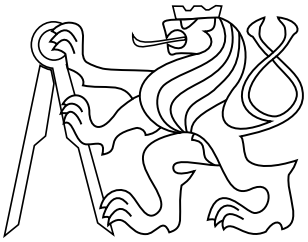
bakalářská práce

Algoritmus pro numerickou integraci Kuzněcovovy rovnice na GPU

Jan Mrňa



May 2014

Ing. Milan Červenka, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra kybernetiky

Poděkování

Rád bych poděkoval svému vedoucímu, Ing. Milanu Červenkovi, Ph.D., jehož podpora mi umožnila zdárně tuto práci dokončit.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jan M r ň a

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Algoritmus pro numerickou integraci Kuzněcovovy rovnice na GPU

Pokyny pro vypracování:

1. Navrhněte semidiskrétní numerický algoritmus pro řešení Kuzněcovovy rovnice nelineární akustiky pro popis stojatých zvukových vln v akustických rezonátorech. K výpočtu derivací podle prostorových souřadnic použijte metodu centrálních konečných diferencí a Fourierovu metodu.
2. Numerický algoritmus otestujte a implementujte pro paralelní běh na grafické kartě s využitím technologie NVIDIA-CUDA.
3. S využitím implementovaného algoritmu proveďte základní numerické experimenty v oblasti stojatých zvukových vln konečných amplitud v akustických rezonátorech proměnného průřezu.

Seznam odborné literatury:

[1] Yu. A. Il'inskii, B. Lipkens, T. S. Lucas, T. W. Van Doren and E. A. Zabolotskaya: Nonlinear standing waves in an acoustical resonator, J. Acoust. Soc. Am. 104, p2664, 1998.
[2] M. Červenka, M. Šoltés, M. Bernařík: Optimal shaping of acoustic resonators for the generation of high-amplitude standing waves, submitted to J. Acoust. Soc. Am., 2013.
[3] N. Albin, O. P. Bruno, T. Y. Cheung, R. O. Cleveland: Fourier continuation methods for high-fidelity simulation of nonlinear acoustic beams, J. Acoust. Soc. Am. 132 (4), p2371, 2012.
[4] D. Zwillinger: Handbook of Differential Equations, Academic Press, 1997.
[5] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery: Numerical Recipes 3rd Edition: The Art of Scientific Computing, Cambridge University Press, 2007.
[6] J. Sanders, E. Kandrot: CUDA by Example: An Introduction to General-Purpose GPU Programming, Addison-Wesley Professional; 1 edition, 2010.
[7] N. Wilt: CUDA Handbook: A Comprehensive Guide to GPU Programming, Addison-Wesley Professional; 1 edition, 2013.
[8] P. Pacheco: An Introduction to Parallel Programming , Morgan Kaufmann, 2011.

Vedoucí bakalářské práce: Ing. Milan Červenka, Ph.D.

Platnost zadání: do konce letního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2014

Abstrakt

Tato práce se zabývá návrhem semidiskrétního numerického algoritmu pro řešení Kuzněcovovy rovnice nelineární akustiky pro popis stojatých zvukových vln v akustických rezonátorech a jeho implementaci na grafických kartách NVIDIA pomocí technologie CUDA. K výpočtu derivací podle prostorových souřadnic je použita metoda centrálních konečných diferencí a Fourierova metoda. Uvedené algoritmy jsou použity pro provedení základních numerických experimentů v oblasti stojatých zvukových vln konečných amplitud v akustických rezonátorech proměnného průřezu. Obě metody jsou porovnány z hlediska přesnosti a rychlosti s verzí pracující jednovláknově na CPU. Uvedený program může být upraven i pro řešení jiných evolučních parciálních diferenciálních rovnic či jejich soustav.

Klíčová slova

CUDA, GPU, numerické řešení, parciální diferenciální rovnice

Abstrakt

This work deals with design of semidiscrete numerical algorithm for solving non-linear acoustic Kuznetsov equation for describing standing waves in acoustic resonators and its implementation on NVIDIA GPUs using CUDA technology. Derivatives with respect to spatial coordinate are calculated using central finite differences and Fourier methods. These algorithms are used to conduct basic numerical experiments with finite-amplitude acoustic waves in axisymmetric resonators. Speed and accuracy of both methods are assesed and compared with single-threaded CPU implementation. The programme can be modified even for solving other types of evolutionary partial differential equations or their sets.

Keywords

CUDA, GPU, numerical solution, partial differential equation

Obsah

Zkratky

Použité zkratky:

PDE	Partial Differential Equation
CFD	Central Finite Differences
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
RK4	Runge-Kutta 4. řádu
PK45	Prediktor 4. řádu – Korektor 5. řádu
GPU	Graphics Processing Unit
GPGPU	General Purpose Graphics Processing Unit
OOP	Objektově orientované programování

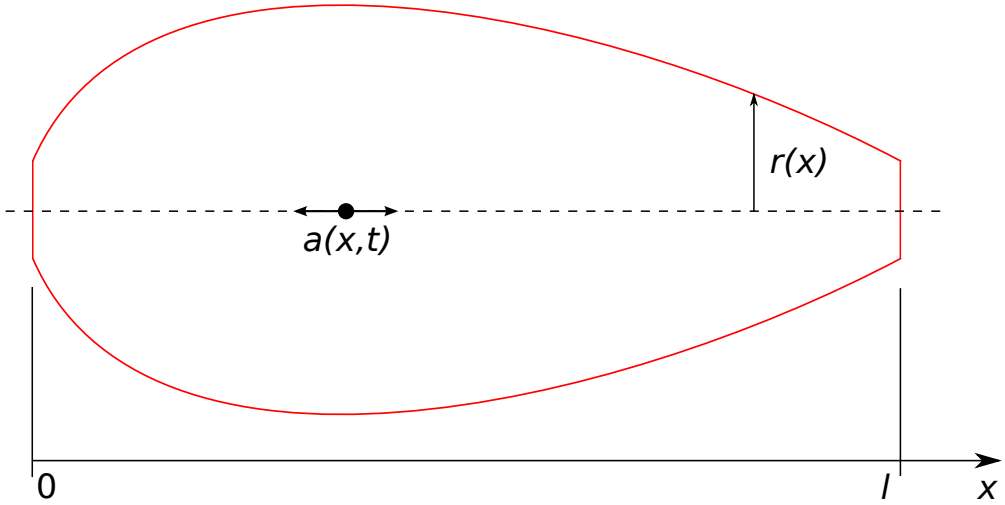
1. Úvod

Generování akustických polí extrémních amplitud v pevnostěnných osově symetrických rezonátorech má využití v mnoha oborech, např. termoakustice či medicíně. Pro maximalizaci efektivity zařízení je nutné zvolit nejvhodnější tvar rezonátoru, aby nedocházelo ke vzniku rázových vln (viz [1]), ve kterých je disipována energie ve formě tepla. Kvalitu návrhu tvaru rezonátoru je nutné nejdříve ověřit výpočtem. Analytické řešení příslušných rovnic (reprezentovaných nelineárními parciálními diferenciálními rovnicemi) však není známo ani pro nejjednodušší případ (rezonátor konstantního průřezu), je proto nutné jej hledat numericky. Tato procedura je však velmi náročná na výpočetní výkon počítače, proto je pro řešení tohoto problému vhodné využít masivní paralelizace. Jedním z komerčně (a cenově) dostupných nástrojů pro masivní paralelizaci je použití GPGPU, tedy grafických procesorů schopných provádět obecné výpočty (ty jsou v dnešní době obsaženy na většině grafických karet).

Tato práce popisuje implementaci řešiče provedenou pomocí technologie CUDA, která umožňuje využít výpočetní potenciál grafických karet NVIDIA na obecné výpočty.

2. Matematický model

V pevnostěnných rezonátorech proměnného průřezu vyplněných plynem je možné generovat akustické pole extrémních amplitud [1]. Obvykle jsou tyto rezonátory osově symetrické, jejich poloměr můžeme vyjádřit jako funkci prostorové souřadnice x, tedy $r = r(x)$. Uvažujeme případy, kdy největší poloměr rezonátoru je mnohem menší než jeho délka l, proto můžeme veličiny popisující akustické pole q_a považovat za jednorozměrné, tedy $q_a = q_a(x,t)$, kde t je čas. Zvukové vlny lze budit například tak, že rezonátor se pohybuje periodicky ve směru své osy se zrychlením $a_s = a_s(t)$, pole je tedy buzeno objemovou setrvačnou silou.



Obrázek 1. Rezonátor buzený objemovou setrvačnou silou.

2.1. Modelová rovnice

Akustické pole vysokých amplitud lze v pevnostěnném rezonátoru proměnného průřezu popsat jednorozměrnou vlnovou rovnicí (viz [2]):

$$\frac{\partial^2\varphi}{\partial t^2} - \frac{c_0^2}{r^2}\frac{\partial}{\partial x}\left(r^2\frac{\partial\varphi}{\partial x}\right) = -\frac{\partial}{\partial t}\left(\frac{\partial\varphi}{\partial x}\right)^2 - \frac{\gamma-1}{2c_0^2}\frac{\partial}{\partial t}\left(\frac{\partial\varphi}{\partial t}\right)^2 - \\ - x\frac{\mathrm{d}a_\mathrm{s}}{\mathrm{d}t} + \frac{\delta}{c_0^2}\frac{\partial^3\varphi}{\partial t^3} - \frac{2c_0^2\varepsilon}{\sqrt{\pi}r^2}\int_{-\infty}^{t}\frac{1}{\sqrt{t-\tau}}\frac{\partial}{\partial x}\left[r\frac{\partial\varphi(x,\tau)}{\partial x}\right]\mathrm{d}\tau, \quad (1)$$

Kde φ je rychlostní potenciál, a_s je zrychlení rezonátoru, t je čas, x prostorová souřadnice ve směru osy rezonátoru, $r = r(x)$ je poloměr rezonátoru, γ je adiabatický exponent, $\delta = [\zeta + 4\eta/3 + \kappa(1/c_V - 1/c_P)]/\rho_0$ je koeficient difúze, η a ζ jsou koeficienty příčné a podélné viskozity, κ je koeficient tepelné vodivosti, c_P a c_V je měrná tepelná kapacita při konstantním tlaku a objemu a ρ_0 je rovnovážná hustota plynu. Koeficient ε je definován jako $\varepsilon = \sqrt{\nu_0}[1+(\gamma-1)/\sqrt{Pr}]$, kde $\nu_0 = \eta/\rho_0$ je kinematická viskozita a $Pr = \eta c_\mathrm{p}/\kappa$ je Prandtlovo číslo.

Členy na levé straně rovnice (1) představují bezeztrátovou Websterovu lineární vlnovou rovnici, první dva členy na pravé straně reprezentují nelineární jevy, třetí člen reprezentuje ztráty akustické energie v objemu plynu, čtvrtý člen ztráty v mezní vrstvě.

Akustický tlak a akustická rychlost mohou být z rovnice (1) spočteny:

$$v = \frac{\partial \varphi}{\partial x}, \tag{2a}$$

$$p' = -\rho_0 \frac{\partial \varphi}{\partial t} - \rho_0 a_\mathrm{s} x + \frac{\rho_0}{2c_0^2}\left(\frac{\partial \varphi}{\partial t}\right)^2 - \frac{\rho_0}{2}\left(\frac{\partial \varphi}{\partial x}\right)^2 + \frac{1}{c_0^2}\left(\zeta + \frac{4}{3}\eta\right)\frac{\partial^2 \varphi}{\partial t^2}. \tag{2b}$$

Na rozhraní plynu a (pevných) stěn rezonátoru má normálová komponenta vektoru akustické rychlosti stejnou hodnotu jako normálová komponenta rychlosti stěny. Z toho můžeme odvodit okrajové podmínky:

$$v(x=0,t) = \left.\frac{\partial \varphi}{\partial x}\right|_{x=0} = 0 \qquad \text{a} \qquad v(x=l,t) = \left.\frac{\partial \varphi}{\partial x}\right|_{x=l} = 0. \tag{3}$$

Uvažujeme, že na počátku není žádná vlna v rezonanční dutině přítomna, a plyn je homogenně rozložen. Z toho můžeme odvodit počáteční podmínky pro celý rezonátor:

$$\varphi(t=0,x) = 0 \quad \text{a} \quad \left.\frac{\partial \varphi}{\partial t}\right|_{t=0} = 0 \qquad \text{pro} \qquad x \in \langle 0, l\rangle. \tag{4}$$

2.2. Úprava rovnice

Analytické řešení rovnice (1) splňující okrajové podmínky (3) nebylo dosud nalezeno ani pro nejjednodušší případ $r(x) = konst$, ale je možné jej najít numericky.

Rovnice (1) je integrována v čase. Z tohoto důvodu je vhodné snížit řád derivací v čase pomocí následujících úprav. Z rovnice (1) můžeme psát:

$$\frac{\partial^2 \varphi}{\partial t^2} = \frac{c_0^2}{r^2}\frac{\partial}{\partial x}\left(r^2 \frac{\partial \varphi}{\partial x}\right) + \mathcal{O}(\mu^2), \tag{5}$$

kde člen $\mathcal{O}(\mu^2)$ reprezentuje všechny členy druhého řádu z rovnice (1). Pokud dosadíme (5) do druhého členu pravé strany rovnice (1), dostaneme:

$$\frac{\gamma-1}{2c_0^2}\frac{\partial}{\partial t}\left(\frac{\partial \varphi}{\partial t}\right)^2 = \frac{\gamma-1}{c_0^2}\frac{\partial \varphi}{\partial t}\frac{\partial^2 \varphi}{\partial t^2} = \frac{\gamma-1}{r^2}\frac{\partial}{\partial x}\left(r^2\frac{\partial \varphi}{\partial x}\right)\frac{\partial \varphi}{\partial t} + \mathcal{O}(\mu^4).$$

Podobně můžeme psát pro čtvrtý člen na pravé straně rovnice (1):

$$\frac{\delta}{c_0^2}\frac{\partial^3 \varphi}{\partial t^3} = \frac{\delta}{r^2}\frac{\partial}{\partial x}\left(r^2 \frac{\partial^2 \varphi}{\partial x \partial t}\right) + \mathcal{O}(\mu^4).$$

Dosadíme zpět do rovnice (1) a členy čtvrtého řádu vypustíme (výsledná chyba bude zanedbatelně malá):

$$\frac{\partial^2 \varphi}{\partial t^2} - \frac{c_0^2}{r^2}\frac{\partial}{\partial x}\left(r^2\frac{\partial \varphi}{\partial x}\right) = -\frac{\partial}{\partial t}\left(\frac{\partial \varphi}{\partial x}\right)^2 - \frac{\gamma-1}{r^2}\frac{\partial}{\partial x}\left(r^2\frac{\partial \varphi}{\partial x}\right)\frac{\partial \varphi}{\partial t} - \\ - x\frac{\mathrm{d}a_\mathrm{s}}{\mathrm{d}t} + \frac{\delta}{r^2}\frac{\partial}{\partial x}\left(r^2\frac{\partial^2 \varphi}{\partial x \partial t}\right) - \frac{2c_0^2\varepsilon}{\sqrt{\pi}r^2}\int_{-\infty}^{t}\frac{1}{\sqrt{t-\tau}}\frac{\partial}{\partial x}\left[r\frac{\partial \varphi(x,\tau)}{\partial x}\right]\mathrm{d}\tau. \tag{6}$$

Poslední člen na pravé straně rovnice (6) představující ztráty v mezní vrstvě komplikuje numerickou integraci rovnice. Naštěstí je možné tento člen vynechat a nahradit absorbční koeficient δ koeficientem $\delta' > \delta$, při zachování dobré shody s naměřenými daty (viz [3]). Rovnice (6) pak má tvar:

$$\frac{\partial^2 \varphi}{\partial t^2} - \frac{c_0^2}{r^2}\frac{\partial}{\partial x}\left(r^2\frac{\partial \varphi}{\partial x}\right) = -\frac{\partial}{\partial t}\left(\frac{\partial \varphi}{\partial x}\right)^2 - \\ - \frac{\gamma-1}{r^2}\frac{\partial}{\partial x}\left(r^2\frac{\partial \varphi}{\partial x}\right)\frac{\partial \varphi}{\partial t} - x\frac{\mathrm{d}a_\mathrm{s}}{\mathrm{d}t} + \frac{\delta'}{r^2}\frac{\partial}{\partial x}\left(r^2\frac{\partial^2 \varphi}{\partial x \partial t}\right). \tag{7}$$

Rovnice (2b) může být stejným způsobem přepsána na:

$$p' = -\rho_0\frac{\partial \varphi}{\partial t} - \rho_0 a_\mathrm{s} x + \frac{\rho_0}{2c_0^2}\left(\frac{\partial \varphi}{\partial t}\right)^2 - \frac{\rho_0}{2}\left(\frac{\partial \varphi}{\partial x}\right)^2 + \left(\zeta + \frac{4}{3}\eta\right)\frac{1}{r^2}\frac{\partial}{\partial x}\left(r^2\frac{\partial \varphi}{\partial x}\right). \tag{8}$$

