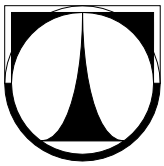
TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií



DIPLOMOVÁ PRÁCE

Liberec 2013 Bc. Jaroslav Ševic

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N3901 - Aplikované vědy v inženýrství
Studijní obor: 3901T025 - Přírodovědné inženýrství

Výpočty analytického řešení systému PDR pro testování XFEM metod

Calculations of analytical solution to a system of PDE for testing XFEM methods

Diplomová práce

Autor: Bc. Jaroslav Ševic

Vedoucí práce: Mgr. Jan Březina, Ph.D.

V Liberci 15. května 2013

[Zadání diplomové práce]

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 - školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce.

Datum: 17. května 2013

Podpis:

Poděkování

Chtěl bych poděkovat vedoucímu diplomové práce Mgr. Janu Březinovi, Ph.D. za obětavou spolupráci i za čas, který mi při konzultacích věnoval a za podporu při řešení problémů.

Abstrakt

Jeden ze způsobů jak lze řešit složité problémy puklinového proudění je homogenizace malých puklin v kombinaci s explicitním popisem větších puklin jako 2D respektive 1D objekty. Aby se předešlo problému při generování sítě, je možné použít nekompatibilní sítě různých dimenzí, což ale vyžaduje úpravu XFEM nebo mortar metod pro popis vazeb rovnic na různých dimenzích. Cílem této práce je navrhnout netriviální testovací problém těchto nových metod a odvodit jeho analytické řešení. Analytické řešení je odvozeno ve formě Fourierovy řady pro dva modely propojením mezi puklinou a kontinuálním prostředím. Řešení je ověřeno metodou konečných diferencí. Výpočet analytického řešení je implementován v jazyku Python jako programovatelný filtr pro software ParaView.

Klíčová slova

Puklinové proudění, Fourierova metoda, metoda sítí, ParaView

Abstract

One way how to deal with complex fracture flow problems is homogenization of the small fractures combined with an explicit description of the larger fractures as 2D or 1D objects respectively. In order to avoid problems during mesh generation, one can use non-compatible meshing of different dimensions, which in turn requires adaptation of XFEM or mortar method to describe the coupling of equations living on different dimensions. The aim of this work is to propose a nontrivial test problem for these new methods and derive its analytical solution. The analytical solution is derived in the form of Fourier series for the two models of coupling between the fracture and the continuum domain. The solution is verified against finite difference method. Calculation of the analytical solution is implemented in Python as a programmable filter into the ParaView software.

Keywords

Fracture flow, Fourier method, finite difference method, ParaView

Obsah

Seznam tabulek

Seznam obrázků

Úvod

Zabýváme se obecnou úlohou puklinového proudění. Modelujeme propustnost a rozložení tlaku kapaliny v horninovém prostředí obsahujícím pukliny. Složité úlohy puklinového proudění řešíme homogenizací malých puklin a velké pukliny popisujeme jako objekty nižší dimenze. Simulace provádíme na kombinované síti, kde elementy sítě nižší dimenze reprezentují pukliny a elementy vyšší dimenze pokrývají okolní homogenizované horninové prostředí.

Problémem je vytvoření vhodné kombinované sítě. V ideálním případě by síť měla být kompatibilní, to znamená, že elementy sítě nižší dimenze, těsně sousedí s elementy sítě vyšší dimenze a nemají žádný společný průnik. Takovou síť, zvláště pro složitější úlohy, je velmi složité vygenerovat. Proto ve většině případů máme síť nekompatibilní, kde elementy nižší dimenze protínají elementy sítě vyšší dimenze, což ale výrazně ztěžuje výpočty.

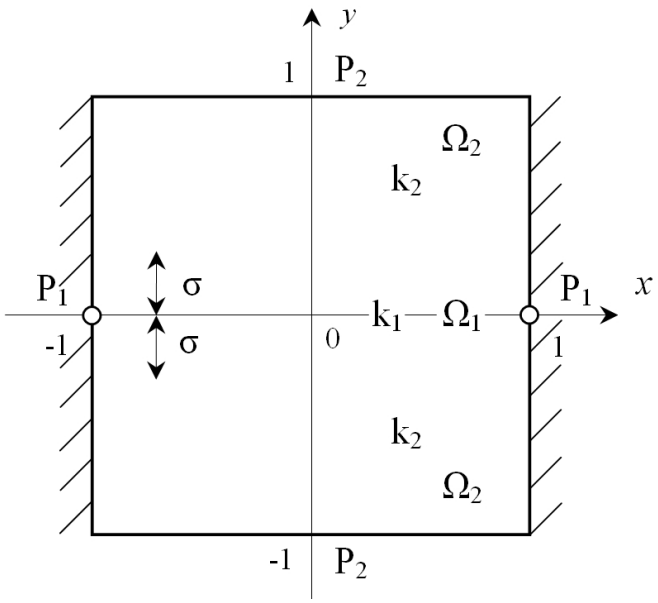
Cílem práce je nalézt analytické řešení úlohy puklinového prostředí, implementovat ho a provádět výpočty s co nejvyšší přesností. Účelem je testování a ověření přesnosti výsledků získaných jinými numerickými metodami. V tomto případě se jedná hlavně o software Flow123d, pro který již existují připravené kompatibilní sítě konkrétní úlohy.

V této práci hledáme analytické řešení zjednodušené úlohy proudění. Úloha je definovaná jako čtvercová oblast o rozměrech $< -1, 1 > \times < -1, 1 >$, která je v polovině horizontálně rozdělena puklinou. Vlastnosti prostředí jsou popsány koeficienty hydraulické vodivosti pukliny k_1 a okolního prostředí k_2. Dále je definována propustnost pukliny koeficientem vodní výměny mezi puklinou a okolím σ. Jsou definovány Dirichletovy okrajové podmínky, hodnota tlaku na dolní a horní hraně oblasti P_2 a velikost tlaku na koncích pukliny P_1. Boční stěny oblasti jsou definovány jako nepropustné nulovou Neumannovu okrajovou podmínkou.

Oblast byla takto definována, protože její jednoduchá geometrie umožňuje řešení pomocí Fourierovy metody a metody sítí. Řešení je v okolí pukliny netriviální, což je vhodné pro testování metod pro popis vazeb mezi sítěmi.

Řešíme dva modely vazby mezi dimenzemi, model se spojitým tlakem a model s nespojitým tlakem kolem pukliny.

Nejprve popíšeme model se spojitým tlakem, kdy pro celou oblast uvažujeme konstantní hodnoty parametrů popisujících vlastnosti pukliny a okolního prostředí. Schéma spojitého modelu je znázorněno na obrázku 1.



Obr. 1: Schéma spojitého modelu

Uvažujeme 2D oblast $\Omega_2 = (-1, 1) \times (-1, 1) \subset R^2$ a na ní Darcyho rovnici proudění vody v prostředí s okrajovými podmínkami

$$-k_2 \Delta p_2(x, y) = 0 \qquad \text{na } \Omega_2. \tag{1}$$

Definujeme Dirichletovy okrajové podmínka zaručující konstantní tlak na horní a dolní hraně oblasti

$$p_2(x, 1) = p_2(x, -1) = P_2. \tag{2}$$

Neumannova okrajová podmínka popisuje komunikaci s puklinou, ta je stejná pro obě poloviny oblasti sousedící s puklinou

$$k_2 \frac{\partial p_2}{\partial y}(x, 0) = \sigma(p_2(x, 0) - p_1(x)). \tag{3}$$

Druhá Neumannova okrajová podmínka určuje nulovou derivaci na obou bočních stěnách, tedy nulový tok stěnami

$$\frac{\partial p_2}{\partial x}(-1, y) = \frac{\partial p_2}{\partial x}(1, y) = 0. \tag{4}$$

Puklinu popíšeme jako 1D oblast $\Omega_1 = (-1, 1) \subset R$. Chování pukliny a její komunikace s okolním prostředím je popsána rovnicí

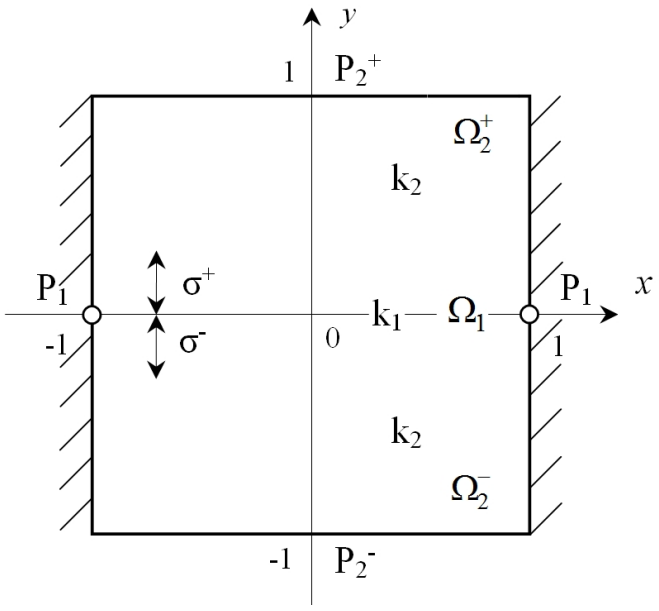
$$-k_1 p_1''(x) = 2\sigma(p_2(x, 0) - p_1(x)) \qquad \text{na } \Omega_1. \tag{5}$$

Dále máme definovanou Dirichletovou okrajovou podmínkou určující konstantní tlak na koncích pukliny a Neumannovou okrajovou podmínkou, která zajišťuje nulový tok na konci pukliny

$$p_1(-1) = p_1(1) = P_1, \qquad p_1'(0) = 0, \tag{6}$$

kde k_1 je hydraulická vodivost na Ω_1 a k_2 hydraulická vodivost na Ω_2, σ je koeficient výměny vody mezi puklinou a okolním prostředím. P_1 a P_2 jsou hodnoty Dirichletových okrajových podmínek na hranicích oblastí Ω_1 a Ω_2.

Obdobným způsobem popíšeme model s nespojitým tlakem, kdy pro každou polovinu oblasti uvažujeme různé hodnoty parametrů. Schéma nespojitého modelu je znázorněno na obrázku 2.



Obr. 2: Schéma nespojitého modelu

Horní polovinu popíšeme jako 2D oblast $\Omega_2^+ = (-1, 1) \times (0, 1) \subset R^2$, dolní polovina je 2D oblast $\Omega_2^- = (-1, 1) \times (-1, 0) \subset R^2$. Pro každou z nich napíšeme Darcyho rovnici proudění s okrajovými podmínkami

$$\begin{aligned} -k_2 \Delta p_2^+(x, y) &= 0 \qquad \text{na } \Omega_2^+, \\ -k_2 \Delta p_2^-(x, y) &= 0 \qquad \text{na } \Omega_2^-. \end{aligned} \tag{7}$$

Definujeme Dirichletovy okrajové podmínka zaručující konstantní tlak na horní a dolní hraně oblasti

$$p_2^+(x,1) = P_2^+, \qquad p_2^-(x,-1) = P_2^-. \tag{8}$$

Neumannova okrajová podmínka popisuje komunikaci s puklinou, musí se uvažovat zvlášť z každé strany pukliny

$$\begin{aligned} k_2 \frac{\partial p_2^+}{\partial y}(x,0^+) &= \sigma(p_2^+(x,0^+) - p_1(x)), \\ k_2 \frac{\partial p_2^-}{\partial y}(x,0^-) &= \sigma(p_2^-(x,0^-) - p_1(x)). \end{aligned} \tag{9}$$

Druhá Neumannova okrajová podmínka určuje nulovou derivaci na obou bočních stěnách, tedy nulový průtok stěnami

$$\frac{\partial p_2^+}{\partial x}(-1,y) = \frac{\partial p_2^+}{\partial x}(1,y) = \frac{\partial p_2^-}{\partial x}(-1,y) = \frac{\partial p_2^-}{\partial x}(1,y) = 0. \tag{10}$$

Puklinu popíšeme stejně jako u spojitého modelu jako 1D oblast $\Omega_1 = (-1,1) \subset R$. Puklina tentokrát komunikuje nezávisle s oběma polovinami okolní oblasti. Její chování popíšeme rovnicí

$$-k_1 p_1''(x) = \sigma^+(p_2^+(x,0^+) - p_1(x)) + \sigma^-(p_2^-(x,0^-) - p_1(x)) \qquad \text{na } \Omega_1. \tag{11}$$

Dirichletovy okrajové podmínky pro puklinu jsou stejné jako v předchozím případě

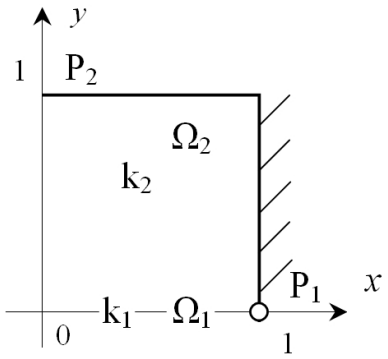
$$p_1(-1) = p_1(1) = P_1, \qquad p_1'(0) = 0. \tag{12}$$

1 Analytické řešení úlohy proudění

V této kapitole budeme hledat analytické řešení obou modelových úloh a odvodíme vztahy pro výpočet rozložení tlaku na puklině a okolní oblasti.

1.1 Řešení modelu se spojitým tlakem

Úloha se spojitým tlakem má na celé oblasti stejné parametry, je tedy symetrická podle obou souřadných os a můžeme tak řešenou oblast omezit jen na jednu čtvrtinu, jak je vidět na obrázku 3.



Obr. 3: Zjednodušená oblast pro spojitý model

Vycházíme z rovnic (1) - (6) popisujících celou oblast, které musíme omezit na námi řešenou zjednodušenou 2D oblast $\Omega_2 = (0,1) \times (0,1) \subset R^2$. Dostaneme tak rovnice s okrajovými podmínkami

$$-k_2 \Delta p_2(x,y) = 0 \qquad \text{na } \Omega_2, \tag{13}$$

$$p_2(x,1) = P_2, \tag{14}$$

$$k_2 \frac{\partial p_2}{\partial y}(x,0) = \sigma(p_2(x,0) - p_1(x)), \tag{15}$$

$$\frac{\partial p_2}{\partial x}(0,y) = \frac{\partial p_2}{\partial x}(1,y) = 0. \tag{16}$$

Dále definujeme 1D oblast $\Omega_1 = (0,1) \subset R$, reprezentující puklinu. Chování pukliny a její komunikace s okolním prostředím je popsána rovnicí s okrajovými podmínkami

$$-k_1 p_1''(x) = \sigma(p_2(x,0) - p_1(x)) \qquad \text{na } \Omega_1, \tag{17}$$

$$p_1(0) = P_1 \qquad p_1'(1) = 0, \tag{18}$$

1.1.1 Analytické řešení

Odvození řešení pro spojitý model již bylo provedeno v práci prof. RNDr. Pavla Burdy, CSc. a Mgr. Jana Březiny, Ph.D., která ale nebyla dosud publikována. Proto zde uvedu jen nejdůležitější body řešení a konečné výsledky. Podrobné odvození bude provedeno v další podkapitole pro složitější nespojitý model, protože ten již nebyl součástí zmíněné práce.

