.177099198985	1847.177076419549	-1.23e-08
10.	1884.955592153876	1884.955592153876	1884.955950632048	1.90e-07
11.	2183.027480871528	2183.027480871528	2183.027547416118	3.05e-08
12.	2513.274122871835	*2506.669190031828*	2513.273953168533	-6.75e-08
13.	2518.877862544071	*2525.594870177522*	2518.878286507523	1.68e-07
14.	2854.728244216614	2854.728244216614	2854.728293158427	1.71e-08
15.	3141.592653589793	3141.592653589793	3141.592787647014	4.27e-08
16.	3190.578625889157	3190.578625889157	3190.578696330291	2.21e-08
17.	3526.429007561699	3526.429007561699	3526.429046135008	1.09e-08
18.	3769.911184307752	3769.911184307752	3769.911309595985	3.32e-08
19.	3862.279389234242	3862.279389234242	3862.279435017535	1.19e-08
20.	4198.129770906785	4198.129770906785	4198.129802558911	7.54e-09
21.	4398.229715025710	4398.229715025710	4398.229826747388	2.54e-08
22.	4533.980152579327	4533.980152579327	4533.980188025994	7.82e-09
23.	4869.830534251871	4869.830534251871	4869.830560823263	5.46e-09
24.	5026.548245743669	5026.548245743669	5026.548345724314	1.99e-08
25.	5205.680915924414	5205.680915924414	5205.680945296431	5.64e-09
26.	5541.531297596956	5541.531297596956	5541.531320034073	4.05e-09
27.	5654.866776461628	5654.866776461628	5654.866867021391	1.60e-08
28.	5877.381679269499	5877.381679269499	5877.381704508239	4.29e-09
29.	6213.232060942042	6213.232060942042	6213.232079296701	2.95e-09
30.	6283.185307179586	6283.185307179586	6283.185390791452	1.33e-08

Tab. 2.3: Hodnoty asymptotických odhadů, jejich modifikace a přesné hodnoty kořenů disperzních křivek pro šíření příčných vln v tlusté desce pro $kd = 0.005$.

Např. příkaz

```
tp_dl_roots(0.3, 250, 100, 10, 'CurveL030', 'CurveL030_log');
```

vypočítává 100 disperzních křivek pro Poissonovo číslo 0.3 a pro $kd = 0.005, 0.010, \ldots 250$. Do souboru `CurveL030.m` se uloží hodnoty disperzních křivek pro $kd = 0.005, 0.055, \ldots 250$, tedy každá desátá. Do souboru `CurveL030_log.m` se uloží hodnoty dosažené přesnosti a počty iterací při výpočtu.

Program pracuje podle následující strategie:

1. začíná se odhadem kořenů z limity pro $kd \to 0$
2. z odhadů se Newtonovou metodou nalezenou kořeny (potřeba derivace),
3. od $kd = 0.4$ se odhady vypočítávají extrapolací
4. z odhadů se Newtonovou metodou nalezenou kořeny (potřeba derivace),
5. na konci kontrola splývání a křížení křivek

Křivka	Původní odhad	Modifikovaný odhad	Přesná hodnota	Rel. chyba odhadu
1.	0.052164053096	0.052164053096	0.051798781929	-7.05e-03
2.	8.396259541814	8.396259541814	8.454449418417	6.88e-03
3.	25.188778625441	25.188778625441	25.187309135648	-5.83e-05
4.	31.415926535898	31.415926535898	31.434475039853	5.90e-04
5.	41.981297709068	41.981297709068	41.985748986713	1.06e-04
6.	58.773816792695	58.773816792695	58.774052078833	4.00e-06
7.	62.831853071796	62.831853071796	62.841436057433	1.52e-04
8.	75.566335876322	75.566335876322	75.568401852954	2.73e-05
9.	92.358854959949	92.358854959949	92.358405585937	-4.87e-06
10.	94.247779607694	94.247779607694	94.254942706168	7.60e-05
11.	109.151374043576	109.151374043576	109.152705017105	1.22e-05
12.	125.663706143592	*125.333459501591*	125.660425982632	-2.61e-05
13.	125.943893127204	*126.279743508876*	125.952258377641	6.64e-05
14.	142.736412210831	142.736412210831	142.737391084629	6.86e-06
15.	157.079632679490	157.079632679490	157.082313390329	1.71e-05
16.	159.528931294458	159.528931294458	159.530340488516	8.83e-06
17.	176.321450378085	176.321450378085	176.322221866859	4.38e-06
18.	188.495559215388	188.495559215388	188.498064868329	1.33e-05
19.	193.113969461712	193.113969461712	193.114885199794	4.74e-06
20.	209.906488545339	209.906488545339	209.907121604903	3.02e-06
21.	219.911485751286	219.911485751286	219.913720132944	1.02e-05
22.	226.699007628966	226.699007628966	226.699716585513	3.13e-06
23.	243.491526712594	243.491526712594	243.492058156441	2.18e-06
24.	251.327412287183	251.327412287183	251.329411865949	7.96e-06
25.	260.284045796221	260.284045796221	260.284633246538	2.26e-06
26.	277.076564879848	277.076564879848	277.077013641567	1.62e-06
27.	282.743338823081	282.743338823081	282.745149987824	6.41e-06
28.	293.869083963475	293.869083963475	293.869588743488	1.72e-06
29.	310.661603047102	310.661603047102	310.661970175521	1.18e-06
30.	314.159265358979	314.159265358979	314.160937553446	5.32e-06

Tab. 2.4: Hodnoty asymptotických odhadů, jejich modifikace a přesné hodnoty kořenů disperzních křivek pro šíření příčných vln v tlusté desce pro $kd = 0.1$.

2.5 Problémy

2.5.1 Problémy s odhady kořenů

Jeden z největších problémů, které se při odlaďování objevily, se týkal odhadu kořenů disperzních křivek pro $kd \to 0$ podle vztahů (2.7) a (2.8) resp. (2.13) a (2.14). Pro jisté hodnoty Poissonova čísla (μ) se odhady setkají nebo leží velice blízko sebe, viz obr. 2.4. Vyšetřeme nyní podrobně tuto problematiku jak pro dilatační, tak pro příčné vlny.

Dilatační vlny

Hledáme, kdy dostaneme odhad podle vztahu (2.7) shodný s odhadem podle (2.8),

$$i\pi\frac{1}{kd} = (2j-1)\,\frac{\pi}{2}\frac{1}{kd}\frac{c_1}{c_2},\ i,\, j = 1,\, 2,\, 3,\, \ldots$$

Po úpravě dostaneme

$$2i\frac{c_2}{c_1} = 2j - 1,$$

tedy hledáme

$$2\frac{c_2}{c_1} = N,$$

kde N je libovolné přirozené liché číslo. Po vyjádření poměru c_2/c_1 pomocí Poissonova čísla a úpravě dostaneme

$$\mu = \frac{N^2 - 2}{N^2 - 4}.$$

Nyní nám zbývá najít taková přirozená lichá čísla N, která splňují podmínku pro Poissonovo číslo, tj. $\mu \subset \langle 0; 0.5)$. Tímto číslem je pouze $N = 1$, pro které vychází $\mu = 1/3$. U dilatačních vln je tedy jedinou problematickou hodnotou vzhledem k odhadu kořenů Poissonovo číslo $\mu = 1/3$ a jeho těsné okolí.

Příčné vlny

Hledáme, kdy dostaneme odhad podle vztahu (2.13) shodný s odhadem podle (2.14),

$$i\pi\frac{1}{kd} = (2j-1)\,\frac{\pi}{2}\frac{1}{kd}\frac{c_2}{c_1},\ i,\, j = 1,\, 2,\, 3,\, \ldots$$

Po úpravě dostaneme

$$2i\frac{c_1}{c_2} = 2j - 1,$$

tedy hledáme

$$2\frac{c_1}{c_2} = N,$$

kde N je libovolné přirozené liché číslo. Po vyjádření poměru c_1/c_2 pomocí Poissonova čísla a úpravě dostaneme

$$\mu = \frac{N^2 - 8}{2N^2 - 8}.$$

Nyní nám zbývá najít taková přirozená lichá čísla N, která splňují podmínku pro Poissonovo číslo, tj. $\mu \subset \langle 0; 0.5)$. Těchto čísel je nekonečné množství, neboť vyhovují všechna přirozená lichá čísla $N > 2$. Prvních deset vyhovujících čísel a jim odpovídající Poissonova čísla jsou uvedena v tab. 2.5.

Řešením je umělé oddálení (modifikace) odhadů, jak demonstrováno na obr. 2.4, kde jsou vykresleny závislosti R-L funkce na ξ pro $kd = 0.1$ a $\mu = 0.28,\ 0.33$ a 0.3333 .

2.5.2 Problémy při přechodu mezí

Někdy se vyskytují problémy při přechodu mezí 1, $1/\sqrt{k_L}$ a $1/\sqrt{k_T}$. V jednotlivých pásmech se počítá podle různých vztahů (viz rovnice (2.4) až (2.6) resp. rovnice (2.10) až (2.12)) a ty na sebe v krajních polohách nenavazují. Důsledkem je ztráta přesnosti, nelze počítat s přesností 5.10^{-16}. Řešením je zpřesnit křivky pomocí *Extended Symbolic Toolboxu MATLABu* (Maple), jak je demonstrováno v kapitole 3.

N	Poissonovo číslo (μ)
3	0.1000
5	0.4048
7	0.4556
9	0.4740
11	0.4829
13	0.4879
15	0.4910
17	0.4930
19	0.4944
21	0.4954

Tab. 2.5: Prvních deset Poissonových čísel, u nichž se vyskytují problémy s odhadem kořenů.

2.5.3 Problém při přechodu na odhad extrapolací

Pokud se přejde z výpočtu odhadu kořenů podle vztahů (2.7) a (2.8) resp. (2.13) a (2.14) na odhad pomocí extrapolace příliš brzy, dochází ke splynutí nebo dokonce křížení křivek. Když k této situaci dojde, vypočtou se odhady kořenů hrubou silou, viz vnitřní funkce `tp_dl_rawguess.m` ve funkci `tp_dl_roots.m` resp. vnitřní funkce `tp_ds_rawguess.m` ve funkci `tp_ds_roots.m`.

2.6 Výpočet grupové rychlosti

Grupová rychlost, $c_g = \partial\omega/\partial k$, může být vypočtena, jakmile je známa fázová rychlost jako funkce vlnové délky. K převodu slouží funkce `tp_dl_cg` (výpis na str. 53) resp. `tp_ds_cg` (výpis na str. 56).

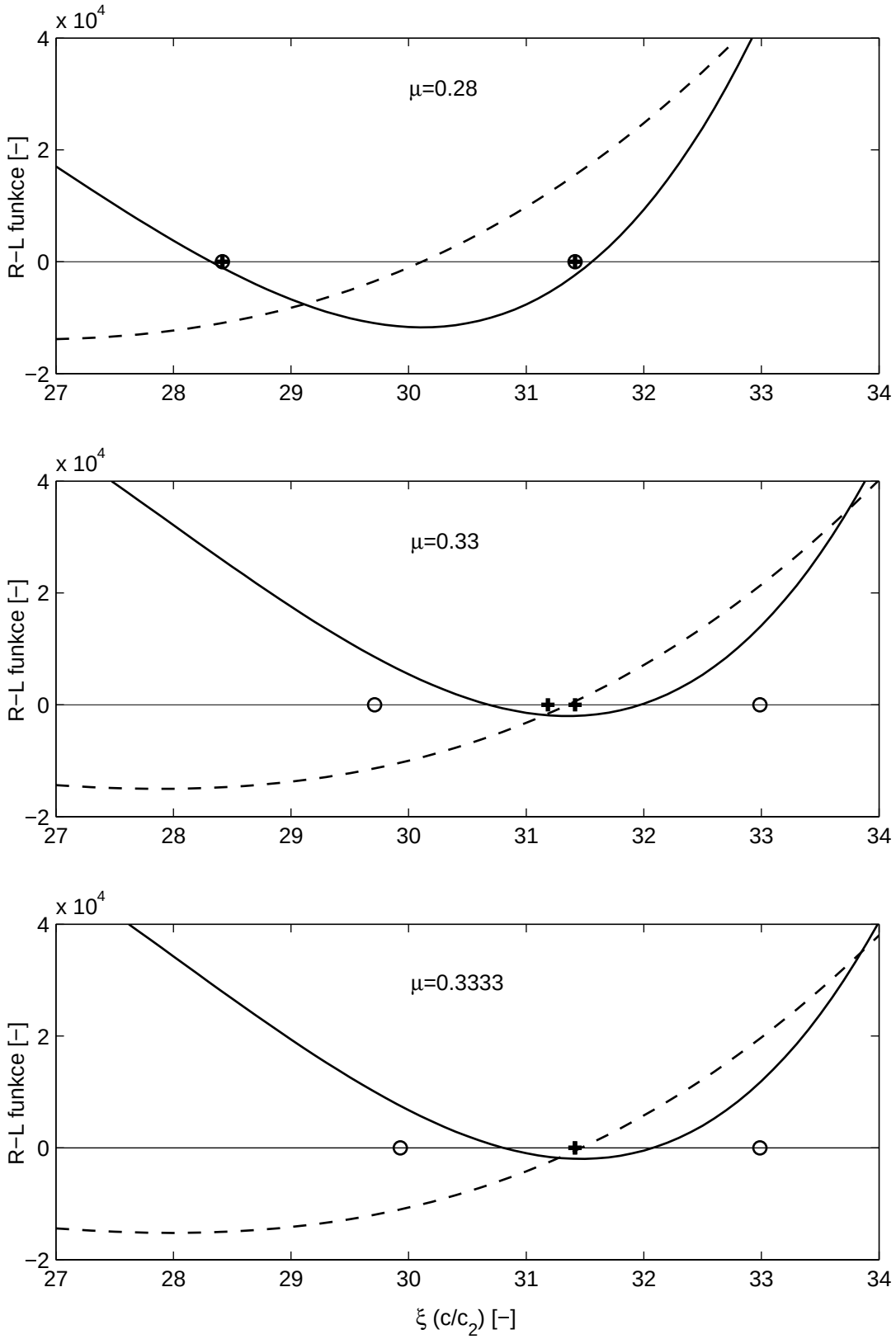
Syntaxe je následující:

```
tp_dl_cg(fname);
```

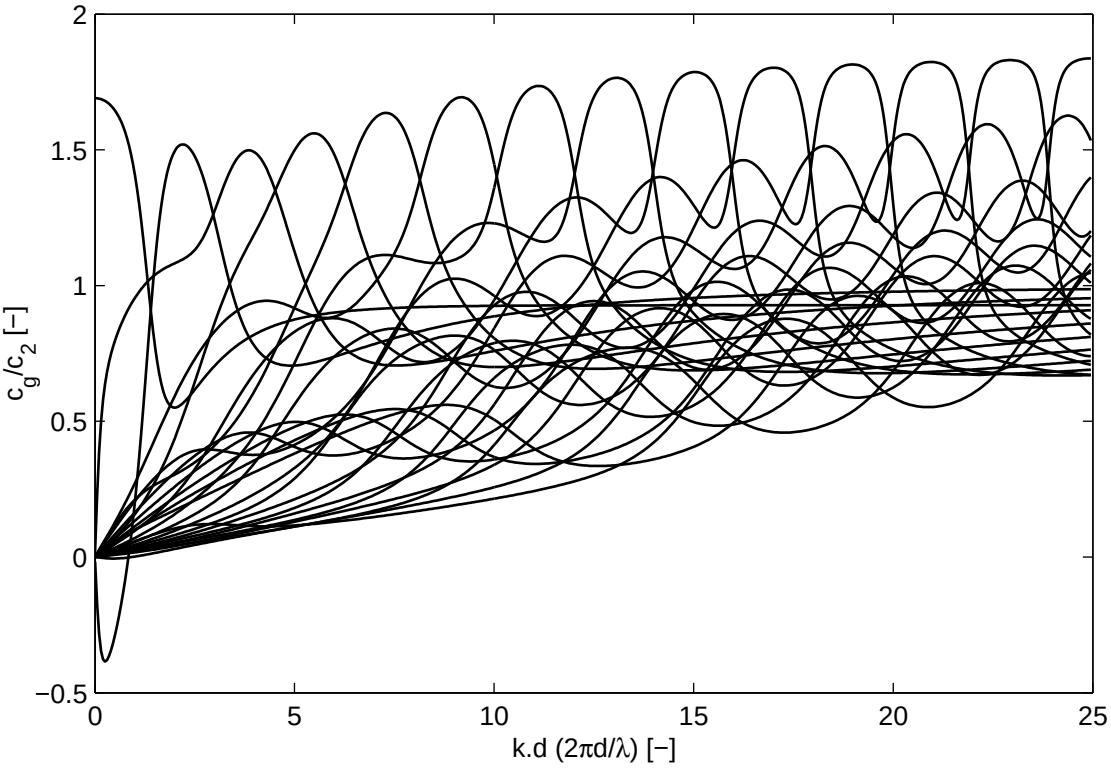
resp.

```
tp_ds_cg(fname);
```

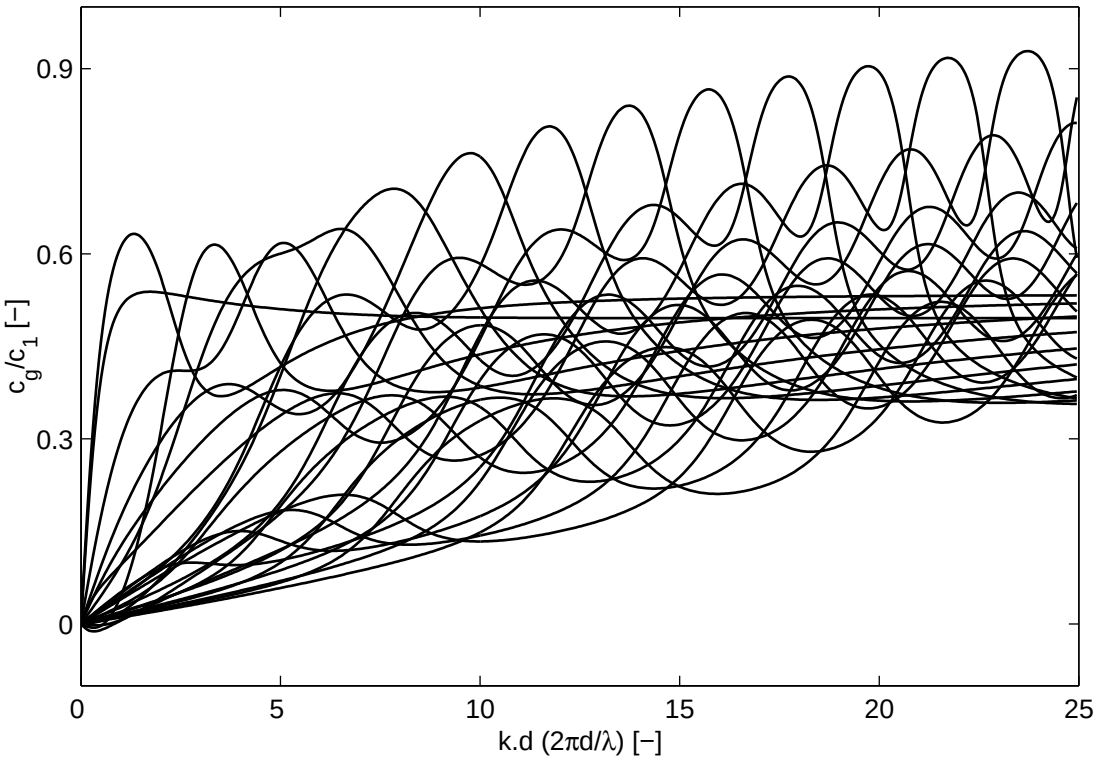
kde `fname` je název souboru s uloženými disperzními křivkami.