2.3. Bezrozměrné veličiny

Pro účely numerické integrace je vhodné použít bezrozměrné veličiny:

$$X = \frac{x}{l}, \quad T = \omega t, \quad R = \frac{r}{l}, \quad A = \frac{a}{l\omega_0^2}, \quad \Phi = \frac{\varphi}{l^2\omega_0}, \quad V = \frac{v}{\pi c_0} \quad \text{a} \quad P = \frac{p'}{\rho_0 \pi^2 c_0^2},$$

kde ω je úhlová rychlost buzení rezonátoru a

$$\omega_0 = \frac{\pi c_0}{l}$$

je úhlová rychlost příslušící první rezonanční frekvenci (odpovídající půlvlně) rezonátoru s konstantním průřezem $r(x) = konst$. Po substituci bezrozměrných veličin do rovnice (7) dostáváme:

$$\frac{\partial^2\Phi}{\partial T^2} - \frac{1}{\pi^2\Omega^2 R^2}\frac{\partial}{\partial X}\left(R^2\frac{\partial\Phi}{\partial X}\right) = -\frac{1}{\Omega}\frac{\partial}{\partial T}\left(\frac{\partial\Phi}{\partial X}\right)^2 - \\ - \frac{\gamma-1}{\Omega R^2}\frac{\partial\Phi}{\partial T}\frac{\partial}{\partial X}\left(R^2\frac{\partial\Phi}{\partial X}\right) - \frac{X}{\Omega}\frac{\mathrm{d}A_\mathrm{s}}{\mathrm{d}T} + \frac{G}{\pi^3\Omega R^2}\frac{\partial}{\partial X}\left(R^2\frac{\partial^2\Phi}{\partial X\partial T}\right), \quad (9)$$

kde $\Omega = \omega/\omega_0$ je bezrozměrná frekvence a $G = \pi\omega_0\delta'/c_0^2$ je atenuační koeficient. Po zavedení nové proměnné $\Psi = \partial\Phi/\partial T$ lze rovnici (9) přepsat do tvaru:

$$\frac{\partial\Psi}{\partial T} = -\frac{2}{\Omega}\frac{\partial\Phi}{\partial X}\frac{\partial\Psi}{\partial X} + \left(\frac{1}{\pi^2\Omega^2} - \frac{\gamma-1}{\Omega}\Psi\right)\frac{1}{R^2}\frac{\partial}{\partial X}\left(R^2\frac{\partial\Phi}{\partial X}\right) - \\ - \frac{X}{\Omega}\frac{\mathrm{d}A_\mathrm{s}}{\mathrm{d}T} + \frac{G}{\pi^3\Omega R^2}\frac{\partial}{\partial X}\left(R^2\frac{\partial\Psi}{\partial X}\right), \quad (10a)$$

$$\frac{\partial\Phi}{\partial T} = \Psi, \quad (10b)$$

Akustický tlak a akustická rychlost s použitím v bezrozměrných proměnných jsou spočteny dosazením do (2a) a (8):

$$v = \pi c_0 \frac{\partial\Phi}{\partial X}, \quad (11a)$$

$$p' = \rho_0\pi^2 c_0^2\left[-\Omega\Psi - A_\mathrm{s}X + \frac{\pi^2\Omega^2}{2}\Psi^2 - \frac{1}{2}\left(\frac{\partial\Phi}{\partial X}\right)^2 + \frac{G'}{\pi^3 R^2}\frac{\partial}{\partial R}\left(R^2\frac{\partial\Phi}{\partial X}\right)\right], \quad (11b)$$

kde $G' = \pi\omega_0(\zeta + 4\eta/3)/(\rho_0 c_0^2)$.

Okrajové podmínky vyjádřené pomocí bezrozměrných veličin mají tvar (z (3)):

$$\left.\frac{\partial\Phi}{\partial X}\right|_{X=0} = 0 \qquad \text{a} \qquad \left.\frac{\partial\Phi}{\partial X}\right|_{X=1} = 0. \quad (12)$$

Jako počáteční podmínku opět uvažujeme homogenní rozložení plynu v dutině v nulovém čase, tedy z (4):

$$\Phi(T=0, X) = 0 \quad \text{and} \quad \Psi(T=0, X) = 0 \qquad \text{pro} \qquad X \in \langle 0, 1\rangle. \quad (13)$$

3. Metody numerického řešení

Tato kapitola se zabývá numerickými metodami použitými k řešení soustavy rovnic (10). Parciální diferenciální rovnice jsou v tomto případě převedeny na soustavu obyčejných diferenciálních rovnic, které jsou následně integrovány pomocí algoritmů pro integraci soustav obyčejných diferenciálních rovnic.

3.1. Diskretizace prostorové souřadnice

Problém řešený na počítači je nutné diskretizovat. Bezrozměrná prostorová souřadnice X nabývá hodnot od 0 do 1. Diskretizujeme s krokem $h = 1/N$:

$$X_i = ih, \qquad i = 0, 1, 2, \ldots, N.$$

Veličiny popisující akustické pole pak přejdou do tvaru $Q(X,T) \to Q(X_i,T) = Q_i(T)$.

3.2. Výpočet derivace podle prostorové souřadnice

Soustava rovnic (10) obsahuje derivace podle prostorové souřadnice. V této práci byly pro jejich výpočet použity dva přístupy: metoda konečných diferencí a Fourierova metoda. Při použití Fourierovy metody je k výpočtu prostorových derivací využita informace z celé výpočetní oblasti, narozdíl od metody konečných diferencí, kdy je k výpočtu derivací využita informace pouze z bezprostředního okolí daného bodu. Při použití metody konečných diferencí je třeba speciálním způsobem ošetřit krajní body výpočetní oblasti, viz níže, k čemuž je použita extrapolace. Fourierova metoda, ve které se pracuje se spektrem, navíc v principu umožňuje jednoduše provádět filtraci dat (pro zajištění stability řešení) či modelovat složitější fyzikální mechanizmy absorpce a disperze.

3.2.1. Metoda konečných diferencí

Tato metoda je založena na aproximaci prostorových derivací (viz dodatek A.1) v rovnici (10a) použitím metody centrálních konečných diferencí:

$$\begin{aligned} \left.\frac{\partial Q}{\partial X}\right|_{X=X_i} &\approx \frac{Q_{i-2} - 8Q_{i-1} + 8Q_{i+1} - Q_{i+2}}{12h}, \\ \left.\frac{\partial^2 Q}{\partial X^2}\right|_{X=X_i} &\approx \frac{-Q_{i-2} + 16Q_{i-1} - 30Q_i + 16Q_{i+1} - Q_{i+2}}{12h^2}. \end{aligned} \tag{14}$$

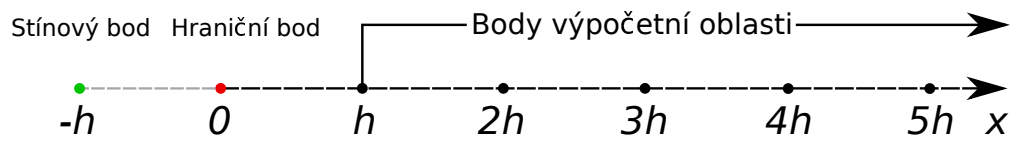
Kde Q představuje akustickou veličinu Φ nebo Ψ. Soustava rovnic (10) se tím zjednoduší na soustavu nelineárních obyčejných diferenciálních rovnic

$$\frac{\mathrm{d}\Psi_i}{\mathrm{d}T} = F(\Phi_{i-2}, \ldots, \Phi_{i+2}, \Psi_{i-2}, \ldots, \Psi_{i+2}, X_i, T), \tag{15a}$$

$$\frac{\mathrm{d}\Phi_i}{\mathrm{d}T} = \Psi_i, \tag{15b}$$

které mohou být vyřešeny některou z metod pro řešení obyčejných diferenciálních rovnic (viz 3.3).

Rovnice (15) nemohou být ovšem integrovány pro $i = 0, 1$ a $N-1, N$, pro výpočet derivací pomocí centrálních konečných diferencí v těchto bodech totiž potřebujeme informaci mimo prostor, ve kterém rovnici řešíme. Tento problém lze obejít úpravou uspořádání bodů, pro něž úlohu řešíme



Obrázek 2. Uspořádání diskretizovaného výpočetního prostoru.

(viz obrázek 2). Derivace pro krajní body X_0 a X_N jsou dané okrajovými podmínkami:

$$\left.\frac{\partial\Phi}{\partial X}\right|_{X_0} = \left.\frac{\partial\Phi}{\partial X}\right|_{X_N} = 0 \qquad \text{a} \qquad \left.\frac{\partial\Psi}{\partial X}\right|_{X_0} = \left.\frac{\partial\Psi}{\partial X}\right|_{X_N} = 0.$$

Akustické veličiny Ψ a Φ v bodě X_0 mohou být spočteny při znalosti derivace (okrajové podmínky) a přilehlých bodů podle vzorce (odvození viz dodatek A.2):

$$Q_0 \approx \frac{18Q_1 - 9Q_2 + 2Q_3}{11}. \tag{16}$$

Podobně je možné spočíst Φ a Ψ v pomocném (stínovém) bodě $X_{-1} = -h$:

$$Q_{-1} \approx \frac{6Q_1 + 8Q_2 - 3Q_3}{11}. \tag{17}$$

Znalost hodnoty v tomto bodě nám umožní vypočítat Ψ a Φ v bodě $X_1 = h$ (pro výpočet derivace potřebujeme znát dvě sousední hodnoty z obou stran). Podobně vyřešíme okrajový bod pro X_N a pomocný bod X_{N+1}

$$Q_N \approx \frac{18_{N-1} - 9_{N-2} + 2_{N-3}}{11}, \tag{18}$$

$$Q_{N+1} \approx \frac{6Q_{N-1} + 8Q_{N-2} - 3Q_{N-3}}{11}. \tag{19}$$

Při použití metody konečných diferencí je tedy soustava rovnic (15) řešena pro $i = 1, 2, \ldots, N-1$, hodnoty v bodě X_0 a X_N jsou extrapolovány (vztahy (16) a (18)). Prostorové derivace jsou aproximovány pomocí vztahů (14), pro body X_0 a X_N jsou první derivace nulové.

3.2.2. Fourierova metoda

Tato metoda využívá faktu, že derivaci funkce můžeme ve frekvenční (kmitočtové) oblasti vypočítat algebraicky pomocí jejího Fourierova obrazu. Rozvoj periodické funkce y Fourierovou řadou má tvar (viz [4])

$$y(x) = \sum_{k=-\infty}^{\infty} Y_k e^{\frac{2\pi\mathrm{i}}{L}kx}, \tag{20}$$

kde i je imaginární jednotka, L je délka jedné periody, a Y_k jsou jednotlivé Fourierovy koeficienty, které lze získat integrací:

$$Y_k = \frac{1}{L}\int_0^L y(x)e^{-\frac{2\pi\mathrm{i}}{L}kx}\mathrm{d}x.$$

Derivace funkce $y(x)$ ve vztahu (20) má podobu:

$$\frac{\mathrm{d}y(x)}{\mathrm{d}x} = \frac{\mathrm{d}}{\mathrm{d}x}\sum_{k=-\infty}^{\infty} Y_k e^{\frac{2\pi\mathrm{i}}{L}kx} = \sum_{k=-\infty}^{\infty}\frac{\mathrm{d}}{\mathrm{d}x}\left(Y_k e^{\frac{2\pi\mathrm{i}}{L}kx}\right) = \sum_{k=-\infty}^{\infty}\left(\frac{2\pi\mathrm{i}}{L}k\right)Y_k e^{\frac{2\pi\mathrm{i}}{L}kx}. \tag{21}$$

Druhou derivaci lze získat opětnou derivací vztahu (21):

$$\frac{\mathrm{d}y(x)}{\mathrm{d}x^2} = \frac{\mathrm{d}}{\mathrm{d}x}\sum_{k=-\infty}^{\infty}\left(Y_k\frac{2\pi\mathrm{i}}{L}k\right)e^{\frac{2\pi\mathrm{i}}{L}kx} = \sum_{k=-\infty}^{\infty}\frac{\mathrm{d}}{\mathrm{d}x}\left(Y_k\frac{2\pi\mathrm{i}}{L}ke^{\frac{2\pi\mathrm{i}}{L}kx}\right) =$$

$$= -\sum_{k=-\infty}^{\infty}\left(\frac{2\pi}{L}k\right)^2 Y_k e^{\frac{2\pi\mathrm{i}}{L}kx}. \tag{22}$$

Pro účely výpočtu na počítači je nutné použít diskrétní algoritmus. Spojitá funkce $y(x)$ je diskretizována v N bodech:

$$y_n = y\left(n\frac{L}{N}\right) \qquad n = 0, 1, 2, 3, ..., N-1$$

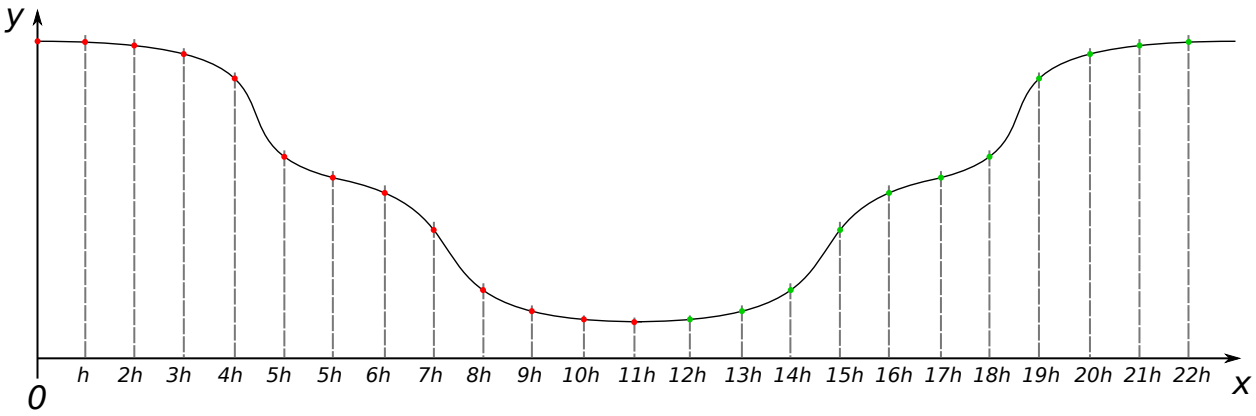
Fourierova řada (20) je pak aproximována pomocí diskrétní Fourierovy transformace (viz [4])

$$Y_k = \frac{1}{N}\sum_{n=0}^{N-1} y_n e^{-\frac{2\pi i}{N}kn}, \tag{23}$$

původní funkci získáme pomocí zpětné Fourierovy transformace:

$$y_n = \sum_{k=0}^{N-1} Y_k e^{\frac{2\pi i}{N}kn}. \tag{24}$$

V tomto případě pracujeme s průběhem, který má vždy na okrajích nulovou první derivaci (to vyplývá z okrajových podmínek (3)). Proto je nutné provést před aplikací Fourierovy transformace sudé periodické prodloužení. Výsledná Fourierova řada bude mít nenulové pouze kosínové koeficienty, je tedy zaručeno, že první derivace na okrajích bude nulová. Na obrázku 3 je znázorněno sudé



Obrázek 3. Sudé periodické prodloužení průběhu pro $N = 12$.

periodické prodloužení diskrétního průběhu. Při sudém prodloužení jsou 'zrcadlově' kopírovány jednotlivé body s vyjímkou prvního a posledního. Výsledný průběh tvoří jednu periodu periodicky prodloužené funkce. Na taková data je již možné přímo aplikovat diskrétní Fourierovu transformaci. Algebraickou úpravou ve frekvenční oblasti získáme obraz derivované funkce. Zpět do prostorové oblasti jej převedeme zpětnou Fourierovou transformací.

Při použití Fourierovy metody je tedy soustava rovnic (15) řešena pro všechny body, tedy $i = 0, 1, 2, \ldots, N$. Pro praktickou implementaci byl využit algoritmus FFT z knihovny cuFFT [5]. První derivaci lze z obrazu získat podle (21). Člen odpovídající Nyquistově frekvenci (maximální obsažené frekvenci) vynásobíme 0, čímž je zajištěn výběr 'minimálně oscilující' řady reprezentující derivaci původní funkce (viz [4]). Druhou derivaci lze z obrazu získat podle (22).

3.3. Integrace v časové oblasti

Numerické řešení počáteční úlohy pro obyčejné diferenciální rovnice a jejich soustavy je iterační proces, ve kterém odhadujeme hodnotu funkce v následujícím časovém kroku na základě předcházejících hodnot, popřípadě počátečních podmínek. Nejjednodušší metodou integrace diferenciální rovnice je Eulerova metoda (viz [6]), jejíž přístup je intuitivní

$$y_{n+1} = y_n + h_t f(t_n, y_n) + \mathcal{O}(h_t^2),$$

kde h_t je krok integrace a f je funkce která vrací derivaci odhadovaného průběhu v bodě t_n (v tomto případě odpovídá rovnicím (10)) Kvůli malé přesnosti a numerické nestabilitě se tato metoda nepoužívá.

3.3.1. Metoda Rungova-Kuttova 4. řádu

Tato metoda aproximuje hodnotu funkce v jednom kroku pomocí čtyř vyhodnocení funkce pravé strany, výsledná hodnota je jejich lineární kombinací. Jednotlivé odhady mají tvar:

$$\begin{aligned} k_1 &= hf\left(t_n, y_n\right), \\ k_2 &= hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right), \\ k_3 &= hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2\right), \\ k_4 &= hf\left(t_n + h, y_n + k_3\right). \end{aligned} \tag{25}$$

Hodnota funkce v následujícím kroku pak je:

$$y_{n+1} = y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 + \mathcal{O}(h^5), \tag{26}$$

viz např: [6], [7].