Nejprve je řešena rovnice (13) pomocí Fourierovy metody. Jsou použity okrajové podmínky (14) a (16), čímž získáme rovnici pro $p_2(x, y)$ na Ω_2

$$p_2(x, y) = P_2 - B_0 + B_0 y - \sum_{n=1}^{\infty} A_n \cos(n\pi x) \sinh(n\pi(1 - y)).$$

Dále je hledáno řešení rovnice (17), popisující komunikace pukliny s okolním prostředím, s okrajovými podmínkami (18). Tak získáme rovnici pro $p_1(x)$ na Ω_1

$$p_1(x) = \Big(P_1 - P_2 + B_0 + \sum_{n=1}^{\infty} \tilde{u}_n\Big) \frac{\cosh(\sqrt{\frac{\sigma}{k_1}}(1 - x))}{\cosh \sqrt{\frac{\sigma}{k_1}}} + P_2 - B_0 - \sum_{n=1}^{\infty} \tilde{u}_n \cos(n\pi x).$$

Poté jsou počítány chybějící neznámé parametry. Po dalších úpravách a dosazení dostaneme konečné vztahy pro řešení $p_1(x)$ na oblasti Ω_1 a $p_2(x, y)$ na oblasti Ω_2

$$p_2(x, y) = P_2 - B_0 + B_0 y - 2B_0 \sum_{n=1}^{\infty} \left(a_n \cos(\pi n x) \sinh(n\pi(1 - y))\right), \tag{19}$$

$$p_1(x) = P_2 - B_0 - u_0 \cosh((1 - x)\sqrt{\frac{\sigma}{k_1}}) - 2B_0 \sum_{n=1}^{\infty} u_n \cos(\pi n x), \tag{20}$$

kde

$$a_n = \frac{\sigma k_2}{k_2 \pi n \cosh(\pi n)(\sigma + k_1 \pi^2 n^2) + \sigma k_1 \pi^2 n^2 \sinh(\pi n)}, \tag{21}$$

$$u_n = \frac{a_n \sigma \sinh(\pi n)}{\sigma + k_1 \pi^2 n^2}\ , \tag{22}$$

$$B_0 = \frac{(P_2 - P_1)\sigma\sqrt{k_1}}{\sqrt{\sigma} k_2 \coth(\sqrt{\frac{\sigma}{k_1}}) + \sigma\sqrt{k_1} + 2\sigma\sqrt{k_1} \sum_{n=1}^{\infty} u_n}, \tag{23}$$

$$u_0 = \frac{k_2 B_0}{\sqrt{k_1}\sqrt{\sigma} \sinh(\sqrt{\frac{\sigma}{k_1}})}. \tag{24}$$

Z důvodu zvýšení přesnosti výpočtu při implementaci vzorců do Matlabu nebo Pythonu, nahradíme hyperbolické funkce exponenciálními funkcemi. Děláme to proto, že v našem případě mají hyperbolické funkce obsažené v sumách díky proměnné n velmi vysoké hodnoty a nepříznivě by nám ovlivňovaly výpočet. Dosadíme tedy do (19), (21) a (22) za

$$\sinh(x) = \frac{e^x - e^{-x}}{2},$$
$$\cosh(x) = \frac{e^x + e^{-x}}{2},$$

$$\alpha_n = a_n e^{\pi n} = \frac{\sigma k_2}{k_2 \pi n \left(\frac{e^{\pi n}+e^{-\pi n}}{2}\right)(\sigma + k_1\pi^2 n^2) + (\sigma k_1 \pi^2 n^2)\left(\frac{e^{\pi n}-e^{-\pi n}}{2}\right)} \frac{1}{\frac{1}{e^{\pi n}}}$$
$$= \frac{\sigma k_2}{k_2 \pi n \left(\frac{1+e^{-2\pi n}}{2}\right)(\sigma + k_1\pi^2 n^2) + (\sigma k_1 \pi^2 n^2)\left(\frac{1-e^{-2\pi n}}{2}\right)},$$

$$u_n = \frac{\alpha_n \sigma \left(e^{\pi n} - e^{-\pi n}\right)}{2\sigma + 2k_1\pi^2 n^2} \frac{1}{e^{\pi n}} = \frac{\alpha_n \sigma \left(1 - e^{-2\pi n}\right)}{2\sigma + 2k_1\pi^2 n^2}.$$

Dostaneme tak upravené vztahy pro $p_2(x, y)$ a u_n

$$p_2(x,y) = P_2 - B_0 + B_0 y - 2B_0 \sum_{n=1}^{\infty} \cos(\pi n x)\, \alpha_n \left(\frac{e^{n\pi(1-y)} - e^{n\pi(y-1)}}{2}\right) \frac{1}{e^{n\pi}}, \quad (25)$$

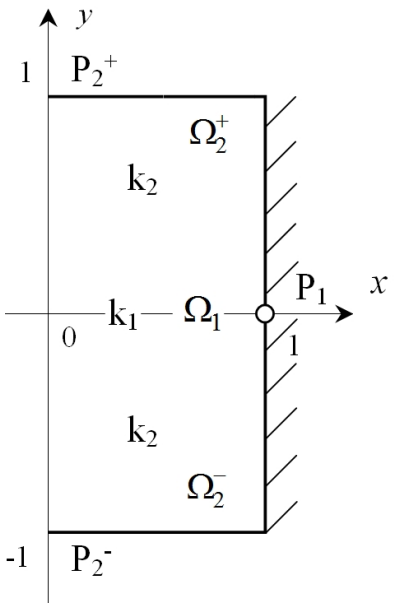
$$u_n = \frac{\alpha_n \sigma \left(1 - e^{-2\pi n}\right)}{2\sigma + 2k_1\pi^2 n^2}, \quad (26)$$

kde

$$\alpha_n = \frac{\sigma k_2}{k_2 \pi n \left(\frac{1+e^{-2\pi n}}{2}\right)(\sigma + k_1\pi^2 n^2) + (\sigma k_1 \pi^2 n^2)\left(\frac{1-e^{-2\pi n}}{2}\right)}. \quad (27)$$

1.2 Řešení modelu s nespojitým tlakem

Úloha s nespojitým tlakem má na každé straně pukliny, to znamená pro každou polovinu 2D oblasti, jiné hodnoty parametrů. Úloha je tak symetrická pouze podle osy y a řešení tedy hledáme jen na jedné polovině oblasti, jak je vidět z obrázku 4.



Obr. 4: Zjednodušená oblast pro nespojitý model

Vycházíme z rovnic (7) - (12) popisujících celou oblast, které musíme omezit na námi řešenou zjednodušenou 2D oblast rozdělenou na dvě poloviny. Nejprve popíšeme část oblasti nad puklinou $\Omega_2^+ = (0,1) \times (0,1) \subset R^2$. Proudění je zde popsáno rovnicí s okrajovými podmínkami

$$-k_2 \Delta p_2^+(x,y) = 0 \qquad \text{na } \Omega_2^+, \tag{28}$$

$$p_2^+(x,1) = P_2^+, \tag{29}$$

$$k_2 \frac{\partial p_2^+}{\partial y}(x,0) = \sigma^+(p_2^+(x,0) - p_1(x)), \tag{30}$$

$$\frac{\partial p_2^+}{\partial x}(0,y) = \frac{\partial p_2^+}{\partial x}(1,y) = 0. \tag{31}$$

Stejným způsobem popíšeme i část oblasti pod puklinou, kde $\Omega_2^- = (0,1) \times (-1,0) \subset R^2$

$$-k_2 \Delta p_2^-(x,y) = 0 \qquad \text{na } \Omega_2^-, \tag{32}$$

$$p_2^-(x,-1) = P_2^-, \tag{33}$$

$$k_2 \frac{\partial p_2^-}{\partial y}(x,0) = \sigma^-(p_2^-(x,0) - p_1(x)), \tag{34}$$

$$\frac{\partial p_2^-}{\partial x}(0,y) = \frac{\partial p_2^-}{\partial x}(1,y) = 0. \tag{35}$$

Nyní ještě zbývá vyjádření proudění na puklině. Oblast $\Omega_1 = (0,1) \subset R$ je popsána rovnicí s okrajovými podmínkami

$$-k_1 p_1''(x) = \sigma^+(p_2^+(x,0) - p_1(x)) + \sigma^-(p_2^-(x,0) - p_1(x)) \qquad \text{na } \Omega_1, \tag{36}$$

$$p_1(0) = P_1 \quad p_1'(1) = 0. \tag{37}$$

1.2.1 Řešení rovnic na Ω_2

Nejprve hledáme řešení pro oblast nad puklinou Ω_2^+. Začneme řešením rovnice (28) pomocí Fourierovy metody [7]. Koeficient k_2 považujeme za konstantu a můžeme ho tedy z rovnice vypustit. Uvažujeme řešení v separovaném tvaru

$$p_2^+(x,y) = X(x) \cdot Y(y),$$

které dosadíme do (28)

$$\Delta\left(X(x) \cdot Y(y)\right) = 0,$$

provedeme druhou derivaci

$$X(x)Y''(y) + X''(x)Y(y) + 2X'(x)Y'(y) = 0,$$

aplikujeme okrajové podmínky (31) a po úpravě dostaneme

$$\frac{X''}{X}(x) = -\frac{Y''}{Y}(y) = L, \tag{38}$$

kde L je reálná konstanta. Výraz (38) tvoří dvě diferenciální rovnice. Nejprve budeme řešit první z nich

$$X''(x) - LX(x) = 0. \tag{39}$$

Použitím charakteristické rovnice hledáme obecné řešení rovnice (39). Uvažujeme $L > 0$

$$x^2 - L = 0,$$

$$x = \pm\sqrt{L},$$

$$X(x) = C_1 e^{\sqrt{L}x} + C_2 e^{\sqrt{L}x},$$

vypočítáním derivace a s použitím okrajových podmínek (31) dostaneme řešení

$$X(x) = C_1 \qquad \text{pro } L = 0.$$

Nyní hledáme řešení rovnice (39) pro $L < 0$

$$X^2 + L = 0,$$

$$X = \pm i\sqrt{L},$$

$$X(x) = C_1 e^{i\sqrt{L}x} + C_2 e^{-i\sqrt{L}x},$$

$$= C_1 \left(\cos(\sqrt{L}x) + i\sin(\sqrt{L}x)\right) + C_1 \left(\cos(\sqrt{L}x) - i\sin(\sqrt{L}x)\right),$$

opět vypočítáme derivaci, použijeme okrajovou podmínku (31) a dostaneme řešení

$$X(x) = C_2 \cos(n\pi x) \qquad \text{pro } L = -n^2\pi^2.$$

Obecné řešení rovnice (39)je ve tvaru

$$\begin{aligned} X(x) &= C_1 && \text{pro } L = 0, \\ X(x) &= C_2 \cos(n\pi x) && \text{pro } L = -n^2\pi^2, \end{aligned} \tag{40}$$

kde C_1 a C_2 jsou reálné konstanty. Okrajové podmínky (31) nám omezí hodnoty L na $L = -n^2\pi^2$, kde n je nezáporné celé číslo. Nyní dosadíme L do rovnice (38), ze které použijeme část pro Y

$$Y''(y) - n^2\pi^2 Y(y) = 0.$$

Vytvoříme charakteristickou rovnici, dosadíme okrajové podmínky a získáme obecné řešení ve tvaru

$$\begin{aligned} Y(y) &= \tilde{A}_n e^{n\pi y} + \tilde{B}_n e^{-n\pi y} && \text{pro } n \in N \quad \text{a} \\ Y(y) &= \tilde{A}_0 + \tilde{B}_0 y && \text{pro } n = 0, \end{aligned} \tag{41}$$

kde $\tilde{A}_0,\ \tilde{B}_0,\ \tilde{A}_n,\ \tilde{B}_n$ jsou reálné parametry.

Nyní dosadíme do (34) řešení (40) a (41) a vyjádříme tak řešení rovnice (28) s okrajovými podmínkami (31) ve tvaru

$$p_2^+(x,y) = \tilde{A}_0 + \tilde{B}_0 y + \sum_{n=1}^{\infty} \cos(n\pi x)(\tilde{A}_n \frac{e^{-n\pi}}{2} e^{n\pi y} + \tilde{B}_n \frac{e^{n\pi}}{2} e^{-n\pi y}). \tag{42}$$

Dosazením do okrajové podmínky (29) dostaneme

$$P_2^+ = \tilde{A}_0 + \tilde{B}_0 + \sum_{n=1}^{\infty} \cos(n\pi x)(\frac{\tilde{A}_n + \tilde{B}_n}{2}) \qquad \text{pro } x \in (0,1),$$

z porovnání obou stran plyne

$$\begin{aligned} \tilde{A}_0 + \tilde{B}_0 &= P_2^+, \\ \tilde{A}_n + \tilde{B}_n &= 0 \qquad \text{pro } n \in N. \end{aligned}$$

Dosazením do (42) získáme řešení rovnice (28) s okrajovými podmínkami (29) a (31) pro tlak p_2^+ na Ω_2^+

$$p_2^+(x,y) = P_2^+ - B_0^+ + B_0^+ y - \sum_{n=1}^{\infty} A_n^+ \cos(n\pi x)\sinh(n\pi(1-y)). \tag{43}$$

Stejným postupem z rovnic (32) - (35) získáme vztah pro výpočet tlaku p_2^- na Ω_2^-

$$p_2^-(x,y) = P_2^- - B_0^- + B_0^- y - \sum_{n=1}^{\infty} A_n^- \cos(n\pi x)\sinh(n\pi(1-y)). \tag{44}$$

1.2.2 Řešení rovnic na Ω_1

Řešíme rovnici (36), do které dosadíme (43) a (44)

$$\begin{aligned} k_1 p_1''(x) - (\sigma^+ + \sigma^-)p_1(x) &= \sigma^+ \left(-P_2^+ + B_0^+ + \sum_{n=1}^{\infty} A_n^+ \cos(n\pi x)\sinh(n\pi) \right) \\ &+ \sigma^- \left(-P_2^- + B_0^- + \sum_{n=1}^{\infty} A_n^- \cos(n\pi x)\sinh(n\pi) \right) \end{aligned} \tag{45}$$

Pravou stranu rovnice (45) roznásobíme, celou rovnici vydělíme $\tilde{\sigma} = (\sigma^+ + \sigma^-)$ a po úpravě dostaneme

$$\frac{k_1}{\tilde{\sigma}} p_1''(x) - p_1(x) = -\tilde{P}_2 + \tilde{B}_0 + \sum_{n=1}^{\infty} \tilde{A}_n \cos(n\pi x)\sinh(n\pi), \tag{46}$$

kde

$$\tilde{P}_2 = \frac{(\sigma^+ P_2^+ + \sigma^- P_2^-)}{(\sigma^+ + \sigma^-)}, \tag{47}$$

$$\tilde{B}_0 = \frac{(\sigma^+ B_0^+ + \sigma^- B_0^-)}{(\sigma^+ + \sigma^-)}, \tag{48}$$

$$\tilde{A}_n = \frac{(\sigma^+ A_n^+ + \sigma^- A_n^-)}{(\sigma^+ + \sigma^-)}. \tag{49}$$

Řešení rovnice (46) je

$$p_1(x) = C_1 e^{Kx} + C_2 e^{-Kx} + \tilde{P}_2 - \tilde{B}_0 - \sum_{n=1}^{\infty} \tilde{u}_n \cos(n\pi x), \tag{50}$$

kde

$$\tilde{u}_n = \frac{\tilde{\sigma} \tilde{A}_n \sinh(n\pi)}{n^2\pi^2 k_1 + \tilde{\sigma}}, \tag{51}$$

$$K = \sqrt{\frac{\tilde{\sigma}}{k_1}}. \tag{52}$$

Dosazením rovnice (50) do okrajových podmínek (37) dostaneme soustavu rovnic

$$C_1 + C_2 + \tilde{P}_2 - \tilde{B}_0 - \sum_{n=1}^{\infty} \tilde{u}_n = P_1,$$

$$K\left(C_1 e^{K} - C_2 e^{-K}\right) = 0.$$

Jejím vyřešením vyjádříme konstanty C_1 a C_2

$$C_1 = \left(P_1 - \tilde{P}_2 + \tilde{B}_0 + \sum_{n=1}^{\infty} \tilde{u}_n\right) \frac{1}{1 + e^{2K}},$$
$$C_2 = \left(P_1 - \tilde{P}_2 + \tilde{B}_0 + \sum_{n=1}^{\infty} \tilde{u}_n\right) \frac{e^{2K}}{1 + e^{2K}}.$$

Dosazením vypočítaných konstant C_1 a C_2 do (50) dostaneme

$$p_1(x) = \left(P_1 - \tilde{P}_2 + \tilde{B}_0 + \sum_{n=1}^{\infty} \tilde{u}_n\right) C + \tilde{P}_2 - \tilde{B}_0 - \sum_{n=1}^{\infty} \tilde{u}_n \cos(n\pi x), \tag{53}$$

kde

$$C = \left(\frac{e^{Kx}}{1 + e^{2K}} + \frac{e^{2K} e^{-Kx}}{1 + e^{2K}}\right). \tag{54}$$

Výraz (54) lze dále upravit

$$C = \left(\frac{e^{Kx}}{1 + e^{2K}} + \frac{e^{2K} e^{-Kx}}{1 + e^{2K}}\right) \frac{e^{-K}}{e^{-K}} = \frac{e^{Kx} e^{-K} + e^{2K} e^{-Kx} e^{-K}}{e^{-K} + e^{2K} e^{-K}}$$

$$= \frac{\frac{1}{2} e^{-K(1-x)} + e^{K(1-x)}}{\frac{1}{2} e^{-K} e^{K}} = \frac{\cosh(K(1-x))}{\cosh K},$$

dosazením zpět do (53) dostaneme vztah pro p_1 na Ω_1

$$p_1(x) = \left(P_1 - \tilde{P}_2 + \tilde{B}_0 + \sum_{n=1}^{\infty} \tilde{u}_n \right) \frac{\cosh(K(1-x))}{\cosh K} + \tilde{P}_2 - \tilde{B}_0 - \sum_{n=1}^{\infty} \tilde{u}_n \cos(n\pi x). \tag{55}$$

1.2.3 Výpočet neznámých koeficientů

Ještě nám zbývá dopočítat neznámé koeficienty $\tilde{B}_0$, B_0^+, B_0^- a A_n^+, A_n^-, které je třeba dosadit do výsledných vztahů pro p_1 na Ω_1, p_2^+ na Ω_2^+ a p_2^- na Ω_2^-. Vycházíme z rovnic (30) a (34) popisujících interakci mezi puklinou Ω_1 a okolním prostředím Ω_2.