Obr. 2.4: Ukázka problému s odhadem kořenů. Pro $\mu = 0.28$ jsou odhady kořene velice přesné, avšak pro $\mu = 0.33$ a $\mu = 0.3333$ (jak se blížíme k hodnotě $\mu = 1/3$) se odhady blíží k sobě a znemožňují aplikovat Newtonovu metodu výpočtu kořene. Řešením je umělé oddálení (modifikace) odhadů. Legenda: plná čára - průběh Rayleigh-Lambovy funkce, čárkovaná čára - průběh derivace Rayleigh-Lambovy funkce, plus - odhad kořene, kolečko - modifikace odhadu.



Obr. 2.5: Prvních 25 disperzních křivek pro podélné vlny pro $\mu = 0.3$, grupové rychlosti.



Obr. 2.6: Prvních 25 disperzních křivek pro příčné vlny pro $\mu = 0.3$, grupové rychlosti.

Kapitola 3

Zpřesnění křivek pro posuvy

Pokud je k dispozici Extended Symbolic Toolbox MATLABu, je možno provést zpřesnění výpočtu disperzních křivek. Toto zpřesnění je zvlášť důležité, pokud chceme použít vypočtené disperzní křivky ke stanovení posuvů tlusté desky.

Pro zpřesnění dilatačních disperzních křivek slouží funkce `tp_dl_fine.m` (výpis na str. 58), pro zpřesnění příčných disperzních křivek `funkce tp_ds_fine.m` (výpis na str. 61).

Syntaxe volání je následující:

```
tp_dl_fine(fname, precision);
```

resp.

```
tp_ds_fine(fname, precision);
```

kde `fname` je název souboru s uloženými disperzními křivkami a `precision` je požadovaná přesnost.

Např.

```
tp_ds_fine('cs030', 1e-14);
```

Kapitola 4

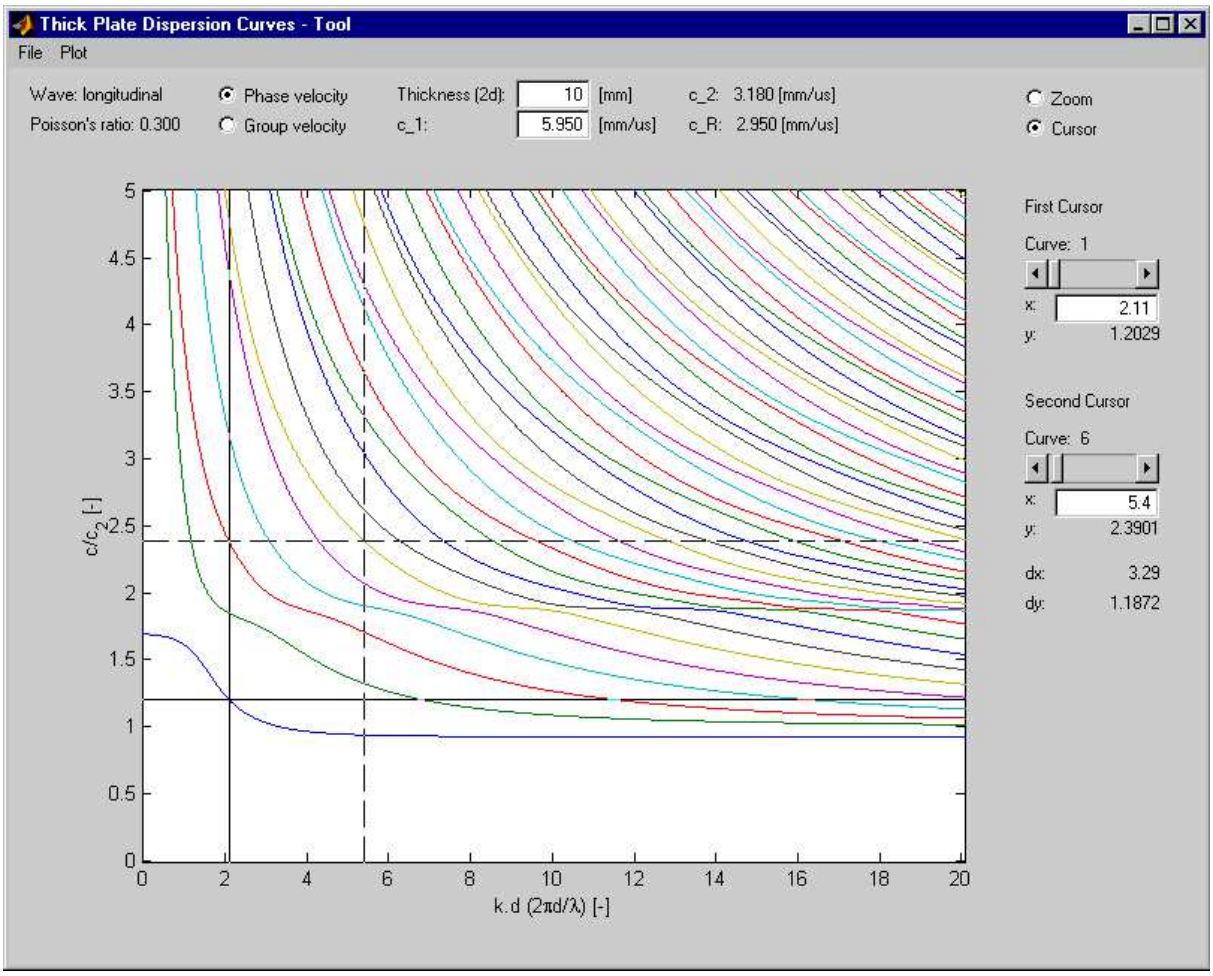
Manager disperzních křivek

Manager disperzních křivek (viz obr. 4.1) slouží k zobrazení vypočtených disperzních křivek. Křivky je možno zobrazovat v několika formátech: fázová rychlost versus vlnové číslo, fázová rychlost versus kmitočet, fázová rychlost versus součin kmitočtu a tloušťky desky, grupová rychlost versus vlnové číslo, atd.

Výpis programu `tp_dc_tool.m` je na str. 68.

Syntaxe volání je následující:

`tp_dc_tool`



Obr. 4.1: Manager disperzních křivek.

Kapitola 5

Simulace šíření napěťových vln

V této kapitole se zabýváme vlivem geometrické disperze na šíření napěťových vln zejména vzhledem k možnosti využití *guided waves* pro kontrolu konstrukcí na velké vzdálenosti. Příspěvek vychází z článku [WLC01].

Široký rozsah prací publikovaných na téma použití *guided waves* pro účely kontroly konstrukcí lze nalézt v [Chi97]. Použití *guided waves* pro účely nedestruktivního vyšetřování spadá s ohledem ke vzdálenosti šíření do dvou kategorií. První kategorie, šíření na krátké vzdálenosti, obsahuje aplikace, u nichž jsou *guided waves* používány k získání informací o testovaném vzorku. Tyto oblasti zahrnují např. určení elastických vlastností materiálu, detekce defektů v okolí rozhraní atd. V těchto aplikacích je rozhodujícím kritériem citlivost. Vzhledem k malým vzdálenostem je vliv disperze vcelku nedůležitý.

Zde se soustředíme na druhou kategorii aplikací *guided waves*, při které je naopak vzdálenost, po níž se vlny šíří, velká. Tyto aplikace zahrnují kontrolu kompozitů, potrubí a desek. V této kategorii je hlavním cílem především rychle zkontrolovat velké oblasti vzorků.

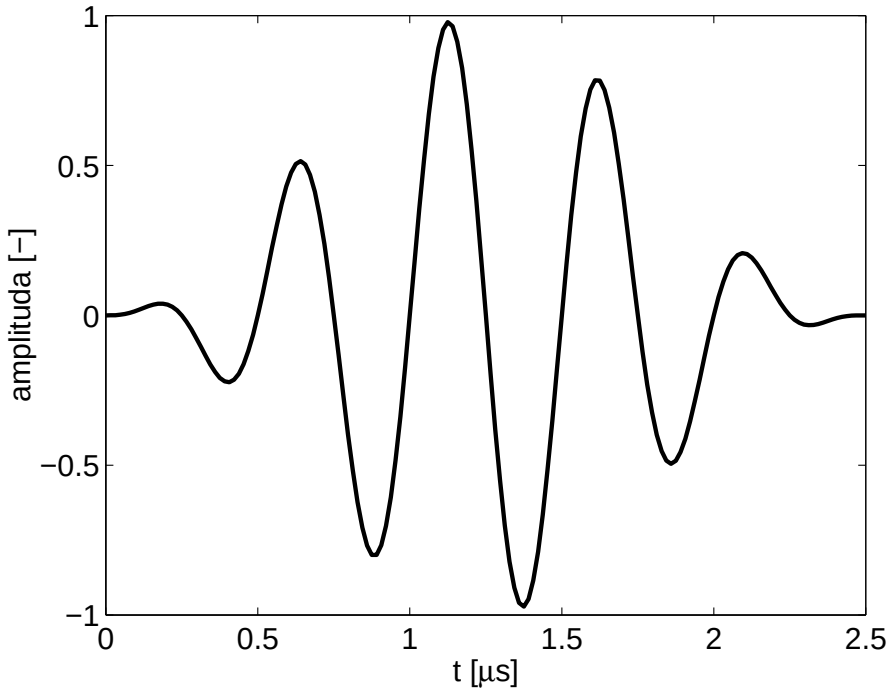
Guided waves jsou vybuzeny krátkým energeticky silným impulzem aplikovaným vhodným budičem do jednoho místa na vzorku. Vybuzení vyvolá balík *guided waves*, který se šíří od snímače do okolního prostředí. K detekci signálů vyvolaných odrazy od okolních rozhraní nebo defektů je použit buď stejný nebo druhý snímač.

Problémy spojené s využitím *guided waves* pro účely kontroly spočívají jak v existenci více módů *guided waves*, tak v disperzním chování (tj. jejich fázové rychlosti jsou závislé na kmitočtu) těchto módů. Abychom ze systému používajícím *guided waves* dostali užitečná data, je nezbytné selektivně vybudit a detekovat pouze jeden mód.

Příklady snímačů, které lze použít k vybuzení a detekci *guided waves* jsou uvedeny v [MWC97] a [AB87]. Vstupními signály jsou radioimpulzy s vhodnou okénkovou funkcí, přesnou střední frekvencí a omezenou šířkou pásma. Důvodem pro omezení šířky pásma je jednak odstranění nežádoucího vybuzení okolních módů a dále redukce vlivu disperze na šíření požadovaného módu, což je téma tohoto příspěvku.

5.1 Geometrická disperze

Disperzi lze definovat jako jev, při kterém se vlnové složky ve vlnovém balíku šíří různými fázovými rychlostmi v závislosti na frekvenci. To se projevuje jako prodlužování vlnového balíku v prostoru a čase při jeho šíření strukturou. Na obr. 5.1 je znázorněn budící vstupní signál ve tvaru radioimpulzu o 5-ti cyklech s Hannovým okénkem a střední frekvencí 2 MHz, tj. šířka pulzu je 2.5 μs.



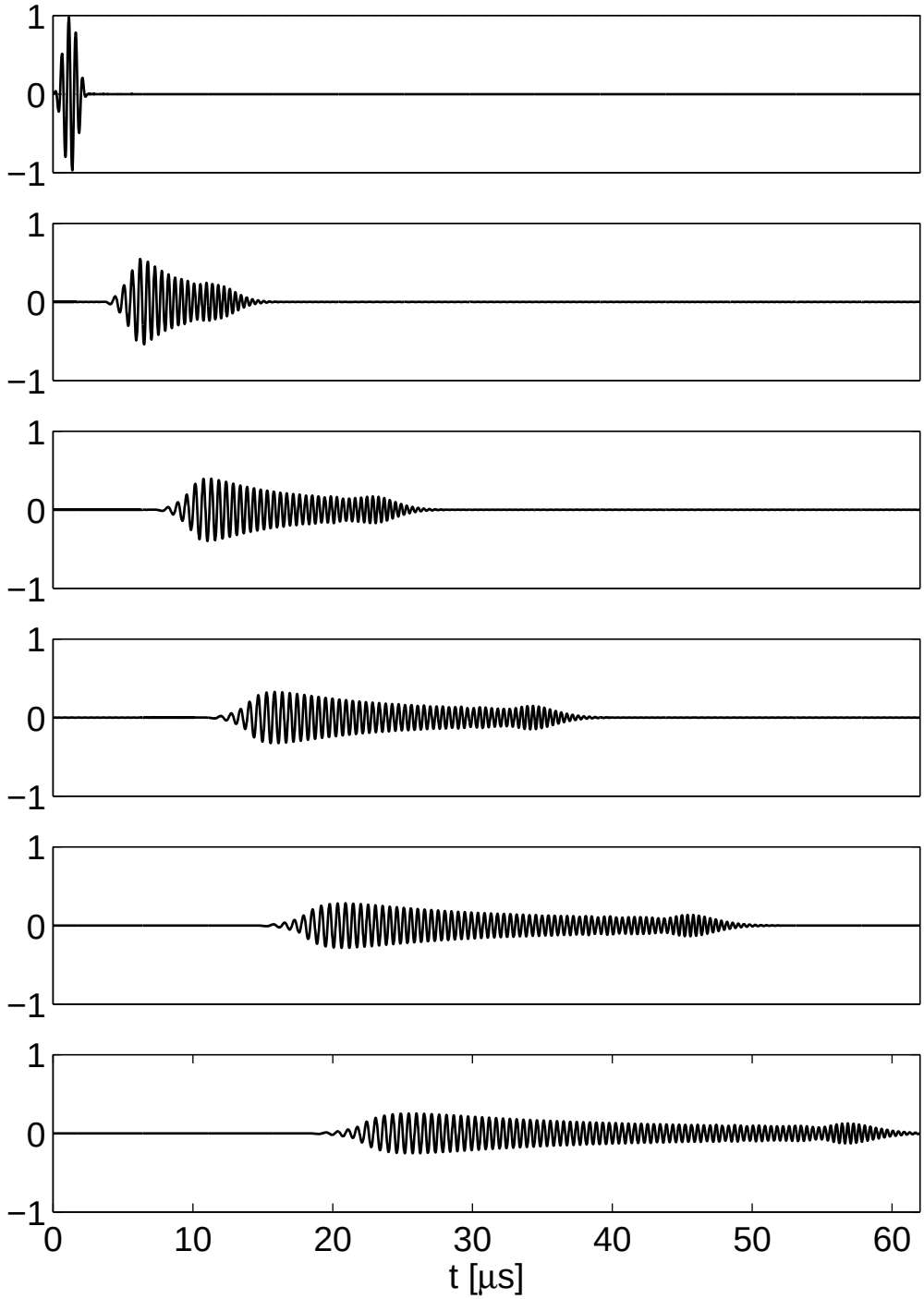
Obr. 5.1: Budící vstupní signál ve tvaru radioimpulzu o 5-ti cyklech s Hannovým okénkem a střední frekvencí 2 MHz.

Na obr. 5.2 je znázorněn jev geometrické disperze na příkladu šíření Lambova módu S0 v 1 mm silné ocelové desce při vstupním signálu dle obr. 5.1. Tuto konfiguraci budeme dále v textu nazývat vzorovým příkladem.

Jevy zvětšování časového trvání vlnového balíku a zmenšování amplitudy vyvolané disperzí jsou pro testování rozsáhlých struktur, které využívá *guided waves*, nežádoucí.

Rozšiřování vlnového balíku v prostoru a čase redukuje dosažitelné rozlišení. Tento problém se často vyskytuje při pokusu detekovat defekt v těsné blízkosti změny struktury, např. v blízkosti svaru. V takovém případě může být defekt detekován, pouze pokud lze jeho odraz spolehlivě odlišit od odrazu vyvolaného změnou struktury.

Redukce amplitudy disperzního vlnového balíku redukuje citlivost testovacího systému. Zmenšování amplitudy vlnového balíku může být odhadnuto použitím zákona o zachování energie. Na jeho základě a při zanedbání jiných ztrát lze v prvním přiblížení předpokládat, že amplituda vlnového balíku se bude zmenšovat úměrně s druhou odmocninou jeho časového trvání.



Obr. 5.2: Numerická simulace jevu geometrické disperze vzorového příkladu pro vzdálenosti 0, 20, 40, 60, 80 a 100 mm (shora dolů).

5.2 Modelování geometrické disperze

Abychom mohli dělat kvantitativní měření šíření napěťových vln, musíme být schopni modelovat, jak se vlnový balík za přítomnosti disperze šíří. Obvyklý způsob spočívá ve využití samotné definice disperzního jevu, tj. jevu, při kterém se energie ve vlnovém balíku šíří různými rychlostmi v závislosti na frekvenci.

Uvažujme případ, kdy vhodný měnič vybudí ve struktuře *guided waves*. Předpokládejme, že

je měnič ideální, tj. vybudí pouze jeden mód a to pouze v jednou směru. Měnič je napájen elektrickým signálem délky $V(t)$, který se mění na akustickou energii. Tato energie se šíří od měniče ve směru osy x jako jednoduchý mód *guided wave*. Předpokládá se, že v místě měniče, $x = 0$, je změna parametru v desce, např. vertikální výchylka u vzhledem k času t, přímo úměrná $V(t)$.

Funkci $u(t)$ lze považovat za řez v časo-prostorové rovině funkce $u(x,t)$, který prochází bodem $x = 0$. Pokud známe disperzní křivky (závislost fázové rychlosti na frekvenci) pro daný systém, pak může být vypočteno $u(x,t)$ v libovolném bodě časo-prostorové roviny dle následujícího algoritmu.

Nejprve provedeme převod do frekvenční oblasti

$$U(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} u(t)\, e^{-i\omega t}\, \mathrm{d}t, \tag{5.1}$$

kde ω je úhlová frekvence. Hodnota $u(x,t)$ odpovídající individuální spektrální složce $U(\omega)$ je dána řešením vlnové rovnice ve frekvenční oblasti

$$U(\omega)\, e^{i(k(\omega)x - \omega t)}, \tag{5.2}$$

kde $k(\omega)$ je vlnové číslo, které lze získat z fázové rychlosti $c_f(\omega)$ vztahem

$$k(\omega) = \frac{\omega}{c_f(\omega)}. \tag{5.3}$$

Hodnota $u(x,t)$ je dána integrací příspěvků všech frekvenčních složek $U(\omega)$

$$u(x,t) = \int_{-\infty}^{\infty} U(\omega)\, e^{i(k(\omega)x - \omega t)}\, \mathrm{d}\omega. \tag{5.4}$$

K vyhodnocení tohoto integrálu lze s výhodou použít FFT.

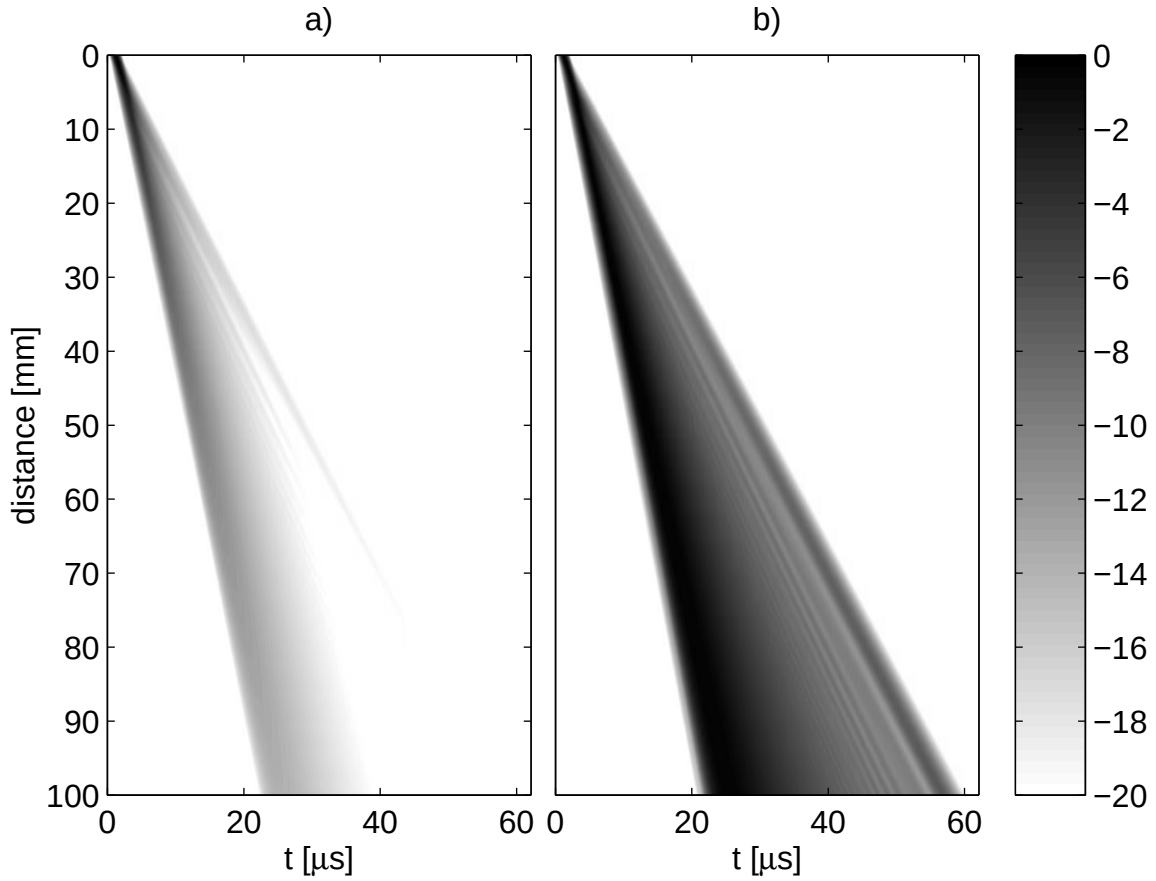
Pro účel predikce hranic vlnového balíku je vhodnější pracovat s obálkou vlnového balíku, kterou získáme pomocí Hilbertovy transformace.