Tato metoda se narozdíl od Eulerovy metody vyznačuje dobrou stabilitou a vyšší přesností. Dá se použít i na soustavy rovnic; f pak přejde na vektorovou funkci $\boldsymbol{f}$, členy y, k_1, k_2, k_3, k_4 přejdou na vektory $\boldsymbol{y}$, $\boldsymbol{k_1}$, $\boldsymbol{k_2}$, $\boldsymbol{k_3}$, $\boldsymbol{k_4}$.

3.3.2. Metoda prediktor-korektor

První část této vícekrokové metody, prediktor, aproximuje hodnotu funkce v dalším kroku na základě několika známých předchozích hodnot. Použita byla metoda Adamsova-Bashforthova 4. řádu (viz [7])

$$\begin{aligned} y_{n+1} = y_n + \frac{h}{24}[55f(t_n, y_n) - 59f(t_n - h, y_{n-1}) + \\ + 37f(t_n - 2h, y_{n-2}) - 9f(t_n - 3h, y_{n-3})] \end{aligned} \tag{27}$$

Pro výpočet hodnoty v dalším kroku potřebujeme znát 4 hodnoty předchozí. Kvůli tomu je nutné na počátku integrace použít jinou jednokrokovou metodu, například Runge-Kutta 4. řádu (viz 3.3.1).

Pro zpřesnění výpočtu je zde ještě druhá část metody, korektor. Ta při znalosti předchozích kroků a aproximovaného kroku následujícího vytvoří nový, přesnější, odhad následujícího kroku. Použita byla metoda Adamsova-Moultonova 5. řádu:

$$\begin{aligned} y_{n+1} = y_n + \frac{h}{720}[251f(t_n + h, y_{n+1}) + 646f(t_n, y_n) - \\ - 264f(t_n - h, y_{n-1}) + 106f(t_n - 2h, y_{n-2}) - 19f(t_n - 3h, y_{n-3})] \end{aligned} \tag{28}$$

Tato metoda pro výpočet dalšího kroku potřebuje vyhodnotit funkci $f(x, y)$ pouze dvakrát, což vede k rychlejšímu výpočtu, pokud je toto vyhodnocení časově náročné. Nicméně PK45 metoda je i méně numericky stabilní. Podobně jako u RK4 je možné PK45 snadno rozšířit na řešení soustav rovnic.

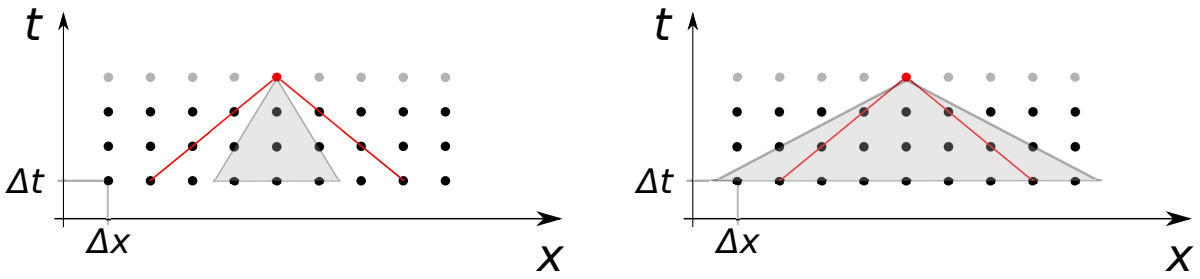
3.4. CFL podmínka

Nutným předpokladem pro stabilitu numerického řešení vlnové rovnice je splnění CFL (Courant-Friedrichs-Lewy) podmínky. Pro evoluční rovnici tvaru

$$\frac{\partial u}{\partial t} = v\frac{\partial u}{\partial x} \tag{29}$$

je CFL podmínka následující

$$\frac{v\Delta t}{\Delta x} \leq 1 \tag{30}$$



(a) Stabilní případ. **(b)** Nestabilní případ.

Obrázek 4. CFL podmínka ve výpočetním prostoru a čase pro metodu 2. řádu.

kde u je veličina závislá na čase a prostorové souřadnici a v je rychlost šíření změny dané veličiny prostorem. Nerovnice (30) vyjadřuje skutečnost, že rychlost šíření informace výpočetním prostorem musí být větší nebo rovna rychlosti šíření informace v matematickém modelu (diferenciální rovnici) (viz [7] a [6]). Situace je ilustrována na obrázcích 4a a 4b. Jednotlivé body výpočetního prostoru jsou zobrazeny jako tečky, černé jsou známé (vypočtené), šedé nevypočtené, červeně je označen bod, pro který výpočet právě probíhá. Pro výpočet derivace podle prostorové souřadnice metodou konečných diferencí druhého řádu je nutné znát z minulého časového kroku dvě okolní hodnoty. Maximální rychlost šíření informace výpočetním prostorem je tedy jedno Δx za jeden časový krok (Δt). Na obrázku je toto vyobrazeno červenými úsečkami. Rychlost šíření informace v matematickém modelu (tedy rychlost šíření vzruchu veličiny u) je označena šedým trojúhelníkem. V tomto trojúhelníku jsou tedy zahrnuty všechny body, na nichž je závislý výpočet hodnoty pro červený bod (na základě matematického modelu). Pokud tato oblast přesahuje přes hranice vyznačené červenými úsečkami, znamená to, že ve vlastním výpočtu nebude tato část brána v potaz, což může způsobit nestability řešení.

Maximální velikost časového kroku je závislá na velikosti kroku podél prostorové souřadnice. Tato maximální velikost časového kroku je pro zajištění konvergentního řešení závislá i na použitém numerickém algoritmu. Experimentálně bylo zjištěno, že pro metodu PK45 je vhodné nastavit $\Delta t = \Delta x/4$, pro metodu RK4 je vhodné nastavit $\Delta t = \Delta x/2$ (Δx je nastaveno pomocí N: $\Delta x = 1/(N-1)$ pro jednotkovou bezrozměrnou délku rezonátoru).

3.5. Shrnutí

Soustava parciálních diferenciálních rovnic (15) byla integrována tak, že prostorové derivace byly aproximovány pomocí metod uvedených v sekci 3.2. Takto vzniklá soustava obyčejných diferenciální rovnic byla integrována metodami RK4 (sekce 3.3.1) a PK45 (sekce 3.3.2). Velikost kroku v časové oblasti je závislá na velikosti prostorového kroku (sekce 3.4). V tomto případě byl numerický algoritmus (vzhledem k velikosti kroku) stabilnější s využitím metody RK4, bylo tedy možné použít větší krok.

4. Implementace

4.1. Použitá technologie

V této sekci jsou zmíněny použité technologie a knihovny.

4.1.1. CUDA

CUDA je technologie pro GPGPU od společnosti NVIDIA, zahrnuje knihovny a kompilátory pro C/C++ a Fortran. Možnou alternativou je OpenCL, které podporuje větší množství zařízení (GPU od AMD i NVIDIA, víceprocesorové systémy apod.). Vzhledem k optimalizaci na jednu platformu by CUDA měla na systémech s grafickými procesory NVIDIA vykazovat lepší výsledky.

Grafické akcelerátory

V prvních počítačích bylo vykreslování obrazu obstaráno samotným CPU, což výrazně zpomalovalo běh ostatních aplikací. Pokusy o řešení tohoto problému vedly k vývoji prvních grafických akcelerátorů, specializovaných procesorů, které obstarávaly vykreslování a snížily tím zátež CPU, který je pouze instruoval a zásoboval daty. První grafické akcelerátory měly pevně danou grafickou pipeline – shadery měly pevně danou funkci. Vývoj GPU byl úzce svázaný s vývojem grafických API jako OpenGL a DirectX, které se objevily v 90. letech. Po roce 2000 se začaly objevovat první GPU s programovatelnými shadery (vertex, geometry, fragment), jejich funkci bylo tedy možné měnit, ovšem stále byly využitelné pouze v oblasti grafických operací. První pokusy o využití potenciálu GPU pro účely obecných výpočtů tedy musely být složitě maskovány jako výpočty grafické (stínování a osvětlení 3D scén apod.).

GPGPU

GPGPU (general-purpose GPU) je grafický procesor schopný provádět výpočty nesouvisející přímo se zpracováním grafických scén. Grafické karty jsou schopné velmi silné paralelizace (architektury SIMD – Single Instruction, Multiple Data a novější SIMT – Single Instruction, Multiple Thread), jsou proto velice efektivní na problémy, kde je na velké množství dat aplikován stejný algoritmus (pro vhodné problémy řádově rychlejší oproti CPU). Paralelizace na CPU je oproti tomu typu MIMD (Multiple Instructions, Multiple Data), tedy souběžně běžící na sobě nezávislá vlákna vykonávající různé algoritmy na různá data (ačkoliv moderní CPU mají k dispozici i vektorové instrukce, které umožňují podobně jako u GPGPU aplikovat v jednom kroku stejnou operaci na větší množství dat, nicméně ne v takovém měřítku). Rychlost růstu výkonu mezi generacemi grafických karet je znatelně rychlejší než je nárůst výkonu CPU (GPGPU přidávají výkon zvýšováním množství jednoduchých, univerzálních výpočetních prvků, výpočetní jádra CPU jsou komplexnější), GPGPU je tedy perspektivní oblast. Zdroje: [8], [9]

4.1.2. HDF

HDF (Hierarchical Data Format) je formát pro ukládání velkých objemů numerických dat, spravovaný organizací HDF Group. Vzhledem k otevřenosti tohoto formátu a liberální licenci (BSD) je implementován v řadě aplikací, např. Matlab a Octave. Seznam projektů využívajících HDF je k nalezení na [10].

4.1.3. Pthreads

Pthreads (POSIX threads) je standard popisující API pro práci s vlákny. Ačkoliv je primárně určen pro UNIXové systémy, existují implementace i pro Microsoft Windows. Možnou alternativou je OpenMP, které ale neposkytuje takovou kontrolu nad funkcí programu (automaticky se stará o synchronizaci přístupu k paměti apod.), což bylo v tomto případě nežádoucí. Pthreads nabízí pro synchronizaci přístupu k proměnným semafory a mutexy. V této práci bylo využito jednoduché schéma běhu programu (dvě vlákna přistupující k jednomu poli, která si předávají informaci zda je možné do pole zapisovat/číst). Pro implementaci byly využity pouze semafory. Pod GNU/Linux je pthread součástí knihovny glibc. Zdroj: [11]

4.1.4. cuFFT

CUDA poskytuje pro výpočet FFT/IFFT knihovnu cuFFT. Pro výpočet FFT jsou použity algoritmy Cooley-Tukey a Bluestein. Knihovna je součástí zdarma distribuovaného CUDAToolkit. Zdroj: [5]

4.2. CPU implementace

Jako první byla provedena implementace na CPU. Tato verze byla použita jako referenční pro měření času, používá objektový přístup (C++). Obsahuje dvě třídy, ODEsolver a RHS. Třída ODEsolver obsahuje metody pro integraci v časové oblasti, třída RHS obsahuje metody pro numerickou diferenciaci podle prostorové souřadnice (metodou CFD) a funkci pravé strany. Zdrojový kód dále obsahuje funkce main() a time_diff(), která je využita pro výpočet času běhu programu. Tato verze je obsažena v souboru *src/cpu/wave_solver.cpp*.

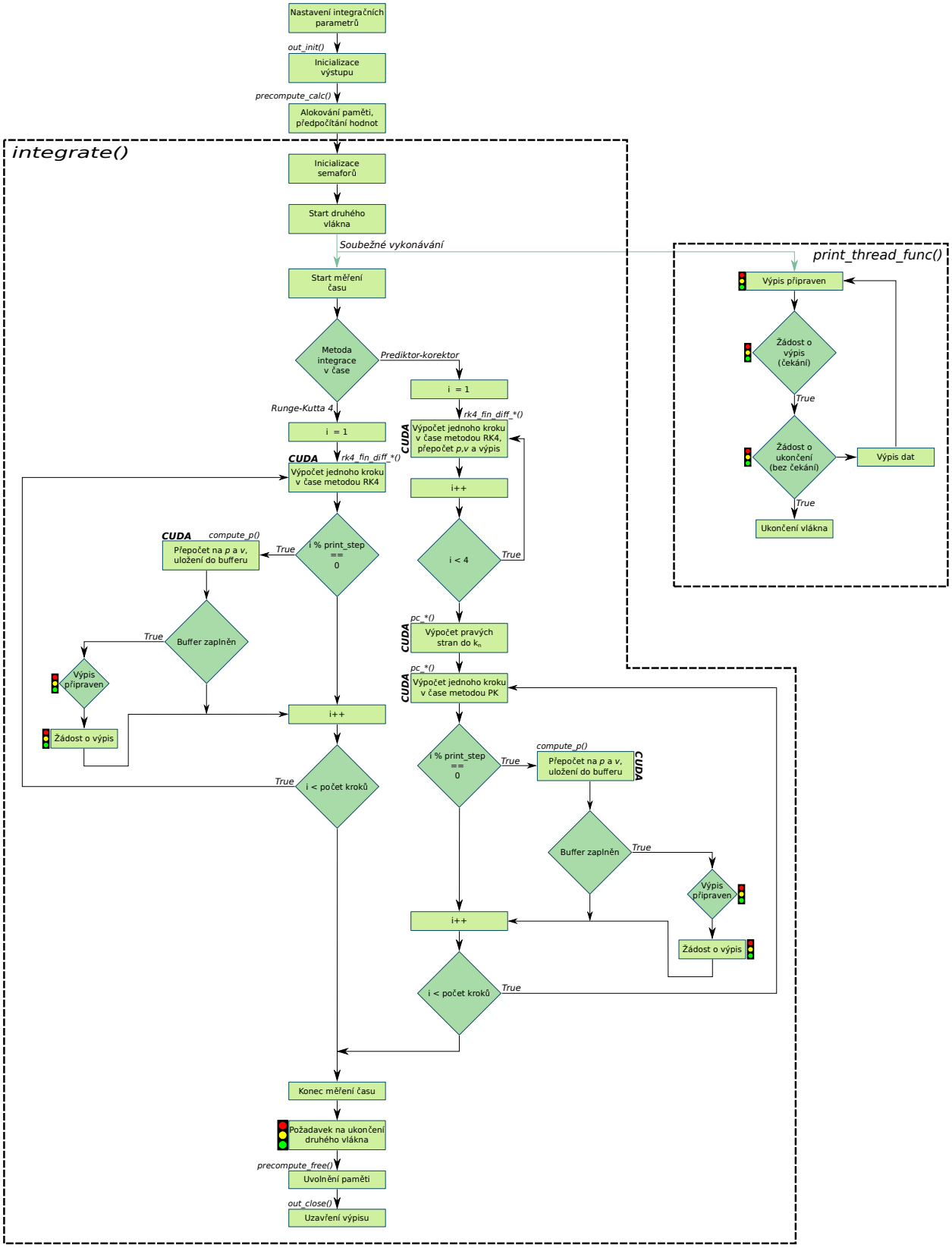
4.3. GPU implementace

GPU verze je psaná bez použití OOP. Disponuje několika odlišnostmi, zavedených zejména kvůli zrychlení běhu programu:

- Volba frekvence výpisu podél prostorové souřadnice (krok výpisu v prostoru)
- Výpis ve formátu HDF
  - rychlejší výpis oproti textovému
  - menší velikost výstupních dat (data uložena v binární podobě) při stejné nebo vyšší přesnosti
  - rychlejší načítání v Matlabu/Octave
- Buffer výpisu
  - snížení počtu volání funkcí pro výpis, zápis většího množství dat najednou
  - paměťově náročnější než verze bez bufferu (což ale s ohledem na velikost paměti moderních grafických karet není překážkou v použití programu)
- Paralelní běh programu
  - dvě vlákna, jedno instruující grafickou kartu, druhé vypisující vypočtená data
  - zrychlení běhu programu (výpis bude prováděn souběžně s výpočtem dalšího kroku)
  - snižuje optimální velikost bufferu (hlavní vlákno musí před zápisem vypočtených dat počkat, než jsou vypsána data z minulého kroku), čímž šetří paměť
- Výpis průběhu výpočtu

Při výpočtu je pro reprezentaci reálných čísel použit typ *double*. GPU jsou schopny se single–precision typem (*float*) pracovat se znatelně vyšší rychlostí (důvodem je nižší počet aritmetických jednotek pro double), nicméně tento typ nemá dostatečnou přesnost pro fyzikální simulace.

Pro větší přehlednost byla implementace na GPU rozdělena do dvou částí: verze CFD a FFT. Na obrázku 5 je znázorněn algoritmus řešiče, společný pro obě verze. Bloky, jejichž vykonávání je provedeno na grafické kartě, jsou označeny textem CUDA. U bloků, v nichž je řešena synchronizace vláken na CPU je zobrazen malý semafor. Samotný integrační krok je popsán v dalších sekcích.



Obrázek 5. Diagram GPU implementace řešiče.