Nejprve použijeme rovnici (43) pro p_2^+ na Ω_2^+ , kterou omezíme na popis hranice oblasti s puklinou dosazením za $y = 0$

$$p_2^+(x,0) = P_2^+ - B_0^+ - \sum_{n=1}^{\infty} A_n^+ \cos(n\pi x) \sinh(n\pi). \tag{56}$$

Vypočítáme její derivaci a také do ní dosadíme za $y = 0$, čímž dostaneme popis toku hranicí mezi Ω_1 a Ω_2^+

$$\frac{\partial p_2^+}{\partial y}(x,y) = B_0^+ - \sum_{n=1}^{\infty} A_n^+ (-n\pi) \cos(n\pi x) \cosh(n\pi(1-y)),$$

$$\frac{\partial p_2^+}{\partial y}(x,0) = B_0^+ + \sum_{n=1}^{\infty} A_n^+ (n\pi) \cos(n\pi x) \cosh(n\pi). \tag{57}$$

Získané rovnice (56) a (57) společně s rovnicí (55) pro p_1 na Ω_1 dosadíme do (30). Sloučením sum, vydělením σ^+ a dalšími úpravami dostaneme

$$\frac{k_2 B_0^+}{\sigma^+} + \sum_{n=1}^{\infty} \left(A_n^+ \left(\frac{k_2(n\pi)}{\sigma^+} \cosh(n\pi) + \sinh(n\pi) \right) - \tilde{u}_n \right) \cos(n\pi x) = \\ = \tilde{\omega} \cosh(K(1-x)) + (P_2^+ - B_0^+ - \tilde{P}_2 + \tilde{B}_0), \tag{58}$$

kde

$$\tilde{\omega} = \frac{-\left(P_1 - \tilde{P}_2 + \tilde{B}_0 + \sum_{n=1}^{\infty} \tilde{u}_n \right)}{\cosh K}. \tag{59}$$

Rovnici (58) nyní přepíšeme do tvaru

$$\frac{\mathcal{A}_0^+}{2} + \sum_{n=1}^{\infty} \mathcal{A}_n^+ \cos(n\pi x) = \tilde{\omega} \cosh(K(1-x)), \tag{60}$$

kde

$$\frac{\mathcal{A}_0^+}{2} = \frac{k_2 B_0^+}{\sigma^+} - (P_2^+ - B_0^+ - \tilde{P}_2 + \tilde{B}_0), \tag{61}$$

$$\mathcal{A}_n^+ = A_n^+ \left(\frac{k_2(n\pi)}{\sigma^+} \cosh(n\pi) + \sinh(n\pi) \right) - \tilde{u}_n. \tag{62}$$

Levá strana (60) představuje rozvoj pravé strany v kosinovou Fourierovu řadu. Rozvojem pravé strany dostaneme rovnice pro prvky řady $\mathcal{A}_0^+$

$$\mathcal{A}_0^+ = \int_0^1 2\tilde{\omega} \cosh(K(1-x))dx = \frac{-2\tilde{\omega}}{K} \left[\sinh(K - Kx)\right]_0^1 = 2\tilde{\omega}K^{-1} \sinh K \tag{63}$$

a pro $\mathcal{A}_n^+$

$$\mathcal{A}_n^+ = \int_0^1 2\tilde{\omega} \cosh(K(1-x)) \cos(n\pi x)dx.$$

Provedeme dvakrát integraci per partes a po dalších úpravách dostaneme řešení integrálu ve tvaru

$$\mathcal{A}_n^+ = \left[\frac{2\tilde{\omega}}{n\pi + \frac{K^2}{n\pi}} \cosh(K - Kx) \sin(n\pi x) - \frac{2\tilde{\omega}K}{n^2\pi^2 + K^2} \sinh(K - Kx) \cos(n\pi x) \right]_0^1,$$

dosazením mezí integrálu dostaneme

$$\mathcal{A}_n^+ = \frac{2\tilde{\omega}Kk_1}{n^2\pi^2 k_1 + \tilde{\sigma}} \sinh K. \tag{64}$$

Podobným postupem odvodíme vztahy pro opačnou stranu pukliny. Vycházíme z rovnice (44) omezené na popis hranice oblasti dosazením $y = 0$. Spočítáme její derivaci a do ní také dosadíme $y = 0$. Obě získané rovnice spolu s (55) dosadíme do (34) a dostaneme

$$\frac{\mathcal{A}_0^-}{2} + \sum_{n=1}^{\infty} \mathcal{A}_n^- \cos(n\pi) = \tilde{\omega} \cosh(K(1-x)), \tag{65}$$

kde

$$\frac{\mathcal{A}_0^-}{2} = \frac{k_2 B_0^-}{\sigma^-} - (P_2^- - B_0^- - \tilde{P}_2 + \tilde{B}_0), \tag{66}$$

$$\mathcal{A}_n^- = A_n^- \left(\frac{k_2(n\pi)}{\sigma^-} \cosh(n\pi) + \sinh(n\pi) \right) - \tilde{u}_n. \tag{67}$$

Rozvojem pravé strany (65) v kosinovou Fourierovu řadu dostaneme rovnice pro prvky řady $\mathcal{A}_0^-$ a $\mathcal{A}_n^-$

$$\mathcal{A}_0^- = 2\tilde{\omega}K^{-1}\sinh K, \tag{68}$$

$$\mathcal{A}_n^- = \frac{2\tilde{\omega}K}{n^2\pi^2k_1+\tilde{\sigma}}\sinh K. \tag{69}$$

Nyní odvodíme vztahy pro výpočet neznámých koeficientů A_n^+ a A_n^-. Do rovnice (62) dosadíme (51), kam ještě dosadíme (49). Po roznásobení a po vytknutí A_n^+ a A_n^- dostaneme

$$\mathcal{A}_n^+ = A_n^+\left(\frac{k_2n\pi}{\sigma^+}\cosh(n\pi)+\sinh(n\pi)-\frac{\sigma^+\sinh(n\pi)}{n^2\pi^2k_1+\tilde{\sigma}}\right)-A_n^-\left(\frac{\sigma^-\sinh(n\pi)}{n^2\pi^2k_1+\tilde{\sigma}}\right). \tag{70}$$

V rovnici (70) nahradíme hyperbolické funkce exponenciálními. Dáme ji do rovnosti s (64) a po roznásobení, vytknutí $e^{n\pi}$ a dalších úpravách dostaneme

$$\alpha_n^+\left(\frac{k_2n\pi}{\sigma^+}\left(1+e^{-2n\pi}\right)+1-e^{-2n\pi}-\left(\frac{\sigma^+\left(1-e^{-2n\pi}\right)}{n^2\pi^2k_1+\tilde{\sigma}}\right)\right)+\alpha_n^-\left(\frac{-\sigma^-\left(1-e^{-2n\pi}\right)}{n^2\pi^2k_1+\tilde{\sigma}}\right)$$

$$= 4k_1K\sinh K, \tag{71}$$

kde

$$\alpha_n^+ = \frac{A_n^+e^{n\pi}(n^2\pi^2k_1+\tilde{\sigma})}{\tilde{\omega}}, \tag{72}$$

$$\alpha_n^- = \frac{A_n^-e^{n\pi}(n^2\pi^2k_1+\tilde{\sigma})}{\tilde{\omega}}. \tag{73}$$

Stejným způsobem z rovnice (67) po dosazení (51) a (49) a výše popsaným nahrazením hyperbolických funkcí exponenciálními získáme

$$\alpha_n^+\left(\frac{-\sigma^+\left(1-e^{-2n\pi}\right)}{n^2\pi^2k_1+\tilde{\sigma}}\right)+\alpha_n^-\left(\frac{k_2n\pi}{\sigma^-}\left(1+e^{-2n\pi}\right)+1-e^{-2n\pi}-\left(\frac{\sigma^-\left(1-e^{-2n\pi}\right)}{n^2\pi^2k_1+\tilde{\sigma}}\right)\right)$$

$$= 4k_1K\sinh K. \tag{74}$$

Rovnice (71) a (74) tvoří soustavu. Obsahují již jen známé koeficienty a lze tedy z nich vypočítat α_n^+ a α_n^-. Soustavu lze zjednodušeně zapsat v maticovém tvaru

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} \alpha_n^+ \\ \alpha_n^- \end{pmatrix} = \begin{pmatrix} g \\ g \end{pmatrix},$$

kde

$$\begin{aligned}
a &= \frac{k_2 n\pi}{\sigma^+}\left(1+e^{-2n\pi}\right)+1-e^{-2n\pi}-\left(\frac{\sigma^+\left(1-e^{-2n\pi}\right)}{n^2\pi^2k_1+\tilde{\sigma}}\right),\\
b &= \frac{-\sigma^-\left(1-e^{-2n\pi}\right)}{n^2\pi^2k_1+\tilde{\sigma}},\\
c &= \frac{-\sigma^+\left(1-e^{-2n\pi}\right)}{n^2\pi^2k_1+\tilde{\sigma}},\\
d &= \frac{k_2 n\pi}{\sigma^-}\left(1+e^{-2n\pi}\right)+1-e^{-2n\pi}-\left(\frac{\sigma^-\left(1-e^{-2n\pi}\right)}{n^2\pi^2k_1+\tilde{\sigma}}\right),\\
g &= 4k_1K\sinh K.
\end{aligned}$$

Z (72) a (73) vyjádříme A_n^+ a A_n^-

$$A_n^+ = \frac{\alpha_n^+\tilde{\omega}e^{-n\pi}}{n^2\pi^2k_1+\tilde{\sigma}}, \tag{75}$$

$$A_n^- = \frac{\alpha_n^-\tilde{\omega}e^{-n\pi}}{n^2\pi^2k_1+\tilde{\sigma}}. \tag{76}$$

Nyní zbývá dopočítat neznámé parametry B_0^+, B_0^-, $\tilde{B}_0$ a také nový parametr $\tilde{\omega}$, který vznikl v průběhu odvozování. Dáme do rovnosti (61) a (63), vytkneme jednotlivé neznámé a dáme na jednu stranu. Tak dostaneme první rovnici pro Ω_2^+

$$\left(\frac{k_2}{\sigma^+}+1\right)B_0^+-\tilde{B}_0-\left(K^{-1}\sinh K\right)\tilde{\omega}=P_2^+-\tilde{P}_2. \tag{77}$$

Stejným způsobem získáme z (66) a (68) druhou rovnici pro Ω_2^-

$$\left(\frac{k_2}{\sigma^-}+1\right)B_0^--\tilde{B}_0-\left(K^{-1}\sinh K\right)\tilde{\omega}=P_2^--\tilde{P}_2. \tag{78}$$

Jako třetí použijeme upravenou rovnici (48) pro $\tilde{B}_0$

$$\left(\frac{\sigma^+}{\tilde{\sigma}}\right)B_0^++\left(\frac{\sigma^-}{\tilde{\sigma}}\right)B_0^--\tilde{B}_0=0. \tag{79}$$

Do rovnice (59) pro $\tilde{\omega}$ dosadíme (51), kam dále dosadíme (49) a poté ještě ze vztahů (75) a (76) za A_n^+ a A_n^-. Po roznásobení a úpravách dostaneme čtvrtou rovnici

$$-\tilde{B}_0-\tilde{\omega}\left(\sum_{n=1}^{\infty}\frac{\left(1-e^{-2n\pi}\right)\left(\alpha^+\sigma^++\alpha^-\sigma^-\right)}{2\left(n^2\pi^2k_1+\tilde{\sigma}\right)^2}+\cosh K\right)=P_1-\tilde{P}_2. \tag{80}$$

Rovnice (77) až (80) tvoří soustavu rovnic, jejímž vyřešením získáme hodnoty neznámých parametrů B_0^+, B_0^-, $\tilde{B}_0$ a $\tilde{\omega}$.

1.2.4 Výsledné vztahy pro výpočet tlaku

Nyní již můžeme vyjádřit vztahy pro výpočet tlaku v řešené oblasti. Pro výpočet tlaku na puklině použijeme rovnici (55)

$$p_1(x) = \left(P_1 - \tilde{P}_2 + \tilde{B}_0 + \sum_{n=1}^{\infty} \tilde{u}_n \right) \frac{\cosh(K(1-x))}{\cosh K} + \tilde{P}_2 - \tilde{B}_0 - \sum_{n=1}^{\infty} \tilde{u}_n \cos(n\pi x), \tag{81}$$

do které za $\tilde{u}_n$ dosadíme

$$\tilde{u}_n = \frac{\tilde{\omega}\left(1 - e^{-2n\pi}\right)\left(\alpha_n^+ \sigma^+ + \alpha_n^- \sigma^-\right)}{2\left(n^2\pi^2 k_1 + \tilde{\sigma}\right)^2}. \tag{82}$$

Pro výpočet tlaku v okolním prostředí použijeme rovnice (43) a (44), do kterých dosadíme (75) a (76). Z důvodu přesnějšího výpočtu nahradíme hyperbolické funkce exponenciálními stejně jako v kapitole 1.1.1. Po dalších úpravách při kterých ze vztahu eliminujeme $e^{n\pi}$ dostaneme

$$p_2^+(x,y) = P_2^+ - B_0^+ + B_0^+ y - \sum_{n=1}^{\infty} \frac{\alpha_n^+ \tilde{\omega}}{2\left(n^2\pi^2 k_1 + \tilde{\sigma}\right)} \left(e^{-n\pi y} - e^{-n\pi(2-y)}\right) \cos(n\pi x) \tag{83}$$

$$p_2^-(x,y) = P_2^- - B_0^- + B_0^- y - \sum_{n=1}^{\infty} \frac{\alpha_n^- \tilde{\omega}}{2\left(n^2\pi^2 k_1 + \tilde{\sigma}\right)} \left(e^{-n\pi y} - e^{-n\pi(2-y)}\right) \cos(n\pi x). \tag{84}$$

Máme již vyjádřeny vztahy pro výpočet všech parametrů, které je třeba vypočítat a následně dosadit do konečných vztahů pro výpočet tlaků. Tyto parametry můžeme rozdělit na dvě skupiny. Ty, které nejsou závislé na souřadnicích ani na proměnné n, používané ve výpočtech součtů. Konkrétně to jsou B_0^+, B_0^-, $\tilde{B}_0$ a $\tilde{\omega}$. Všechny dostaneme ze soustavy rovnic (77) až (80), a jelikož se jedná o konstanty, stačí je vypočítat pouze jednou. Dále pak máme parametry, které používáme ve výpočtech sum a jsou tedy závislé na proměnné n. Sem patří $\tilde{u}_n$, α_n^+ a α_n^-. I ty si můžeme předpočítat a jejich hodnoty pro různá n uložit do pole. K jejich výpočtu použijeme soustavu rovnic (71) a (74). K výpočtu $\tilde{u}_n$ použijeme vztah (82).