5.3 Definování trvání vlnového balíku

Abychom dovedli kvantifikovat disperzi, je nezbytné umět změřit trvání vlnového balíku, což znamená vědět, kde vlnový balík začíná a kde končí.

Jednou z možností je definovat začátek a konec vlnového balíku pomocí časo-prostorové mapy $u(x,t)$, ve které si definujeme začátek a konec body, ve kterých obálka klesne pod danou referenční úroveň. Na obr. 5.3 jsou znázorněny dva případy volby referenční úrovně pro náš vzorový příklad: a) za referenční úroveň se volí globální maximum obálky (špičková hodnota

obálky zdrojového signálu), b) pro každou vzdálenost se volí nová reference. Druhá možnost je pro účely sledování disperzního chování vhodnější.



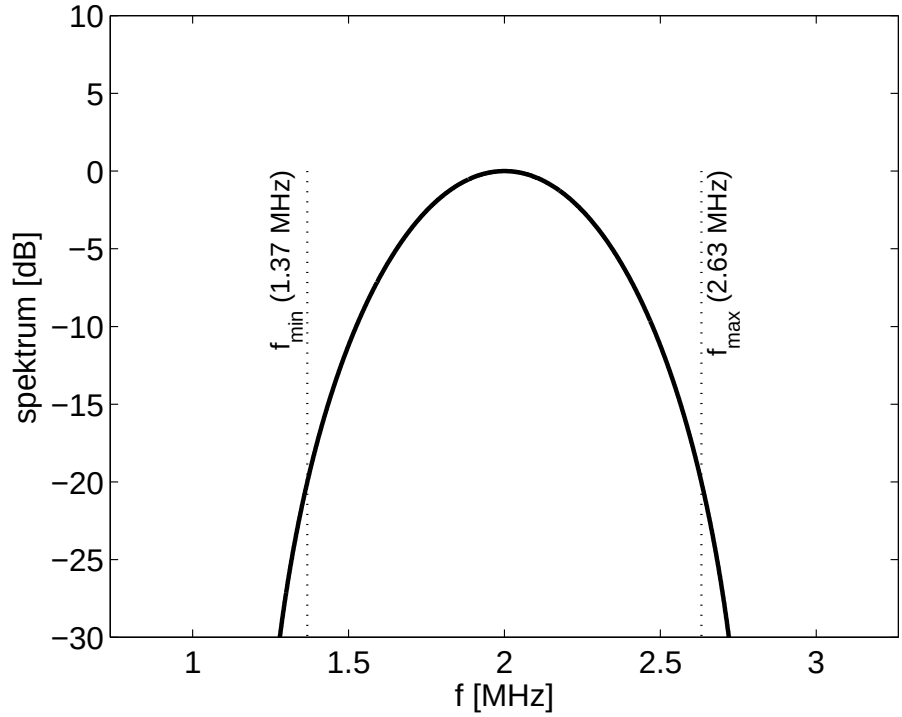
Obr. 5.3: Dvě různé definice referenční úrovně: a) reference (0 dB) je špičková hodnota obálky pro distanci rovnou nule, b) reference je přepočítána pro každou vzdálenost jako špičková hodnota obálky pro danou vzdálenost.

Procedura Fourierovy dekompozice je pro získání trvání vlnového balíku velice neefektivní i přes použití FFT, proto uvedeme jinou možnost, která se opírá o znalost frekvenční závislosti grupových rychlostí. Mějme náš vzorový příklad. Šířku pásma vstupního signálu získáme nalezením frekvencí, ve kterých amplituda spektra padá o 20 dB, viz obr. 5.4.

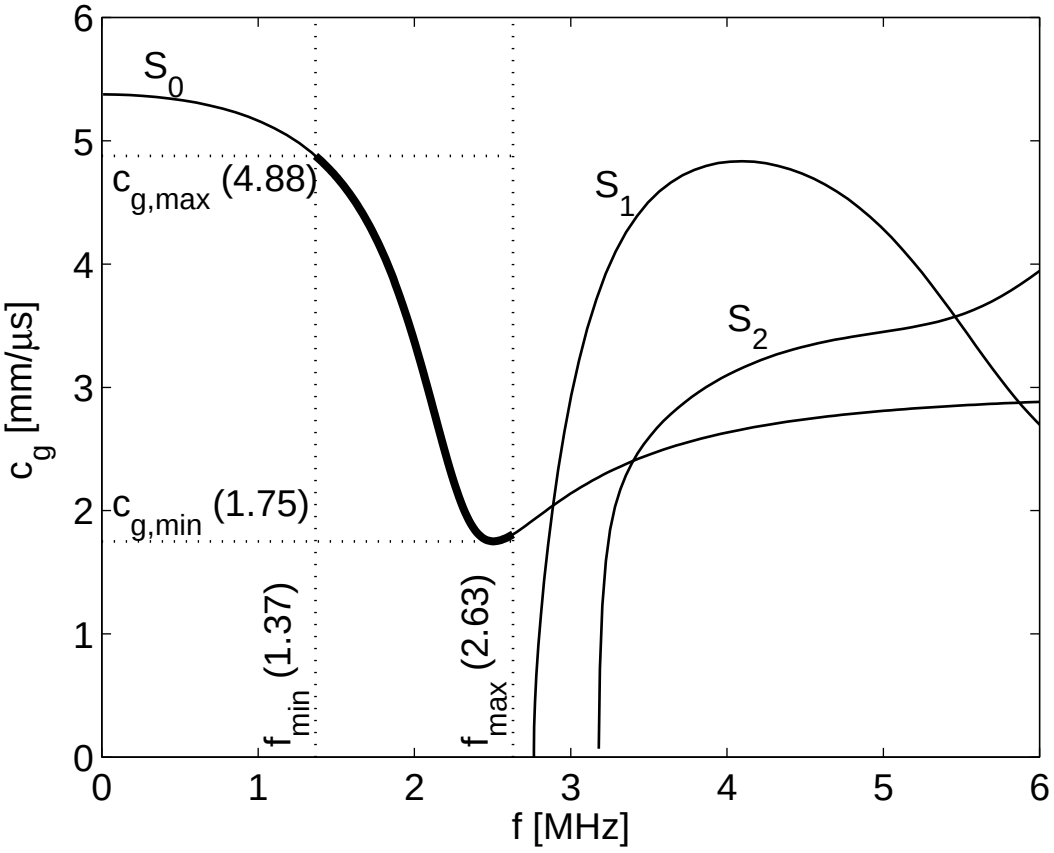
Frekvenční závislosti grupových rychlostí pro náš vzorový příklad jsou znázorněny na obr. 5.5. Vertikální čáry v tomto obrázku vyznačují šířku pásma dle obr. 5.4 a vodorovné čáry vyznačují minimální a maximální grupovou rychlost nacházející se v daném frekvenčním pásmu, tj. $c_{g,min} = 1.75$ mm/μs a $c_{g,max} = 4.88$ mm/μs.

Na obr. 5.6 je porovnání metody Fourierovy dekompozice a metody grupové rychlosti (čárkovaná čára) pro náš vzorový příklad.

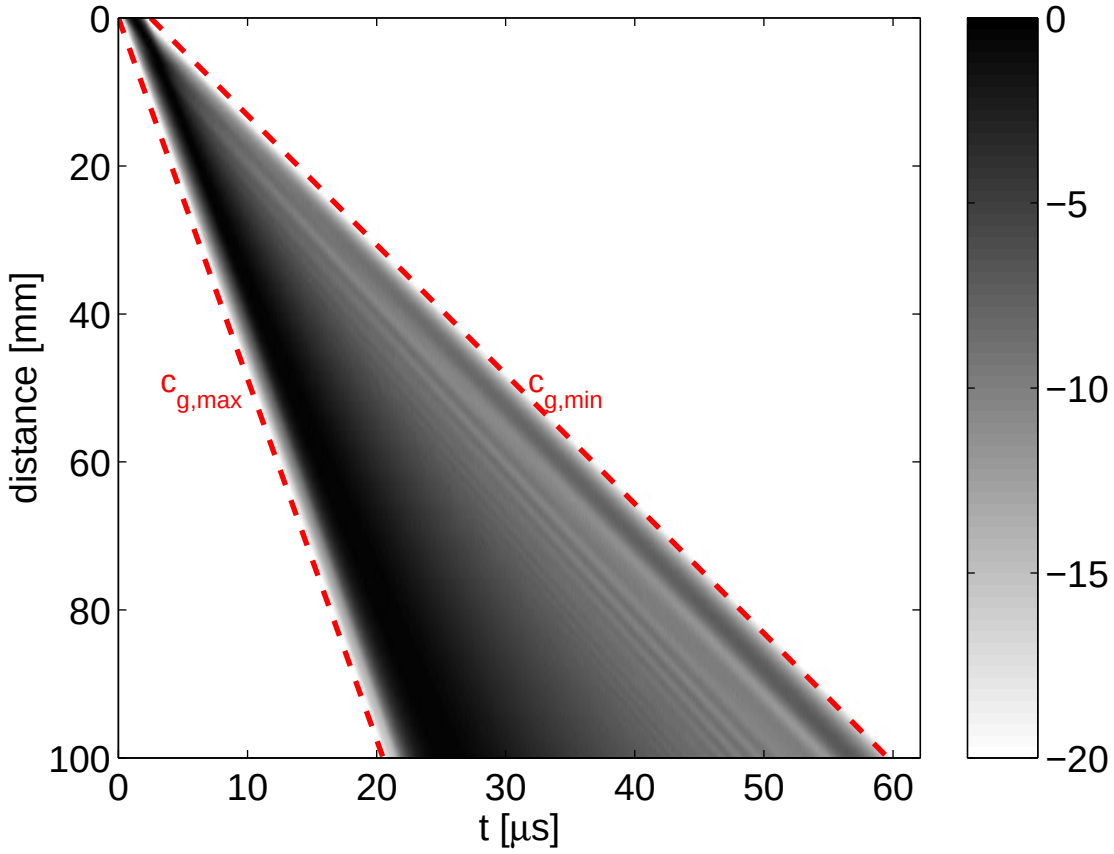
Výpis programu je uveden na str. 90.



Obr. 5.4: Spektrum vstupního signálu s vyznačením -20 dB frekvenčního rozsahu.



Obr. 5.5: Frekvenční závislosti grupových rychlostí pro 1 mm silnou ocelovou desku ve vakuu. Vertikální čáry vyznačují šířku pásma dle obr. 5.4. Vodorovné čáry vyznačují minimální a maximální grupovou rychlost nacházející se v daném frekvenčním pásmu.



Obr. 5.6: Porovnání dvou metod predikce trvání vlnového balíku: metoda Fourierovy dekompozice a metoda grupové rychlosti.

Kapitola 6

2D-FFT pro stanovení křivek z měření nebo simulace

Aplikace tradičních ultrazvukových metod, např. pulsní echo, jsou omezeny na testování relativně jednoduchých geometrií nebo podrobně zkoumají pouze oblast v bezprostředním okolí snímače. Nové ultrazvukové metody využívají pro vyšetřování konstrukčních prvků tzv. vlnovodné vlny (*guided waves*). Výhody těchto metod spočívají ve schopnosti otestovat celý konstrukční prvek jediným měřením a ve schopnosti testovat i nepřístupné oblasti složitých komponent.

Šíření *guided waves* ve složitých strukturách je komplikovaný proces, který se obtížně popisuje a interpretuje. Neustále se vyvíjí prostředky pro modelování tohoto šíření.

Jeden z přístupů k modelování šíření *guided waves* spočívá v analytickém řešení diferenciálních pohybových rovnic s příslušnými okrajovými a počátečními podmínkami. Tento postup byl již aplikován na řadě jednoduchých geometrií, s uvažováním homogenního a izotropního materiálu (viz [Gra75] a [Mik78]). Tyto rovnice se však pro komplikovanější geometrie nebo nehomogenní materiály stávají neřešitelnými.

Jiný přístup k této problematice zahrnuje numerická řešení. Existují tři hlavní numerické metody, které mohou být pro tento problém použity: metoda konečných diferencí (MKD), metoda konečných prvků (MKP) nebo metoda hraničních prvků (MHP). MKD byla první numerická metoda, která byla aplikována na studium šíření napěťových vln. MHP je výhodná v tom, že potřebuje diskretizovat pouze povrch zkoumaného vzorku; numerický problém se tím o jednu dimenzi redukuje. Naopak primární výhoda MKP spočívá v dostupnosti řady komerčních MKP programů, tedy odpadá potřeba vývoje vlastního programového kódu.

Cílem tohoto výzkumu je porovnat známé analytické řešení problému šíření *guided waves* v tlusté desce s řešením získaným numericky.

6.1 Numerické modelování metodou konečných prvků

Časové a prostorové rozlišení konečně-prvkového modelu je kritické pro konvergenci numerického řešení. Volba odpovídajícího integračního časového kroku, Δt, je velice důležitá pro přesnost řešení. Obecně, zmenšováním integračních časových kroků můžeme model zpřesňovat. S časovými kroky, která jsou příliš dlouhé, nejsou vysokofrekvenční složky dostatečně přesně rozlišeny. Naopak, příliš malé časové kroky jsou plýtváním výpočetního času. Nezbývá tedy nic jiného, než nalézti nějaký kompromis. Podle našich zkušeností je dostatečné volit 20 bodů na periodu nejvyšší frekvenční složky. Toto pravidlo lze vyjádřit vztahem:

$$\Delta t = \frac{1}{20 f_{\max}} \ , \tag{6.1}$$

kde $f_{\max}$ je nejvyšší frekvence, která nás zajímá. Určením nejvyšší frekvence vln šířících se strukturou a použitím vztahu (6.1) dostaneme časový krok,Δt, který je dostatečně malý pro modelování časového chování šíření *guided waves*. Pokud se vstupní funkce blíží skokové funkci, nemusí poměr daný vztahem (6.1) zajistit dostatečné časové rozlišení. V některých případech musí být tento poměr zvýšen až na desetinásobek. Potřebný časový krok může být také odvozen z času, který potřebuje nejrychlejší vlna na překonání vzdálenosti dvou nejbližších bodů sítě.

Velikost prvků se volí takovým způsobem, aby se zachovalo prostorové rozlišení šířících se vln. Ze zkušenosti lze říci, že je potřeba uvažovat alespoň 20 bodů na nejkratší vlnovou délku. Toto pravidlo lze vyjádřit vztahem:

$$l_{\mathrm{e}} = \frac{\lambda_{\min}}{20} \ , \tag{6.2}$$

kde l_{e} je délka prvku a $\lambda_{\min}$ je nejkratší vlnová délka, která nás zajímá. Pokud jsou zapotřebí vysoce přesné numerické výsledky, nemusí být vztah (6.2) dostatečný a je třeba uvažovat vyšší úroveň diskretizace.

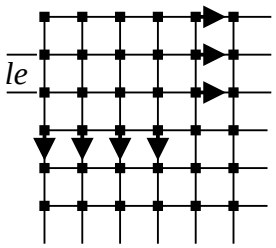
Ze vztahů (6.1) a (6.2) vyplývá, že problémy vysokofrekvenčního vlnového šíření vyžadují enormní výpočetní zdroje. Výpočty těchto problémů vedou na vysoké hodnoty $f_{\max}$ a malé hodnoty $\lambda_{\min}$, což znamená velice hustou síť a velmi malý integrační časový krok.

Abychom pochopili chování MKP aplikované na řešení problému *guided waves*, uvažovali jsme relativně jednoduchou geometrii: 2 mm silnou a 100 mm dlouhou ocelovou desku, jaká byla použita v [MJQ99]. Pro tuto geometrii existuje známé analytické řešení (Rayleigh-Lambova rovnice, viz následující kapitola). MKP program používaný pro tuto práci byl MARC ver. K7.3.2 s pre- a post-procesorem MENTAT ver. 3.2.0, který byl instalován na pracovní stanici SGI OCTANE (procesor R 10000, 195 MHz, 256 MB RAM, 4+9 GB HD). Geometrické a materiálové vlastnosti konečně-prvkového modelu jsou uvedeny v tabulce 6.1. Horní levý roh desky, která je modelována čtvercovými prvky ($l_{\mathrm{e}} = 0.1$ mm), je zatížen výchylkou v x-ovém a y-ovém směru. Obrázek 6.1 znázorňuje aplikované výchylky v různých uzlech horního levého rohu desky. Časový průběh těchto výchylek je trojúhelníkový puls se šířkou 0.2 μs. Způsob zatížení nemá žádný

praktický význam, jde jen o to, aby se vybudily vysokofrekvenční vlny. Cílem tohoto modelu je ukázat disperzní jevy až do frekvence, f, 5 MHz. Podle výše uvedených doporučení je tento transientní problém řešen s integračním časovým krokem, $\Delta t = 10^{-8}$ s. Při MKP řešení bylo použito metody centrálních diferencí.

Geometrické vlastnosti			**Materiálové vlastnosti**		
Šířka	2	[mm]	Youngův modul	2.10^{11}	[Pa]
Délka	100	[mm]	Poissonovo číslo	0.29	[-]
Typ prvku	4-uzlový		Hustota	7850	[kg/m^3]
Velikost prvku	0.1	[mm]	Rychlost dilatační vlny	5778	[m/s]
Počet prvků	20000	[-]	Rychlost smykové vlny	3142	[m/s]
Počet uzlů	21021	[-]	Rychlost Rayleighovy vlny	2909	[m/s]

Tab. 6.1: MKP model a materiálové vlastnosti



Obr. 6.1: Aplikované zatížení.

6.2 Dvojrozměrná spektrální metoda

Klíčový problém týkající se kvantitativního měření charakteristik šíření Lambových vln spočívá ve skutečnosti, že pro libovolnou frekvenci může existovat více módů šíření. Dvojrozměrná FFT metoda popsaná v [ACl91] je rozšířením jednorozměrné spektrální metody vyvinuté Sachsem a Paoem [SP78] pro měření rychlosti napěťových vln.

Šířící se Lambovy vlny jsou harmonické jak ve frekvenční, tak v prostorové oblasti, jak lze nahlédnout ze vztahu (2.1). Provedením časové Fourierovy transformace přejdeme z časové do frekvenční oblasti. Následným provedením prostorové Fourierovy transformace přejdeme do oblasti frekvence-vlnové číslo, kde lze měřit amplitudy a vlnová čísla jednotlivých módů.