Po startu programu jsou nastaveny parametry integrace:

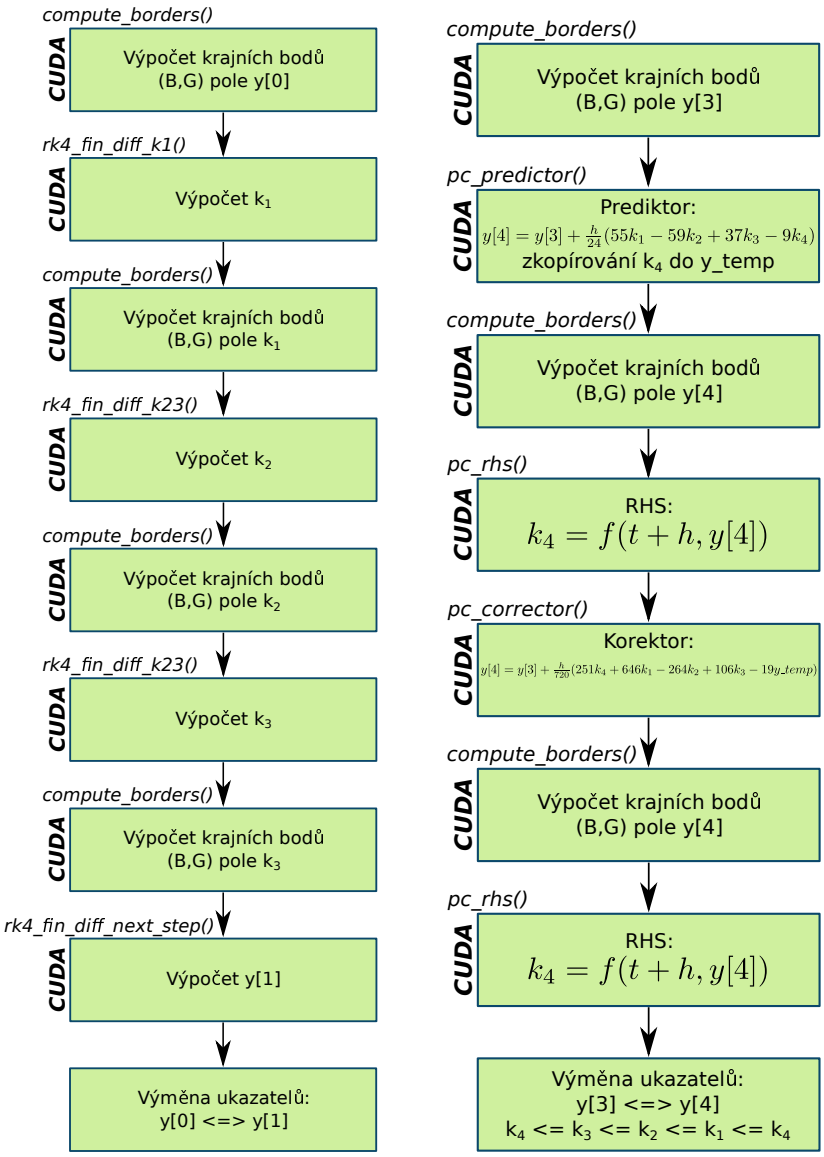
- **N**: jemnost diskretizace prostorové souřadnice
- **print_type**: typ výpisu vypočtených dat, možné hodnoty jsou TEXT (výstup do textového souboru) nebo HDF (Hierarchical Data Format, viz sekce 4.1.2)
- **integration_method**: metoda integrace v čase, možné hodnoty jsou PC pro prediktor-korektor a RK4 pro Runge-Kutta 4
- **print_step**: frekvence výpisu v čase – hodnota 2 znamená, že se vypisuje každý druhý krok
- **print_step_X**: frekvence výpisu v podél prostorové souřadnice – hodnota 2 znamená, že se bude vypisovat každý druhý diskretizovaný bod v rezonátoru
- **buff_steps**: velikost bufferu, do kterého jsou ukládány vypočtené hodnoty tlaku

Dále jsou nastaveny fyzikální parametry simulace:

- **G**: atenuační koeficient
- **GAMMA**: adiabatický exponent
- **OMEGA**: bezrozměrná frekvence
- **c0**: rychlost šíření akustických vln v daném prostředí
- **rho0**: klidová hustota daného prostředí
- **max_T**: bezrozměrný čas, do nějž je integrováno
- **A0**: amplituda bezrozměrného budícího zrychlení

4.3.1. CFD verze

Pro výpočet funkce pravé strany (derivace průběhu podle času) je nutné znát derivace podle prostorové souřadnice (viz (10)). Ty je možné spočítat metodou konečných diferencí (CFD), která je poměrně nenáročná (derivace diskretizovaného průběhu v daném bodě je lineární kombinací hodnot průběhu v sousedních bodech). Pro krajní body je však nutné znát hodnoty bodů mimo výpočetní prostor. Ty je možné aproximovat, nicméně jejich výpočet pomocí CUDA je nepříliš efektivní, protože zaměstnává minimální množství vláken, a každé z nich vykonává jiný výpočet. Výpočet na CPU by ovšem byl ještě náročnější, neboť by bylo nutné velice často předávat data mezi grafickou kartou a CPU. Proto byl výpočet krajních hodnot implementován pomocí kernelu (CUDA funkce vykonávané na GPU). Bloková schémata výpočtu jednoho kroku touto metodou jsou na obrázku 6, zdrojový kód je v souboru *src/gpu/wave_solver.cpp*.



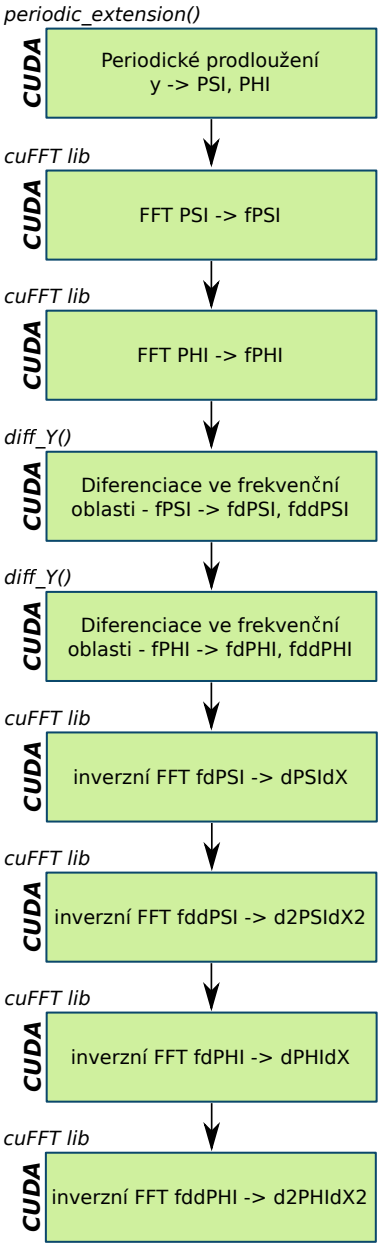
(a) Výpočet jednoho kroku metodou Runge-Kutta 4.

(b) Výpočet jednoho kroku metodou prediktor-korektor.

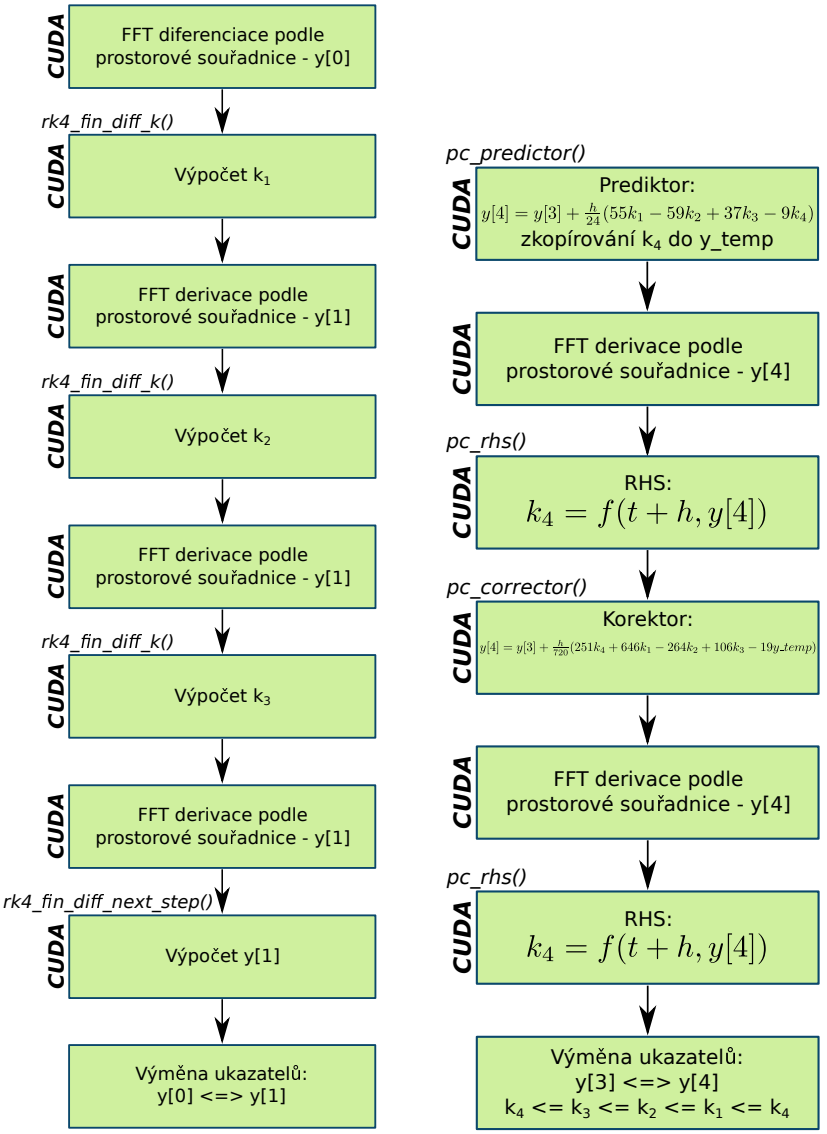
Obrázek 6. Výpočet jednoho kroku integrace pro CFD verzi.

4.3.2. FFT verze

Pro výpočet derivace podle prostorové souřadnice je též možno využít poznatku, že ve frekvenční oblasti je derivaci možno provést přenásobením obrazu signálu příslušným koeficientem (viz [4]). Toho bylo využito v FFT verzi řešiče, která průběh veličin převádí do frekvenční oblasti pomocí FFT (rychlé Fourierovy transformace), v té provede derivaci (je nutné znát první i druhou derivaci podle prostorové souřadnice), a poté provede inverzní FFT, čímž je získán časový průběh derivovaných signálů. Výhodou tohoto přístupu je, že se výpočet derivace v každém bodě používá informace z celé výpočetní oblasti, tato metoda je tedy přesnější (i proto, že nepoužívá méně přesné hodnoty krajních a stínových bodů). Je také snadno možné implementovat filtrování dat ve frekvenční oblasti a tím docílit lepší stability (v současné podobě je v programu filtrováná nejvyšší přítomná (Nyquistova) frekvence pro první derivaci). Zdrojové kódy demonstrující tento postup jsou v adresáři *testy_knihoven/fft_diff*. Byla využita implementace FFT z knihovny cuFFT. Blokové schémata výpočtu jednoho kroku touto metodou jsou na obrázku 7, zdrojový kód je v souboru *src/gpu/wave_solver_fft.cpp*.



Obrázek 7. Derivace podle prostorové souřadnice pomocí FFT.



(a) Výpočet jednoho kroku metodou Runge-Kutta 4.

(b) Výpočet jednoho kroku metodou prediktor-korektor.

Obrázek 8. Výpočet jednoho kroku integrace pro FFT verzi.

4.3.3. Kontrola přístupu k paměti

cuda-memcheck

Obě verze byly zkontrolovány programem *cuda-memcheck* na přítomnost chyb v přístupu k paměti na grafické kartě (device). Výsledky byly negativní.

Valgrind

Obě verze byly zkontrolovány programem *Valgrind* na přítomnost chyb v přístupu k paměti na systému (host). Kromě chyb hlášených kvůli použití CUDA knihoven (Valgrind není uzpůsoben pro použití s CUDA) byly výsledky negativní.

4.3.4. Historie verzí

Při psaní GPU implementace řešiče byl využit verzovací systém Git. Historie verzí se nachází v adresáři *src/gpu/.git*.

4.4. Práce s aplikací

V této sekci je uvedena základní obsluha programu. Uživatelské rozhraní je čistě textové (bez GUI) z důvodu jednodušší kompilace a snadného použití na vzdálených výpočetních serverech.

4.4.1. Modifikace parametrů

Veškeré parametry jsou specifikovány přímo v kódu, ve funkci main() (s vyjímkou N – parametrem jemnosti prostorové diskretizace, který je specifikován v záhlaví programu formou pojmenované konstanty). Výpis je možné vypnout definováním makra DO_NOT_PRINT, nejlépe v záhlaví programu. Procentuální výpis výpočtu je zapnut definováním makra PRINT_PROGRESS (ve výchozím nastavení je definován v hlavičkovém souboru wave_solver.h).

4.4.2. Kompilace

Projekt obsahuje Makefile pro automatizaci překladu. Pro kompilaci nutné se přepnout do adresáře *src/gpu/* a spustit:

```
> make
nvcc -o wave_solver -arch=sm_20 -lhdf5 wave_solver.cu print_lib.cpp
nvcc -o wave_solver_fft -arch=sm_20 -lhdf5 -lcufft wave_solver_fft.cu \
print_lib.cpp
```

Pro rychlost běhu programu může být výhodné vytvořit symbolický odkaz na soubory na tmpfs (filesystem na RAM paměti, ve většině distribucí je /tmp v podobě tmpfs):

```
> mkdir /tmp/ws && cd /tmp/ws
> ln -s $PROJECT_ROOT/src/gpu/* ./
> make
```

Kde $PROJECT_ROOT je adresář, kde je uložena tato práce. Je nutné mít dostatečné množství volné paměti na /tmp. Práce s velkými daty je na tmpfs znatelně rychlejší, a v případě SSD disků šetří i životnost.

4.4.3. Spuštění

Po kompilaci je možné program spustit pomocí příkazu:

```
> ./wave_solver
```

Pro FFT verzi pak:

```
> ./wave_solver_fft
```

Program oznámí přibližně množství alokované paměti a velikost výstupních dat:

```
> ./wave_solver
Pouzita metoda: Runge-Kutta 4
Pocet vypsanych kroku: 12880
Velikost bufferu: 1000 kroku (15 MB)
Odhad velikosti na disku: 2 x 25.2 MB = 50.4 MB
100 %
Cas integrace: 2.517291838 s
```

Výstup je v souborech *T.hd5* (čas), *p.hd5* (pro tlak) a *v.hd5* (pro rychlost).

4.4.4. Práce s daty

V aplikaci Octave je možné data nahrát pomocí příkazu:

```
> p = load('p.hd5'); v = load('v.hd5'); T = load('T.hd5');
```

V aplikaci Matlab je možné data nahrát pomocí příkazu:

```
> p = h5read('p.hd5','/p'); v = h5read('v.hd5','/v'); \
T = h5read('T.hd5','/time');
```

5. Měření výkonu

Tato sekce se zabývá měřením času běhu různých metod CPU a GPU verzí programu. Časy jsou porovnány a je zkoumán vliv parametrů integrace na rychlost a paměťovou náročnost aplikace.

5.1. HW sestava

V této sekci jsou uvedeny systémové parametry zařízení, na nichž byla měření provedena.

5.1.1. Zařízení A

Zařízení A je školní server (herodes) umístěný v budově ČVUT FEL.

- OS: OpenSUSE 13.2 'Harlequin' 64-bit
- Motherboard: MSI P67A-GD55 (MS-7681)
- Procesor: Intel i7-2600
  - Počet jader: 4
  - Nominální frekvence: 3,4 GHz
- RAM: 16 GB
- GPU: 2× GeForce GTX 580 (GPU0: 3071 MB, GPU1: 1535 MB)
  - Frekvence GPU: 772 MHz [12]
  - Počet CUDA jader: 512 [12]
  - Propustnost paměti: 192,4 Gb/s [12]
  - Compute capability: 2.0 [13]

5.1.2. Zařízení B

Zařízení B je běžný desktop.

- OS: Archlinux 64-bit
- Motherboard: Asus M5A99FX PRO R2.0
- Procesor: AMD FX-6300
  - Počet jader: 6
  - Nominální frekvence: 3,5 GHz
- RAM: 16 GB
- GPU: GeForce GTX 750 (2048 MB)
  - Frekvence GPU: 1020 MHz [14]
  - Počet CUDA jader: 512 [14]
  - Propustnost paměti: 80 Gb/s [14]
  - Compute capability: 5.0 [13]

5.2. Metoda měření

5.2.1. Automatizace

Měření bylo automatizováno pomocí skriptu v programovacím jazyce Python. Skript umožňuje modifikaci parametrů integrace ve zdrojovém kódu, jeho kompilaci a spuštění. Výsledky měření pak uloží do souboru 'out'. Základní verze zdrojového kódu se nachází v *mereni/measure.py*, jednotlivé jeho mutace a výsledky měření jsou v podsložkách adresáře *mereni/*. Skript očekává veškeré zdrojové kódy ve složce, kde se sám nachází. Aby se omezil vliv nerovnoměrného zatížení systému (vliv běhu jiných aplikací apod.), je každé měření opakováno a do výstupního souboru je zapsán průměrný čas. Počet opakování je možné nastavit.

5.2.2. Metoda měření času

Po dokončení výpočtu vypisuje program čas běhu, měřený pomocí POSIXové funkce clock_gettime(). Tato funkce umožňuje měření strojového času (na celý proces či jedno vlákno) nebo reálného času běhu aplikace na základě použitého parametru. Pro měření v této aplikaci byl použit parametr CLOCK_MONOTONIC, který měří reálný běh času. Pokud tento parametr není k dispozici, je použit CLOCK_REALTIME, který má oproti MONOTONIC nevýhodu v tom, že je ovlivněn případnou změnou nastavení systémového času v průběhu měření. Rozhodnutí, který parametr je použit, je učiněno v průběhu vykonávání programu.