Konkrétní hodnotu tlaku v libovolném místě pukliny získáme dosazením požadované souřadnice x do rovnice pro $p_1(x)$. Tlak v libovolném místě 2D oblasti získáme dosazením souřadnic x a y do rovnic $p_2^+(x,y)$ pro část nad puklinou nebo do rovnice $p_2^-(x,y)$ pro část oblasti pod puklinou.

2 Implementace v Matlabu

V předcházející kapitole jsme odvodili vztahy pro výpočet rozložení tlaku v horninovém prostředí s puklinou pro dva modely vazby mezi dimenzemi. Pro model se spojitým tlakem, který je symetrický podle pukliny, jsou parametry pro celou oblast konstantní. Naopak pro model s nespojitým tlakem, který je nesymetrický podle pukliny, má prostředí na každé polovině jiné hodnoty parametrů a na každé straně pukliny je tak jiné rozložení tlaku. Na puklině tak dochází k tlakové nespojitosti. V této kapitole popíšeme způsob implementace výpočtu odvozených řešení v programu Matlab.

2.1 Způsob sčítání nekonečných řad

Při implementaci vztahů bylo hlavním problémem vyřešit sčítání nekonečných řad, kterých se ve vztazích objevuje hned několik. Jelikož reálně nelze sečíst nekonečné množství prvků řady, je třeba otestovat konvergenci řad a zjistit minimální počet prvků, které je nutné sečíst, abychom dosáhli co možná nejvyšší přesnosti. K tomuto účelu jsme vytvořili testovací program, který generoval po sobě jdoucí prvky řady. Ty byly postupně sčítány a vždy se počítal rozdíl mezi dílčím součtem před a po přičtení dalšího prvku. Toto probíhalo až do chvíle, kdy rozdíl dosáhl hranice strojové přesnosti ϵ. Tento test jsme provedli na všech řadách obsažených ve vzorcích a došli jsme k závěru, že by se mělo vždy sčítat minimálně 3000 prvků řady.

Dále jsme řešili otázku způsobu sčítání velkého množství členů řad. V našem případě se jedná o řady, ve kterých se členy zmenšují řádově $1/n^2$. Uvažovali jsme dva základní způsoby. Sčítání členů standardním způsobem po předu, to je v našem případě od největšího po nejmenší. Druhou možností je sčítat členy řady odzadu, tedy od nejmenšího členu k největšímu. Tento způsob by teoreticky měl být přesnější, protože počítače počítají jen na určitý počet platných cifer, takže při klasickém sčítání může být dílčí součet již tak velké číslo, že řádově výrazně menší členy posloupnosti se již do součtu nemusí projevit. Naopak při sčítání odzadu, tedy od nejmenších členů, máme jistotu, že i tato malá čísla se v celkovém součtu projeví. V rámci testů jsme provedli součet 5000 členů řady oběma směry. Rozdíl výsledků byl velmi malý, řádově 10^{-15}, což je na hranici strojové přesnosti. V implementovaných výpočtech je sčítáno 5000 členů způsobem odzadu.

2.2 Model se spojitým tlakem

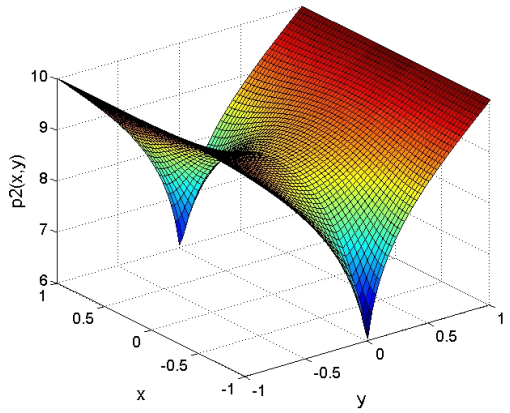
2.2.1 Tvorba programu

Používáme vztahy popsané v kapitole 1.1.1. Z důvodu zvýšení rychlosti výpočtu si nejprve vypočítáme parametry B_0 a u_0, které nejsou závislé na souřadnicích a ani se nevyskytují v sumách. To znamená, že jsou konstantní a stačí nám je spočítat jen jednou. Dále je vhodné hned na začátku do n-prvkového pole předpočítat parametry α_n a u_n, které se vyskytují v součtech řad. Tím se zamezí jejich opakovanému počítání, neboť se ve vzorcích také vyskytují vícekrát. Rovněž si musíme předem vygenerovat síť, na které budeme úlohu řešit. Jelikož nám jde hlavně o testy a ladění programu, stačí nám obyčejná pravidelná čtvercová síť. Tu vytvoříme rozdělením oblasti v osách x a y na n dílků. Souřadnice takto získaných uzlů sítě si uložíme do pole.

Nyní je již vše připraveno k dosazení do vzorců pro výpočet tlaku $p_1(x)$ a $p_2(x, y)$. Postupně vypočítáme tlak pro všechny uzly sítě a výsledky tlaku v 2D oblasti uložíme do matice. Jako vektor pak ukládáme výsledné hodnoty tlaku na 1D puklině.

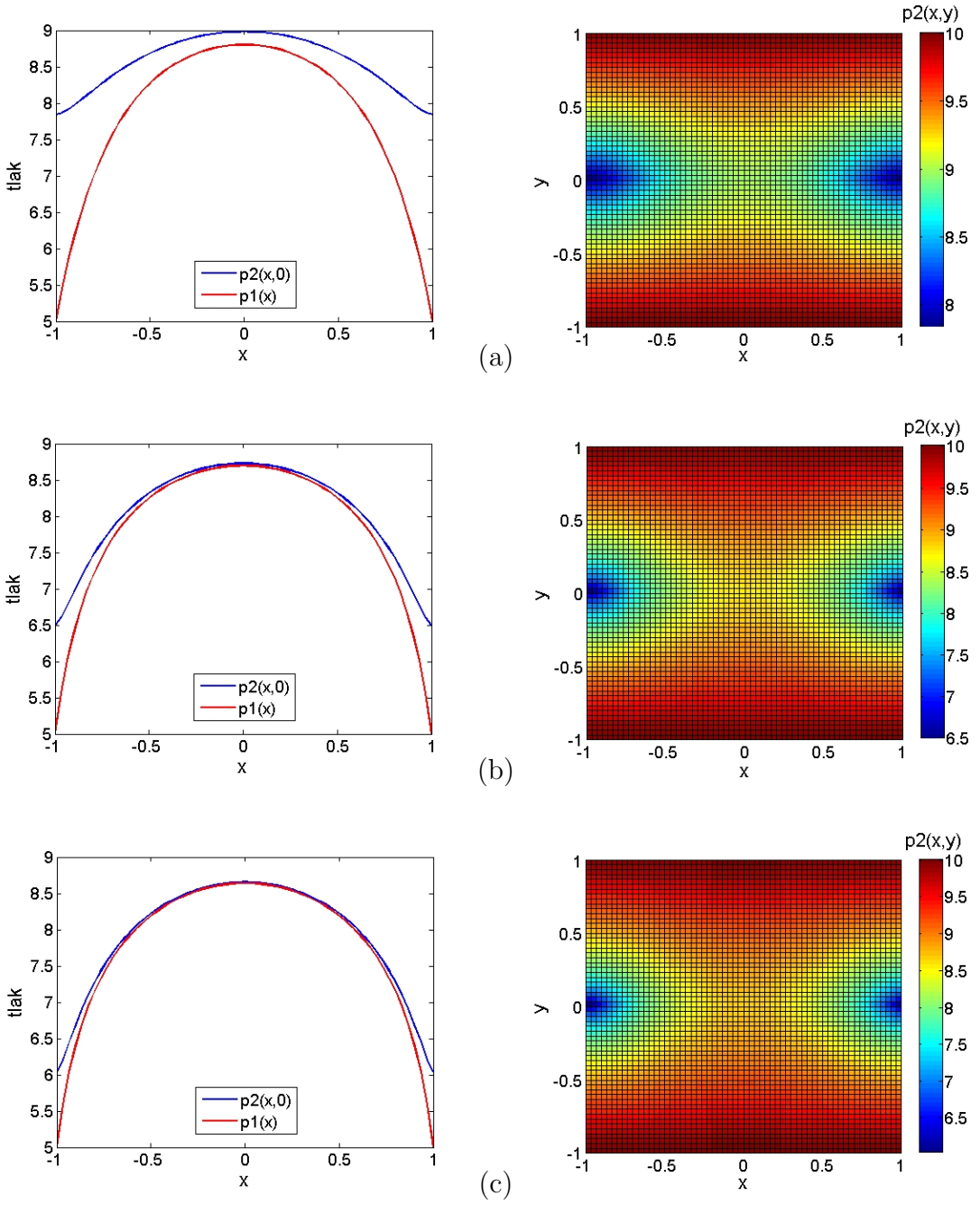
2.2.2 Výsledky

Na obrázku 5 je prostorové zobrazení rozložení tlaku na řešené 2D oblasti, kde je vidět, že i kolem pukliny je rozložení skutečně spojité. Na dalších příkladech ukážeme vliv koeficientu výměny vody σ mezi puklinou a okolním prostředím. Ostatní pa-



Obr. 5: Příklad výsledného rozložení tlaku pro spojitý model

rametry jsou konstantní a jejich hodnoty jsou zvoleny pro Dirichletovy okrajové podmínky $P_1 = 5$ a $P_2 = 10$, hydraulické vodivosti $k_1 = 1$ a $k_2 = 5$.



Obr. 6: Zobrazení vlivu koeficientu σ na rozložení tlaku. Hodnoty σ jsou zvoleny (a) $\sigma = 10$, (b) $\sigma = 50$, (c) $\sigma = 100$. Vlevo je srovnání tlaku na puklině s tlakem na sousedící hranici oblasti, vpravo je zobrazeno rozložení tlaku na celé oblasti.

Na obrázku 6 vidíme výsledné průběhy pro tři různé hodnoty koeficientu σ. Vpravo je znázorněno rozložení tlaku na celé ploše řešené 2D oblasti. Nejvíce je vliv σ vidět na levém obrázku, kde je vždy porovnání průběhu tlaku na puklině se sousední hranou 2D oblasti. Vidíme, že čím větší koeficient σ je, tím dochází k lepší výměně a velikosti tlaků se k sobě blíží. Pro $\sigma > 500$ už jsou oba tlaky v podstatě totožné a dále už nemá na výsledek výrazný vliv.

2.3 Model s nespojitým tlakem

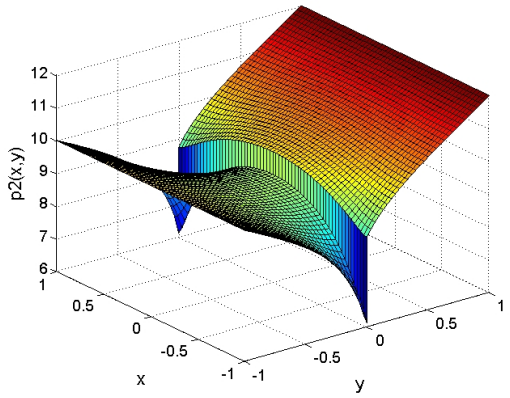
2.3.1 Tvorba programu

Používáme vztahy popsané v kapitole 1.2.4. Postup je obdobný jako v předchozím případě, jen o něco složitější. Nejprve si opět předpočítáme konstantní parametry, což obnáší vyřešení soustavy čtyř lineárních rovnic, čímž dostaneme hodnoty $\tilde{B_0}$, B_0^+, B_0^- a ω. Dále vytvoříme n-prvkové pole, do kterého předpočítáme parametry α_n^+ a α_n^- závislé na n a používané v součtech řad. Úlohu řešíme na totožné zkušební síti jako v předchozím případě.

Nyní již můžeme dosadit do vzorců pro výpočet tlaku. V tomto případě však musíme testovat souřadnici y, abychom určili, zda se jedná o oblast nad nebo pod puklinou. Tedy pokud $y > 0$, musíme použít vzorec pro $p_2^+(x, y)$ a pro $y < 0$ použijeme vzorec $p_2^-(x, y)$. Výsledky opět ukládáme do jedné matice na souřadnici příslušného uzlu. Tlak na puklině počítáme podle vzorce pro $p_1(x)$ a výsledné hodnoty ukládáme jako vektor.

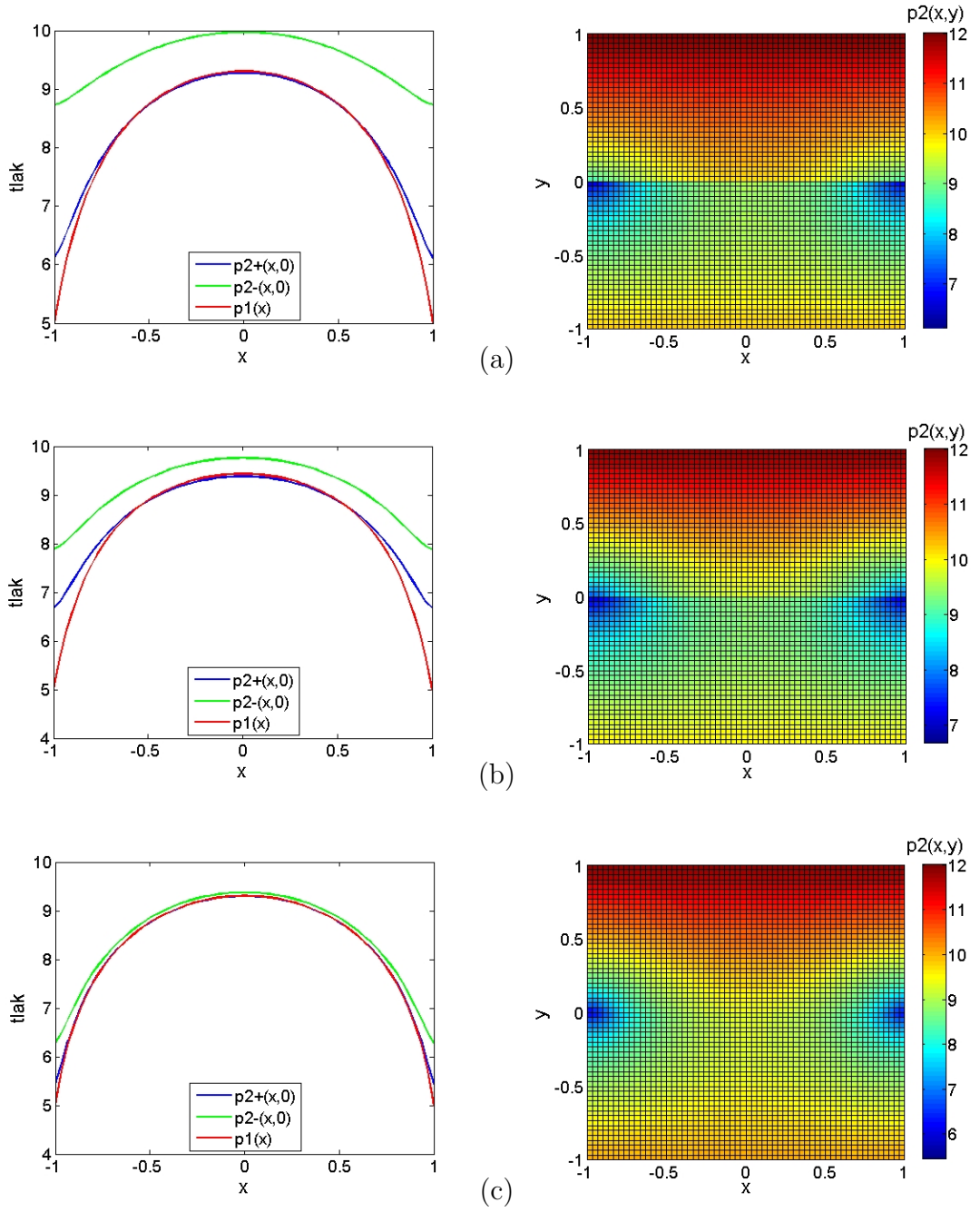
2.3.2 Výsledky

Na obrázku 7 je pro ilustraci příklad výpočtu nespojitého modelu, kde je zobrazeno rozložení tlaku na celé 2D oblasti a je vidět výrazný skok hodnot tlaku v okolí pukliny.