Dvojrozměrná Fourierova transformace vztahu (2.1) je dána vztahem

$$H(k,f) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} u(x,t)\, e^{-i(kx+\omega t)} dx dt \ . \tag{6.3}$$

Jelikož vypočtené i naměřené výchylky budou dány v diskrétních bodech, použijeme diskrétní Fourierovu transformaci. Diskrétní dvojrozměrná Fourierova transformace může být definována podobným způsobem jako jednorozměrná DFT. Výsledkem této transformace bude dvojrozměrné pole amplitud v diskrétních frekvencích a vlnových číslech. Jako v jednorozměrném

případě musí být odstraněn *aliasing* vzorkováním dat dostatečně vysokou frekvencí v časové oblasti i v oblasti vlnových čísel. Jelikož signály nebudou obvykle periodické vzhledem k časovému a prostorovému vzorkovacímu okénku, objeví se rozptyly, které mohou být redukovány okénkovými funkcemi, např. Hannovo okénko. K přesnějšímu určení frekvence a vlnového čísla maximálních amplitud mohou být za konce signálů doplněny nuly.

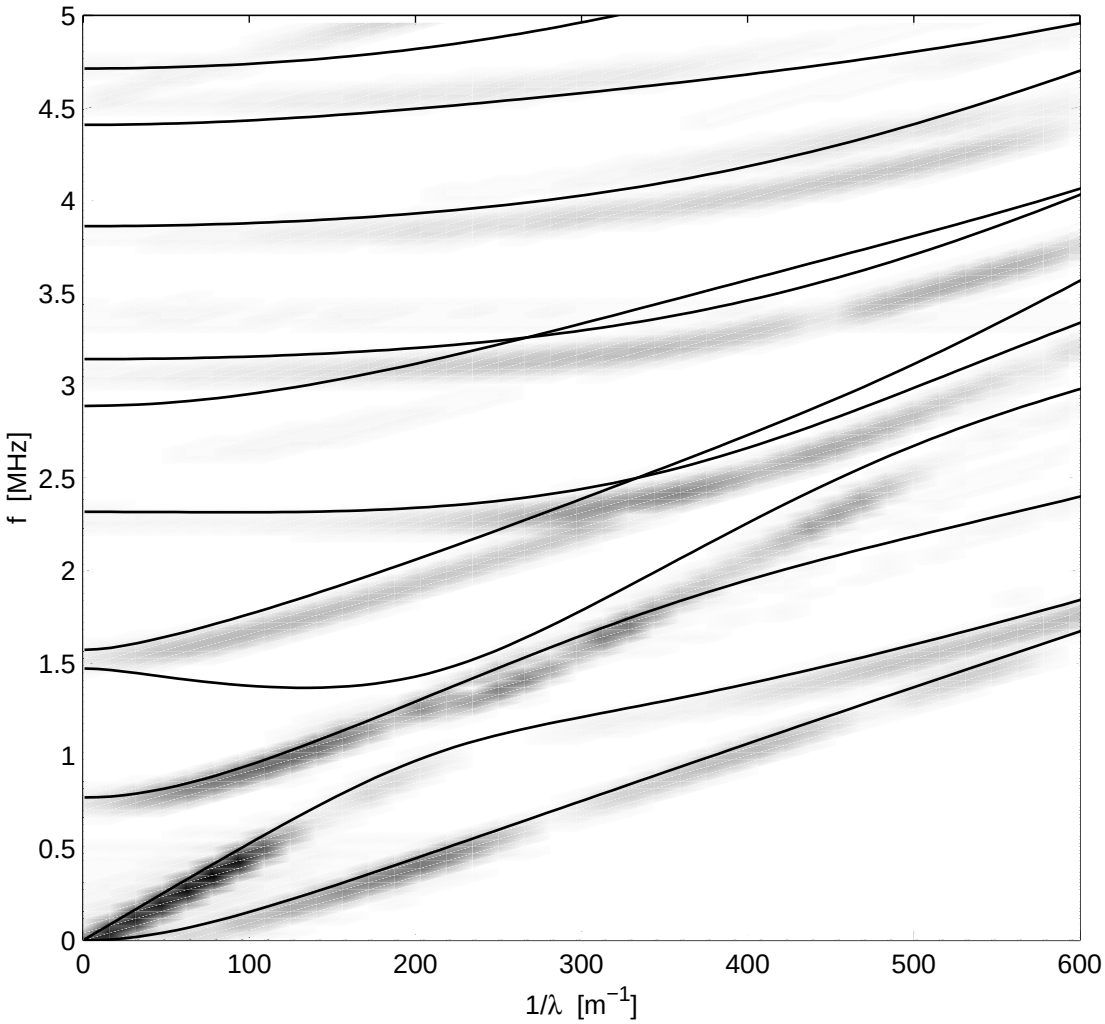
Algoritmus:

1. Vytvoří se pole (sloupcově) z experimentálně nebo numericky získaných časových historií výchylek sejmutých z řady ekvidistantně rozmístěných bodů podél cesty šíření.
2. Provede se časová Fourierova transformace každého sloupce pole, čímž se získá frekvenční spektrum pro každou časovou historii.
3. Provede se prostorová Fourierova transformace každého řádku (nyní sestaveného ze složek stejné frekvence), čímž se získá informace o amplitudách nad oblastí frekvence-vlnové číslo.

Pro demonstraci 2D-FFT budeme uvažovat pouze x-ové výchylky v uzlech na horním povrchu desky. Aby se odstranil *aliasing* pro uvažovaný rozsah vlnových čísel a frekvencí, musí být časová i prostorová vzorkovací rychlost pro 2D-FFT vybrána dosti vysoká. Poněvadž uvažovaná horní frekvenční mez je 5 MHz, je použita vzorkovací rychlost $\Delta T = 10^{-7}$ s. Z pohledu MKP vedou prvky délky $l_\mathrm{e} = 0.1$ mm k přesným výsledkům pro $\lambda > \lambda_\mathrm{min} = 2$ mm [viz vztah (6.2)]. Z toho plyne maximální hodnota pro $1/\lambda = 500$ m^{-1}. Proto je použit prostorový vzorkovací krok $\Delta x = 0.5$ mm. To znamená, že pro 2D-FFT jsou zapotřebí pouze řešení z každého pátého bodu na povrchu desky a to pro každý desátý časový krok. Abychom dostali signál bez odrazů od pravého konce desky, ořízneme časový signál Hannovo okénkem o šířce 17 μs, což odpovídá času, který potřebuje dilatační vlna k dostižení pravého okraje desky. Řešení pro časy větší než 17 μs jsou ignorována a nahrazena nulami pro zvýšení frekvenčního rozlišení časové FFT.

Obrázek 6.2 znázorňuje pseudobarevný graf amplitud nad oblastí frekvence-vlnové číslo. Z obrázku je zřejmé, že významnější amplitudy jsou pouze pro jisté kombinace vlnového čísla a frekvence; tyto hodnoty jsou řešením vztahu (2.2). Přesná řešení vztahu (2.2) jsou do obrázku zakreslena plnými čarami. Povšimněte si, že existují také nějaké rušivé špičky vyvolané chybami vzorkování a chybami numerickými.

Pro tento MKP model je doporučeného poměru $\lambda/l_\mathrm{e} = 20$ dosaženo pro $1/\lambda = 500$ m^{-1}. Skutečnost, že existuje dobrá shoda i pro vyšší hodnoty $1/\lambda$, vede k závěru, že toto omezení vlnové délky není tak kritické. Avšak poměr mezi integračním časovým krokem Δt a frekvencí f_max je mnohem kritičtější; numerická řešení se zhoršují, jak se poměr $1/(\Delta t\ f_\mathrm{max})$ blíží k doporučené hodnotě 20. Závěrem lze konstatovat, že tento model desky prokázal použitelnost komerčního MKP systému pro modelování disperzní povahy *guided waves*.



Obr. 6.2: Pseudobarevný graf spektra $1/\lambda - f$.

Příloha A

tp_dl_roots.m

```
function tp_dl_roots(mi, kd_max, no, frequency, fname);

%
% TP - Thick Plate toolbox
% DL - Dispersion for Longitudinal waves
% Dispersion curve roots finding
%
% Parameters:  mi     ... Poisson's ratio (scalar)
%              kd_max ... max. k.d (scalar)
%              no     ... number of dispersion curves (scalar)
%              frequency ... save frequency of k.d (scalar)
%              fname  ... name of file for the dispersion curves (string)
%
%              DC     ... dispersion curves (matrix),
%                         rows    ... k.d,
%                         columns ... curves
%
% Syntax:      tp_dl_roots(mi, kd_max, no, frequency, fname);
%
% Example:     tp_dl_roots(0.3, 250, 100, 10, 'CurveL030');
%

% (c) 2001, Petr Hora



kl=(1-2*mi)/(2*(1-mi));

KD_COMP=[0.0025:0.0025:kd_max];
KD=[0.0025:0.0025*frequency:kd_max];

DC_tmp=zeros([3, no]);

tmpfname_dc=tempname;
fidDC = fopen(tmpfname_dc, 'w');
```

```
tmpfname_log=tempname;
fidLOG = fopen(tmpfname_log, 'w');


h = waitbar(0,'Please wait...');
n=length(KD_COMP);

wrong=0;
i=0; j=0;
for kd=KD_COMP,
   i=i+1;

   if kd<0.4,  % very sensitive value (0.4) !!!
      guess=tp_dl_guess(kd,mi,no);
      [result,delta,iter]=tp_dl_precising(no,guess,kd,kl);
   else
      guess=extrapol_quad(DC_tmp);
      [result,delta,iter]=tp_dl_precising(no,guess,kd,kl);
   end

   if ~all(diff(result)>0),
      fprintf('Raw guess for kd = %g\n', kd);
      guess=tp_dl_rawguess(kd,mi,no);
      [result,delta,iter]=tp_dl_precising(no,guess,kd,kl);
   end

   if ~all(diff(result)>0),
      fprintf(['Error: Curves intersect or contact each other.\n\n',...
               'kd = %g\n\nWrong curves:'], kd);
      disp(find(diff(result)<=0))
      wrong=1;
      break,
   end

   DC_tmp([1 2],:)=DC_tmp([2 3],:);
   DC_tmp(3,:)=result;

   if mod(i-1, frequency) == 0,
      j=j+1;
      fwrite(fidDC,result,'double');
      fwrite(fidLOG,delta,'double');
      fwrite(fidLOG,iter,'uchar');
   end

   waitbar(i/n, h)
end

fclose(fidDC);
```

```
fclose(fidLOG);


% transform: tmp -> curve
fidDC = fopen(tmpfname_dc, 'r');
fidLOG = fopen(tmpfname_log, 'r');

for i=1:no,
   fid_r(i)=fopen(sprintf('curve_r_%d', i),'a');
   fid_d(i)=fopen(sprintf('curve_d_%d', i),'a');
   fid_i(i)=fopen(sprintf('curve_i_%d', i),'a');
end

for kd=1:length(KD),
   result=fread(fidDC, no, 'double');
   delta=fread(fidLOG, no, 'double');
   iter=fread(fidLOG, no, 'uchar');
   for i=1:no,
      fwrite(fid_r(i), result(i), 'double');
      fwrite(fid_d(i), delta(i), 'double');
      fwrite(fid_i(i), iter(i), 'uchar');
   end
end

fclose(fid_r);
fclose(fid_d);
fclose(fid_i);
fclose(fidDC);
fclose(fidLOG);
delete(tmpfname_dc)
delete(tmpfname_log)

% transform: curve -> fname
FLAG_TPD=1;
wave='longitudinal';
velocity='phase';
save(fname, 'FLAG_TPD', 'wave', 'velocity', 'mi', 'KD', 'no')
save([fname '_log'], 'FLAG_TPD', 'wave', 'velocity', 'mi', 'KD', 'no')
for i=1:no,
   fid_r=fopen(sprintf('curve_r_%d', i),'r');
   eval(['X' sprintf('%d',i) '=fread(fid_r,length(KD),''double'');'])
   save(fname, ['X' sprintf('%d',i)], '-append')
   clear(['X' sprintf('%d',i)])
   fclose(fid_r);
   delete(sprintf('curve_r_%d', i))

   fid_d=fopen(sprintf('curve_d_%d', i),'r');
   eval(['D' sprintf('%d',i) '=fread(fid_d,length(KD),''double'');'])
   save([fname '_log'], ['D' sprintf('%d',i)], '-append')
```

```
   clear(['D' sprintf('%d',i)])
   fclose(fid_d);
   delete(sprintf('curve_d_%d', i))

   fid_i=fopen(sprintf('curve_i_%d', i),'r');
   eval(['I' sprintf('%d',i) '=fread(fid_i,length(KD),''uchar'');'])
   save([fname '_log'], ['I' sprintf('%d',i)], '-append')
   clear(['I' sprintf('%d',i)])
   fclose(fid_i);
   delete(sprintf('curve_i_%d', i))
end


if ~wrong, disp('O.K. Curves don''t intersect.'), end

close(h)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y = tp_dl_f(x, kd, kl);
%
% TP - Thick Plate toolbox
% DL - Dispersion for Longitudinal waves
% Dispersion curve function
%
% Parameters:  x  ... c/c2 (vector)
%              kd ... k.d (scalar)
%              kl ... (c2/c1)^2 (scalar)
%
%              y  ... output (vector)
%
% Syntax:      y = tp_dl_f(x, kd, kl);
%
% Example:     y = tp_dl_f([0.5:0.1:500], 1.5, (1-2*0.3)/2/(1-0.3));
%

% (c) 2001, Petr Hora


x2=x.^2;

x2_1=x2(x<1);
x2_2=x2(x>=1 & x<=1/sqrt(kl));
x2_3=x2(x>1/sqrt(kl));

o_1=sqrt(1-x2_1);
ok_1=sqrt(1-kl*x2_1);
```

```
o_2=sqrt(x2_2-1);
ok_2=sqrt(1-kl*x2_2);
o_3=sqrt(x2_3-1);
ok_3=sqrt(kl*x2_3-1);

y(x<1)=(2-x2_1).^2.*(1+exp(-2*kd*ok_1)).*(1-exp(-2*kd*o_1))./4 ...
   - o_1.*ok_1.*(1-exp(-2*kd*ok_1)).*(1+exp(-2*kd*o_1));

y(x>=1 & x<=1/sqrt(kl))=(2-x2_2).^2.*(1+exp(-2*kd*ok_2)).*sin(kd*o_2)./2 ...
   -2*o_2.*ok_2.*(1-exp(-2*kd*ok_2)).*cos(kd*o_2);

y(x>1/sqrt(kl))=(2-x2_3).^2.*cos(kd*ok_3).*sin(kd*o_3) ...
   +4*o_3.*ok_3.*sin(kd*ok_3).*cos(kd*o_3);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y = tp_dl_df(x, kd, kl);
%
% TP - Thick Plate toolbox
% DL - Dispersion for Longitudinal waves
% Derivation of dispersive curve function
%
% Parameters:  x  ... c/c2 (vector)
%              kd ... k.d (scalar)
%              kl ... (c2/c1)^2 (scalar)
%
%              y  ... output (vector)
%
% Syntax:      y = tp_dl_df(x, kd, kl);
%
% Example:     y = tp_dl_df([0.5:0.1:500], 1.5, (1-2*0.3)/2/(1-0.3));
%

% (c) 2001, Petr Hora


x_1=x(x<1);
x_2=x(x>=1 & x<=1/sqrt(kl));
x_3=x(x>1/sqrt(kl));

x2=x.^2;

x2_1=x2(x<1);
x2_2=x2(x>=1 & x<=1/sqrt(kl));
x2_3=x2(x>1/sqrt(kl));


o_1=sqrt(1-x2_1);
```

```
ok_1=sqrt(1-kl*x2_1);
co_1=0.5*(1+exp(-2*kd*o_1));
cok_1=0.5*(1+exp(-2*kd*ok_1));
so_1=0.5*(1-exp(-2*kd*o_1));
sok_1=0.5*(1-exp(-2*kd*ok_1));

y(x<1)=-4*(2-x2_1).*cok_1.*so_1.*x_1-(2-x2_1).^2.*sok_1*kd./ok_1.*kl.*x_1.*so_1 ...
   -(2-x2_1).^2.*cok_1.*co_1.*kd./o_1.*x_1+4./o_1.*ok_1.*sok_1.*co_1.*x_1 ...
   +4*o_1./ok_1.*sok_1.*co_1.*kl.*x_1+4*o_1.*cok_1.*kd.*kl.*x_1.*co_1 ...
   +4*ok_1.*sok_1.*so_1.*kd.*x_1;

o_2=sqrt(x2_2-1);
ok_2=sqrt(1-kl*x2_2);
co_2=cos(kd*o_2);
cok_2=0.5*(1+exp(-2*kd*ok_2));
so_2=sin(kd*o_2);
sok_2=0.5*(1-exp(-2*kd*ok_2));

y(x>=1 & x<=1/sqrt(kl))=-4*(2-x2_2).*cok_2.*so_2.*x_2 ...
   -(2-x2_2).^2.*sok_2*kd./ok_2.*kl.*x_2.*so_2+(2-x2_2).^2.*cok_2.*co_2.*kd./o_2.*x_2 ...
   -4./o_2.*ok_2.*sok_2.*co_2.*x_2+4*o_2./ok_2.*sok_2.*co_2.*kl.*x_2 ...
   +4*o_2.*cok_2.*kd.*kl.*x_2.*co_2+4*ok_2.*sok_2.*so_2.*kd.*x_2;

o_3=sqrt(x2_3-1);
ok_3=sqrt(kl*x2_3-1);
co_3=cos(kd*o_3);
cok_3=cos(kd*ok_3);
so_3=sin(kd*o_3);
sok_3=sin(kd*ok_3);

y(x>1/sqrt(kl))=-4*(2-x2_3).*cok_3.*so_3.*x_3 ...
   -(2-x2_3).^2.*sok_3*kd./ok_3.*kl.*x_3.*so_3+(2-x2_3).^2.*cok_3.*co_3.*kd./o_3.*x_3 ...
   +4./o_3.*ok_3.*sok_3.*co_3.*x_3+4*o_3./ok_3.*sok_3.*co_3.*kl.*x_3 ...
   +4*o_3.*cok_3.*kd.*kl.*x_3.*co_3-4*ok_3.*sok_3.*so_3.*kd.*x_3;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y=extrapol_quad(yi);
%
% Quadratic equidistant extrapolation
%
%          f_n+1 = f_n-2 - 3.f_n-1 + 3.f_n
%
%
% Parameters:  yi ... column vector [or matrix]
%              y  ... scalar [or row vector]
%
% Syntax:      y = extrapol_quad(yi);
```

```
%

% (c) 2001, Petr Hora


y=3*yi(end,:)-3*yi(end-1,:)+yi(end-2,:);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function g = tp_dl_guess(kd, mi, no);
%
% TP - Thick Plate toolbox
% DL - Dispersion for Longitudinal waves
% Guess of roots for k.d -> 0
%
% Parameters:  kd ... k.d (scalar)
%              mi ... Poisson's ratio (scalar)
%              no ... number of dispersion curves (scalar)
%
%              g  ... guess (vector)
%
% Syntax:      g = tp_dl_guess(kd, mi, no);
%
% Example:     o = tp_dl_guess(0.005, 0.3, 10);
%

% (c) 2001, Petr Hora


c1toc2=sqrt(2*(1-mi)/(1-2*mi));

kl=1/c1toc2^2;
first_curve=2*sqrt(1-kl)*sqrt(1-1/3*(1-2*kl)*kd.^2);


k=1:no;
pk=k/kd*pi;

s=1:no;
ps=0.5*(2*s-1)/kd*c1toc2*pi;

g=sort([pk ps]);

g=[first_curve g];

% (Ex. mi=0.333)
d=diff(g);
warning('off');
```

```
i=find(d(2:end)./d(1:end-1)<10e-2);
warning('on');
g(i+1)=g(i+1)-d(i)/20;
g(i+2)=g(i+2)+d(i+2)/20;

g=g(1:no);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function g = tp_dl_rawguess(kd, mi, no);
%
% TP - Thick Plate toolbox
% DL - Dispersion for Longitudinal waves
% Raw guess of roots for k.d -> 0
%
% Parameters:  kd ... k.d (scalar)
%              mi ... Poisson's ratio (scalar)
%              no ... number of dispersion curves (scalar)
%
%              g  ... guess (vector)
%
% Syntax:      g = tp_dl_rawguess(kd, mi, no);
%
% Example:     o = tp_dl_rawguess(0.005, 0.3, 10);
%

% (c) 2001, Petr Hora


kl=(1-2*mi)/(2*(1-mi));
n=10;


g_old = tp_dl_guess(kd, mi, no+1);

g_old=[0 g_old];

dg=diff(g_old);

sg=dg/n;

g_new=[];
for i=1:no+1,
   g_new=[g_new g_old(i):sg(i):g_old(i+1)-sg(i)];
end
g_new=[g_new g_old(no+2)];
```

```
f=tp_dl_f(g_new,kd,kl);
s=abs(diff(sign(f)));
i=find(s==2);
a=g_new(i);
b=g_new(i+1);

g=(a+b)./2;

g=g(1:no);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [new, delta, iter] = tp_dl_precising(no, old, kd, kl);
%
% TP - Thick Plate toolbox
% DL - Dispersion for Longitudinal waves
% Dispersion curve roots precising
%
% Parameters:  no    ... number of dispersion curves (scalar)
%              old   ... old roots (vector)
%              kd    ... k.d (scalar)
%              kl    ... (c2/c1)^2 (scalar)
%
%              new   ... new refine roots (vector)
%              delta ... precision of roots (vector)
%              iter  ... number of iteration (vector)
%
% Syntax:      new = tp_dl_precising(no, old, kd, kl);
%
% Example:     new = tp_dl_precising(10, old, 1.5, (1-2*0.3)/2/(1-0.3));
%

% (c) 2001, Petr Hora


PREC=5e-16;
MAX_ITER=20;

index=1:length(old);
delta=PREC+zeros(size(old));
iter=zeros(size(old));

while 1,
   iter(index)=iter(index)+1;

   new(index)=old(index)-tp_dl_f(old(index),kd,kl)./tp_dl_df(old(index),kd,kl);

   delta(index)=abs(new(index)-old(index))./new(index);
```

```
   af=abs(tp_dl_f(new(index),kd,kl));

   i=delta(index)<PREC | af<eps;
   index=index(~i);

   if isempty(index) | iter(index(1))==MAX_ITER,
      break,
   end

   old(index)=new(index);
end
```