5.2.3. Aproximace funkčních závislostí

Aproximace funkce doby integrace v závislosti na různých integračních parametrech umožní odhad času nutného pro běh programu. To je výhodné v situaci, kdy chceme maximalizovat nějaký parametr (např. N, T_{max}) při zachování 'rozumných' časů. Byla provedena aproximace obecnou mocninou ($y = ax^b$). Ta je provedena metodou polynomiální regrese s jistými obměnami (viz níže).

Metoda polynomiální regrese

Neznámá funkce $f(x)$ je aproximována polynomem n–tého stupně:

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_nx^n + \epsilon(x) \tag{31}$$

Kde $\epsilon(x)$ je rozdíl aproximované a skutečné hodnoty funkce v daném bodě. Při znalosti hodnoty funkce $f(x_i)$ v k bodech x_i chceme získat koeficienty $a_0, a_1, ..., a_n$ pro zvolené n. Tuto situaci můžeme maticově zapsat jako:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 & ... & x_1^n \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 & ... & x_2^n \\ 1 & x_3 & x_3^2 & x_3^3 & x_3^4 & ... & x_3^n \\ \vdots & & & & & & \\ 1 & x_k & x_k^2 & x_k^3 & x_k^4 & ... & x_k^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \\ \vdots \\ f(x_k) \end{bmatrix} + \boldsymbol{\epsilon}(x) \tag{32}$$

Což po zanedbání $\boldsymbol{\epsilon}(x)$ odpovídá maticové rovnici

$$\boldsymbol{X}\boldsymbol{a} = \boldsymbol{y}, \tag{33}$$

jejímž řešením pro $\boldsymbol{a}$ získáme:

$$\boldsymbol{a} = (\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{34}$$

Čímž obdržíme hledané koeficienty. Zdroj: [15]

Aproximace obecnou mocninou

Aproximace na funkci $y = ax^b$ je provedena metodou polynomiální regrese, nejdříve je ale nutné zlogaritmovat naměřené hodnoty:

$$f(x) = ax^b \tag{35}$$

$$\ln(f(x)) = \ln(ax^b) = \ln(a) + b \cdot \ln(x) \tag{36}$$

Tím jsme problém hledání parametrů a a b zjednodušili na hledání koeficientů polynomu prvního stupně, tedy lineární funkce. Na tento problém je možné aplikovat polynomiální regresi. Upravíme rovnici (32):

$$\begin{bmatrix} 1 & \ln(x_1) \\ 1 & \ln(x_2) \\ 1 & \ln(x_3) \\ \vdots & \\ 1 & \ln(x_k) \end{bmatrix} \begin{bmatrix} \ln(a) \\ b \end{bmatrix} = \begin{bmatrix} \ln(f(x_1)) \\ \ln(f(x_2)) \\ \ln(f(x_3)) \\ \ln(f(x_4)) \\ \vdots \\ \ln(f(x_k)) \end{bmatrix} \tag{37}$$

Řešením této rovnice (dle (34)) je možné získat přímo parametr b. Parametr a je nutné nejdříve odlogaritmovat:

$$a = \mathrm{e}^{\ln(a)} \tag{38}$$

Čímž jsme získali všechny parametry nutné pro aproximaci obecnou mocninou.

5.3. Porovnání CPU a GPU verzí metody CFD

V porovnání šlo o změření rychlosti samotného výpočtu, bylo proto vypnuto vypisování, které by stejnoměrně zatěžovalo všechny verze. Tím pádem odpadl vliv parametrů print_step_X (krok výpisu podél prostorové souřadnice), buff_step (velikost bufferu pro výpis) a print_type (výpis do textového formátu nebo HDF). Parametr print_step_T byl nastaven na 1, aby docházelo k přepočtu na bezrozměrný tlak a rychlost v každém kroku. Hodnoty N, pro něž byly zjišteny časy běhu, byly voleny jako celočíselné násobky množství CUDA jader na daném zařízení, tedy 512 (k N je nutno ještě přičíst 2 kvůli krajním bodům). Toto zarovnání by mělo mít za následek efektivnější využití výpočetního potenciálu GPU. Testováno tedy bylo pro N:

$$N = 512k + 2 \qquad k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\} \tag{39}$$

Počet vláken v jednom bloku byl nastaven na 32, počet bloků byl dopočítán jako $(N - 2)/32$. Dva krajní body nejsou zahrnuty, ty jsou přepočítány v jiném kernelu (volaném s 4 bloky po 2 vláknech). Volání kernelů s tak nízkým počtem vláken není příliš efektivní (výpočet zatěžuje velmi malou část GPU), nicméně přenos dat a výpočet na CPU by byl velice pomalý, byla proto zvolena tato metoda.

Čas integrace T_{max} byl nastaven na 10π, poměr $\Delta X/\,\Delta T$ byl nastaven na 6.5. Tato hodnota byla zvolena tak, aby byl výpočet numericky stabilní a konvergoval v celém intervalu měřených N (u obou metod). Pro nízké hodnoty N toto však není nejefektivnější přístup, při řešení konkrétního problému je vhodné experimentálně nalézt nejmenší možnou hodnotu poměru $\Delta X/\,\Delta T$ tak, aby výpočet byl ještě stabilní (čas výpočtu je přímo úměrný velikosti $\Delta X/\,\Delta T$).

Veškeré soubory k testu (zdrojové kódy, automatizující skript a výsledky) jsou v adresáři *mereni/4.3*.

5.3.1. Zařízení A

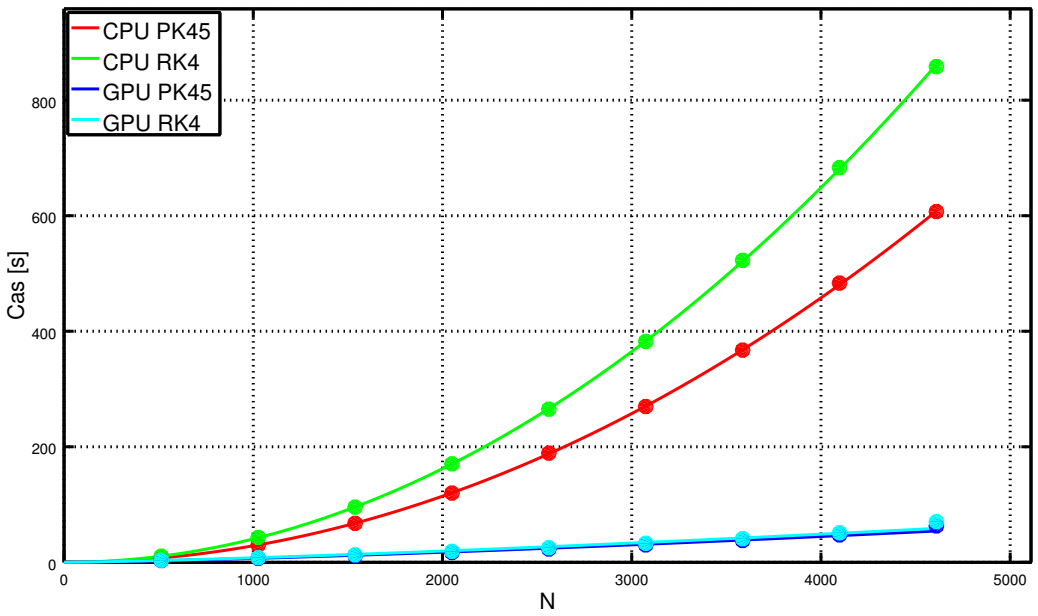
V tabulce 1 jsou naměřené časy pro zařízení A (tabulka byla z prostorových důvodů zkrácena). Z těch byly aproximovány funkce závislosti času běhu na parametru N (viz tabulka 2). Naměřené časy proložené aproximační křivkou jsou na obrázku 9.

N	514	1026	1538	2050	2562	3074	3586	4098
CPU PK45	7,6 s	30,0 s	67,5 s	119,8 s	189,0 s	269,7 s	367,4 s	483,4 s
CPU RK4	10,7 s	43,1 s	95,5 s	170,5 s	265,1 s	382,4 s	522,8 s	683,2 s
GPU PK45	3,0 s	6,8 s	11,4 s	17,0 s	23,0 s	30,2 s	37,7 s	47,4 s
GPU RK4	3,6 s	7,9 s	12,9 s	19,1 s	25,2 s	33,0 s	40,9 s	51,1 s

Tabulka 1. Časy běhů CFD verze pro metody RK4 a PK45 na zařízení A.

CPU PK45	$t(N) = 29 \cdot N^{1,999}\,\mu s$
CPU RK4	$t(N) = 41 \cdot N^{2,000}\,\mu s$
GPU PK45	$t(N) = 570 \cdot N^{1,359}\,\mu s$
GPU RK4	$t(N) = 872 \cdot N^{1,318}\,\mu s$

Tabulka 2. Aproximované funkce závislosti času běhu na N.



Obrázek 9. Časy běhů CFD verze pro metody RK4 a PK45 na zařízení A.

5.3.2. Zařízení B

V tabulce 3 jsou naměřené časy pro zařízení B (tabulka byla z prostorových důvodů zkrácena). Z těch byly aproximovány funkce závislosti času běhu na parametru N (viz tabulka 4). Naměřené časy proložené aproximační křivkou jsou na obrázku 10.

N	514	1026	1538	2050	2562	3074	3586	4098
CPU PK45	8,9 s	35,3 s	79,7 s	142,7 s	234,9 s	317,8 s	435,5 s	570,3 s
CPU RK4	12,5 s	50,1 s	111,8 s	201,2 s	310,2 s	459,1 s	609,3 s	809,3 s
GPU PK45	5,6 s	12,1 s	19,6 s	28,7 s	39,0 s	51,1 s	64,2 s	79,2 s
GPU RK4	6,4 s	14,1 s	23,2 s	34,5 s	47,5 s	63,0 s	80,3 s	100,0 s

Tabulka 3. Časy běhů CFD verze pro metody RK4 a PK45 na zařízení B.

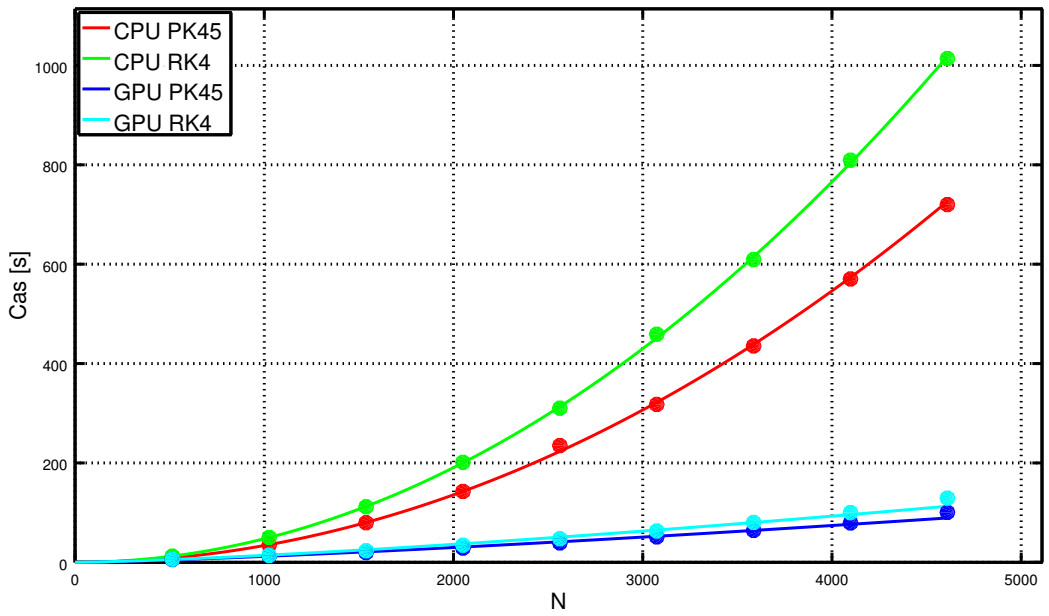
CPU PK45	$t(N) = 32 \cdot N^{2,007}\,\mu s$
CPU RK4	$t(N) = 46 \cdot N^{2,006}\,\mu s$
GPU PK45	$t(N) = 1388 \cdot N^{1,312}\,\mu s$
GPU RK4	$t(N) = 1155 \cdot N^{1,362}\,\mu s$

Tabulka 4. Aproximované funkce závislosti času běhu na N.

5.3.3. Závěr

Doba běhu CPU implementace vykazovala ve všech případech téměř přesnou kvadratickou závislost na velikosti N. GPU implementace oproti tomu vykazuje závislost na N s exponentem v intervalu $(1{,}3 - 1{,}4)$. Masivní paralelizace tohoto problému na GPGPU tedy přinesla nezanedbatelné zlepšení časové závislosti.

Ačkoliv grafické karty na obou zařízení obsahují stejný počet CUDA jader, je GPU na zařízení A znatelně rychlejší. To je způsobeno větším počtem aritmetických jednotek pro double precision a vyšší propustností paměti (šířka paměťové sběrnice u GTX 580 je 384 bitů oproti 128 bitům na GTX 750).



Obrázek 10. Časy běhů CFD verze pro metody RK4 a PK45 na zařízení B.

5.4. Porovnání CFD a FFT verze na GPU

Stejně jako u porovnání 5.3 bylo i v tomto případě vypnuto vypisování, což zamezilo vlivu parametrů print_step_X (krok výpisu podél prostorové souřadnice), buff_step (velikost bufferu pro výpis) a print_type (výpis do textového formátu nebo HDF). Parametr print_step_T byl nastaven na 1, aby docházelo k přepočtu na bezrozměrný tlak a rychlost v každém kroku. Čas integrace T_{max} byl nastaven na 10π.

Veškeré soubory k testu (zdrojové kódy, automatizující skript a výsledky) jsou v adresáři *mereni/4.4*.

5.4.1. Časová závislost na N verze CFD

Testování pouze na GPU umožnilo ověřit časové závislosti pro vyšší N:

$$N = 512k + 2 \qquad k \in \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19\} \tag{40}$$

Počet vláken v jednom bloku byl nastaven na 32, počet bloků byl dopočítán jako $(N-2)/32$. Poměr $\Delta X/\Delta T$ byl nastaven na hodnotu 12,5. Tato hodnota byla zvolena tak, aby byly všechny metody stabilní a konvergovaly v celém rozsahu N.

Zařízení A

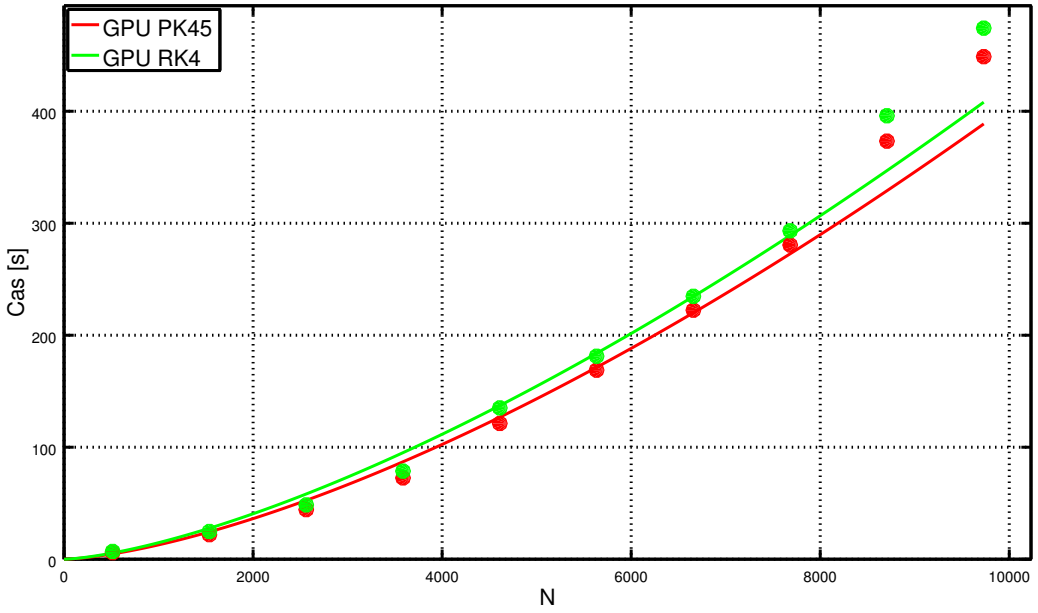
V tabulce 5 jsou naměřené časy pro zařízení A (tabulka byla z prostorových důvodů zkrácena). Z těch byly aproximovány funkce závislosti času běhu na parametru N (viz tabulka 6). Naměřené časy proložené aproximační křivkou jsou na obrázku 11. Na obrázku je patrné, že poslední dvě naměřené hodnoty nesouhlasí s odhadem – čas běhu pro tyto N je znatelně vyšší, než bychom očekávali. To je pravděpodobně způsobeno tím, že od určitého N nedochází k maskování latence přístupu k paměti. Při výpočtech na GPGPU je nejpomalejším článkem přístup k paměti, proto pokud warp (nejmenší organizační jednotka vláken technologie CUDA) požádá o přístup k paměti, jsou během jeho vyřizování tato vlákna (která pouze čekají na data z paměti) zastoupena jinými vlákny, která mohou pokračovat v práci. Tímto způsobem je docíleno vyšší efektivity využití GPU. Od určité úrovně dojde k „saturaci“, tedy běh dalších vláken již nekompenzuje latenci paměti. To způsobí, že závislost času běhu na N by měla být opět úměrná N^2.