Obr. 7: Příklad výsledného rozložení tlaku pro nespojitý model

Na obrázku 8 ukážeme vliv různých kombinací hodnot koeficientů σ^+ a σ^-. Ostatní parametry jsou opět konstantní a jejich hodnoty jsou zvoleny $P_1 = 5$, $P_2^+ = 12$, $P_2^+ = 10$, $k_1 = 1$ a $k_2 = 5$.



Obr. 8: Zobrazení vlivu koeficientů σ^+ a σ^- na rozložení tlaku. Hodnoty jsou voleny zvlášť pro výměnu s horní a dolní polovinou oblasti (a) $\sigma^+ = 10$, $\sigma^- = 100$, (b) $\sigma^+ = 20$, $\sigma^- = 50$, (c) $\sigma^+ = 100$, $\sigma^- = 500$. Vlevo je srovnání tlaku na puklině s tlaky na obou sousedících hranicích oblasti, vpravo je zobrazeno rozložení tlaku na celé oblasti.

Na obrázcích vpravo je znázorněno rozložení tlaku na celé 2D oblasti, kde jsou vidět různé hodnoty okrajových podmínek na horní a dolní hraně, a také nespojitost v místě pukliny. V levém grafu je znázorněno porovnání tlaku na puklině a na sousedních hranicích obou polovin oblasti. Opět vidíme jak se zvyšujícími se koeficienty σ se průběhy tlaků na hranách blíží puklině. Na posledním obrázku již

vidíme že přestože jsou hodnoty σ^+ a σ^- výrazně odlišné, obě strany mají již téměř stejný průběh a nespojitost není příliš výrazná. To potvrzuje zjištění, že pro velkou hodnotu σ již nemá na výsledek příliš velký vliv.

3 Řešení pomocí metody konečných diferencí

Pro numerické řešení parciálních diferenciálních rovnic jsme použili metodu sítí (nebo-li metodu konečných diferencí). Základní myšlenka metody spočívá v tom, že v oblasti, ve které hledáme řešení dané diferenciální rovnice zvolíme konečnou množinu bodů, kterou nazveme sítí a příslušné body jejími uzly. Derivace hledané funkce, které se vyskytují v dané diferenciální rovnici a v okrajových podmínkách nahradíme diferenčními podíly v uzlech. Diferenčním podílem rozumíme lineární kombinaci funkčních hodnot v daném bodě a v okolních bodech, která příslušnou derivaci aproximuje [4], [8].

Výpočet metodou sítí provádíme, abychom měli kontrolu k ověření výsledků analytického řešení. Byla zvolena jednoduchá geometrie modelů, aby je bylo možné metodou sítí snadno řešit. Výsledky jsou pro nás jen orientační, jelikož jsme při výpočtu použili řídkou síť a také metoda nemá vysoký řád konvergence. Jde nám pouze o porovnání průběhu tlaku, ale ne o ověřování přesnosti výpočtů.

3.1 Popis diskretizace

Podstatou diskretizace je náhrada parciálních derivací podle jejich definice diferencemi. Na základě původní diferenciální úlohy sestavujeme konečný systém rovnic pro konečný počet neznámých, jehož vyřešením pak dospějeme k hledanému přibližnému řešení dané okrajové úlohy.

Vycházíme z parciální diferenciální rovnice (1) s okrajovými podmínkami (2) a (4). Rovnici rozepíšeme do tvaru

$$-k_2\frac{\partial^2 u(x_i,y_i)}{\partial(x^2)} - k_2\frac{\partial^2 u(x_i,y_i)}{\partial(y^2)} = 0$$

a vyjádříme pomocí centrálních diferencí

$$\frac{2k_2u_{i,i} - k_2u_{i-1,i} - k_2u_{i+1,i}}{h^2} + \frac{2k_2u_{i,i} - k_2u_{i,i-1} - k_2u_{i,i+1}}{h^2} = 0. \tag{85}$$

Úloha je řešena na síti, která se skládá z M bodů na šířku a $N = M + 2$ na výšku s krokem $h = 2/(M-1)$. Na výšku jsou v síti dva řádky navíc, jeden z nich je přidaný prostřední řádek uzlů, který reprezentuje puklinu. Tím vlastně síť pro celou oblast rozděluje na dvě poloviny a proto musíme přidat další řádek zdvojením řádku uzlů pro souřadnici $y = 0$. Teď ale máme dva řádky uzlů se stejnými souřadnicemi,

což by dělalo problémy hlavně při vizualizaci výsledku. Proto jsme nahradili nulovou souřadnici minimální možnou nenulovou hodnotou, tedy hodnotou strojové přesnosti $\epsilon = 2.2204 \cdot 10^{-16}$. Tím jsme z nich udělali dva nové řádky s různými souřadnicemi, pro horní polovinu sítě $y = \epsilon$ a pro dolní polovinu sítě $y = -\epsilon$.

3.2 Vzorce pro naplnění matice

Vytvoříme čtvercovou matici A o rozměrech $[NM, NM]$ reprezentující soustavu rovnic a vektor pravé strany soustavy b o velikosti $[NM]$, které teď budeme naplňovat hodnotami příslušejícími jednotlivým uzlům sítě.

V síti se nachází několik typů uzlů, hraniční uzly, vnitřní uzly oblasti, uzly se zadanou okrajovou podmínkou apod. Pro každý z nich je třeba napsat rovnici a popsat jeho vztah se sousedními uzly. Tento popis musíme zanést na příslušné místo do matice. Síť procházíme postupně uzel po uzlu, proměnná i nám označuje kolikátý uzel právě zpracováváme a zároveň udává index řádku a sloupce matice.

Pro hraniční uzly 2D oblasti, na kterých je definovaná Dirichletova okrajová podmínka, vložíme na příslušné místo na diagonále matice hodnotu 1 a do vektoru pravé strany zapíšeme hodnotu okrajové podmínky

$$A(i,i) = 1, \qquad b(i) = P_2^+ \text{ resp. } P_2^-.$$

Vnitřní uzly, které leží uvnitř 2D oblasti a mají čtyři sousední uzly, jsou popsány přímo výše uvedenou aproximací (85) původní diferenciální rovnice

$$\begin{aligned} A(i,i) &= 4 \cdot \frac{k_2}{h^2}, \\ A(i,i-1) &= -\frac{k_2}{h^2}, \\ A(i,i+1) &= -\frac{k_2}{h^2}, \\ A(i,i-M) &= -\frac{k_2}{h^2}, \\ A(i,i+M) &= -\frac{k_2}{h^2}. \end{aligned}$$

Hraniční uzly ležící na okrajích 2D oblasti mají pouze tři sousedy, protože z jedné strany mají definovanou nulovou Neumannovu okrajovou podmínku reprezentující nepropustnou stěnu oblasti. Tyto uzly jsou, podle toho, zda leží na pravé nebo levé hranici, popsány rovnicí

$$\frac{k_2 u_{i,i} - k_2 u_{i+1,i}}{h^2} + \frac{2k_2 u_{i,i} - k_2 u_{i,i-1} - k_2 u_{i,i+1}}{h^2} = 0$$

resp.

$$\frac{k_2 u_{i,i} - k_2 u_{i-1,i}}{h^2} + \frac{2k_2 u_{i,i} - k_2 u_{i,i-1} - k_2 u_{i,i+1}}{h^2} = 0.$$

Do matice A zapíšeme

$$\begin{aligned} A(i,i) &= 3 \cdot \frac{k_2}{h^2}, \\ A(i,i-1) &= -\frac{k_2}{h^2} \qquad \text{resp.} \qquad A(i,i+1) = -\frac{k_2}{h^2}, \\ A(i,i-M) &= -\frac{k_2}{h^2}, \\ A(i,i+M) &= -\frac{k_2}{h^2}. \end{aligned}$$

Dále síť obsahuje uzly, které sousedí s puklinou. Do jejich popisu musíme zahrnout komunikaci s uzly reprezentujícími puklinu. I zde je však musíme rozdělit na uzly hraniční a vnitřní.

Vztah pro vnitřní body sousedící s puklinou je

$$\frac{2k_2 u_{i,i} - k_2 u_{i-1,i} - k_2 u_{i+1,i}}{h^2} + \frac{k_2 u_{i,i} - k_2 u_{i,i-1}}{h^2} - \frac{\sigma\left(u_{i,i} - u_{i,i+1}\right)}{h} = 0,$$

pro hraniční body upravíme (3.2) a získáme

$$\frac{k_2 u_{i,i} - k_2 u_{i-1,i}}{h^2} + \frac{k_2 u_{i,i} - k_2 u_{i,i-1}}{h^2} - \frac{\sigma\left(u_{i,i} - u_{i,i+1}\right)}{h} = 0$$

resp. pro body na druhé straně

$$\frac{k_2 u_{i,i} - k_2 u_{i+1,i}}{h^2} + \frac{k_2 u_{i,i} - k_2 u_{i,i-1}}{h^2} - \frac{\sigma\left(u_{i,i} - u_{i,i+1}\right)}{h} = 0.$$

Do matice doplníme

$$\begin{aligned} A(i,i) &= 3 \cdot \frac{k_2}{h^2} + \frac{\sigma^+}{h}, \\ A(i,i-1) &= -\frac{k_2}{h^2}, \\ A(i,i+1) &= -\frac{k_2}{h^2}, \\ A(i,i-M) &= -\frac{k_2}{h^2}, \\ A(i,i+M) &= -\frac{\sigma^+}{h} \end{aligned}$$

a

$$\begin{aligned} A(i,i) &= 2 \cdot \frac{k_2}{h^2} + \frac{\sigma^+}{h}, \\ A(i,i+1) &= -\frac{k_2}{h^2} \qquad \text{resp.} \qquad A(i,i-1) = -\frac{k_2}{h^2}, \\ A(i,i-M) &= -\frac{k_2}{h^2}, \\ A(i,i+M) &= -\frac{\sigma^+}{h}. \end{aligned}$$

Tímto jsme prošli a popsali všechny uzly sítě horní poloviny 2D oblasti. Úplně stejným způsobem jsou popsány uzly i na spodní polovině, jen je nutné správně nahradit specifické parametry, tedy σ^- místo σ^+ a P_2^- zaměníme za P_2^+.

Nyní ještě zbývá popsat uzly sítě na puklině, které jsou ve vztahu s hraničními uzly obou polovin 2D oblasti. Na obou koncových bodech pukliny jsou definovány Dirichletovy okrajové podmínky, tedy do matice A na pozici (i,i) vložíme hodnotu 1 a na i-tou pozici vektoru pravé strany dosadíme hodnotu okrajové podmínky P_1

$$A(i,i) = 1, \qquad\qquad b(i) = P_1.$$

Vztah vnitřních uzlů pukliny s jejich sousedními uzly na 2D oblastech je

$$\frac{k_1 u_{i,i} - k_1 u_{i-1,i} - k_1 u_{i+1,i}}{h^2} + \sigma^+(u_{i,i} - u_{i,i-1}) + \sigma^-(u_{i,i} - u_{i,i+1}) = 0.$$

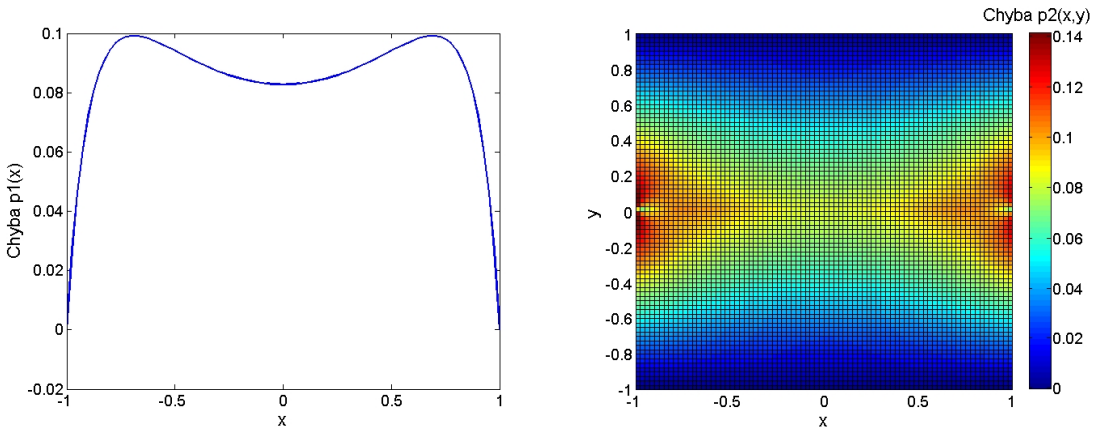
Do matice A vložíme

$$\begin{aligned} A(i,i) &= 2 \cdot \frac{k_1}{h^2} + (\sigma^+ + \sigma^-), \\ A(i,i-1) &= -\frac{k_1}{h^2}, \\ A(i,i+1) &= -\frac{k_1}{h^2}, \\ A(i,i-M) &= -\sigma^+, \\ A(i,i+M) &= -\sigma^-. \end{aligned}$$

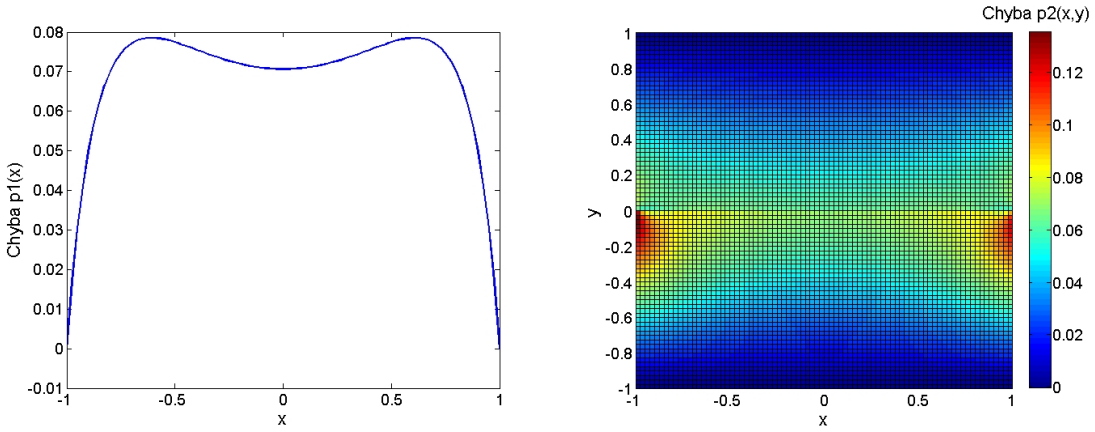
Nyní již máme naplněnou matici A i vektor pravé strany b, které tvoří soustavu rovnic $Ax = b$. Tu již stačí vyřešit a dostaneme vektor řešení x, v němž i-tý prvek obsahuje hodnotu tlaku i-tého uzlu sítě. Výsledný vektor musíme přetvořit do výsledné matice odpovídající rozložení uzlů v síti, abychom mohli výsledek vizualizovat a také porovnávat s analytickým řešením. Kompletní kód programu v Matlabu je uveden v příloze 1.