Příloha B

tp_ds_roots.m

```
function tp_ds_roots(mi, kd_max, no, frequency, fname);

%
% TP - Thick Plate toolbox
% DS - Dispersion for Shear waves
% Dispersion curve roots finding
%
% Parameters:  mi     ... Poisson's ratio (scalar)
%              kd_max ... max. k.d (scalar)
%              no     ... number of dispersion curves (scalar)
%              frequency ... save frequency of k.d (scalar)
%              fname  ... name of file for the dispersion curves (string)
%
%              DC     ... dispersion curves (matrix),
%                         rows    ... k.d,
%                         columns ... curves
%
% Syntax:      tp_ds_roots(mi, kd_max, no, frequency, fname);
%
% Example:     tp_ds_roots(0.3, 250, 100, 10, 'CurveS030');
%

% (c) 2001, Petr Hora



kt=(2*(1-mi))/(1-2*mi);

KD_COMP=[0.0025:0.0025:kd_max];
KD=[0.0025:0.0025*frequency:kd_max];

DC_tmp=zeros([3, no]);

tmpfname_dc=tempname;
fidDC = fopen(tmpfname_dc, 'w');
```

```
tmpfname_log=tempname;
fidLOG = fopen(tmpfname_log, 'w');


h = waitbar(0,'Please wait...');
n=length(KD_COMP);

wrong=0;
i=0; j=0;
for kd=KD_COMP,
   i=i+1;

   if kd<0.4,  % very sensitive value (0.4)!!!
      guess=tp_ds_guess(kd,mi,no);
      [result,delta,iter]=tp_ds_precising(no,guess,kd,kt);
   else
      guess=extrapol_quad(DC_tmp);
      [result,delta,iter]=tp_ds_precising(no,guess,kd,kt);
   end

   if ~all(diff(result)>0),
      fprintf('Raw guess for kd = %g\n', kd);
      guess=tp_ds_rawguess(kd,mi,no);
      [result,delta,iter]=tp_ds_precising(no,guess,kd,kt);
   end

   if ~all(diff(result)>0),
      fprintf(['Error: Curves intersect or contact each other.\n\n',...
               'kd = %g\n\nWrong curves:'], kd);
      disp(find(diff(result)<=0))
      wrong=1;
      break,
   end

   DC_tmp([1 2],:)=DC_tmp([2 3],:);
   DC_tmp(3,:)=result;

   if mod(i-1, frequency) == 0,
      j=j+1;
      fwrite(fidDC,result,'double');
      fwrite(fidLOG,delta,'double');
      fwrite(fidLOG,iter,'uchar');
   end

   waitbar(i/n, h)
end

fclose(fidDC);
```

```
fclose(fidLOG);


% transform: tmp -> curve
fidDC = fopen(tmpfname_dc, 'r');
fidLOG = fopen(tmpfname_log, 'r');

for i=1:no,
   fid_r(i)=fopen(sprintf('curve_r_%d', i),'a');
   fid_d(i)=fopen(sprintf('curve_d_%d', i),'a');
   fid_i(i)=fopen(sprintf('curve_i_%d', i),'a');
end

for kd=1:length(KD),
   result=fread(fidDC, no, 'double');
   delta=fread(fidLOG, no, 'double');
   iter=fread(fidLOG, no, 'uchar');
   for i=1:no,
      fwrite(fid_r(i), result(i), 'double');
      fwrite(fid_d(i), delta(i), 'double');
      fwrite(fid_i(i), iter(i), 'uchar');
   end
end

fclose(fid_r);
fclose(fid_d);
fclose(fid_i);
fclose(fidDC);
fclose(fidLOG);
delete(tmpfname_dc)
delete(tmpfname_log)

% transform: curve -> fname
FLAG_TPD=1;
wave='shear';
velocity='phase';
save(fname, 'FLAG_TPD', 'wave', 'velocity', 'mi', 'KD', 'no')
save([fname '_log'], 'FLAG_TPD', 'wave', 'velocity', 'mi', 'KD', 'no')
for i=1:no,
   fid_r=fopen(sprintf('curve_r_%d', i),'r');
   eval(['E' sprintf('%d',i) '=fread(fid_r,length(KD),''double'');'])
   save(fname, ['E' sprintf('%d',i)], '-append')
   clear(['E' sprintf('%d',i)])
   fclose(fid_r);
   delete(sprintf('curve_r_%d', i))

   fid_d=fopen(sprintf('curve_d_%d', i),'r');
   eval(['D' sprintf('%d',i) '=fread(fid_d,length(KD),''double'');'])
   save([fname '_log'], ['D' sprintf('%d',i)], '-append')
```

```
   clear(['D' sprintf('%d',i)])
   fclose(fid_d);
   delete(sprintf('curve_d_%d', i))

   fid_i=fopen(sprintf('curve_i_%d', i),'r');
   eval(['I' sprintf('%d',i) '=fread(fid_i,length(KD),''uchar'');'])
   save([fname '_log'], ['I' sprintf('%d',i)], '-append')
   clear(['I' sprintf('%d',i)])
   fclose(fid_i);
   delete(sprintf('curve_i_%d', i))
end


if ~wrong, disp('O.K. Curves don''t intersect.'), end

close(h)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y = tp_ds_f(x, kd, kt);
%
% TP - Thick Plate toolbox
% DS - Dispersion for Shear waves
% Dispersion curve function
%
% Parameters:  x  ... c/c1 (vector)
%              kd ... k.d (scalar)
%              kt ... (c1/c2)^2 (scalar)
%
%              y  ... output (vector)
%
% Syntax:      y = tp_ds_f(x, kd, kt);
%
% Example:     y = tp_ds_f([0.5:0.1:500], 1.5, 2*(1-0.3)/(1-2*0.3));
%

% (c) 2001, Petr Hora


x2=x.^2;

x2_1=x2(x<1/sqrt(kt));
x2_2=x2(x>=1/sqrt(kt) & x<=1);
x2_3=x2(x>1);

o_1=sqrt(1-x2_1);
ok_1=sqrt(1-kt*x2_1);
o_2=sqrt(1-x2_2);
```

```
ok_2=sqrt(kt*x2_2-1);
o_3=sqrt(x2_3-1);
ok_3=sqrt(kt*x2_3-1);

y(x<1/sqrt(kt))=(2-kt*x2_1).^2.*(1+exp(-2*kd*ok_1)).*(1-exp(-2*kd*o_1))./4 ...
   -o_1.*ok_1.*(1-exp(-2*kd*ok_1)).*(1+exp(-2*kd*o_1));

y(x>=1/sqrt(kt) & x<=1)=(2-kt*x2_2).^2.*cos(kd*ok_2).*(1-exp(-2*kd*o_2))./2 ...
   +2*o_2.*ok_2.*sin(kd*ok_2).*(1+exp(-2*kd*o_2));

y(x>1)=(2-kt*x2_3).^2.*cos(kd*ok_3).*sin(kd*o_3) ...
   +4*o_3.*ok_3.*sin(kd*ok_3).*cos(kd*o_3);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y = tp_ds_df(x, kd, kt);
%
% TP - Thick Plate toolbox
% DS - Dispersion for Shear waves
% Derivation of dispersive curve function
%
% Parameters:  x  ... c/c1 (vector)
%              kd ... k.d (scalar)
%              kt ... (c1/c2)^2 (scalar)
%
%              y  ... output (vector)
%
% Syntax:      y = tp_ds_df(x, kd, kt);
%
% Example:     y = tp_ds_df([0.5:0.1:500], 1.5, 2*(1-0.3)/(1-2*0.3));
%

% (c) 2001, Petr Hora


x_1=x(x<1/sqrt(kt));
x_2=x(x>=1/sqrt(kt) & x<=1);
x_3=x(x>1);

x2=x.^2;

x2_1=x2(x<1/sqrt(kt));
x2_2=x2(x>=1/sqrt(kt) & x<=1);
x2_3=x2(x>1);


o_1=sqrt(1-x2_1);
ok_1=sqrt(1-kt*x2_1);
```

```
co_1=0.5*(1+exp(-2*kd*o_1));
cok_1=0.5*(1+exp(-2*kd*ok_1));
so_1=0.5*(1-exp(-2*kd*o_1));
sok_1=0.5*(1-exp(-2*kd*ok_1));

y(x<1/sqrt(kt))=-4*(2-kt*x2_1).*cok_1.*so_1*kt.*x_1 ...
   -(2-kt*x2_1).^2.*sok_1*kd./ok_1*kt.*x_1.*so_1-(2-kt*x2_1).^2.*cok_1.*co_1*kd./o_1.*x_1 ...
   +4./o_1.*ok_1.*sok_1.*co_1.*x_1+4*o_1./ok_1.*sok_1.*co_1*kt.*x_1 ...
   +4*o_1.*cok_1*kd*kt.*x_1.*co_1+4*ok_1.*sok_1.*so_1*kd.*x_1;


o_2=sqrt(1-x2_2);
ok_2=sqrt(kt*x2_2-1);
co_2=0.5*(1+exp(-2*kd*o_2));
cok_2=cos(kd*ok_2);
so_2=0.5*(1-exp(-2*kd*o_2));
sok_2=sin(kd*ok_2);

y(x>=1/sqrt(kt) & x<=1)=-4*(2-kt*x2_2).*cok_2.*so_2*kt.*x_2 ...
   -(2-kt*x2_2).^2.*sok_2*kd./ok_2*kt.*x_2.*so_2-(2-kt*x2_2).^2.*cok_2.*co_2*kd./o_2.*x_2 ...
   -4./o_2.*ok_2.*sok_2.*co_2.*x_2+4*o_2./ok_2.*sok_2.*co_2*kt.*x_2 ...
   +4*o_2.*cok_2*kd*kt.*x_2.*co_2-4*ok_2.*sok_2.*so_2*kd.*x_2;


o_3=sqrt(x2_3-1);
ok_3=sqrt(kt*x2_3-1);
co_3=cos(kd*o_3);
cok_3=cos(kd*ok_3);
so_3=sin(kd*o_3);
sok_3=sin(kd*ok_3);

y(x>1)=-4*(2-kt*x2_3).*cok_3.*so_3*kt.*x_3-(2-kt*x2_3).^2.*sok_3*kd./ok_3*kt.*x_3.*so_3 ...
   +(2-kt*x2_3).^2.*cok_3.*co_3*kd./o_3.*x_3+4./o_3.*ok_3.*sok_3.*co_3.*x_3 ...
   +4*o_3./ok_3.*sok_3.*co_3*kt.*x_3+4*o_3.*cok_3*kd*kt.*x_3.*co_3-4*ok_3.*sok_3.*so_3*kd.*x_3;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y=extrapol_quad(yi);
%
% Quadratic equidistant extrapolation
%
%          f_n+1 = f_n-2 - 3.f_n-1 + 3.f_n
%
%
% Parameters:  yi ... column vector [or matrix]
%              y  ... scalar [or row vector]
%
% Syntax:      y = extrapol_quad(yi);
```

```
%

% (c) 2001, Petr Hora


y=3*yi(end,:)-3*yi(end-1,:)+yi(end-2,:);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function g = tp_ds_guess(kd, mi, no);
%
% TP - Thick Plate toolbox
% DS - Dispersion for Shear waves
% Guess of roots for k.d -> 0
%
% Parameters:  kd ... k.d (scalar)
%              mi ... Poisson's ratio (scalar)
%              no ... number of dispersion curves (scalar)
%
%              g  ... guess (vector)
%
% Syntax:      g = tp_ds_guess(kd, mi, no);
%
% Example:     o = tp_ds_guess(0.005, 0.3, 10);
%

% (c) 2001, Petr Hora


c2toc1=sqrt((1-2*mi)/(2*(1-mi)));

first_curve=kd*sqrt(2/3/(1-mi))*c2toc1;


k=1:no;
pk=k/kd*pi;

s=1:no;
ps=0.5*(2*s-1)/kd*c2toc1*pi;

g=sort([pk ps]);

g=[first_curve g];

% (Ex. mi=0.1)
d=diff(g);
warning('off');
i=find(d(2:end)./d(1:end-1)<10e-2);
```

```
warning('on');
g(i+1)=g(i+1)-d(i)/50;
g(i+2)=g(i+2)+d(i+2)/50;

g=g(1:no);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function g = tp_ds_rawguess(kd, mi, no);
%
% TP - Thick Plate toolbox
% DS - Dispersion for Shear waves
% Raw guess of roots for k.d -> 0
%
% Parameters:  kd ... k.d (scalar)
%              mi ... Poisson's ratio (scalar)
%              no ... number of dispersion curves (scalar)
%
%              g  ... guess (vector)
%
% Syntax:      g = tp_ds_rawguess(kd, mi, no);
%
% Example:     o = tp_ds_rawguess(0.005, 0.3, 10);
%

% (c) 2001, Petr Hora


kt=(2*(1-mi))/(1-2*mi);
n=10;



g_old = tp_ds_guess(kd, mi, no+1);

g_old=[0 g_old];

dg=diff(g_old);

sg=dg/n;

g_new=[];
for i=1:no+1,
   g_new=[g_new g_old(i):sg(i):g_old(i+1)-sg(i)];
end
g_new=[g_new g_old(no+2)];


f=tp_ds_f(g_new,kd,kt);
```

```
s=abs(diff(sign(f)));
i=find(s==2);
a=g_new(i);
b=g_new(i+1);

g=(a+b)./2;

g=g(1:no);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [new, delta, iter] = tp_ds_precising(no, old, kd, kt);
%
% TP - Thick Plate toolbox
% DS - Dispersion for Shear waves
% Dispersion curve roots precising
%
% Parameters:  no    ... number of dispersion curves (scalar)
%              old   ... old roots (vector)
%              kd    ... k.d (scalar)
%              kt    ... (c1/c2)^2 (scalar)
%
%              new   ... new refine roots (vector)
%              delta ... precision of roots (vector)
%              iter  ... number of iteration (vector)
%
% Syntax:      new = tp_ds_precising(no, old, kd, kt);
%
% Example:     new = tp_ds_precising(10, old, 1.5, 2*(1-0.3)/(1-2*0.3));
%

% (c) 2001, Petr Hora


PREC=5e-16;
MAX_ITER=20;

index=1:length(old);
delta=PREC+zeros(size(old));
iter=zeros(size(old));

while 1,
   iter(index)=iter(index)+1;

   new(index)=old(index)-tp_ds_f(old(index),kd,kt)./tp_ds_df(old(index),kd,kt);

   delta(index)=abs(new(index)-old(index))./new(index);
   af=abs(tp_ds_f(new(index),kd,kt));
```

```
   i=delta(index)<PREC | af<eps;
   index=index(~i);

   if isempty(index) | iter(index(1))==MAX_ITER,
      break,
   end;

   old(index)=new(index);
end
```

Příloha C

tp_dl_cg.m

```
function tp_dl_cg(fname);
%
% TP - Thick Plate toolbox
% DL - Dispersion for Longitudinal waves
% Transformation to group velocity
%
% Parameters:  fname  ... name of file with dispersion curves (string)
%
% Syntax:      tp_dl_cg(fname);
%
% Example:     tp_dl_cg('CurveL029');
%

% (c) 2001, Petr Hora


load(fname, 'FLAG_TPD', 'wave', 'velocity', 'mi', 'KD', 'no')

if ~exist('FLAG_TPD','var') | ~exist('wave','var') | ~exist('velocity','var')
   msgbox(['Sorry, file ' fname ' is not a valid TPD file.'],...
           'Missing Variable: FLAG_TPD | wave | velocity','error')
   return
end

if ~strcmp(velocity, 'phase')
   msgbox(['Sorry, file ' fname ' does not contain the phase velocities.'],...
           'Missing phase velocities','error')
   return
end

if ~strcmp(wave, 'longitudinal')
   msgbox(['Sorry, file ' fname ' does not contain the longitudinal waves.'],...
           'Missing longitudinal waves','error')
   return
end
```

```
velocity='group';
save([fname '_g'], 'FLAG_TPD', 'wave', 'velocity', 'mi','KD', 'no')

x=KD';
kl=(1-2*mi)/2/(1-mi);

h = waitbar(0,'Please wait...');

for i=1:no,
   load(fname, ['X' sprintf('%d',i)])
   eval(['y=X' sprintf('%d',i) ';'])

   x_1=x(y<1);
   x_2=x(y>=1 & y<=1/sqrt(kl));
   x_3=x(y>1/sqrt(kl));

   y_1=y(y<1);
   y_2=y(y>=1 & y<=1/sqrt(kl));
   y_3=y(y>1/sqrt(kl));

   y2=y.^2;
   y2_1=y2(y<1);
   y2_2=y2(y>=1 & y<=1/sqrt(kl));
   y2_3=y2(y>1/sqrt(kl));

   o_1=sqrt(1-y2_1);
   ok_1=sqrt(1-y2_1*kl);
   to_1=tanh(x_1.*o_1);
   tok_1=tanh(x_1.*ok_1);

   num(y<1)=-(1-tok_1.^2).*ok_1./to_1+tok_1./to_1.^2.*(1-to_1.^2).*o_1;
   den(y<1)=-(1-tok_1.^2).*x_1./ok_1.*y_1.*kl./to_1 ...
      +tok_1./to_1.^2.*(1-to_1.^2).*x_1./o_1.*y_1 ...
      -(y2_1-2)./o_1./ok_1.*y_1-1/4.*(y2_1-2).^2./o_1.^3./ok_1.*y_1 ...
      -1/4.*(y2_1-2).^2./o_1./ok_1.^3.*y_1.*kl;

   o_2=sqrt(y2_2-1);
   ok_2=sqrt(1-y2_2*kl);
   to_2=tan(x_2.*o_2);
   tok_2=tanh(x_2.*ok_2);

   num(y>=1 & y<=1/sqrt(kl))=(1-tok_2.^2).*ok_2./to_2-tok_2./to_2.^2.*(1+to_2.^2).*o_2;
   den(y>=1 & y<=1/sqrt(kl))=(1-tok_2.^2).*x_2./ok_2.*y_2.*kl./to_2 ...
      +tok_2./to_2.^2.*(1+to_2.^2).*x_2./o_2.*y_2 ...
      +(y2_2-2)./o_2./ok_2.*y_2-1/4.*(y2_2-2).^2./o_2.^3./ok_2.*y_2 ...
      +1/4.*(y2_2-2).^2./o_2./ok_2.^3.*y_2.*kl;

   o_3=sqrt(y2_3-1);
```

```
   ok_3=sqrt(y2_3*kl-1);
   to_3=tan(x_3.*o_3);
   tok_3=tan(x_3.*ok_3);

   num(y>1/sqrt(kl))=-(1+tok_3.^2).*ok_3./to_3+tok_3./to_3.^2.*(1+to_3.^2).*o_3;
   den(y>1/sqrt(kl))=+(1+tok_3.^2).*x_3./ok_3.*y_3.*kl./to_3 ...
      -tok_3./to_3.^2.*(1+to_3.^2).*x_3./o_3.*y_3 ...
      +(y2_3-2)./o_3./ok_3.*y_3-1/4.*(y2_3-2).^2./o_3.^3./ok_3.*y_3 ...
      -1/4.*(y2_3-2).^2./o_3./ok_3.^3.*y_3.*kl;


   y=y+x.*num'./den';

   eval(['X' sprintf('%d',i) '=y;']);
   save([fname '_g'], ['X' sprintf('%d',i)], '-append')
   clear(['X' sprintf('%d',i)])

   waitbar(i/no, h)
end

close(h)
```