N	514	1538	2562	3586	4610	5634	6658	7682	8706
GPU PK45	5,9 s	21,9 s	44,3 s	72,6 s	121,2 s	168,8 s	222,4 s	280,6 s	373,3 s
GPU RK4	7,0 s	24,8 s	48,5 s	78,7 s	135,1 s	181,3 s	234,8 s	293,2 s	396,0 s

Tabulka 5. Časy běhů CFD verze pro metody RK4 a PK45 na zařízení A.

$$\begin{aligned} &\text{CPU PK45} \quad t(N) = 407 \cdot N^{1,499}\,\mu s \\ &\text{GPU RK4} \quad t(N) = 620 \cdot N^{1,459}\,\mu s \end{aligned}$$

Tabulka 6. Aproximované funkce závislosti času běhu na N zařízení A.



Obrázek 11. Časy běhů CFD verze pro metody RK4 a PK45 na zařízení A.

Zařízení B

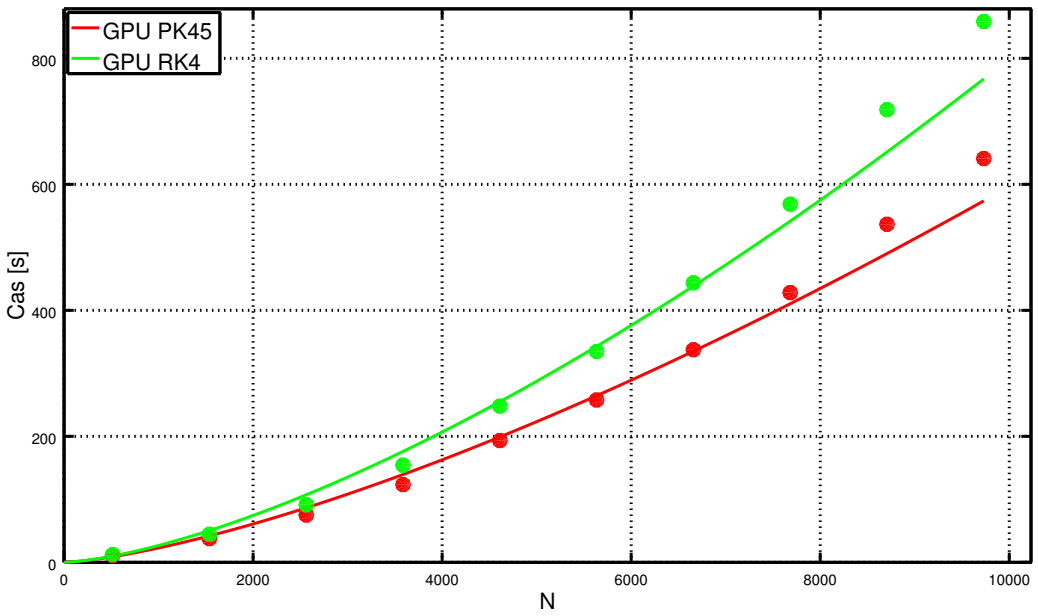
V tabulce 7 jsou naměřené časy pro zařízení B (tabulka byla z prostorových důvodů zkrácena). Z těch byly aproximovány funkce závislosti času běhu na parametru N (viz tabulka 8). Naměřené časy proložené aproximační křivkou jsou na obrázku 12. U této křivky také pozorujeme efekt jako v minulém případě – pro vysoká N se objeví skok neodpovídající aproximovaným hodnotám. V tomto případě efekt není tak patrný, je možné, že je to způsobeno výrazně nižší propustností paměti grafické karty na tomto zařízení. Plná saturace (kdy už nedochází k maskování latence paměti) by se tedy projevila až u vyšších N.

N	514	1538	2562	3586	4610	5634	6658	7682	8706
GPU PK45	10,6 s	37,6 s	75,0 s	123,3 s	193,3 s	257,7 s	337,4 s	428,1 s	536,6 s
GPU RK4	12,2 s	44,6 s	91,3 s	154,3 s	248,0 s	334,7 s	443,7 s	568,5 s	718,5 s

Tabulka 7. Časy běhů CFD verze pro metody RK4 a PK45 na zařízení B.

$$\begin{aligned} &\text{GPU PK45} \quad t(N) = 1275 \cdot N^{1,418}\,\mu s \\ &\text{GPU RK4} \quad t(N) = 1007 \cdot N^{1,475}\,\mu s \end{aligned}$$

Tabulka 8. Aproximované funkce závislosti času běhu na N zařízení B.



Obrázek 12. Časy běhů CFD verze pro metody RK4 a PK45 na zařízení B.

5.4.2. Numerická stabilita CFD verze v závislosti na hodnotě $\Delta X/\,\Delta T$

Poměr $\Delta X/\,\Delta T$ je nutno nastavit tak, aby byl výpočet numericky stabilní (vyšší hodnota zlepší stabilitu). Příliš vysoká hodnota ale zbytečně zvyšuje počet iterací výpočtu a tím pádem jej zpomaluje. Protože nejmenší stabilní hodnota tohoto poměru závisí na mnoha faktorech (mimo jiné např. na CFL podmínce, viz 3.4), je v praxi vhodnější hodnotu nastavit zkusmo a ověřit experimenty.

V této sekci jsou uvedeny výsledky měření nejmenší stabilní hodnoty poměru $\Delta X/\,\Delta T$ pro pevně dané fyzikální a integrační parametry (uvedené níže) a různá N:

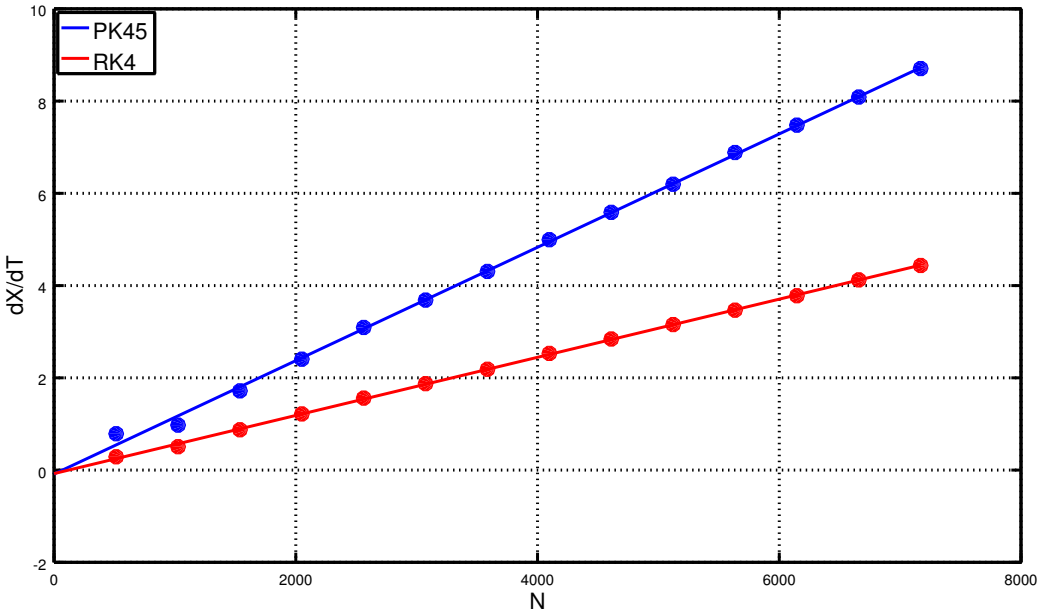
$$N = 512k + 2 \qquad k \in \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19\} \tag{41}$$

Čas integrace T_{max} byl nastaven na 10π. Parametr print_step_X byl nastaven na 4, print_step_T na 5. Oba tyto parametry na stabilitu nemají vliv, pouze ovlivňují, jak často (a kolik dat) je vypisováno. Nastavení fyzikálních parametrů bylo následující:

- $G = 0{,}01$
- $\gamma = 1{,}4$
- $\omega = 1{,}0$
- $c_0 = 345{,}0$
- $\rho_0 = 1{,}2$
- $A_0 = 0{,}0005$

Testování bylo částečně automatizováno, skript upravovil hodnoty poměru $\Delta X/\,\Delta T$, zkompiloval a spustil program, poté vyzval uživatele k intervenci. Ten vypočtená data načetl do externí aplikace (např. Octave nebo Matlab) a ohodnotil, zda jsou výsledky správné (a výpočet byl tedy numericky stabilní), či ne. Na základě vstupu od uživatele skript dále upravuje hodnotu poměru, dokud se nedostane na hranici stability. Hledání je provedeno dělením intervalů – je určen interval, kde se přesná hodnota nachází, poté je rozdělen na poloviny, a je určeno ve které polovině se nachází hledaná hodnota (binární hledání). Tento postup je iterativně opakován, dokud není interval menší než stanovený limit (v tomto případě 0,0625). Skript poté nalezenou hodnotu (horní limit intervalu) vypíše, a pokračuje k dalšímu N.

Z důvodu nutnosti časté uživatelské interakce (a vykreslování velkého množství dat) byl test proveden pouze na lokálním zařízení (B). V tabulce 9 jsou naměřené časy pro zařízení B (tabulka byla z prostorových důvodů zkrácena). Z těch byly aproximovány funkce závislosti $\Delta X/\,\Delta T$ na parametru N (viz tabulka 10). Aproximováno bylo metodou polynomiální regrese na mnohočlen



Obrázek 13. Grafy nalezených minimálních hodnot pro stabilní a konvergentní výpočet $\Delta X/\Delta T$ pro zařízení B.

prvního stupně (lineární funkci). Naměřené hodnoty proložené aproximační křivkou jsou na obrázku 13. Veškeré soubory k testu (zdrojové kódy, automatizující skript a výsledky) jsou v adresáři *mereni/dt_dx*.

N	514	1538	2562	3586	4610	5634	6658
GPU PK45	0,79	1,72	3,09	4,31	5,59	6,89	8,09
GPU RK4	0,29	0,87	1,56	2,19	2,84	3,47	4,12

Tabulka 9. Nalezené minimální hodnoty pro stabilní a konvergentní výpočet $\Delta X/\Delta T$ pro metody RK4 a PK45 na zařízení B.

GPU PK45	$\frac{\Delta x}{\Delta t}(N) = 12{,}291 \cdot 10^{-4} N - 8{,}575 \cdot 10^{-2}$	$N \geq 70$
GPU RK4	$\frac{\Delta x}{\Delta t}(N) = 6{,}303 \cdot 10^{-4} N - 7{,}623 \cdot 10^{-2}$	$N \geq 121$

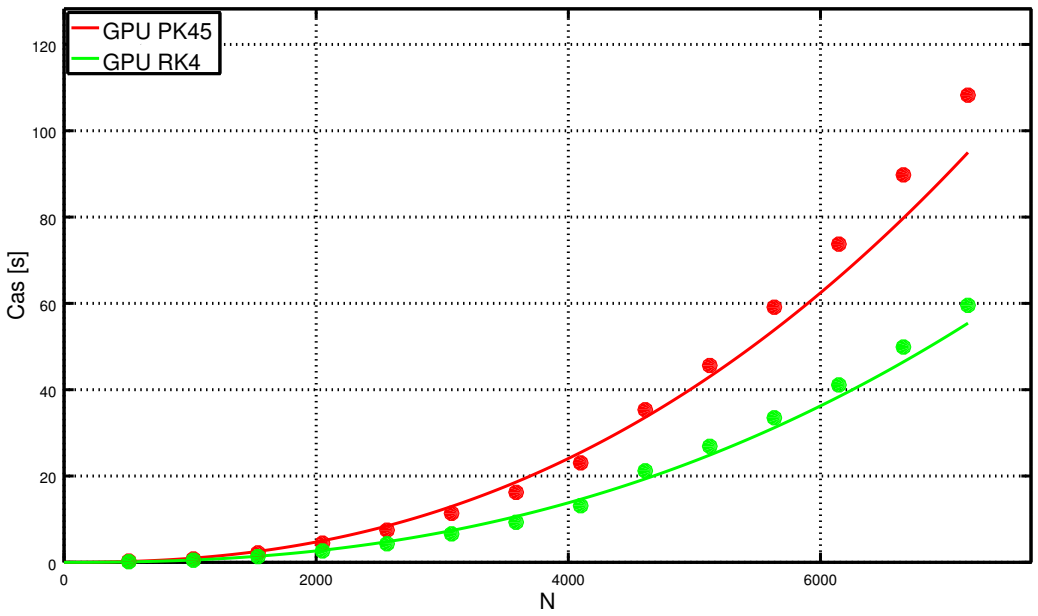
Tabulka 10. Aproximované funkce závislosti $\Delta X/\Delta T$ na N pro zařízení B.

5.4.3. Porovnání metod PK45 a RK4 verze CFD s minimalizovaným $\Delta X/\Delta T$

Znalost minimálních stabilních poměrů $\Delta X/\Delta T$ umožnila provést porovnání metod PK45 a RK4 z hlediska rychlosti při zohlednění stability každé metody. Metoda prediktor-korektor je aritmeticky méně náročná, metoda Rungova–Kuttova 4. řádu je oproti tomu numericky stabilnější (viz [6]), což umožňuje použít menší $\Delta X/\Delta T$ a tím běh zrychlit. Test byl proveden na obou zařízeních, pro stejná N jako v případě předcházejícího testu. Hodnoty $\Delta X/\Delta T$ byly brány z tabulky 9. Nastavení parametrů bylo stejné jako v předchozím testu (vypisování však bylo vypnuto). Veškeré soubory k testu (zdrojové kódy, automatizující skript a výsledky) jsou v adresáři *mereni/dt_dx/pc_rk_compare*.

Zařízení A

Naměřené výsledky jsou v tabulce 11. Z těch byly aproximovány funkce v tabulce 12. Časová závislost na N odpovídá $N^{2,4}$, což je způsobeno tím, že s růstem N ($t(N) \approx N^{1,4}$) roste i $\Delta X/\Delta T$ ($t(\Delta X/\Delta T) \approx N^1$) . Výsledná časová závislost vznikne složením těchto dvou závislostí. Graf



Obrázek 14. Naměřené časy s minimalizovaným $\Delta X/\,\Delta T$ na zařízení A.

s vynesenými hodnotami a aproximačními křivkami je na obrázku 14. Pro srovnání s rychlostmi běhu výpočtu s neoptimalizovaným $\Delta X/\,\Delta T$ je možno nahlédnout do tabulky 5. Je patrné, že správné nastavení $\Delta X/\,\Delta T$ má významný vliv na rychlost běhu aplikace.

N	514	1538	2562	3586	4610	5634	6658
GPU PK45	0,3 s	2,2 s	7,5 s	16,2 s	35,3 s	59,1 s	89,8 s
GPU RK4	0,1 s	1,3 s	4,3 s	9,3 s	21,2 s	33,5 s	49,9 s

Tabulka 11. Naměřené časy s minimalizovaným $\Delta X/\,\Delta T$ na zařízení A.

GPU PK45	$t(N) = 0.08N^{2,35}\,\mu s$
GPU RK4	$t(N) = 0.04N^{2,38}\,\mu s$

Tabulka 12. Aproximované funkce závislosti času na N pro zařízení A.

Zařízení B

Naměřené výsledky jsou v tabulce 13. Z těch byly aproximovány funkce v tabulce 14. Graf s vynesenými hodnotami a aproximačními křivkami je na obrázku 15. Pro srovnání s rychlostmi běhu výpočtu s neoptimalizovaným $\Delta X/\,\Delta T$ je možno nahlédnout do tabulky 7. Je patrné, že správné nastavení $\Delta X/\,\Delta T$ má významný vliv na rychlost běhu aplikace.

N	514	1538	2562	3586	4610	5634	6658
GPU PK45	0,6 s	4,2 s	14,7 s	33,7 s	69,4 s	112,7 s	172,8 s
GPU RK4	0,2 s	2,6 s	9,5 s	22,4 s	47,7 s	78,3 s	123,4 s

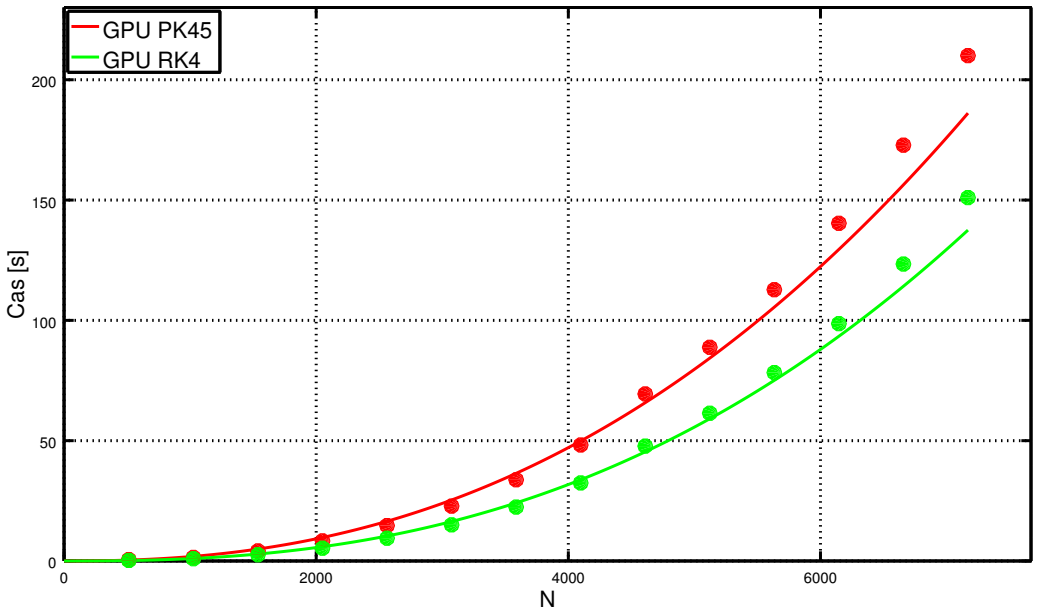
Tabulka 13. Naměřené časy s minimalizovaným $\Delta X/\,\Delta T$ na zařízení B.