3.3 Porovnání výsledků

Vypočítáme rozdíl řešení získaného metodou konečných diferencí a analytického řešení stejné úlohy. Výpočty provedeme na síti o velikosti $N = 61$, tedy s krokem $h = 0,0323$. Do jednoho grafu vyneseme rozdíl hodnot na puklině a do druhého zobrazíme rozložení chyby pro výsledky na 2D oblasti. Takto porovnáme výsledky pro spojitý i nespojitý model.



Obr. 9: Chyba pro spojitý model



Obr. 10: Chyba pro nespojitý model

Pro oba modely jsme ještě, pro různou hustotu sítě, vypočítali ze získaných rozdílů obou metod aproximované normy L_2 pro 1D puklinu i okolní 2D oblast podle vztahů

$$L_{22D} = \sqrt{\sum h^2 \left(V_{A2D}(i,j) - V_{S2D}(i,j)\right)^2}, \tag{86}$$

$$L_{21D} = \sqrt{\sum h \left(V_{A1D}(i) - V_{S1D}(i)\right)^2}. \tag{87}$$

Z nich jsme pak vypočítali řád konvergence

$$\mu = \frac{-\log\left(\frac{A_2}{A_1}\right)}{\log\left(\frac{N_2}{N_1}\right)} \tag{88}$$

a získané výsledky jsme sestavili do tabulek.

N	11	21	41	81
L_2 2D	0,9518	0,4973	0,2495	0,1236
L_2 1D	0,8468	0,4709	0,2443	0,1231
μ 2D	—	1,0039	1,0309	1,0316
μ 1D	—	0,9075	0,9809	1,0066

Tab. 1: Normy a řád konvergence pro spojitý model

Z obrázku 9 a 10 je vidět, že pro oba modely je chyba velmi výrazná, zejména v oblasti okolo konců pukliny, kde je chyba větší než 0,1. To je způsobeno zejména malou hustotou sítě. Jelikož používáme přímý řešič soustavy rovnic, je práce se sítí

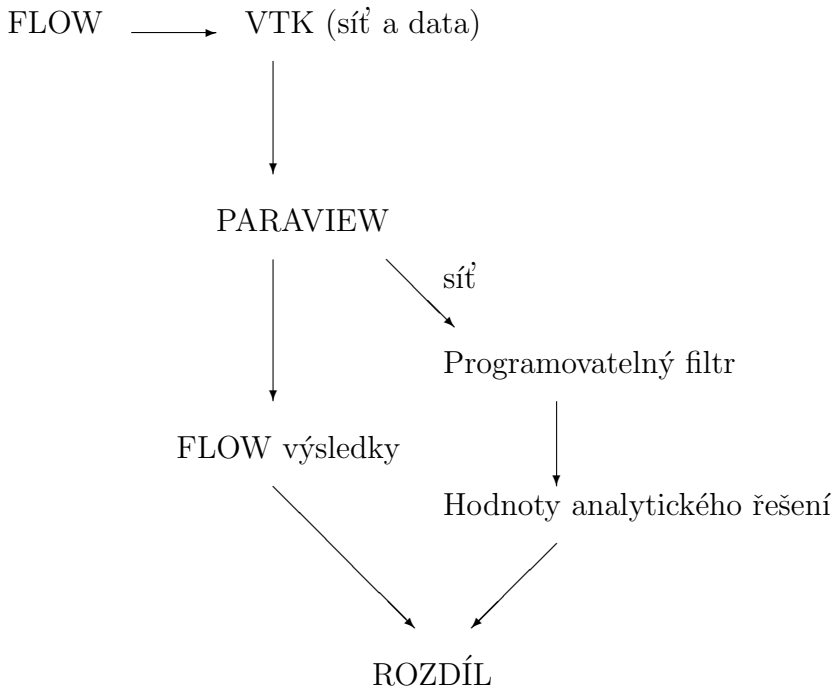
N	11	21	41	81
L_2 2D	0,8054	0,3994	0,1952	0,0957
L_2 1D	0,7520	0,3959	0,1996	0,0994
μ 2D	—	1,0847	1,0701	1,0469
μ 1D	—	0,9921	1,0236	1,0239

Tab. 2: Normy a řád konvergence pro nespojitý model

větší než 81×81 problematická. Dalším faktorem je malý řád konvergence, který jak je vidět z tabulek 1 a 2, se pohybuje kolem jedné. To je způsobeno problematickou aproximací Neumannovy okrajové podmínky vyššího řádu. Výsledky získané metodou sítí jsou sice jen orientační, ale ověřili jsme správnost odvozeného analytického řešení.

4 Implementace v ParaView

Naším hlavním cílem je možnost porovnat výsledky získané některou numerickou metodou, v našem případě programem Flow123d, s analytickým řešením. Aby bylo porovnání možné, musíme oběma způsoby spočítat stejnou úlohu za stejných podmínek. K tomuto účelu jsme použili software ParaView. Síť a výsledky získané Flow123d se dají importovat do ParaView, tam vypočítat analytické řešení a provést porovnání výsledků, jak je znázorněno na následujícím schématu.



Pro výpočet analytického řešení úlohy v ParaView používáme programovatelný filtr, který ale pracuje s programovacím jazykem Python a proto je nutné všechny programy z Matlabu přepracovat. Programovatelný filtr, ale není vhodný pro tvorbu a odladění programu, pouze do něj načítáme již hotové skripty. S jazykem Python tedy pracujeme v prostředí Enthought Python, ve kterém implementujeme a odzkoušíme kód skriptů. Ten teprve pak poupravíme pro potřeby ParaView a importujeme do programovatelného filtru. Takto přepracujeme řešení pro spojitý i nespojitý model. Dále popsané postupy jsou pro oba modely stejné.

Důležité je používat novější verze ParaView (používána verze 3.14.1). Důvodem je verze obsaženého Pythonu a jeho knihoven, což je v našem případě důležité zejména pro práci s maticemi při řešení soustav rovnic.

4.1 Práce s ParaView

Jako vstup používáme datové soubory typu `vtu`, které obsahují předdefinovanou síť a jí přiřazené hodnoty. Jsou zde uloženy souřadnice uzlů sítě, jim je možno přiřadit hodnoty `node_scalars`. Pomocí uzlů jsou definovány trojúhelníkové elementy, které obsahují hodnoty `element_vectors` a `element_scalars`.

4.1.1 Systém filtrů

Na zobrazená načtená data lze aplikovat filtry. Jedná se o předdefinované funkce, provádějící různé výpočty, analýzy, vizualizaci dat, atd.

Jedním z filtrů je i programovatelný filtr, do kterého lze pomocí skriptu jazyku Python naprogramovat vlastní funkce. Tento filtr jsme použili pro implementaci zadaných vztahů.

Každý filtr po svém vytvoření automaticky převezme všechna data z objektu, na který je aplikován a dále pracuje jen s těmito daty. Výsledky ukládá do vlastní nové proměnné, nepřepisuje tedy původní hodnoty.

Filtry lze řetězit, tzn. že filtr lze aplikovat i na výsledky získané jiným filtrem.

4.1.2 Popis řešení

Jelikož v ParaView probíhají výpočty výrazně pomaleji než v Matlabu, což se projevilo hlavně u nespojitého modelu, je struktura programu v Pythonu výrazněji pozměněna. Zejména jsou v programu v podstatě všechny jednotlivé výpočty nadefinovány jako funkce a pak vyvolávány. Byl upraven i způsob sčítání nekonečných řad. Použili jsme způsobu sčítání po balíčcích, se zadanou tolerancí. To znamená, že nesčítáme konkrétní počet členů řady, ale odzadu sečteme vždy balíček sta členů, tento mezivýsledek si uložíme do pole a takto pokračujeme, dokud dva po sobě jdoucí mezisoučty nemají rozdíl na úrovni tolerance. Poté proces skončí a my sečteme odzadu všechny dílčí součty balíčků, čímž dostaneme konečný součet.

Nyní popíšeme některé důležité části skriptu, jako je práce s daty a sítí, které jsou specifické pro ParaView a programovatelný filtr [1], [6]. Kompletní kód skriptu je pak uveden v příloze 2.

Proměnné `pdi` je přiřazen objekt se vstupní sítí a daty. Proměnné `pdo` je přiřazen výstupní objekt a jsou do ní ukládány výsledky, které pak lze vizualizovat

```
pdi = self.GetPolyDataInput(),
pdo = self.GetOutput().
```

Kód pro vytvoření pole, do kterého jsou ukládány výsledné hodnoty pro jednotlivé elementy

```
newData = vtk.vtkDoubleArray(),
newData.SetName("pressure").
```

Zjištění počtu elementů sítě

```
nc = pdi.GetNumberOfCells().
```

Nyní pomocí for cyklu procházíme všechny elementy a pro každý zjistíme počet uzlů. Pokud mají 3 uzly, jedná se o 2D elementy prostředí, nebo mohou mít jen dva uzly, pak jde o 1D elementy na puklině

```
cell = pdi.GetCell(i),
pb = cell.GetNumberOfPoints().
```

Zjištění souřadnic uzlů (vrcholů trojúhelníků) a výpočet souřadnic těžiště daného elementu, které jsou dosazovány do vzorců pro výpočet tlaku

```
p1 = pdi.GetPoint(cell.GetPointId(0)),
x1, y1, z1 = p1[:3],
p2 = pdi.GetPoint(cell.GetPointId(1)),
x2, y2, z2 = p2[:3],
p3 = pdi.GetPoint(cell.GetPointId(2)),
x3, y3, z3 = p3[:3],
tx = 1 - math.fabs((x1+x2+x3)/(3)),
ty = math.fabs((y1+y2+y3)/(3)).
```

Po dosazení a výpočtu tlaku, jsou výsledné hodnoty ukládány do pole, které je nakonec přiřazeno výstupní proměnné

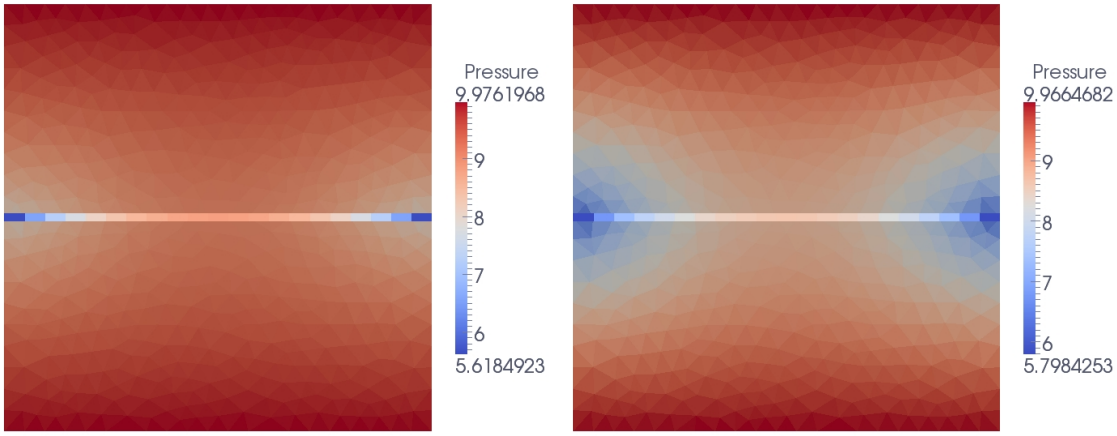
```
newData.InsertNextValue(p2),
pdo.GetCellData().AddArray(newData).
```

4.1.3 Spouštění skriptu

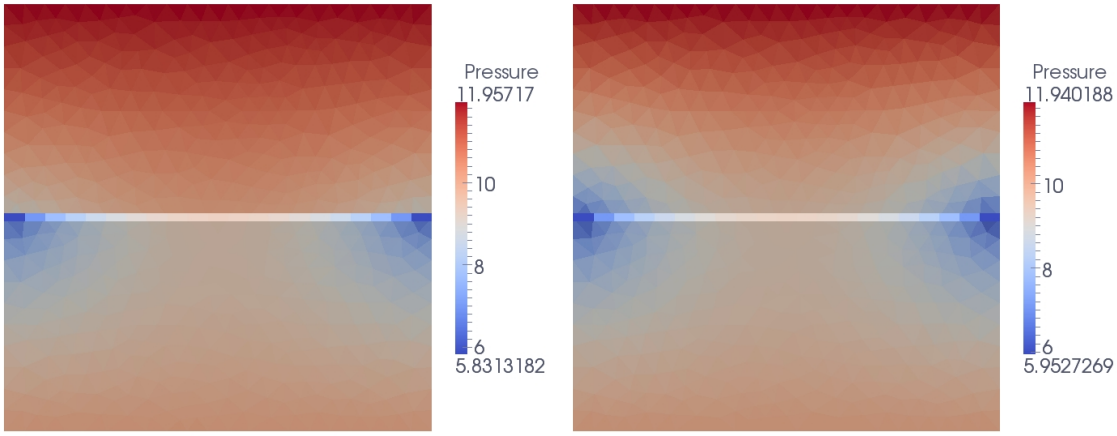
Spustíme software ParaView a načteme vstupní datový soubor `vtu`. Musíme jej aktivovat stiskem tlačítka `Apply`. Z nabídky `Filters/Alphabetical` vybereme `Programmable filter`, který aplikujeme na načtená data. Do okna `Script` tohoto programovatelného filtru musíme ručně z textového souboru zkopírovat kód skriptu. Poté se stiskem `Apply` provede výpočet. Pro požadované zobrazení výsledků musíme v horní části okna z rozbalovacích nabídek vybrat způsob zobrazení, vlevo vybereme název proměnné s uloženými hodnotami a v pravé nabídce vybereme možnost `Surface` nebo `Surface with edges` pro zobrazení se sítí. Na témže řádku první ikonou zleva lze zobrazit barevnou stupnici hodnot.

4.2 Výsledky

Stejně jako v Matlabu jsme i zde provedli několik vzorových výpočtů, tentokrát jsme však použili konkrétní kompatibilní trojúhelníkovou síť, pro kterou je implementace určená. Stejně jako předtím jsme i zde použili konstantní parametry prostředí a zobrazujeme jen vliv koeficientu σ.



Obr. 11: Výsledné průběhy tlaku pro spojitý model, hodnoty koeficientu σ jsou na levém obrázku $\sigma = 10$ a na pravém $\sigma = 100$.



Obr. 12: Výsledné průběhy tlaku pro nespojitý model, hodnoty koeficientů σ^+ a σ^- jsou na levém obrázku $\sigma^+ = 10$ a $\sigma^- = 100$, na pravém $\sigma^+ = 100$ a $\sigma^- = 500$.

Zobrazené výsledky potvrzují naše dřívější zjištění, že pro velké hodnoty σ se již jeho vliv do výsledku výrazně neprojevuje.

5 Test metod pro akceleraci konvergence

V této kapitole vyzkoušíme některé metody akcelerace konvergence na součty nekonečných řad. Využijeme k tomu již implementované algoritmy v jazyku Python. Slibujeme si od nich zpřesnění, ale především zrychlení výpočtu součtu řady.

5.1 Knihovna Mpmath pro Python

Mpmath je knihovna pro Python pro výpočty s libovolnou přesností aritmetiky s plovoucí desetinnou čárkou. Poskytuje rozsáhlý soubor transcendentních funkcí, umožňuje práci s komplexními čísly, exponenty neomezené velikosti, intervalovou aritmetikou a lineární algebrou. Obsahuje nástroje pro hledání kořenů rovnic a soustav, výpočet numerických integrálů a derivací a mnoho dalšího. Téměř každý výpočet lze provádět stejně dobře v přesnosti na 10 nebo i 1000 desetinných míst a v mnoha případech Mpmath implementuje asymptoticky rychlé algoritmy, které lze využívat také pro velmi vysoké přesnosti práce. Knihovna Mpmath je volně k dispozici. Je určena pro verzi Pythonu 2.5 a vyšší [5].