Příloha D

tp_ds_cg.m

```
function tp_ds_cg(fname);
%
% TP - Thick Plate toolbox
% DS - Dispersion for Shear waves
% Transformation to group velocity
%
% Parameters:  fname  ... name of file with dispersion curves (string)
%
% Syntax:      tp_ds_cg(fname);
%
% Example:     tp_ds_cg('CurveS029');
%

% (c) 2001, Petr Hora


load(fname, 'FLAG_TPD', 'wave', 'velocity', 'mi', 'KD', 'no')

if ~exist('FLAG_TPD','var') | ~exist('wave','var') | ~exist('velocity','var')
   msgbox(['Sorry, file ' fname ' is not a valid TPD file.'],...
           'Missing Variable: FLAG_TPD | wave | velocity','error')
   return
end

if ~strcmp(velocity, 'phase')
   msgbox(['Sorry, file ' fname ' does not contain the phase velocities.'],...
           'Missing phase velocities','error')
   return
end

if ~strcmp(wave, 'shear')
   msgbox(['Sorry, file ' fname ' does not contain the shear waves.'],...
           'Missing shear waves','error')
   return
end
```

```
velocity='group';
save([fname '_g'], 'FLAG_TPD', 'wave', 'velocity', 'mi','KD', 'no')

x=KD';
kt=2*(1-mi)/(1-2*mi);

h = waitbar(0,'Please wait...');

for i=1:no,
   load(fname, ['E' sprintf('%d',i)])
   eval(['y=E' sprintf('%d',i) ';'])

   y2=y.^2;
   o=sqrt(1-y2);
   ok=sqrt(1-y2*kt);
   to=tanh(x.*o);
   tok=tanh(x.*ok);

   num=-(1-to.^2).*o./tok+to./tok.^2.*(1-tok.^2).*ok;
   den=-(1-to.^2).*x./o.*y./tok+to./tok.^2.*(1-tok.^2).*x./ok.*y.*kt ...
      +4./o.*ok./(y2.*kt-2).^2.*y+4.*o./ok./(y2.*kt-2).^2.*y.*kt ...
      +16.*o.*ok./(y2.*kt-2).^3.*y.*kt;

   y=real(y+x.*num./den);

   eval(['E' sprintf('%d',i) '=y;']);
   save([fname '_g'], ['E' sprintf('%d',i)], '-append')
   clear(['E' sprintf('%d',i)])

   waitbar(i/no, h)
end

close(h)
```

Příloha E

tp_dl_fine.m

```
function tp_dl_fine(fname, prec);
%
% TP - Thick Plate toolbox
% DL - Dispersion for Longitudinal waves
% Dispersion curve roots precising
%
% Parameters:
%             fname ... name of file for the dispersion curves (string)
%             prec    ... precision
%
% Syntax:     tp_dl_fine(fname, precision);
%
% Example:    tp_dl_fine('cl030', 1e-14);
%

% (c) 2001, Petr Hora



if length(ver('symbolic'))==0,
   error('Symbolic Toolbox not found')
end

procread('l_curve.src');


load(fname, 'FLAG_TPD', 'wave', 'velocity', 'mi', 'KD', 'no')

if ~exist('FLAG_TPD','var') | ~exist('wave','var') | ~exist('velocity','var')
   msgbox(['Sorry, file ' fname ' is not a valid TPD file.'],...
           'Missing Variable: FLAG_TPD | wave | velocity','error')
   return
end

if ~strcmp(velocity, 'phase')
```

```
   msgbox(['Sorry, file ' fname ' does not contain the phase velocities.'],...
           'Missing phase velocities','error')
   return
end

if ~strcmp(wave, 'longitudinal')
   msgbox(['Sorry, file ' fname ' does not contain the longitudinal waves.'],...
           'Missing longitudinal waves','error')
   return
end

h = waitbar(0, 'Please wait...');

for i=1:no,
   load(fname, sprintf('X%d', i))
   eval(sprintf('X=X%d;', i))
   clear(sprintf('X%d', i))

   load([fname '_log'], sprintf('I%d', i), sprintf('D%d', i))
   eval(sprintf('I=I%d; D=D%d;', i, i))
   clear(sprintf('I%d', i), sprintf('D%d', i))

   k=find(D>prec);

   if ~isempty(k),
      for j=1:length(k),
         [result, status]=maple('l_curve', X(k(j)), KD(k(j)), mi, prec);
         disp(sprintf('Curve: %d  kd: %g\n', i, KD(k(j))))
         if status>0,
            status
            error(result);
         else
            x=sscanf(result, '%g,%g,%g');
            X(k(j))=x(1);
            D(k(j))=x(2);
            I(k(j))=x(3)*100+I(k(j));
         end
      end

      eval(sprintf('X%d=X;', i))
      save(fname, sprintf('X%d', i), '-append')
      clear('X')

      eval(sprintf('D%d=D;', i))
      save([fname '_log'], sprintf('D%d', i), '-append')
      clear('D')

      eval(sprintf('I%d=I;', i))
      save([fname '_log'], sprintf('I%d', i), '-append')
```

```
      clear('I')
   end

   waitbar(i/no, h)
end

close(h)
```

Příloha F

tp_ds_fine.m

```
function tp_ds_fine(fname, prec);
%
% TP - Thick Plate toolbox
% DL - Dispersion for Shear waves
% Dispersion curve roots precising
%
% Parameters:
%              fname ... name of file for the dispersion curves (string)
%              prec    ... precision
%
% Syntax:      tp_ds_fine(fname, precision);
%
% Example:     tp_ds_fine('cs030', 1e-14);
%

% (c) 2001, Petr Hora


if length(ver('symbolic'))==0,
   error('Symbolic Toolbox not found')
end

procread('s_curve.src');

load(fname, 'FLAG_TPD', 'wave', 'velocity', 'mi', 'KD', 'no')

if ~exist('FLAG_TPD','var') | ~exist('wave','var') | ~exist('velocity','var')
   msgbox(['Sorry, file ' fname ' is not a valid TPD file.'],...
           'Missing Variable: FLAG_TPD | wave | velocity','error')
   return
end

if ~strcmp(velocity, 'phase')
   msgbox(['Sorry, file ' fname ' does not contain the phase velocities.'],...
           'Missing phase velocities','error')
```

```
   return
end

if ~strcmp(wave, 'shear')
   msgbox(['Sorry, file ' fname ' does not contain the shear waves.'],...
           'Missing shear waves','error')
   return
end

h = waitbar(0, 'Please wait...');

for i=1:no,
   load(fname, sprintf('E%d', i))
   eval(sprintf('E=E%d;', i))
   clear(sprintf('E%d', i))

   load([fname '_log'], sprintf('I%d', i), sprintf('D%d', i))
   eval(sprintf('I=I%d; D=D%d;', i, i))
   clear(sprintf('I%d', i), sprintf('D%d', i))

   k=find(D>prec);

   if ~isempty(k),
      for j=1:length(k),
         [result, status]=maple('s_curve', E(k(j)), KD(k(j)), mi, prec);
         disp(sprintf('Curve: %d  kd: %g\n', i, KD(k(j))))
         if status>0,
            status
            error(result);
         else
            e=sscanf(result, '%g,%g,%g');
            E(k(j))=e(1);
            D(k(j))=e(2);
            I(k(j))=e(3)*100+I(k(j));
         end
      end

      eval(sprintf('E%d=E;', i))
      save(fname, sprintf('E%d', i), '-append')
      clear('E')

      eval(sprintf('D%d=D;', i))
      save([fname '_log'], sprintf('D%d', i), '-append')
      clear('D')

      eval(sprintf('I%d=I;', i))
      save([fname '_log'], sprintf('I%d', i), '-append')
      clear('I')
   end
```

```
   waitbar(i/no, h)
end

close(h)
```

Příloha G

l_curve.src

```
l_curve := proc(a,kd,pc,pr)

# a  ... old root
# kd ... k*d
# pc ... Poisson's ratio
# pr ... precision  (1e-15)


local mi, kl, x, i, x2, o, ok, co, cok, so, sok, y, dy, new_x, delta;

Digits:=32:

mi:=evalf(pc);
kl:=(1-2*mi)/(2*(1-mi)):

x:=evalf(a);

for i from 1 by 1 to 20 do

   x2:=x^2;

   o:=sqrt(1-x2):
   ok:=sqrt(1-kl*x2):
   co:=cosh(kd*o):
   cok:=cosh(kd*ok):
   so:=sinh(kd*o):
   sok:=sinh(kd*ok):

   y:=(2-x2)^2*cok*so - 4*o*ok*sok*co:

   if x>1.0 then y:=Im(y): fi;

   dy:=-4*(2-x2)*cok*so*x - (2-x2)^2*sok*kd/ok*kl*x*so\
      - (2-x2)^2*cok*co*kd/o*x + 4/o*ok*sok*co*x\
      + 4*o/ok*sok*co*kl*x + 4*o*cok*kd*kl*x*co + 4*ok*sok*so*kd*x:
```

```
   if x>1.0 then dy:=Im(dy): fi;

   new_x:=x-y/dy;

   delta:=abs((new_x-x)/new_x);

   if delta < pr then
      break
   fi;

   x:=new_x:

od;

RETURN(new_x, delta, i)

end;
```

Příloha H

s_curve.src

```
s_curve := proc(a,kd,pc,pr)

# a  ... old root
# kd ... k*d
# pc ... Poisson's ratio
# pr ... precision  (1e-15)


local mi, kt, x, i, x2, o, ok, co, cok, so, sok, y, dy, new_x, delta;

Digits:=32:

mi:=evalf(pc);
kt:=2*(1-mi)/(1-2*mi):

x:=evalf(a);

for i from 1 by 1 to 20 do

   x2:=x^2:

   o:=sqrt(1-x2):
   ok:=sqrt(1-kt*x2):
   co:=cosh(kd*o):
   cok:=cosh(kd*ok):
   so:=sinh(kd*o):
   sok:=sinh(kd*ok):

   y:=(2-kt*x2)^2*cok*so - 4*o*ok*sok*co:

   if x>1.0 then y:=Im(y): fi;

   dy:=-4*(2-kt*x2)*cok*so*kt*x - (2-kt*x2)^2*sok*kd/ok*kt*x*so\
         - (2-kt*x2)^2*cok*co*kd/o*x + 4/o*ok*sok*co*x + 4*o/ok*sok*co*kt*x\
        + 4*o*cok*kd*kt*x*co + 4*ok*sok*so*kd*x:
```

```
   if x>1.0 then dy:=Im(dy): fi;

   new_x:=x-y/dy;

   delta:=abs((new_x-x)/new_x);

   if delta < pr then
      break
   fi;

   x:=new_x:

od;

RETURN(new_x, delta, i)

end;
```