Závěr

Pro konkrétní fyzikální a integrační parametry výpočtu je metoda RK4 rychlejší, ačkoliv je aritmeticky náročnější. Je to způsobeno její vyšší stabilitou, což umožňuje nastavit výhodnější (menší)

GPU PK45	$t(N) = 0.16N^{2,35}\,s$
GPU RK4	$t(N) = 0.03N^{2,51}\,s$

Tabulka 14. Aproximované funkce závislosti času na N pro zařízení B.



Obrázek 15. Naměřené časy s minimalizovaným $\Delta X/\,\Delta T$ na zařízení B.

poměr kroku prostorového a časového ($\Delta X/\,\Delta T$).

5.4.4. Časová závislost na N verze FFT

Testováno bylo pro N:

$$N = 512k \qquad k \in \{1, 2, 3, 4, 5\} \tag{42}$$

Počet vláken v jednom bloku byl nastaven na 32, počet bloků byl dopočítán jako $(N+31)/32$. Poměr $\Delta X/\,\Delta T$ byl nastaven na hodnotu 5,5. Tato hodnota byla zvolena tak, aby byly všechny metody stabilní a konvergovaly v celém rozsahu N. Funkce FFT z knihovny cuFFT pracuje nejefektivněji s daty, jejichž velikost je mocninou 2 (viz [5]). Závislost času běhu na N byla proto aproximována pouze z N, které jsou sudými násobky 512.

Zařízení A

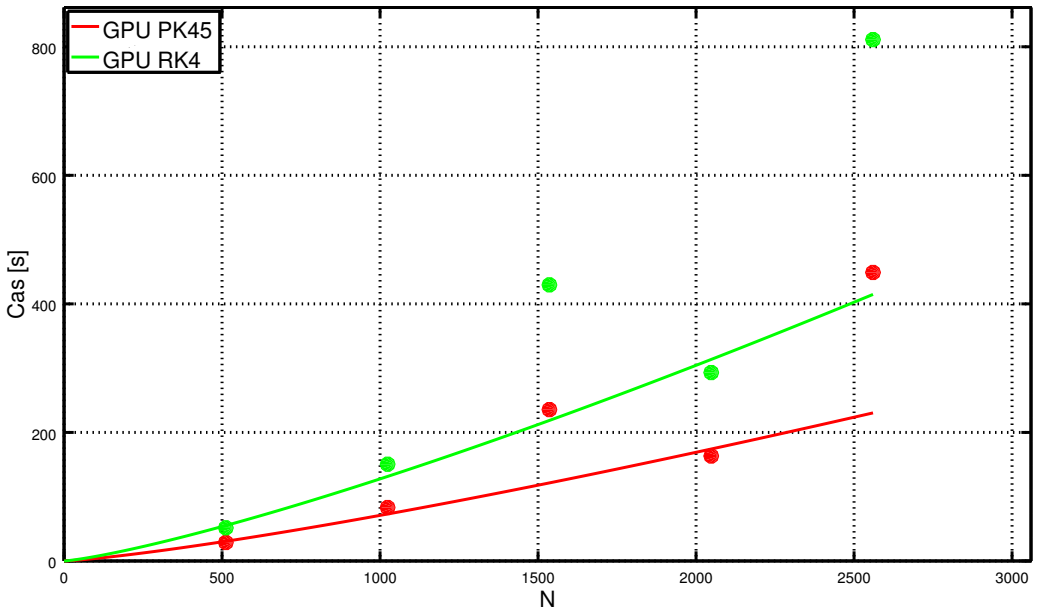
V tabulce 15 jsou naměřené časy pro zařízení A. Z těch byly aproximovány funkce závislosti času běhu na parametru N (viz tabulka 16). Naměřené časy proložené aproximační křivkou jsou na obrázku 16. Pro liché násobky (s vyjímkou 1) 512 je patrné, že výpočet bežel výrazně pomaleji než pro sudé. To je způsobeno knihovnou cuFFT, která pracuje efektivněji s mocninami 2.

N	512	1024	1536	2048	2560
GPU PK45	28,8 s	83,2 s	235,5 s	163,3 s	448,9 s
GPU RK4	51,6 s	150,6 s	429,6 s	293,1 s	811,1 s

Tabulka 15. Časy běhů FFT verze pro metody RK4 a PK45 na zařízení A.

GPU PK45	$t(N) = 0{,}012 \cdot N^{1{,}25}\, s$
GPU RK4	$t(N) = 0{,}022 \cdot N^{1{,}25}\, s$

Tabulka 16. Aproximované funkce závislosti času běhu na N zařízení A.



Obrázek 16. Časy běhů FFT verze pro metody RK4 a PK45 na zařízení A.

Zařízení B

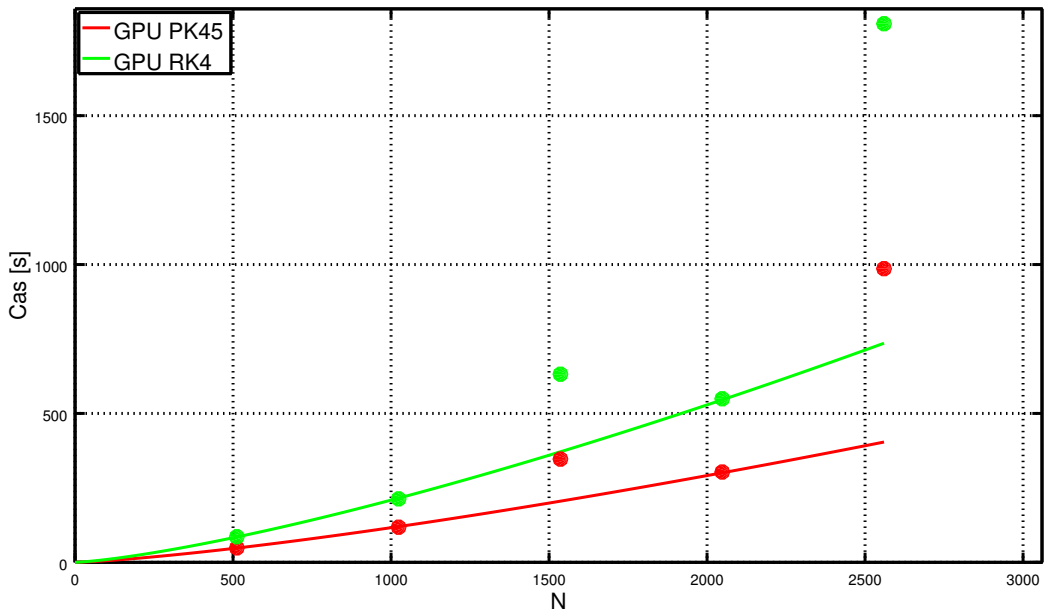
V tabulce 17 jsou naměřené časy pro zařízení B. Z těch byly aproximovány funkce závislosti času běhu na parametru N (viz tabulka 18). Naměřené časy proložené aproximační křivkou jsou na obrázku 17.

N	512	1024	1536	2048	2560
GPU PK45	48,4 s	118,0 s	346,7 s	303,2 s	986,1 s
GPU RK4	85,8 s	213,0 s	631,8 s	549,2 s	1808,6 s

Tabulka 17. Časy běhů FFT verze pro metody RK4 a PK45 na zařízení B.

GPU PK45	$t(N) = 0{,}012 \cdot N^{1{,}32}\, s$
GPU RK4	$t(N) = 0{,}020 \cdot N^{1{,}34}\, s$

Tabulka 18. Aproximované funkce závislosti času běhu na N zařízení B.



Obrázek 17. Časy běhů FFT verze pro metody RK4 a PK45 na zařízení B.

5.5. Vliv parametrů integrace na paměťovou náročnost a rychlost výpočtu

V této sekci je prozkoumán vliv integračních parametrů na rychlost výpočtu a velikost výstupních dat. Nalezená minima jsou závislá na konkrétním případě, proto je vhodné při použití ověřit jejich platnost experimenty. Výpočty derivací podle prostorové souřadnice byly provedeny metodou CFD, integrace metodou RK4. Tyto testy byly prováděny pouze na výkonnějším zařízení A. Výpis nebyl prováděn na ramfs, ale na HDD se souborovým systémem ext4. Veškeré soubory k testu (zdrojové kódy, automatizující skript a výsledky) jsou v adresáři *mereni/integration_params*.

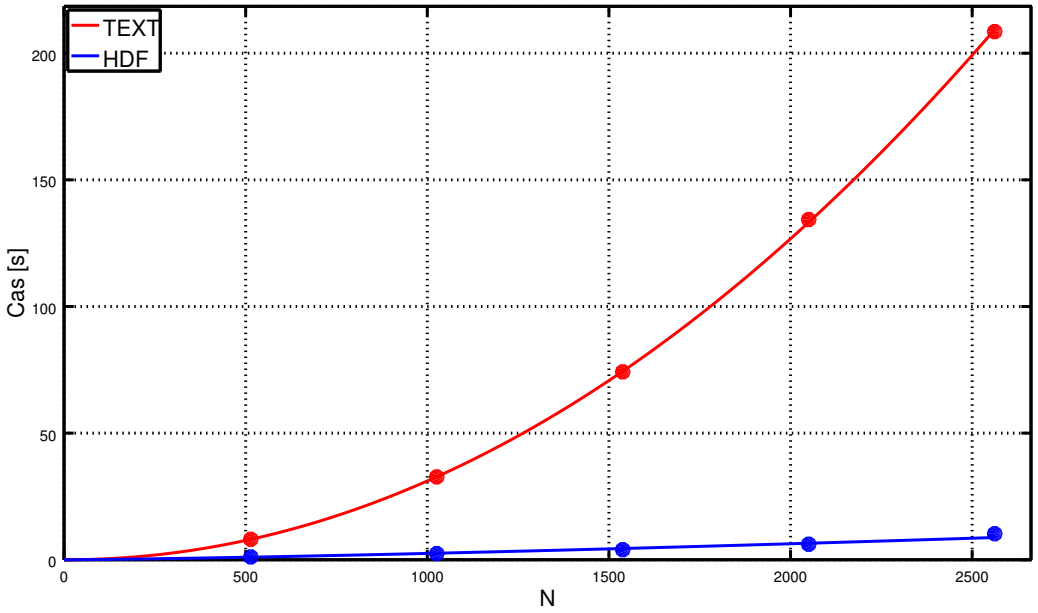
5.5.1. Formát výpisu

Výpis do textového formátu byl využit zejména při odlaďování programu. Výstup v této podobě je možné číst pomocí textových editorů. Implementace tohoto výpisu je jednoduchá, avšak nevhodná pro použití v praxi, neboť tento výpis je neefektivní jednak prostorově (k zápisu jednoho racionálního čísla s přesností na 16 platných číslic je nutné použít přibližně 17 B, oproti tomu typ double, který má podobnou přesnost, využívá 8 B), a jednak i rychlostí (dlouhý čas načítání do dalších aplikací, který je způsoben konverzí z textového formátu). Výpis do formátu HDF je sice implementačně složitější, avšak je efektivnější. Oproti textovému výpisu je i znatelně rychlejší (což je opět způsobeno konverzí z nativního formátu double do textového), viz tabulky 21 a 22. Naměřené hodnoty času proložené aproximační křivkou jsou na obrázku 18. Formát HDF je také úspornější, jak vyplývá z tabulek 21 a 22.

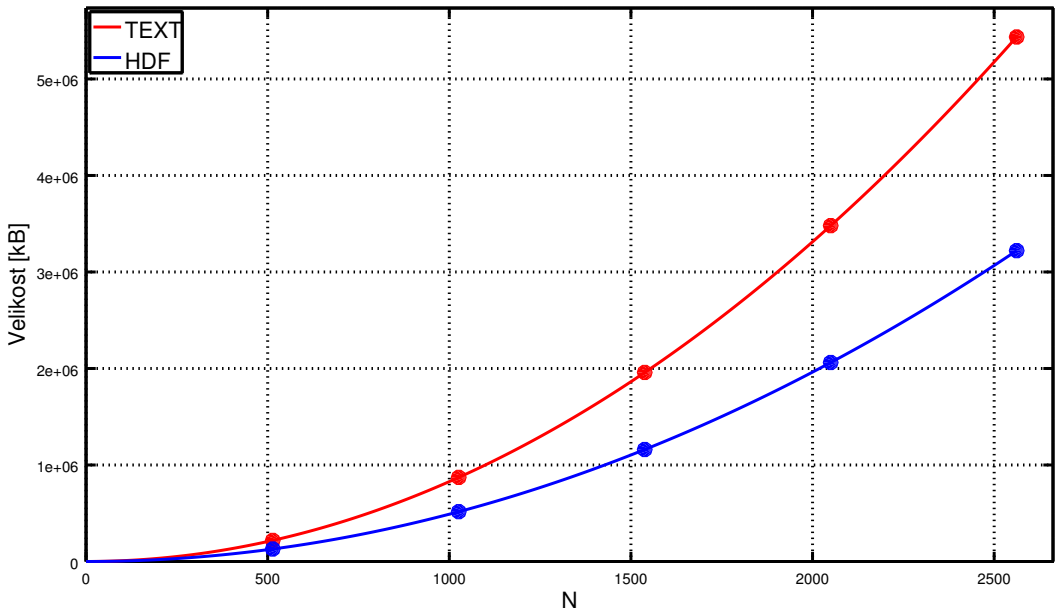
Nastavený krok výpisu podél časové i prostorové souřadnice byl 1, velikost bufferu byla 1000 kroků, poměr $\Delta X/\Delta T$ byl 2,0. V programu byl pro odhad velikosti výstupního souboru použit

N	514	1026	1538	2050	2562
TEXT	8,1 s	32,7 s	74,2 s	134,3 s	208,5 s
HDF	1,1 s	2,4 s	4,0 s	6,1 s	10,3 s

Tabulka 19. Tabulka závislosti rychlosti běhu na N pro různé formáty výpisu.



Obrázek 18. Graf závislosti rychlosti běhu na N pro různé formáty výpisu.



Obrázek 19. Graf závislosti velikosti výstupních dat na N pro různé formáty výpisu.

TEXT	$t(N) = 26 \cdot N^{2,03}\,\mu s$
HDF	$t(N) = 248 \cdot N^{1,34}\,\mu s$

Tabulka 20. Aproximované funkce závislosti rychlosti běhu na N pro různé formáty výpisu.

N	514	1026	1538	2050	2562
TEXT	213,5 MB	851,4 MB	1913,0 MB	3398,4 MB	5307,6 MB
HDF	126,4 MB	504,1 MB	1133,2 MB	2013,6 MB	3145,3 MB

Tabulka 21. Tabulka závislosti velikosti výstupních dat na N pro různé formáty výpisu.

TEXT	$s(N) = 0.82N^2\,\mathrm{kB}$
HDF	$s(N) = 0.48N^2\,\mathrm{kB}$

Tabulka 22. Aproximované funkce závislosti velikosti výstupních dat na N pro různé formáty výpisu.

empiricky odvozený vztah:

$$S\left(N, T_{max}, \frac{\Delta X}{\Delta T}\right) = \frac{T_{max}}{h_T} \cdot N \cdot \frac{1}{k_T} \cdot \frac{1}{k_X} \cdot \frac{1}{\alpha}\,\mathrm{kB} \tag{43}$$

Kde S je velikost výsledného souboru v kB, k_T je krok výpisu podél časové souřadnice, k_X je krok výpisu podél prostorové souřadnice a α je konstanta, jež pro HDF nabývá hodnoty 128 a pro textový výpis hodnoty 75,87. Koeficient h_T je krok integrace, spočtený jako:

$$h_T = \frac{1}{N-1} \cdot \frac{1}{\frac{\Delta X}{\Delta T}} \tag{44}$$

6. Příklady numerických výsledků

V této sekci jsou uvedeny příklady výstupů programu. Veškeré soubory (obrázky, skripty pro Matlab/Octave, zdrojové kódy řešiče) jsou v adresáři *obrazky*.