Knihovna umožňuje práci s některými speciálními datovými typy. Lze ukládat a pracovat s daty ve tvaru matic (`matrix`) a komplexních čísel (`mpc`). Dalším užitečným datovým typem je `mpf`, který je analogický s obyčejným typem `float` vestavěným v Pythonu. Navíc ale oproti němu může kromě reálných čísel nabývat i speciálních hodnot `inf` ($+\infty$), `-inf` ($-\infty$) a `nan` (not-a-number), která označuje neurčitý výsledek.

Nás však z této knihovny nejvíce zajímají obsažené funkce určené pro sčítání řad používající extrapolační algoritmy, které chceme testovat.

5.1.1 Nastavení přesnosti

Mpmath používá globální pracovní přesnost. Nezasahuje do přesnosti jednotlivých čísel, ale nejprve provádí aritmetickou operaci a pak zaokrouhlí výsledek. Pracovní přesnost lze nastavit dvěma způsoby, parametrem `mp.prec`, který určuje přesnost počtem bitů, nebo `mp.dps` určujícím počet desetinných míst výsledku. Standardní vztah mezi nimi je `prec = 3,33 dps`, tedy přesnost na 333 bitů určuje 100 desetinných míst. Neexistuje žádné omezení na velikost čísel typu `mpf`.

5.2 Výpočet sumy

Jelikož naším hlavním zájmem je sčítání řad, použijeme z knihovny Mpmath funkci `nsum` určenou ke sčítání konečných i nekonečných řad [3]

$$S = \sum_{k=0}^{\infty} f(k).$$

Nejdůležitější volitelné parametry funkce `nsum` jsou

`tol` – určuje maximální konečnou chybu, standardně je stanovena jako pracovní přesnost hodnota epsilon,

`method` – vybereme extrapolační metodu, která se použije (výchozí volba je 'Richardson + Shanks'),

`maxterms` – ukončí výpočet po sečtení zadaného počtu členů (výchozí nastavení je 10*dps),

`steps` – parametr ve formě vektoru udává, kolik prvků řady se vždy sečte než se vyvolá proces extrapolace.

5.2.1 Použité metody

Jelikož neexistuje algoritmus, který by dokázal sečíst nekonečné množství členů řady, funkce `mpmath.nsum` implementuje několik sčítacích algoritmů, z nichž každý by měl být vhodný na sčítání jiného typu řady. Pokud sami nevybereme, je přednastavené použití kombinace metod `'r+s'`. Pomalejší, ale použitelná ve většině případů je kombinace všech metod `'r+s+e'`. Pro vysokou přesnost součtu nebo pro splnění požadavku na rychlejší výpočet je vhodné vyzkoušet každou metodu jednotlivě a určit, která z nich je nejvhodnější, a pak použít jen tu jednu.

Může použít některou z následujících metod

`'richardson' / 'r'` – Používá Richardsonovu extrapolaci. Užitečná když $f(k) \sim P(k)/Q(k)$ nebo $f(k) \sim (-1)^k P(k)/Q(k)$ pro polynomiální P a Q .

`'shanks' / 's'` – Používá Shanksovu transformaci. Typicky užitečná metoda když $f(k) \sim c^k$. Je vhodná i pro sčítání některých divergentních řad.

`'euler-maclaurin' / 'e'` – Používá Eulerovu–Maclaurinovu sumační formuli k aproximaci sumy pomocí integrálu. Vyžaduje numerickou derivaci a integraci.

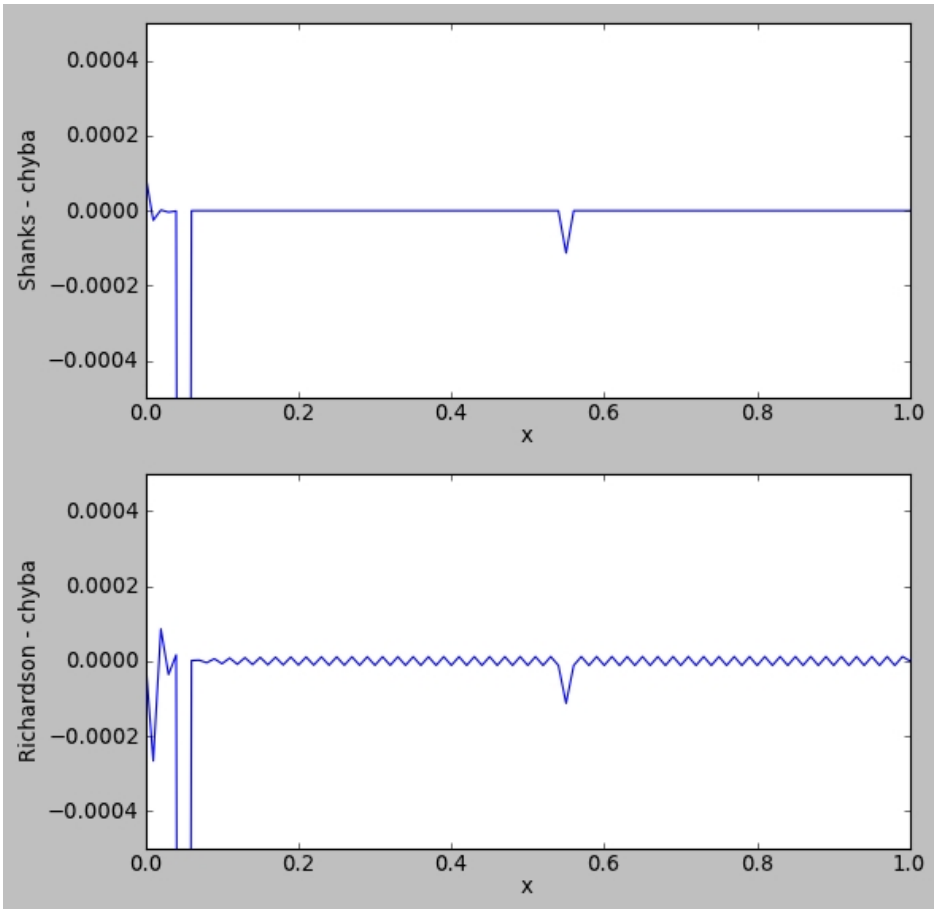
`'direct' / 'd'` – Nevyužívá žádný způsob extrapolace, jedná se o obyčejný přímý součet. Sčítání se ukončí po dosažení požadované tolerance.

5.3 Výsledky

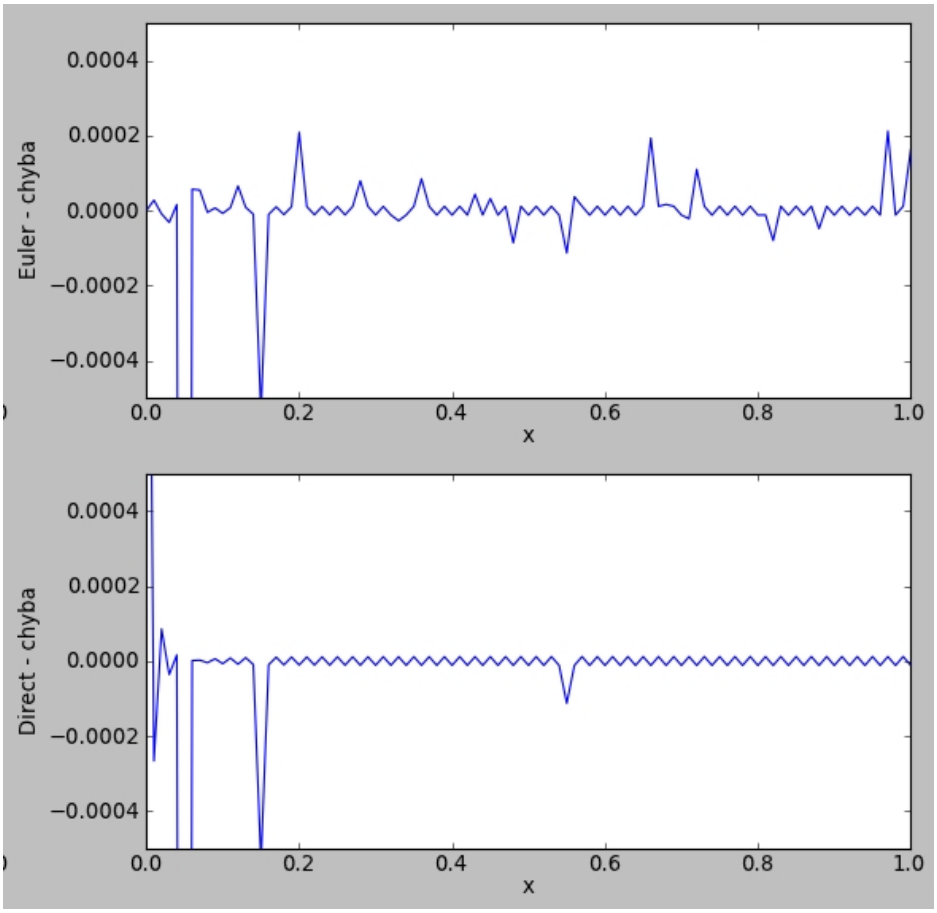
Ověření funkčnosti a efektivnosti jednotlivých metod provádíme tak, že jsme si zvolili vhodný vzorec (60), který obsahuje součet kosinové řady. Do něj jsme dosadili (63) a (64) a za parametry σ a k_1 dali jedničku, získali jsme tak vztah

$$\cosh(1-x) = \sinh(1) + \sum_{n=1}^{\infty} \frac{2\sinh(1)\cos(n\pi x)}{(n^2\pi^2)+1}.$$

Vyjádřili jsme chybu závislou na souřadnici x. Tu jsme získali tak, že pro každé x na intervalu $< 0,1 >$ s krokem $0,01$ jsme vypočítali součet testovací řady pomocí jednotlivých metod a výsledky jsme vždy odečetli od přesné hodnoty funkce $\cosh(1-x)$. Takto získané rozdíly jsou zaneseny do následujících grafů.



Obr. 13: Chyba - metody Shanks a Richardson



Obr. 14: Chyba - metody Euler a Direct

V grafech na obrázcích 13 a 14 vidíme, že všechny metody jsou nepřesné hlavně pro velmi malá x na začátku intervalu, zejména pro $x = 0,005$ je chyba ve všech případech velmi výrazná, až $0,008$. Dále je to již podstatně lepší, jen místy jsou vidět výraznější chyby. V tomto ohledu je pro naši testovací řadu ze všech nejhorší Eulerova–Maclaurinova metoda, naopak jako nejlepší se jeví Shanksova metoda. Tyto velké nepřesnosti jsou způsobeny nevhodným ukončovacím kritériem nastaveným v použitých metodách, kdy v některých případech dochází k příliš brzkému ukončení součtu. Naše testovací řada obsahuje funkci kosinus, která je periodická a když se její hodnota blíží nule, je špatně vyhodnocena ukončovacím kritériem a sumační proces je předčasně přerušen.

Po dalších testech, kdy jsme zkoušeli měnit počet sčítaných členů a toleranci, se potvrdilo, že Eulerova–Maclaurinova metoda je pro náš případ nejméně vhodná a z další testů jsme ji vyřadili. Ve snaze eliminovat velké chyby na začátku intervalu jsme přistoupili k zmenšení tolerance a sčítání po balíčcích, tedy klasickým způsobem jsme vždy přímo sčítali skupinu 100 členů a mezisoučty pak dále sčítali

pomocí testovaných metod. Po těchto úpravách dávaly všechny metody v podstatě totožný výsledek. Ten se sice výrazně zlepšil, ovšem pro malé hodnoty x je stále chyba výraznější. Zvolený postup měl ale za následek, že převládlo přímé sčítání a potlačil se vliv akceleračních metod, ale hlavně se sčítalo velké množství členů (řádově až statisíce) a došlo k značnému nárůstu výpočetního času, což je pro nás nežádoucí, neboť metody nám měly především výpočet urychlit. Po těchto závěrech jsme metody do našich programů nepoužili, je nutné je ještě více otestovat.

Závěr

Provedli jsme modifikaci vzorců popisujících spojitý model proudění v horninovém prostředí s puklinou na popis obecnějšího nespojitého modelu, kdy každá polovina řešené oblasti má jiné hodnoty parametrů popisujících vlastnosti prostředí. Řešili jsme diferenciální rovnice s okrajovými podmínkami, čímž jsme odvodili vzorce pro výpočet rozložení tlaku na 1D puklině a okolní 2D oblasti.

Získané vztahy jsme implementovali v Matlabu, kde jsme programy odladili především z hlediska přesnosti výpočtu. Zejména jsme otestovali několik způsobů sčítání nekonečných řad, které se staly největším problémem řešení úlohy. Totožnou úlohu jsme ještě v Matlabu vyřešili metodou konečných diferencí, která byla vzhledem k jednoduché geometrii řešeného modelu dobře použitelná. Z důvodu použití řídké sítě nejsou výsledky příliš přesné, takže jsme pouze porovnali rozložení tlaku v oblasti a ověřili tak správnost odvozených vzorců analytického řešení, ale ne přesnost výpočtu.

Poté jsme programy přepsali do programovacího jazyku Python za účel provedení výpočtu v prostředí ParaView, pro které jsou připravené konkrétní kompatibilní 1D–2D kombinované sítě s trojúhelníkovými elementy. Do ParaView lze importovat výsledky získané numerickými metodami, například ze softwaru Flow123d, které tak lze snadno porovnávat s výsledky našeho analytického řešení úlohy se stejnými parametry a počítané na stejné síti, což bylo i hlavním cílem práce.

Dále jsme se pokusili o použití metod pro akceleraci konvergence za účelem zrychlení výpočtů, který je v ParaView oproti Matlabu výrazně pomalejší. Testovali jsme tři metody implementované v Pythonovské knihovně Mpmath - Shanksovu, Richardsonovu a Eulerovu–Maclaurinovu. Metody byly testovány na konkrétní řadě obsažené v odvozených vztazích. Hlavní problém se vyskytl s přesností výsledků. Po sérii testů se podařilo najít dosti přesné řešení, ale použitými parametry byl omezen vliv metod a hlavně výpočet se stal výpočetně náročný a pomalý, což je nežádoucí. Z tohoto důvodu nebyly tyto akcelerační metody v programech použity a mohou se tak stát předmětem dalšího zkoumání a testování.

Seznam literatury

[1] AMY HENDERSON SQUILLACOTE, ETC.: *The ParaView Guide: A Parallel Visualization Application*, 2002.

[2] P. DRÁBEK, G. HOLUBOVÁ: *Parciální diferenciální rovnice* [online], učební text Západočeské univerzity v Plzni, 2011.
URL:<`http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/parcialni_diferencialni_rovnice.pdf`>

[3] F. JOHANSSON: *Sums, products, limits and extrapolation* [online], 2010.
URL:<`http://mpmath.googlecode.com/svn/trunk/doc/build/calculus/sums_limits.html`>

[4] S. MÍKA, P. PŘIKRYL, M. BRANDNER: *Speciální numerické metody - Numerické metody řešení okrajových úloh pro diferenciální rovnice*, ISBN, Plzeň, 2006.

[5] MPMATH: *Python library for arbitrary-precision floating-point arithmetic* [online], 2011.
URL:<`https://code.google.com/p/mpmath/`>

[6] PARAVIEW: *Scripting with Python* [online], 2007.
URL:<`http://www.paraview.org/Wiki/images/f/f9/Servermanager2.pdf`>

[7] K. REKTORYS a spolupracovníci: *Přehled užité matematiky II*, Prometheus, Praha, 2000.

[8] E. VITÁSEK: *Numerické metody*, SNTL, Praha, 1987.