Příloha I

tp_dc_tools

```
function tp_dc_tool(action);
%
% TP   - Thick Plate toolbox
% DC   - Dispersion Curves
% TOOL - Tool
%
% Parameters:  action ... string, (optional)
%
% Syntax:      tp_dc_tool(action)
%
% Ex.          tp_dc_tool

% (c) 2001, Petr Hora


global hfTPDTOOL htWAVE htPOISSON hrPHASE hrGROUP htTHICKNESS heTHICKNESS htTHICKNESS_UNITS
global htC1 heC1 htC1_UNITS htC2 htCR hrZOOM hrCURSOR haAXES hNewAxes hmFIGURE hmADD hmDATA
global hmPLOT hmPLOT1 hmPLOT2 hmPLOT3 hmPLOT4 hmPLOT5
global wave mi KD DC DC_G
global C1 C2 HALF_THICKNESS loadName
global hlCURSOR1VER hlCURSOR1HOR hlCURSOR2VER hlCURSOR2HOR XP YP XC1 YC1 XC2 YC2 FIVE_PIXELS
global htMARKER1 htMARKER1CURVE htMARKER1XLABEL heMARKER1X htMARKER1YLABEL htMARKER1Y
global htMARKER2 htMARKER2CURVE htMARKER2XLABEL heMARKER2X htMARKER2YLABEL htMARKER2Y
global htMARKER_DXLABEL htMARKER_DX htMARKER_DYLABEL htMARKER_DY hsMARKER1CURVE hsMARKER2CURVE
global STEP_GAMMA_D

if nargin == 0,
   action = 'init';
end


switch action

case 'init'               % initialization
```

```
   create_tpdtool_gui;

   STEP_GAMMA_D =  1;


case 'open'
%  tp_dc_tool('open')   <-- uses uigetfile to get filename and path

   set(gcbf,'WindowButtonMotionFcn','');
   if ishandle(hlCURSOR1VER),
      set(hrZOOM, 'Value',1)
      set(hrCURSOR, 'Value',0)
      delete([hlCURSOR1VER hlCURSOR1HOR hlCURSOR2VER hlCURSOR2HOR]);
      set(findobj(gcbf, 'Tag', 'MARKERS'), 'Visible', 'off')
   end
   cla

   set([htWAVE htPOISSON hrPHASE hrGROUP htTHICKNESS heTHICKNESS htTHICKNESS_UNITS, ...
        htC1 heC1 htC1_UNITS htC2 htCR hrZOOM hrCURSOR haAXES hmFIGURE hmADD hmDATA hmPLOT], ...
        'Visible', 'off')

   [fname,fpath]=uigetfile('*.mat','Load Dispersion Curves');
   if fname == 0, break, end
   loadName = fullfile(fpath,fname);

   set(hfTPDTOOL, 'Pointer', 'watch')
   if ~isempty(findstr(loadName, '_g.')),
      loadName=strrep(loadName, '_g.', '.');
   end

   load(loadName, 'FLAG_TPD', 'wave', 'velocity', 'mi', 'KD', 'no')

   if ~exist('FLAG_TPD','var') | ~exist('wave','var') | ~exist('velocity','var')
      msgbox(['Sorry, file ' loadName ' is not a valid TPD file.'],...
              'Missing Variable: FLAG_TPD | wave | velocity','error')
      return
   end

   if strcmp(wave,'longitudinal'), ch='X'; else ch='E'; end

   DC=zeros(length(KD),no);
   for i=1:no,
      load(loadName, [ch sprintf('%d',i)])
      eval(['DC(:,i)=' ch sprintf('%d',i) ';'])
      clear([ch sprintf('%d',i)])
   end

   DC_G=zeros(length(KD),no);
   for i=1:no,
```

```
      load(strrep(loadName, '.', '_g.'), [ch sprintf('%d',i)])
      eval(['DC_G(:,i)=' ch sprintf('%d',i) ';'])
      clear([ch sprintf('%d',i)])
   end


   set(hfTPDTOOL, 'Pointer', 'arrow')

   if ~exist('FLAG_TPD','var')  % variable
      msgbox('Sorry, this file is not a valid TPD file.',...
             'Missing Variable: FLAG_TPD','error')
      return
   end

   KD=KD';

   set(htWAVE, 'String', sprintf('Wave: %s', wave))
   set(htPOISSON, 'String', sprintf('Poisson''s ratio: %5.3f', mi))
   C1=str2num(get(heC1, 'String'));
   C2=C1*sqrt((1-2*mi)/2/(1-mi));
   set(htC2, 'String', sprintf('c_2: %7.3f [mm/us]', C2))
   c=[1 -8 8*(2-mi)/(1-mi) -8/(1-mi)];
   cR=C2*sqrt(roots(c));
   cR=cR(3);
   set(htCR, 'String', sprintf('c_R: %7.3f [mm/us]', cR))
   HALF_THICKNESS=str2num(get(heTHICKNESS, 'String'))/2;

   set([hsMARKER1CURVE hsMARKER2CURVE], 'Max', size(DC,2),...
       'SliderStep', [1/(size(DC,2)-1) 10/(size(DC,2)-1)])

   set([htWAVE htPOISSON hrPHASE hrGROUP htTHICKNESS heTHICKNESS htTHICKNESS_UNITS,...
        htC1 heC1 htC1_UNITS htC2 htCR hmPLOT], 'Visible', 'on')

   if strcmp(wave,'longitudinal'),
      set(hmPLOT1, 'Label', 'c/c_2 versus k.d');
   else
      set(hmPLOT1, 'Label', 'c/c_1 versus k.d');
   end


case 'phase-group'                     % changing phase/group velocity

   set(gcbf,'WindowButtonMotionFcn','');
   if ishandle(hlCURSOR1VER),
      set(hrZOOM, 'Value',1)
      set(hrCURSOR, 'Value',0)
      delete([hlCURSOR1VER hlCURSOR1HOR hlCURSOR2VER hlCURSOR2HOR]);
      set(findobj(gcbf, 'Tag', 'MARKERS'), 'Visible', 'off')
   end
```

```
   cla

   if get(hrPHASE, 'Value'),
      if strcmp(wave,'longitudinal'),
         set(hmPLOT1, 'Label', 'c/c_2 versus k.d');
      else
         set(hmPLOT1, 'Label', 'c/c_1 versus k.d');
      end
      set(hmPLOT2, 'Label', 'c versus f.d');
      set(hmPLOT3, 'Label', 'c versus f');
      set([hmPLOT4 hmPLOT5], 'Visible', 'on');
   else
      if strcmp(wave,'longitudinal'),
         set(hmPLOT1, 'Label', 'c_g/c_2 versus k.d');
      else
         set(hmPLOT1, 'Label', 'c_g/c_1 versus k.d');
      end
      set(hmPLOT2, 'Label', 'c_g versus f.d');
      set(hmPLOT3, 'Label', 'c_g versus f');
      set([hmPLOT4 hmPLOT5], 'Visible', 'off');
   end

   set([hrZOOM hrCURSOR haAXES hmFIGURE hmADD hmDATA], 'Visible', 'off')


case 'edit thickness'

   set(gcbf,'WindowButtonMotionFcn','');
   if ishandle(hlCURSOR1VER),
      set(hrZOOM, 'Value',1)
      set(hrCURSOR, 'Value',0)
      delete([hlCURSOR1VER hlCURSOR1HOR hlCURSOR2VER hlCURSOR2HOR]);
      set(findobj(gcbf, 'Tag', 'MARKERS'), 'Visible', 'off')
   end
   cla

   HALF_THICKNESS=str2num(get(heTHICKNESS, 'String'))/2;

   set([hrZOOM hrCURSOR haAXES hmFIGURE hmADD hmDATA], 'Visible', 'off')


case 'edit C1'

   set(gcbf,'WindowButtonMotionFcn','');
   if ishandle(hlCURSOR1VER),
      set(hrZOOM, 'Value',1)
      set(hrCURSOR, 'Value',0)
      delete([hlCURSOR1VER hlCURSOR1HOR hlCURSOR2VER hlCURSOR2HOR]);
      set(findobj(gcbf, 'Tag', 'MARKERS'), 'Visible', 'off')
```

```
   end
   cla

   C1=str2num(get(heC1, 'String'));
   C2=C1*sqrt((1-2*mi)/2/(1-mi));
   set(htC2, 'String', sprintf('c_2: %7.3f [mm/us]', C2))
   c=[1 -8 8*(2-mi)/(1-mi) -8/(1-mi)];
   cR=C2*sqrt(roots(c));
   cR=cR(3);
   set(htCR, 'String', sprintf('c_R: %7.3f [mm/us]', cR))

   set([hrZOOM hrCURSOR haAXES hmFIGURE hmADD hmDATA], 'Visible', 'off')


case 'plot cc to kd'                % plotting c/c_x versus k.d

   set(hfTPDTOOL, 'Pointer', 'watch')

   set(gcbf,'WindowButtonMotionFcn','');
   if ishandle(hlCURSOR1VER),
      set(hrZOOM, 'Value',1)
      set(hrCURSOR, 'Value',0)
      delete([hlCURSOR1VER hlCURSOR1HOR hlCURSOR2VER hlCURSOR2HOR]);
      set(findobj(gcbf, 'Tag', 'MARKERS'), 'Visible', 'off')
   end
   set([haAXES hrZOOM hrCURSOR hmFIGURE hmADD hmDATA], 'Visible', 'on')
   cla

   XP=KD;
   if get(hrPHASE, 'Value'), YP=DC; else YP=DC_G; end

   plot(XP(1:STEP_GAMMA_D:end),YP(1:STEP_GAMMA_D:end,1:end), 'Tag', 'Curve')
   xlabel('k.d (2\pid/\lambda) [-]')
   if strcmp(wave,'longitudinal'),
      if get(hrPHASE, 'Value'), ylabel('c/c_2 [-]'), else ylabel('c_g/c_2 [-]'), end
   else
      if get(hrPHASE, 'Value'), ylabel('c/c_1 [-]'), else ylabel('c_g/c_1 [-]'), end
   end

   zoom on

   set(hfTPDTOOL, 'Pointer', 'arrow')


case 'plot c to fd'                % plotting c versus f.d

   set(hfTPDTOOL, 'Pointer', 'watch')

   set(gcbf,'WindowButtonMotionFcn','');
```

```
   if ishandle(hlCURSOR1VER),
      set(hrZOOM, 'Value',1)
      set(hrCURSOR, 'Value',0)
      delete([hlCURSOR1VER hlCURSOR1HOR hlCURSOR2VER hlCURSOR2HOR]);
      set(findobj(gcbf, 'Tag', 'MARKERS'), 'Visible', 'off')
   end
   set([haAXES hrZOOM hrCURSOR hmFIGURE hmADD hmDATA], 'Visible', 'on')
   cla

   if strcmp(wave,'longitudinal'), cx=C2; else cx=C1; end

   if get(hrPHASE, 'Value'), YP=DC*cx; else YP=DC_G*cx; end
   XP=zeros(size(DC));
   for i=1:size(DC,2),
      XP(:,i)=KD.*DC(:,i)*cx;
   end
   XP=XP./(2*pi);

   plot(XP(1:STEP_GAMMA_D:end,1:end),YP(1:STEP_GAMMA_D:end,1:end), 'Tag', 'Curve')
   xlabel('f.d [MHz.mm]')
   if get(hrPHASE, 'Value'), ylabel('c [mm/us]'), else ylabel('c_g [mm/us]'), end

    zoom on

   set(hfTPDTOOL, 'Pointer', 'arrow')


case 'plot c to f'                % plotting c versus f

   set(hfTPDTOOL, 'Pointer', 'watch')

   set(gcbf,'WindowButtonMotionFcn','');
   if ishandle(hlCURSOR1VER),
      set(hrZOOM, 'Value',1)
      set(hrCURSOR, 'Value',0)
      delete([hlCURSOR1VER hlCURSOR1HOR hlCURSOR2VER hlCURSOR2HOR]);
      set(findobj(gcbf, 'Tag', 'MARKERS'), 'Visible', 'off')
   end
   set([haAXES hrZOOM hrCURSOR hmFIGURE hmADD hmDATA], 'Visible', 'on')
   cla

   if strcmp(wave,'longitudinal'), cx=C2; else cx=C1; end

   if get(hrPHASE, 'Value'), YP=DC*cx; else YP=DC_G*cx; end
   XP=zeros(size(DC));
   for i=1:size(DC,2),
      XP(:,i)=KD.*DC(:,i)*cx;
   end
   XP=XP./(2*pi*HALF_THICKNESS);
```

```
   plot(XP(1:STEP_GAMMA_D:end,1:end),YP(1:STEP_GAMMA_D:end,1:end), 'Tag', 'Curve')
   xlabel('f [MHz]')
   if get(hrPHASE, 'Value'), ylabel('c [mm/us]'), else ylabel('c_g [mm/us]'), end

   zoom on

   set(hfTPDTOOL, 'Pointer', 'arrow')


case 'plot f to kd'                % plotting f versus k.d

   set(hfTPDTOOL, 'Pointer', 'watch')

   set(gcbf,'WindowButtonMotionFcn','');
   if ishandle(hlCURSOR1VER),
      set(hrZOOM, 'Value',1)
      set(hrCURSOR, 'Value',0)
      delete([hlCURSOR1VER hlCURSOR1HOR hlCURSOR2VER hlCURSOR2HOR]);
      set(findobj(gcbf, 'Tag', 'MARKERS'), 'Visible', 'off')
   end
   set([haAXES hrZOOM hrCURSOR hmFIGURE hmADD hmDATA], 'Visible', 'on')
   cla

   if strcmp(wave,'longitudinal'), cx=C2; else cx=C1; end

   XP=KD;
   YP=DC*cx;
   for i=1:size(DC,2),
      YP(:,i)=KD.*YP(:,i);
   end
   YP=YP./(2*pi*HALF_THICKNESS);

   plot(XP(1:STEP_GAMMA_D:end),YP(1:STEP_GAMMA_D:end,1:end), 'Tag', 'Curve')
   xlabel('k.d (2\pid/\lambda) [-]')
   ylabel('f [MHz]')

   zoom on

   set(hfTPDTOOL, 'Pointer', 'arrow')


case 'plot f to k'                % plotting f versus k

   set(hfTPDTOOL, 'Pointer', 'watch')

   set(gcbf,'WindowButtonMotionFcn','');
   if ishandle(hlCURSOR1VER),
      set(hrZOOM, 'Value',1)
```

```
      set(hrCURSOR, 'Value',0)
      delete([hlCURSOR1VER hlCURSOR1HOR hlCURSOR2VER hlCURSOR2HOR]);
      set(findobj(gcbf, 'Tag', 'MARKERS'), 'Visible', 'off')
   end
   set([haAXES hrZOOM hrCURSOR hmFIGURE hmADD hmDATA], 'Visible', 'on')
   cla

   if strcmp(wave,'longitudinal'), cx=C2; else cx=C1; end

   YP=DC*cx;
   for i=1:size(DC,2),
      YP(:,i)=KD.*YP(:,i);
   end
   YP=YP./(2*pi*HALF_THICKNESS);
   XP=KD./HALF_THICKNESS;

   plot(XP(1:STEP_GAMMA_D:end),YP(1:STEP_GAMMA_D:end,1:end), 'Tag', 'Curve')
   xlabel('k (2\pi/\lambda) [mm^{-1}]')
   ylabel('f [MHz]')

   zoom on

   set(hfTPDTOOL, 'Pointer', 'arrow')


case 'zoom-cursor'

   if get(hrZOOM, 'Value'),
      zoom on,
      delete([hlCURSOR1VER hlCURSOR1HOR hlCURSOR2VER hlCURSOR2HOR]);
      set(findobj(gcbf, 'Tag', 'MARKERS'), 'Visible', 'off')
   else
      zoom off,

      xlim=get(haAXES, 'XLim');
      ylim=get(haAXES, 'YLim');

      mpos = get(haAXES,'position'); % in pixels
      FIVE_PIXELS = 3.5*diff(xlim)/mpos(3);

      c1=get(hsMARKER1CURVE, 'Value');
      c2=get(hsMARKER2CURVE, 'Value');
      if size(XP,2) ==1, % XP is vector
         XC1=XP;
         XC2=XP;
      else
         XC1=XP(:,c1);
         XC2=XP(:,c2);
      end
```

```
      i1=XC1>xlim(1) & XC1<=xlim(2);
      XC1=XC1(i1);
      YC1=YP(i1,c1);
      i2=XC2>xlim(1) & XC2<=xlim(2);
      XC2=XC2(i2);
      YC2=YP(i2,c2);

      c1x=xlim(1)+(xlim(2)-xlim(1))/3;
      [m,i1]=min(abs(XC1-c1x));

      hlCURSOR1VER=line('XData',[XC1(i1) XC1(i1)],'YData',[ylim(1) ylim(2)],...
                        'EraseMode','xor','ButtonDownFcn','tp_dc_tool(''down'');',...
                        'Tag', 'Cursor1');
      hlCURSOR1HOR=line('XData',[xlim(1) xlim(2)],'YData',[YC1(i1) YC1(i1)], 'EraseMode','xor');

      c2x=xlim(2)-(xlim(2)-xlim(1))/3;
      [m,i2]=min(abs(XC2-c2x));

      hlCURSOR2VER=line('XData',[XC2(i2) XC2(i2)],'YData',[ylim(1) ylim(2)],...
                        'LineStyle', '--', 'EraseMode','xor',...
                        'ButtonDownFcn','tp_dc_tool(''down'');', 'Tag', 'Cursor2');
      hlCURSOR2HOR=line('XData',[xlim(1) xlim(2)],'YData',[YC2(i2) YC2(i2)],...
                        'LineStyle', '--', 'EraseMode','xor');

      set(gcbf,'WindowButtonMotionFcn','tp_dc_tool(''hand'');');
      set(findobj(gcbf, 'Tag', 'MARKERS'), 'Visible', 'on')

      set(heMARKER1X, 'String', num2str(XC1(i1)))
      set(htMARKER1Y, 'String', num2str(YC1(i1)))
      set(heMARKER2X, 'String', num2str(XC2(i2)))
      set(htMARKER2Y, 'String', num2str(YC2(i2)))
      set(htMARKER_DX, 'String', num2str(XC2(i2)-XC1(i1)))
      set(htMARKER_DY, 'String', num2str(YC2(i2)-YC1(i1)))
   end


case 'hand'

   currPt=get(gca,'CurrentPoint');
   if is_out(currPt, gca),
      setptr(hfTPDTOOL, 'arrow');
      return,
   end

   c1x=get(hlCURSOR1VER, 'XData');
   c2x=get(hlCURSOR2VER, 'XData');

   if abs(currPt(1,1)-c1x(1))<=FIVE_PIXELS,
      setptr(hfTPDTOOL, 'hand1');
```

```
   elseif abs(currPt(1,1)-c2x(1))<=FIVE_PIXELS,
      setptr(hfTPDTOOL, 'hand2');
   else
      setptr(hfTPDTOOL, 'arrow');
   end


case 'down'


   if strcmp(get(gco, 'Tag'), 'Cursor1'),
      set(haAXES, 'UserData', 1);
   else
      set(haAXES, 'UserData', 2);
   end

   set(gcbf,'WindowButtonMotionFcn','tp_dc_tool(''move'');')
   set(gcbf,'WindowButtonUpFcn','tp_dc_tool(''up'');')
   setptr(hfTPDTOOL, 'closedhand');


case 'move'

   currPt=get(gca,'CurrentPoint');

   if get(haAXES, 'UserData')  == 1,
      [m,i1]=min(abs(XC1-currPt(1,1)));

      set([hlCURSOR1VER],'XData',[XC1(i1) XC1(i1)]);
      set([hlCURSOR1HOR],'YData',[YC1(i1) YC1(i1)]);

      set(heMARKER1X, 'String', num2str(XC1(i1)))
      set(htMARKER1Y, 'String', num2str(YC1(i1)))

      x=get(hlCURSOR2VER,'XData');
      y=get(hlCURSOR2HOR,'YData');
      set(htMARKER_DX, 'String', num2str(x(1)-XC1(i1)))
      set(htMARKER_DY, 'String', num2str(y(2)-YC1(i1)))
   else
      [m,i2]=min(abs(XC2-currPt(1,1)));

      set([hlCURSOR2VER],'XData',[XC2(i2) XC2(i2)]);
      set([hlCURSOR2HOR],'YData',[YC2(i2) YC2(i2)]);

      set(heMARKER2X, 'String', num2str(XC2(i2)))
      set(htMARKER2Y, 'String', num2str(YC2(i2)))

      x=get(hlCURSOR1VER,'XData');
      y=get(hlCURSOR1HOR,'YData');
```

```
      set(htMARKER_DX, 'String', num2str(XC2(i2)-x(1)))
      set(htMARKER_DY, 'String', num2str(YC2(i2)-y(1)))
   end


case 'up'

   if get(haAXES, 'UserData')  == 1,
      setptr(hfTPDTOOL, 'hand1');
   else
      setptr(hfTPDTOOL, 'hand2');
   end
   set(gcbf,'WindowButtonMotionFcn','tp_dc_tool(''hand'');')
   set(gcbf,'WindowButtonUpFcn','')


case 'set cursor 1'

   x=str2num(get(gco, 'string'));
   [m,i1]=min(abs(XC1-x));

   set([hlCURSOR1VER],'XData',[XC1(i1) XC1(i1)]);
   set([hlCURSOR1HOR],'YData',[YC1(i1) YC1(i1)]);

   set(heMARKER1X, 'String', num2str(XC1(i1)))
   set(htMARKER1Y, 'String', num2str(YC1(i1)))

   x=get(hlCURSOR2VER,'XData');
   y=get(hlCURSOR2HOR,'YData');
   set(htMARKER_DX, 'String', num2str(x(1)-XC1(i1)))
   set(htMARKER_DY, 'String', num2str(y(2)-YC1(i1)))


case 'set cursor 2'

   x=str2num(get(gco, 'string'));
   [m,i2]=min(abs(XC2-x));

   set([hlCURSOR2VER],'XData',[XC2(i2) XC2(i2)]);
   set([hlCURSOR2HOR],'YData',[YC2(i2) YC2(i2)]);

   set(heMARKER2X, 'String', num2str(XC2(i2)))
   set(htMARKER2Y, 'String', num2str(YC2(i2)))

   x=get(hlCURSOR1VER,'XData');
   y=get(hlCURSOR1HOR,'YData');
   set(htMARKER_DX, 'String', num2str(XC2(i2)-x(1)))
   set(htMARKER_DY, 'String', num2str(YC2(i2)-y(2)))
```

```
case 'move slider 1'

   c1=round(get(hsMARKER1CURVE, 'Value'));
   set(htMARKER1CURVE, 'String', sprintf('Curve:  %d', c1))
   set(hsMARKER1CURVE, 'Value', c1)

   xlim=get(haAXES, 'XLim');
   ylim=get(haAXES, 'YLim');

   if size(XP,2) ==1, % XP is vector
      XC1=XP;
   else
      XC1=XP(:,c1);
   end
   i1=XC1>xlim(1) & XC1<=xlim(2);
   XC1=XC1(i1);
   YC1=YP(i1,c1);

   x=get(hlCURSOR1VER, 'XData');
   x=x(1);
   [m,i1]=min(abs(XC1-x));

   set(hlCURSOR1VER, 'XData', [XC1(i1) XC1(i1)], 'YData', [ylim(1) ylim(2)]);
   set(hlCURSOR1HOR, 'XData', [xlim(1) xlim(2)], 'YData',[YC1(i1) YC1(i1)]);

   set(heMARKER1X, 'String', num2str(XC1(i1)))
   set(htMARKER1Y, 'String', num2str(YC1(i1)))

   x=get(hlCURSOR2VER,'XData');
   set(htMARKER_DX, 'String', num2str(x(1)-XC1(i1)))
   y=get(hlCURSOR2HOR,'YData');
   set(htMARKER_DY, 'String', num2str(y(1)-YC1(i1)))


case 'move slider 2'

   c2=round(get(hsMARKER2CURVE, 'Value'));
   set(htMARKER2CURVE, 'String', sprintf('Curve:  %d', c2))
   set(hsMARKER2CURVE, 'Value', c2)

   xlim=get(haAXES, 'XLim');
   ylim=get(haAXES, 'YLim');

   if size(XP,2) ==1, % XP is vector
      XC2=XP;
   else
      XC2=XP(:,c2);
   end
```

```
   i2=XC2>xlim(1) & XC2<=xlim(2);
   XC2=XC2(i2);
   YC2=YP(i2,c2);

   x=get(hlCURSOR2VER, 'XData');
   x=x(1);
   [m,i2]=min(abs(XC2-x));

   set(hlCURSOR2VER, 'XData', [XC2(i2) XC2(i2)], 'YData', [ylim(1) ylim(2)]);
   set(hlCURSOR2HOR, 'XData', [xlim(1) xlim(2)], 'YData',[YC2(i2) YC2(i2)]);

   set(heMARKER2X, 'String', num2str(XC2(i2)))
   set(htMARKER2Y, 'String', num2str(YC2(i2)))

   x=get(hlCURSOR1VER,'XData');
   set(htMARKER_DX, 'String', num2str(XC2(i2)-x(1)))
   y=get(hlCURSOR1HOR,'YData');
   set(htMARKER_DY, 'String', num2str(YC2(i2)-y(1)))


case 'new figure'
% create new figure with the current plot copy

   hNewFigure=figure;
   hNewAxes = copyobj(haAXES, hNewFigure);
   delete(findobj(hNewAxes, 'EraseMode', 'xor'))

   set(findobj(hNewAxes, 'Type', 'line'), 'Color', 'b')
   drawnow

   set(hmADD, 'Visible', 'on')


case 'add to figure'
% add lines to figure

   hLines = findobj(haAXES, 'Type', 'line');
   if ishandle(hNewAxes),
      hCopyLines = copyobj(hLines, hNewAxes);
      set(hCopyLines, 'Color', 'r')
      delete(findobj(hNewAxes, 'EraseMode', 'xor'))
   else
      waitfor(msgbox('Sorry, lines cannot be add to figure.',...
                     'New Figure doesn''t exist', 'error', 'modal'))
      return
   end


case 'save data'
```

```
% save current plot data

   [fname,fpath]=uiputfile('*.mat','Save Dispersion Curves');
   if fname ==0, break, end

   set(hfTPDTOOL, 'Pointer', 'watch')

   saveName = fullfile(fpath,fname);

   hLines = findobj(gcbf, 'Tag', 'Curve');

   x=get(hLines(1), 'XData');
   row=length(x);
   col=length(hLines);
   X=zeros([row,col]);
   Y=zeros([row,col]);

   for i=1:length(hLines),
      x = get(hLines(i), 'XData');
      y = get(hLines(i), 'YData');
      X(:,i)=x';
      Y(:,i)=y';
   end

   label_x = get(get(haAXES, 'XLabel'), 'String');
   label_y = get(get(haAXES, 'YLabel'), 'String');

   save(saveName, 'X', 'Y', 'label_x', 'label_y')

   set(hfTPDTOOL, 'Pointer', 'arrow')


case 'close'
   delete(gcbf)

   clear global hfTPDTOOL htWAVE htPOISSON hrPHASE hrGROUP htTHICKNESS heTHICKNESS htTHICKNESS_UNITS
   clear global htC1 heC1 htC1_UNITS htC2 htCR hrZOOM hrCURSOR haAXES hNewAxeshmFIGURE hmADD hmDATA
   clear global hmPLOT hmPLOT1 hmPLOT2 hmPLOT3 hmPLOT4 hmPLOT5
   clear global wave mi KD DC DC_G
   clear global C1 C2 HALF_THICKNESS loadName
   clear global hlCURSOR1VER hlCURSOR1HOR hlCURSOR2VER hlCURSOR2HOR XP YP XC YC FIVE_PIXELS
   clear global htMARKER1 htMARKER1CURVE htMARKER1XLABEL heMARKER1X htMARKER1YLABEL htMARKER1Y
   clear global htMARKER2 htMARKER2CURVE htMARKER2XLABEL heMARKER2X htMARKER2YLABEL htMARKER2Y
   clear global htMARKER_DXLABEL htMARKER_DX htMARKER_DYLABEL htMARKER_DY hsMARKER1CURVE hsMARKER2CURVE
   clear global STEP_GAMMA_D


otherwise
   fprintf('The action: %s is wrong.', action)
```

```
end



%---------------------------------------------------------------------
function create_tpdtool_gui;
% CREATE_TPDTOOL_GUI Creates the GUI for tp_dc_tool - the data manager

global hfTPDTOOL htWAVE htPOISSON hrPHASE hrGROUP htTHICKNESS heTHICKNESS htTHICKNESS_UNITS
global htC1 heC1 htC1_UNITS htC2 htCR hrZOOM hrCURSOR haAXES hmFIGURE hmADD hmDATA
global hmPLOT hmPLOT1 hmPLOT2 hmPLOT3 hmPLOT4 hmPLOT5
global htMARKER1 htMARKER1CURVE htMARKER1XLABEL heMARKER1X htMARKER1YLABEL htMARKER1Y
global htMARKER2 htMARKER2CURVE htMARKER2XLABEL heMARKER2X htMARKER2YLABEL htMARKER2Y
global htMARKER_DXLABEL htMARKER_DX htMARKER_DYLABEL htMARKER_DY hsMARKER1CURVE hsMARKER2CURVE


bkcolor=[0.8 0.8 0.8];

pos=get(0, 'ScreenSize');
x1=(pos(3)-800)/2+1;
x2=(pos(4)-600)/2+1;

hfTPDTOOL = figure('Color',bkcolor, ...
   'Position',[x1 x2 800 600], ...
   'ToolBar','none',...
   'MenuBar','none',...
   'NumberTitle','off',...
   'Name','Thick Plate Dispersion Curves - Tool');


htWAVE = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[15 570 110 20], ...
   'String','Wave', ...
   'Visible', 'off');

htPOISSON = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[15 550 110 20], ...
   'String','Poisson''s Ratio', ...
   'Visible', 'off');


hrPHASE = uicontrol('Style','radiobutton', ...
   'BackgroundColor',bkcolor, ...
```

```
   'HorizontalAlignment','left', ...
   'Position',[140 572 150 20], ...
   'String','Phase velocity', ...
   'Value', 1, ...
   'Visible', 'off');

hrGROUP = uicontrol('Style','radiobutton', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[140 552 150 20], ...
   'String','Group velocity', ...
   'Value', 0, ...
   'Visible', 'off');

RADIO=[hrPHASE hrGROUP];
n=length(RADIO);
for i=1:n,
  set(RADIO(i), 'UserData', RADIO(:,[1:(i-1),(i+1):n]));
end

call=['me=get(gcf, ''CurrentObject'');',...
      'if (get(me, ''Value'') == 1),',...
         'set(get(me, ''UserData''), ''Value'', 0),',...'
      'else,',...
         'set(me, ''Value'', 1),',...
      'end;',...
      'clear me;', ...
      'tp_dc_tool(''phase-group'');'];

set(RADIO, 'CallBack', call);


htTHICKNESS = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[260 570 80 20], ...
   'String','Thickness (2d):', ...
   'Visible', 'off');

heTHICKNESS = uicontrol('Style','edit',...
   'BackgroundColor',[1 1 1], ...
   'Callback', 'tp_dc_tool(''edit thickness'');', ...
   'HorizontalAlignment','right', ...
   'Position',[340 573 50 20], ...
   'String','10', ...
   'TooltipString','Set plate thickness in mm', ...
   'Visible', 'off');

htTHICKNESS_UNITS = uicontrol('Style','text', ...
```

```
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[395 570 30 20], ...
   'String','[mm]', ...
   'Visible', 'off');


htC1 = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[260 550 80 20], ...
   'String','c_1:', ...
   'Visible', 'off');

heC1 = uicontrol('Style','edit', ...
   'BackgroundColor',[1 1 1], ...
   'Callback', 'tp_dc_tool(''edit C1'');', ...
   'HorizontalAlignment','right', ...
   'Position',[340 553 50 20], ...
   'String','5.950', ...
   'Visible', 'off');

htC1_UNITS = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[395 550 40 20], ...
   'String','[mm/us]', ...
   'Visible', 'off');

htC2 = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[455 570 100 20], ...
   'Visible', 'off');


htCR = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[455 550 100 20], ...
   'Visible', 'off');


hrZOOM = uicontrol('Style','radiobutton', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[680 570 105 20], ...
   'String','Zoom', ...
   'Value', 1, ...
```

```
   'Visible', 'off');

hrCURSOR = uicontrol('Style','radiobutton', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[680 550 105 20], ...
   'String','Cursor', ...
   'Value', 0, ...
   'Visible', 'off');

RADIO=[hrZOOM hrCURSOR];
n=length(RADIO);
for i=1:n,
  set(RADIO(i), 'UserData', RADIO(:,[1:(i-1),(i+1):n]));
end

call=['me=get(gcf, ''CurrentObject'');',...
      'if (get(me, ''Value'') == 1),',...
         'set(get(me, ''UserData''), ''Value'', 0),',...'
      'else,',...
         'set(me, ''Value'', 1),',...
      'end;',...
      'clear me;', ...
      'tp_dc_tool(''zoom-cursor'');'];

set(RADIO, 'CallBack', call);


htMARKER1 = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[680 495 90 20], ...
   'String','First Cursor', ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

htMARKER1CURVE = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[680 470 90 20], ...
   'String','Curve:  1', ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

hsMARKER1CURVE = uicontrol('Style','slider', ...
   'BackgroundColor',bkcolor, ...
   'Position',[680 453 90 20], ...
   'Min', 1, ...
   'Value', 1, ...
```

```
   'Callback', 'tp_dc_tool(''move slider 1'');', ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

htMARKER1XLABEL = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[680 430 15 20], ...
   'String','x:', ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

heMARKER1X = uicontrol('Style','edit', ...
   'Callback', 'tp_dc_tool(''set cursor 1'')', ...
   'BackgroundColor',[1 1 1], ...
   'HorizontalAlignment','right', ...
   'Position',[700 430 70 20], ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

htMARKER1YLABEL = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[680 410 15 20], ...
   'String','y:', ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

htMARKER1Y = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','right', ...
   'Position',[700 410 70 20], ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

htMARKER2 = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[680 365 90 20], ...
   'String','Second Cursor', ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

htMARKER2CURVE = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[680 340 90 20], ...
   'String','Curve:  1', ...
   'Visible', 'off', ...
```

```
   'Tag', 'MARKERS');

hsMARKER2CURVE = uicontrol('Style','slider', ...
   'BackgroundColor',bkcolor, ...
   'Position',[680 323 90 20], ...
   'Min', 1, ...
   'Value', 1, ...
   'Callback', 'tp_dc_tool(''move slider 2'');', ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

htMARKER2XLABEL = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[680 300 15 20], ...
   'String','x:', ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

heMARKER2X = uicontrol('Style','edit', ...
   'Callback', 'tp_dc_tool(''set cursor 2'')', ...
   'BackgroundColor',[1 1 1], ...
   'HorizontalAlignment','right', ...
   'Position',[700 300 70 20], ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

htMARKER2YLABEL = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[680 280 15 20], ...
   'String','y:', ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

htMARKER2Y = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','right', ...
   'Position',[700 280 70 20], ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

htMARKER_DXLABEL = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[680 250 15 20], ...
   'String','dx:', ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');
```

```
htMARKER_DX = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','right', ...
   'Position',[700 250 70 20], ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

htMARKER_DYLABEL = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','left', ...
   'Position',[680 230 15 20], ...
   'String','dy:', ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');

htMARKER_DY = uicontrol('Style','text', ...
   'BackgroundColor',bkcolor, ...
   'HorizontalAlignment','right', ...
   'Position',[700 230 70 20], ...
   'Visible', 'off', ...
   'Tag', 'MARKERS');


haAXES = axes('Units','pixels', ...
   'CameraUpVector',[0 1 0], ...
   'Color',[1 1 1], ...
   'Position',[90 70 550 450], ...
   'XColor',[0 0 0], ...
   'YColor',[0 0 0], ...
   'ZColor',[0 0 0], ...
   'Visible', 'off');


hmFILE = uimenu('Label', 'File');
uimenu(hmFILE, 'Label', '&Open...', 'Callback', 'tp_dc_tool(''open'')');
hmFIGURE = uimenu(hmFILE, 'Label', 'New &Figure', 'Callback', 'tp_dc_tool(''new figure'');',...
   'Separator', 'on', 'Visible', 'off');
hmADD = uimenu(hmFILE, 'Label', '&Add to Figure', 'Callback', 'tp_dc_tool(''add to figure'');',...
   'Visible', 'off');
hmDATA = uimenu(hmFILE, 'Label', 'Save &Data As...', 'Callback', 'tp_dc_tool(''save data'');',...
   'Visible', 'off');
uimenu(hmFILE, 'Label', '&Close', 'Callback', 'tp_dc_tool(''close'')', 'Separator', 'on');

hmPLOT = uimenu('Label', 'Plot', 'Visible', 'off');
hmPLOT1 = uimenu(hmPLOT, 'Label', 'c/c_x versus k.d', 'Callback', 'tp_dc_tool(''plot cc to kd'')');
hmPLOT2 = uimenu(hmPLOT, 'Label', 'c versus f.d', 'Callback', 'tp_dc_tool(''plot c to fd'')');
hmPLOT3 = uimenu(hmPLOT, 'Label', 'c versus f', 'Callback', 'tp_dc_tool(''plot c to f'')');
hmPLOT4 = uimenu(hmPLOT, 'Label', 'f versus k.d', 'Callback', 'tp_dc_tool(''plot f to kd'')');
```

```
hmPLOT5 = uimenu(hmPLOT, 'Label', 'f versus k', 'Callback', 'tp_dc_tool(''plot f to k'')');

set([findobj('type', 'axes'); findobj('type','uicontrol')], 'units', 'normalized')



% ====================================
function result = is_out(p, ha);

xlim=get(ha, 'XLim');
ylim=get(ha, 'YLim');

if p(1,1)<xlim(1) | p(1,1)>xlim(2) | p(1,2)<ylim(1) | p(1,2)>ylim(2),
   result = 1;
else
   result = 0;
end
```