[9] E. J. WENIGER: *Nonlinear sequence transformations for the acceleration of convergence and the summation of divergent series* [online], Computer Physics Reports Vol. 10, 189 - 371, 1989.
URL:<`http://arxiv.org/pdf/math/0306302v1.pdf`>

Přílohy

1. Řešení metodou konečných diferencí (program v Matlabu)

```
%Vstupní parametry
P1p = 5; P1m = 5; P2p = 12; P2m = 10;
k1 = 1; k2 = 5;
sigmap = 10; sigmam = 100;

% sit
M=61;

N=M+2;
h=2/(M-1);
h2=h*h;

xi=-1:h:1;
y1=-1:h:0;
y1((M+1)/2)=y1((M+1)/2)-eps;
y2=0:h:1;
y2(1)=y2(1)+eps;
yj=[y1,y2];

L=M*N;
A=zeros(L,L);
b=zeros(L,1);

% horni polovina
% první řádek
pos=1;
for i=1:1:M
  b(pos)=P2p;
  A(pos,pos)=1.0;
  pos=pos+1;
end;

% následující řádky
```

```
34 for i=2:1:((N-3)/2)
35
36   % první sloupec
37     A(pos,pos)=3*k2/h2;
38     A(pos,pos+1)=-k2/h2;
39     A(pos,pos-M)=-k2/h2;
40     A(pos,pos+M)=-k2/h2;
41     pos=pos+1;
42   % další sloupce
43   for j=2:1:M-1
44     A(pos,pos)=4*k2/h2;
45     A(pos,pos-1)=-k2/h2;
46     A(pos,pos+1)=-k2/h2;
47     A(pos,pos-M)=-k2/h2;
48     A(pos,pos+M)=-k2/h2;
49     pos=pos+1;
50   end;
51   % poslední sloupec
52     A(pos,pos)=3*k2/h2;
53     A(pos,pos-1)=-k2/h2;
54     A(pos,pos-M)=-k2/h2;
55     A(pos,pos+M)=-k2/h2;
56     pos=pos+1;
57 end;
58
59 % poslední řádek
60     A(pos,pos)=2*k2/h2+sigmap/h;
61     A(pos,pos+1)=-k2/h2;
62     A(pos,pos-M)=-k2/h2;
63     A(pos,pos+M)=-sigmap/h;
64     pos=pos+1;
65 for i=2:1:M-1
66     A(pos,pos)=3*k2/h2+sigmap/h;
67     A(pos,pos-1)=-k2/h2;
68     A(pos,pos+1)=-k2/h2;
69     A(pos,pos-M)=-k2/h2;
70     A(pos,pos+M)=-sigmap/h;
```

```
71      pos=pos+1;
72  end;
73      A(pos,pos)=2*k2/h2+sigmap/h;
74      A(pos,pos-1)=-k2/h2;
75      A(pos,pos-M)=-k2/h2;
76      A(pos,pos+M)=-sigmap/h;
77      pos=pos+1;
78
79  %1D
80    % levá Dirichletova podmínka
81    b(pos)=P1m;
82    A(pos,pos)=1.0;
83    pos=pos+1;
84
85    for j=2:1:M-1
86      A(pos,pos)=2*k1/h2+(sigmap+sigmam);
87      A(pos,pos-1)=-k1/h2;
88      A(pos,pos+1)=-k1/h2;
89      A(pos,pos-M)=-sigmap;
90      A(pos,pos+M)=-sigmam;
91      pos=pos+1;
92    end;
93
94    % pravá Dirichletova podmínka
95    b(pos)=P1p;
96    A(pos,pos)=1.0;
97    pos=pos+1;
98
99  % dolni polovina
100 % první řádek
101     A(pos,pos)=2*k2/h2+sigmam/h;
102     A(pos,pos+1)=-k2/h2;
103     A(pos,pos-M)=-sigmam/h;
104     A(pos,pos+M)=-k2/h2;
105     pos=pos+1;
106 for i=2:1:M-1
107     A(pos,pos)=3*k2/h2+sigmam/h;
```

```
108      A(pos,pos-1)=-k2/h2;
109      A(pos,pos+1)=-k2/h2;
110      A(pos,pos-M)=-sigmam/h;
111      A(pos,pos+M)=-k2/h2;
112      pos=pos+1;
113  end;
114      A(pos,pos)=2*k2/h2+sigmam/h;
115      A(pos,pos-1)=-k2/h2;
116      A(pos,pos-M)=-sigmam/h;
117      A(pos,pos+M)=-k2/h2;
118      pos=pos+1;
119
120  % následující řádky
121  for i=(((N+1)/2)+2):1:(N-1)
122     % první sloupec
123      A(pos,pos)=3*k2/h2;
124      A(pos,pos+1)=-k2/h2;
125      A(pos,pos-M)=-k2/h2;
126      A(pos,pos+M)=-k2/h2;
127      pos=pos+1;
128     % další sloupce
129    for j=2:1:M-1
130      A(pos,pos)=4*k2/h2;
131      A(pos,pos-1)=-k2/h2;
132      A(pos,pos+1)=-k2/h2;
133      A(pos,pos-M)=-k2/h2;
134      A(pos,pos+M)=-k2/h2;
135      pos=pos+1;
136    end;
137     % poslední sloupec
138      A(pos,pos)=3*k2/h2;
139      A(pos,pos-1)=-k2/h2;
140      A(pos,pos-M)=-k2/h2;
141      A(pos,pos+M)=-k2/h2;
142      pos=pos+1;
143  end;
144
```

```
145  % poslední řádek
146 for i=1:1:M
147   b(pos)=P2m;
148   A(pos,pos)=1.0;
149   pos=pos+1;
150 end;
151
152 % řešení soustavy rovnic
153 num_x=A\b;
154
155 c=1; d=0;
156
157 for i=(M*N):-1:1
158     d=d+1;
159     num_p2xy(c,d)=num_x(i);
160     if d==M
161         c=c+1;
162         d=0;
163     end;
164 end;
165
166 V2 = num_p2xy;
167 V1 = V2((N+1)/2,:);
168 V2((N+1)/2,:)=[];
169 V2 = V2';
170
171 % Vizualizace
172 sz=size (yj);
173 sz=sz(2);
174 D1=V2(:,(sz/2));
175 D2=V2(:,(sz/2)+1);
176 figure(1)
177 plot(xi,D1,xi,D2,xi,V1)
178 figure(2)
179 surf(yj,xi,V2)
```

2. Výpočet nespojitého modelu (skript pro ParaView)

```
import math
import numpy

class Parameters:
    k1 = 1.0; k2 = 5.0
    P1 = 5.0; P2p = 12.0; P2m = 10.0
    sigmap = 10.0; sigmam = 100.0

    # dopocitane parametry
    sigmav = sigmap + sigmam
    P1v = (sigmap * P1 + sigmam * P1) / sigmav
    P2v = (sigmap * P2p + sigmam * P2m) / sigmav
    K = math.sqrt(sigmav/k1)

    # parametry pro soucty rad
    N = 100000
    interval_size = 100
    tolerance = 1e-10

    sx = []; sy = []
    Vp = []; Vm = []; Un = []

    B0v = 0.0; B0p = 0.0; B0m = 0.0
    omega = 0.0

    # vypocet soustavy rovnic pro alfa+ a alfa-
    def alfa_pm(self,n):
        ah = ((self.k2*n*math.pi)/self.sigmap) * (1+math.
           exp(-2*n*math.pi)) + 1 - math.exp(-2*n*math.pi
           ) - ((self.sigmap*(1-math.exp(-2*n*math.pi)))
           /(n*n*math.pi*math.pi*self.k1+self.sigmav))
        bh = (-self.sigmam*(1-math.exp(-2*n*math.pi))) /
           (n*n*math.pi*math.pi*self.k1+self.sigmav)
        ch = (-self.sigmap*(1-math.exp(-2*n*math.pi))) /
           (n*n*math.pi*math.pi*self.k1+self.sigmav)
```

```
31          dh = ((self.k2*n*math.pi)/self.sigmam) * (1+math.
               exp(-2*n*math.pi)) + 1 - math.exp(-2*n*math.pi
               ) - ((self.sigmam*(1-math.exp(-2*n*math.pi)))
               /(n*n*math.pi*math.pi*self.k1+self.sigmav))
32          gh = 4 * self.k1 * self.K * math.sinh(self.K)
33
34          Lr = numpy.mat([[ah, bh], [ch, dh]],dtype=float)
35          Pr = numpy.mat([[gh], [gh]],dtype=float)
36
37          Va = numpy.linalg.solve(Lr,Pr)
38          return Va
39
40      # vypocet soustavy rovnic pro omega, B0v, B0+, B0-
41      def vyp_koef(self):
42
43          # I.rovnice
44          I = numpy.mat([(self.k2/self.sigmap)+1, 0, -1, -(
               math.sinh(self.K)/self.K)], dtype=float)
45
46          # II.rovnice
47          II = numpy.mat([0.0, (self.k2/self.sigmam)+1,
               -1.0, -(math.sinh(self.K)/self.K)])
48
49          # III.rovnice
50          III = numpy.mat([(self.sigmap/self.sigmav), (self
               .sigmam/self.sigmav), -1.0, 0.0])
51
52          # IV.rovnice
53          suma1 = self.summarize(1, 0, 0)
54          IV = numpy.mat([0.0, 0.0, -1.0, -(suma1 + math.
               cosh(self.K))])
55
56          # soustava rovnic
57          L = numpy.bmat('I;II;III;IV')
58          R = numpy.mat([[self.P2p-self.P2v], [self.P2m-
               self.P2v], [0.0], [self.P1v-self.P2v]])
59          Vk = numpy.linalg.solve(L,R)
```

```
60          return Vk
61
62      # vypocet Un
63      def u_n(self, n):
64          Vu = (self.omega * (1-math.exp(-2*n*math.pi)) *
               ((self.Vp[n-1]*self.sigmap) + (self.Vm[n-1]*
               self.sigmam))) / (2*(n*n*math.pi*math.pi*self.
               k1 + self.sigmav)*(n*n*math.pi*math.pi*self.k1
                + self.sigmav))
65          return Vu
66
67      # predpocitani hodnot koeficientu
68      def predpocet(self):
69
70          # vypocet Vp, Vm
71          for n in range(1,self.N+1,1):
72              tmp1 = self.alfa_pm(n)
73              self.Vp.append(tmp1[0])
74              self.Vm.append(tmp1[1])
75
76          # vypocet omega a B0
77          tmp3 =  self.vyp_koef()
78          self.omega = tmp3[3]
79          self.B0v = tmp3[2]
80          self.B0p = tmp3[0]
81          self.B0m = tmp3[1]
82
83          # vypocet Un
84          for n in range(1,self.N+1,1):
85              tmp2 = self.u_n(n)
86              self.Un.append(tmp2)
87
88      # vypocet hodnoty p1(x)
89      def p_1d(self, x):
90          suma2 = self.summarize(2, 0, 0)
91          x = 1 - math.fabs(x)
92          suma3 = self.summarize(3, x, 0)
```

```
93          p1 = ( (self.P1v - self.P2v + self.B0v + suma2) *
                (math.cosh(self.K*(1-x))/math.cosh(self.K)) )
                + self.P2v - self.B0v - suma3
94          return p1
95
96      # vypocet hodnoty p2(x,y)
97      def p_2d_p(self, x, y):
98          x = 1 - math.fabs(x)
99          y = math.fabs(y)
100         suma4 = self.summarize(4, x, y)
101         p2 = self.P2p - self.B0p + self.B0p*y - suma4
102         return p2
103
104     def p_2d_m(self, x, y):
105         x = 1 - math.fabs(x)
106         y = math.fabs(y)
107         suma5 = self.summarize(5, x, y)
108         p2 = self.P2m - self.B0m + self.B0m*y - suma5
109         return p2
110
111 # funkce pro vypocet sum
112     def suma1(self, n):  # pouzita ve vypoctu koeficientu
            omega a B0
113         s1 = ((1-math.exp(-2*n*math.pi)) * (self.Vp[n-1]*
               self.sigmap + self.Vm[n-1]*self.sigmam)) /
               (2*(n*n*math.pi*math.pi*self.k1 + self.sigmav)
               *(n*n*math.pi*math.pi*self.k1 + self.sigmav))
114         return s1
115
116     def suma2(self, n):  # pouzita ve vypoctu p1x
117         s2 = self.Un[n-1]
118         return s2
119
120     def suma3(self, n, x):  # pouzita ve vypoctu p1x
121         s3 = (self.Un[n-1]*math.cos(n*math.pi*x))
122         return s3
123
```

```
124     def suma4(self, n, x, y):  # pouzita ve vypoctu p2xy
125         s4 = (self.Vp[n-1] * self.omega * (math.exp(-n*
               math.pi*y) - math.exp(-n*math.pi*(2-y))) *
               math.cos(n*math.pi*x)) / (2*(n*n*math.pi*math.
               pi*self.k1+self.sigmav))
126         return s4
127
128     def suma5(self, n, x, y):  # pouzita ve vypoctu p2xy
129         s5 = (self.Vm[n-1] * self.omega * (math.exp(-n*
               math.pi*y) - math.exp(-n*math.pi*(2-y))) *
               math.cos(n*math.pi*x)) / (2*(n*n*math.pi*math.
               pi*self.k1+self.sigmav))
130         return s5
131
132     def summarize(self, sum_num, x, y):
133         local_sum = []
134         interval_begin = 0
135
136         for interval_end in range(self.interval_size,
               self.N, self.interval_size):
137             interval_sum = 0.0
138
139             for h in range(interval_end-1,interval_begin
                   -1,-1):
140                 if sum_num == 1:
141                     interval_sum = interval_sum + self.
                           suma1(h)
142                 elif sum_num == 2:
143                     interval_sum = interval_sum + self.
                           suma2(h)
144                 elif sum_num == 3:
145                     interval_sum = interval_sum + self.
                           suma3(h, x)
146                 elif sum_num == 4:
147                     interval_sum = interval_sum + self.
                           suma4(h, x, y)
148                 elif sum_num == 5:
```

```
149                     interval_sum = interval_sum + self.
                           suma5(h, x, y)
150             if math.fabs(interval_sum) < self.tolerance :
151                 break
152             local_sum.append(interval_sum)
153             interval_begin = interval_begin + self.
                   interval_size
154         total_sum = 0.0
155
156         for number in reversed(local_sum):
157             total_sum = total_sum + number
158         return total_sum
159
160 # konec tridy Parameters
161
162 par=Parameters()
163
164 # provedeni predpoctu
165 par.predpocet()
166
167 pdi = self.GetPolyDataInput()
168 pdo = self.GetOutput()
169
170 newData = vtk.vtkDoubleArray()
171 newData.SetName("Pressure")
172
173 nc = pdi.GetNumberOfCells()
174
175 for i in range (nc):
176       cell = pdi.GetCell(i)
177       pb = cell.GetNumberOfPoints()
178       if pb==3:
179             p1 = pdi.GetPoint(cell.GetPointId(0))
180             x1, y1, z1 = p1[:3]
181             p2 = pdi.GetPoint(cell.GetPointId(1))
182             x2, y2, z2 = p2[:3]
183             p3 = pdi.GetPoint(cell.GetPointId(2))
```

```
184             x3, y3, z3 = p3[:3]
185             tx=math.fabs((x1+x2+x3)/(3*5.5))
186             ty=(y1+y2+y3)/(3*5.5)
187
188             if ty > 0 :
189                  ty=math.fabs(ty)
190                  v1 = par.p_2d_p(tx,ty)
191
192             if ty < 0 :
193                  ty=math.fabs(ty)
194                  v1 = par.p_2d_m(tx,ty)
195
196             newData.InsertNextValue(v1)
197             pdo.GetCellData().AddArray(newData)
198
199         if pb==2:
200             p1 = pdi.GetPoint(cell.GetPointId(0))
201             x1, y1, z1 = p1[:3]
202             p2 = pdi.GetPoint(cell.GetPointId(1))
203             x2, y2, z2 = p2[:3]
204             tx=math.fabs((x1+x2)/(2*5.5))
205
206             v2=par.p_1d(tx)
207             newData.InsertNextValue(v2)
208             pdo.GetCellData().AddArray(newData)
```

Seznam příloh na CD

1. **Programy v Matlabu**
   (a) Metoda_konecnych_diferenci.m
   (b) Nespojity_model.m
   (c) Spojity_model.m
2. **Programy v Pythonu**
   (a) Nespojity_model.py
   (b) Extrapolacni_metody_chyby.py
3. **Programy v ParaView**
   (a) sit.vtu
   (b) Spojity_model.py
   (c) Nespojity_model.py