Příloha J

wilcox.m

```
function [t, y, yn, ynob, o, fy, h]=wilcox(T);
%
%
% [t, y, yn, ynob, o, fy, h]=wilcox(T);
%
% T ... max time [us] (100)
% for fine evaluation T=1000 !!!!
%

% (c) 2001, Petr Hora


% Prediction of dispersive propagation
% using Fourier decomposition

% load the first five dispersion curves
load('CurveL')
DCL=DC(:,1:5);
load('CurveL_g');
DCL_G=DC(:,1:5);
load('CurveS');
DCS=DC(:,1:5);
load('CurveS_g');
DCS_G=DC(:,1:5);
clear DC

GD=GD';
% mi

f_centre=2;   % MHz
No_cycles=5;
T_puls=No_cycles/f_centre; % us
f_sampl=f_centre*(32);  % MHz
dt_sampl=1/f_sampl;  % us
N=2.^nextpow2(round(T/dt_sampl)+1);
```

```
t=[0:N-1]*dt_sampl;   % us
T=t(end);  % us

% y=sin(2*pi*f_centre*t).*sin(2*pi*f_centre*t/(2*No_cycles));
w=[hann(T_puls/dt_sampl); zeros([N-T_puls/dt_sampl,1])]';
y=w.*sin(2*pi*f_centre*t);

y(t>T_puls)=0;

fy=fft(y);

o=2*pi/T*[0:N-1];  % MHz

C1=5.95;  % mm/us
C2=C1*sqrt((1-2*mi)/2/(1-mi));
HALF_THICKNESS=0.5;  % mm

curve=1;
% longitudinal (S)
DC=DCL(:,curve);
C=DC*C2;   % mm/us

% shear (A)
% DC=DCS(:,curve);
% C=DC*C1;   % mm/us

O=GD.*C./(HALF_THICKNESS);   % MHz

% for the first curve
c=interp1(O,C,o,'spline');

% for others
% c=interp1(O,C,o,'cubic', Inf);

index=find(isnan(c));
if ~isempty(index), c(index)=c(index(end)+1); end

H=zeros(size(N));
H(2:N/2)=j;
H(N/2+2:N)=-j;

distance=0:0.5:100;  % mm
yn=zeros(length(distance),N);
yn2max=yn;
ynob=yn;
ynob2max=yn;
row=1;
hwait = waitbar(0,'Please wait...');
for x=distance,
```

```
  % x
  h=exp(-i*o./c*x);
  h(N/2+2:N)=conj(h(N/2:-1:2));
  h(N/2+1)=real(h(N/2+1));

  yn(row,:)=real(ifft(fy.*h));
  yn2max(row,:)=yn(row,:)/max(yn(row,:));
  ynob(row,:)=sqrt(real(ifft(fft(yn(row,:)).*H)).^2+yn(row,:).^2);
  ynob2max(row,:)=ynob(row,:)/max(ynob(row,:));

  row=row+1;
  waitbar(row/length(distance),hwait)
end

close(hwait)


% Prediction of dispersive propagation
% using group velocity dispersion curves

% get f_min and f_max
fya=abs(fy(1:length(fy)/2));
fyla=10*log(fya/max(fya));
i=find(fyla>-20);
i1=[i(1)-1:i(1)];
f_min=interp1(fyla(i1),o(i1)/2/pi,-20);
i2=[i(end):i(end)+1];
f_max=interp1(fyla(i2),o(i2)/2/pi,-20);


% get cg_min and cg_max
YP=DCL_G*C2;
XP=zeros(size(DCL));
for i=1:size(DCL,2),
  XP(:,i)=GD.*DCL(:,i)*C2;
end
XP=XP./(2*pi*HALF_THICKNESS);

X=linspace(f_min,f_max,100);
Y=interp1(XP(:,curve), YP(:,curve), X,'spline');
cg_min=min(Y);
cg_max=max(Y);



% PLOT

figure
plot(t,y,'b')
```

```
axis([0 T_puls -1 1])
xlabel('time [\mus]')
ylabel('amplitude [-]')
title('Source signal')


figure
plot(o(1:length(o)/2)/2/pi,fyla,'b',...
   [f_min f_min NaN f_max f_max],[0 -30 NaN 0 -30],'r')
text(f_min, 0, sprintf('f_{min} (%4.2f MHz)', f_min),...
     'vertical','bottom', 'horizontal', 'right','rotation',90)
text(f_max, 0, sprintf('f_{max} (%4.2f MHz)', f_max),...
     'vertical','top', 'horizontal', 'right','rotation',90)
axis([f_min-(f_max-f_min)/2 f_max+(f_max-f_min)/2 -30 10])
xlabel('frequency [MHz]')
ylabel('velocity [dB]')
title('Source signal spectrum')


figure
plot(XP(1:1:end,1:end),YP(1:1:end,1:end))
hold on
plot([f_min f_min NaN f_max f_max],[0 6 NaN 0 6],'r:')
plot([0 f_max NaN 0 f_max],[cg_max cg_max NaN cg_min cg_min],'g:')
plot(X,Y,'LineWidth', 3)
text(f_min, 0, sprintf(' f_{min} (%4.2f)', f_min),...
     'vertical','bottom', 'horizontal', 'left','rotation',90)
text(f_max, 0, sprintf(' f_{max} (%4.2f)', f_max),...
     'vertical','bottom', 'horizontal', 'left','rotation',90)
text(0, cg_max, sprintf(' c_{g,max} (%4.2f)', cg_max),...
     'vertical','top', 'horizontal', 'left','rotation',0)
text(0, cg_min, sprintf(' c_{g,min} (%4.2f)', cg_min),...
     'vertical','bottom', 'horizontal', 'left','rotation',0)
axis([0 6 0 6])
xlabel('f [MHz]')
ylabel('c_g [mm/us]')
title('Group velocity dispersion curves')


% figure
% plot(O,C,'b', o,c,'r+')
% xlabel('\omega [MHz]')
% ylabel('velocity [mm/ \mus]')

figure
imagesc(t,distance,yn)
colormap(1-gray)
caxis([-1 1])
axis([0 2*T_puls+distance(end)/cg_min 0 distance(end)])
```

```
colorbar
xlabel('time [\mus]')
ylabel('distance [mm]')
title('Time traces')

% figure
% pcolor(t,distance,yn)
% set(gca,'YDir','reverse')
% shading interp
% colormap(1-gray)
% xlabel('time [\mus]')
% ylabel('distance [mm]')
% title('Time traces')

figure
imagesc(t,distance,10*log(ynob))
hold on
plot([0 distance(end)/cg_max], [0 distance(end)],'r')
plot([T_puls T_puls+distance(end)/cg_min], [0 distance(end)],'r')
colormap(1-gray)
caxis([-20 0])
axis([0 2*T_puls+distance(end)/cg_min 0 distance(end)])
colorbar
xlabel('time [\mus]')
ylabel('distance [mm]')
title('Relative envelope amplitude, references = global max.')

figure
imagesc(t,distance,10*log(ynob2max))
hold on
d=distance(end);
plot([0 d/cg_max NaN T_puls T_puls+d/cg_min], [0 d NaN 0 d],'r')
text(d/2/cg_max, d/2, 'c_{g,max} ',...
   'vertical','middle', 'horizontal', 'right', 'rotation', 0)
text(T_puls+d/2/cg_min, d/2, ' c_{g,min}',...
   'vertical','middle', 'horizontal', 'left', 'rotation', 0)
colormap(1-gray)
caxis([-20 0])
axis([0 2*T_puls+distance(end)/cg_min 0 distance(end)])
colorbar
xlabel('time [\mus]')
ylabel('distance [mm]')
title('Relative envelope amplitude, references = local max.')

% figure
% pcolor(t,distance,ynob)
% set(gca,'YDir','reverse')
% shading interp
% colormap(1-gray)
```

```
% xlabel('time [\mus]')
% ylabel('distance [mm]')
% title('Pcolor, envelope of signal')

% figure
% for i=1:length(distance),
%   plot(t,yn(i,:))
%   axis([t(1) t(end) -1 1])
%   drawnow
% end
% xlabel('time [\mus]')
% title('Movie, signal')

% figure
% for i=1:length(distance),
%   plot(t,ynob(i,:))
%   axis([t(1) t(end) 0 1])
%   drawnow
% end
% xlabel('time [\mus]')
% title('Movie, envelope of signal')
```

Literatura

[AB87] Alers, G. A.; Burns, L. R.: EMAT designs for special applications. Mater. Eval., 45, 1987, 1184-1194

[Chi97] Chimenti, D. E.: Guided waves in plates and their use in materials characterization. Appl. Mech. Rev., 50, 1997, 247-284

[MWC97] Monkhouse, R. S. C.; Wilcox, P.; Cawley, P.: Flexible inter-digital PVDF transducers for the generation of Lamb waves in structures. Ultrasonics, 35, 1997, 489-498

[WLC01] Wilcox, P.; Lowe, M.; Cawley, P.: The effect of dispersion on long-range inspection using ultrasonic guided waves. NDT&E International, 34, 2001, 1-9

[ACl91] Alleyne D., Cawley P.: A two-dimensional Fourier transform method for the measurement of propagating multimode signals. J. Acoust. Soc. Am., 1991, 89, 3,1159-1168, March

[Mik78] Miklowitz J.: The theory of elastic waves and waveguides. North-Holland Publishing Company, Amsterdam, 1978

[Gra75] Graff K. F.: Wave motion in elastic solids. New York: Dover, 1975

[Bre60] Brekhovskikh L. M.: Waves in layered media. Academic, New York, 1960

[SP78] Sachse W., Pao Y-H.: On determination of phase and group velocities of dispersive waves in solids. J. Appl. Phys., 1978, 49, 4320-4327

[MJQ99] Moser F., Jacobs L. J., Qu J.: Modeling elastic wave propagation in waveguides with the finite element method. NDT&E International, 1999, 32, 225-234