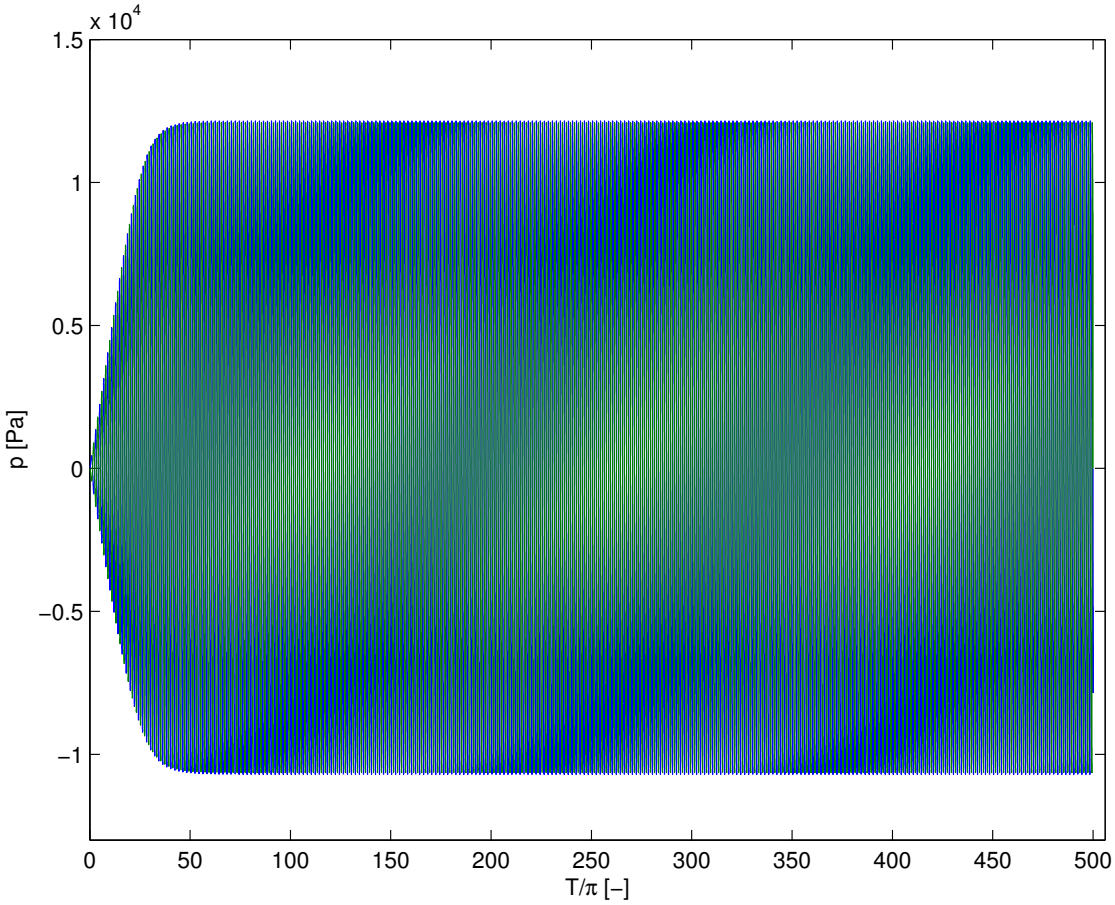
Parametry byly nastaveny:

- $G = 0{,}01$
- $\gamma = 1{,}4$
- $\Omega = 1{,}0$
- $c_0 = 345{,}0\, m/s$
- $\rho_0 = 1{,}2\, kg/m^3$
- $A_0 = 0{,}0005$
- $\frac{\Delta X}{\Delta T} = 0{,}5$
- $T_{max} = 500\pi$
- $N = 1026$
- $print\_step = 10$
- $print\_step\_X = 1$

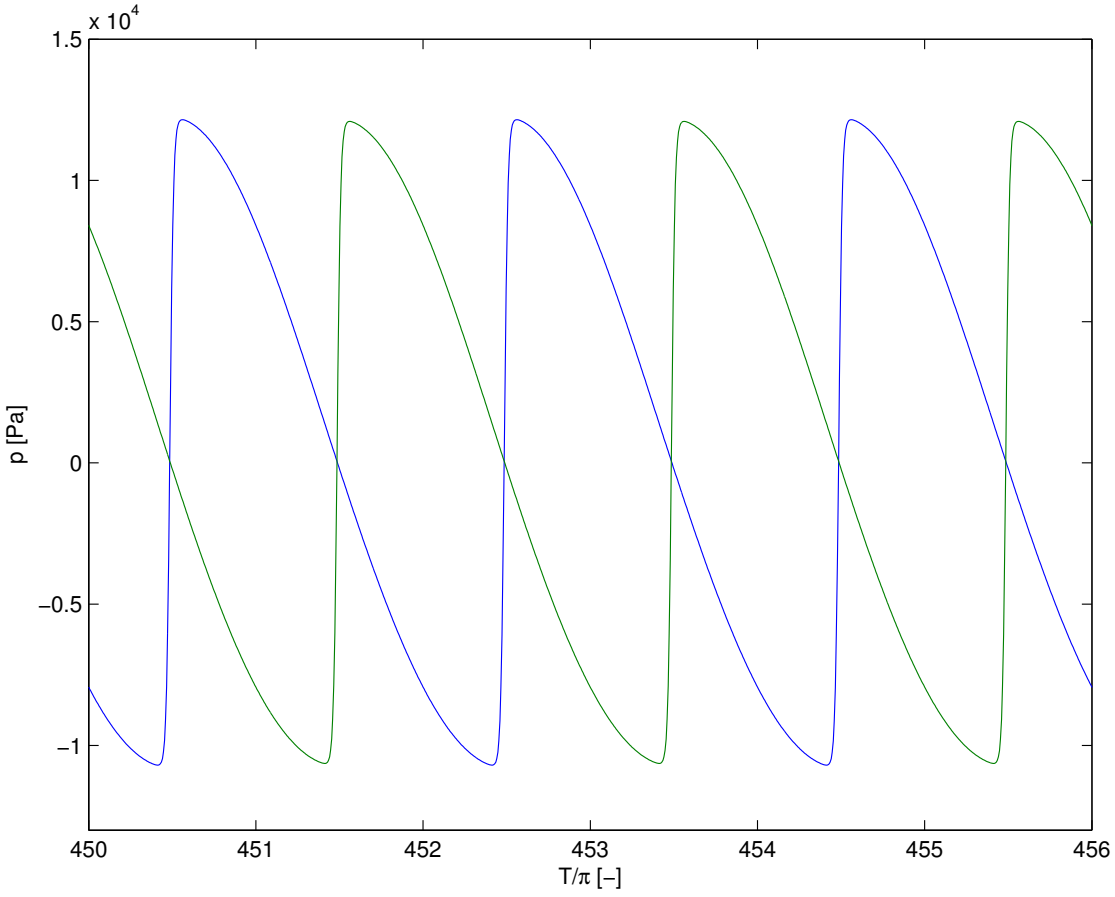
6.1. Rezonátor s konstantním průřezem

V této sekci jsou uvedeny výsledky běhu programu pro konstantní průřez ($R(X) = 1$) rezonátoru vyplněného vzduchem za normálních podmínek. Rezonátor je buzen na prvním vlastním kmitočtu ($\Omega = 1{,}0$).

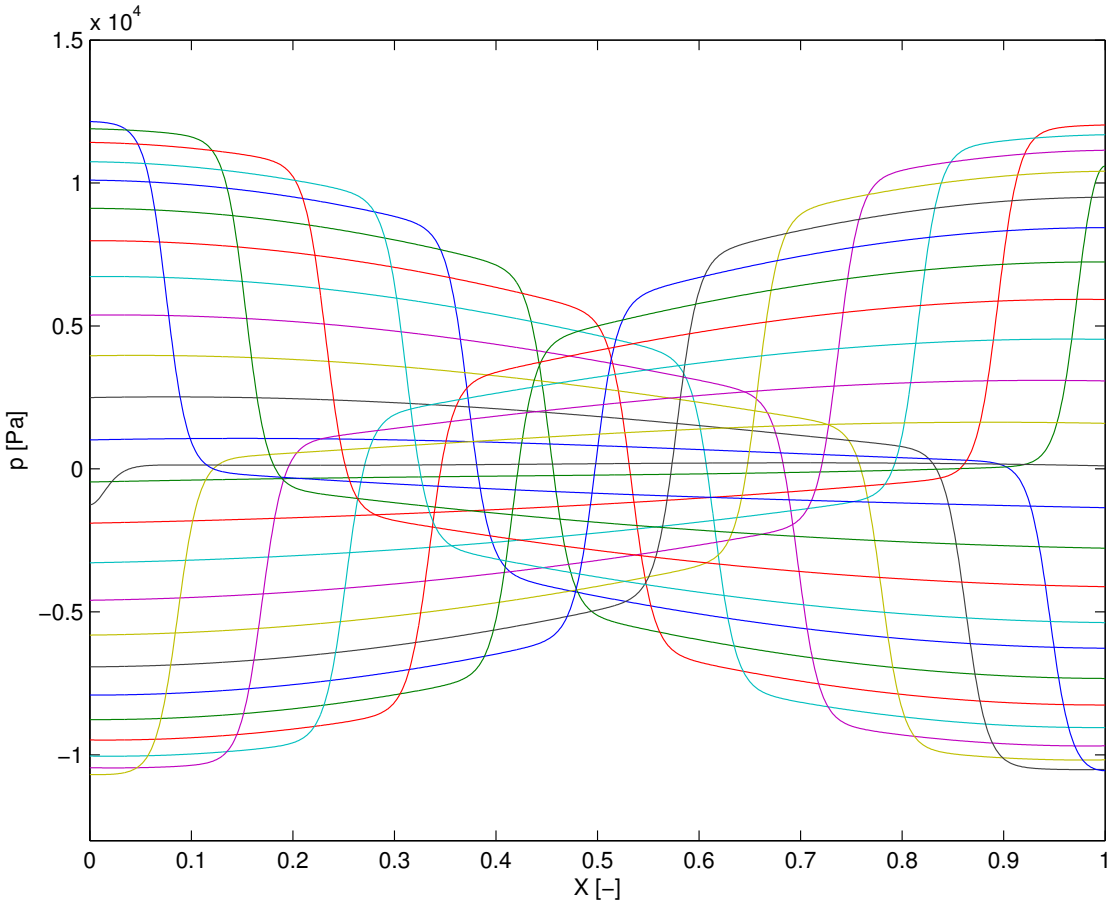
Na obrázku 20 je zobrazen akustický tlak na koncích rezonátoru (grafy se překrývají). Po krátkém přechodovém jevu je dosažen ustálený stav. Vidíme, že tento stav je stabilní a nejsou přítomny žádné výchylky, které by svědčily o numerických nestabilitách výpočtu či kumulaci numerických chyb. Na obrázku 21 je detail průběhu akustického tlaku v ustáleném stavu na koncích rezonátoru. Je vidět, že tlak na opačných koncích rezonátoru má stejnou amplitudu a je v protifázi. Jasně patrná je také rázová vlna (periodicky se opakující prudká náběžná hrana), která způsobuje zvýšenou disipaci akustické energie. Obrázek 22 zachycuje pohyb rázové vlny (tlaku) rezonátorem v pravidelných časových intervalech v průběhu jedné periody (v ustáleném stavu). Vlna se pohybuje od počátku rezonátoru (X = 0) k jeho konci (X = 1), kde se odrazí v protifázi a pohybuje se zpět k počátku. Na obrázku 23 je zobrazena akustická rychlost uprostřed rezonátoru. Je opět patrná rázová vlna (prudká náběžná a sestupná hrana). Symetrie průběhu je způsobena buzením na rezonanční frekvenci. Obrázek 24 zachycuje v pravidelných časových intervalech pohyb rázové vlny (rychlost) rezonátorem v průběhu jedné periody.



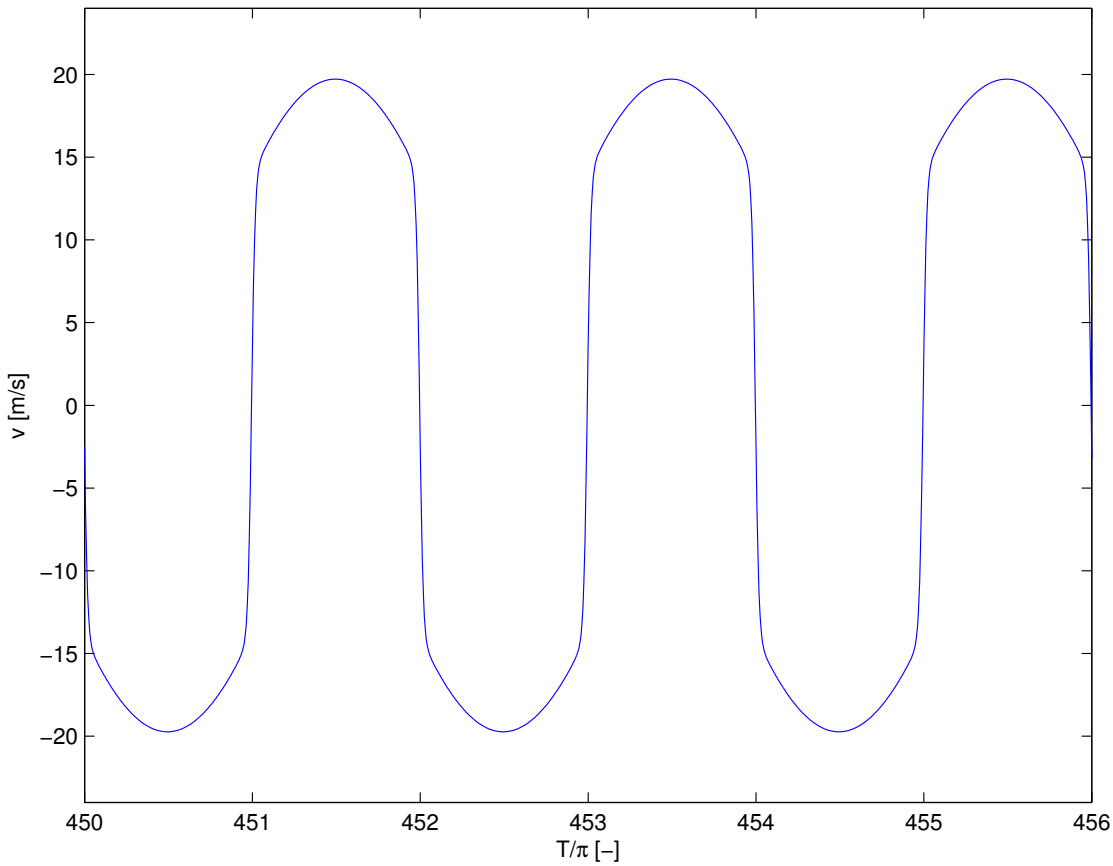
Obrázek 20. Časový vývoj akustického tlaku na koncích rezonátoru.



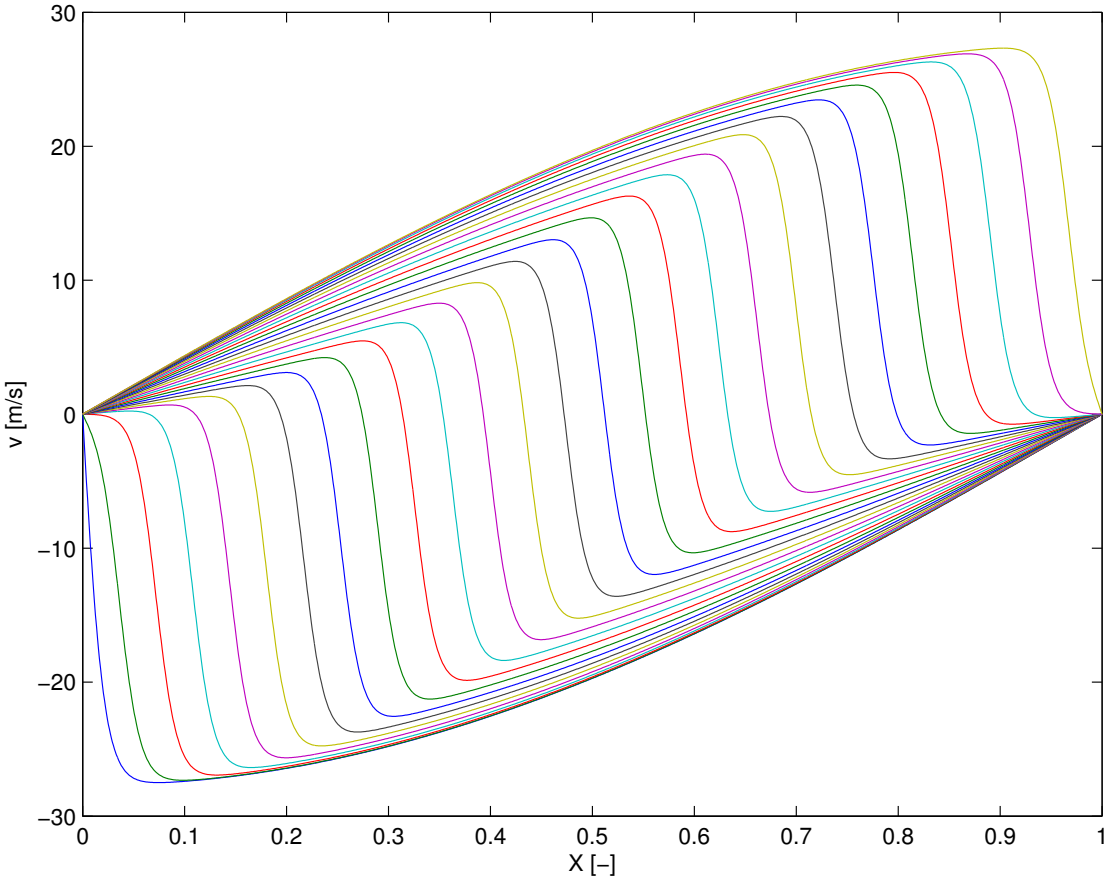
Obrázek 21. Časový vývoj akustického tlaku na koncích rezonátoru - ustálený stav.



Obrázek 22. Rozložení akustického tlaku podél rezonátoru během jedné periody.



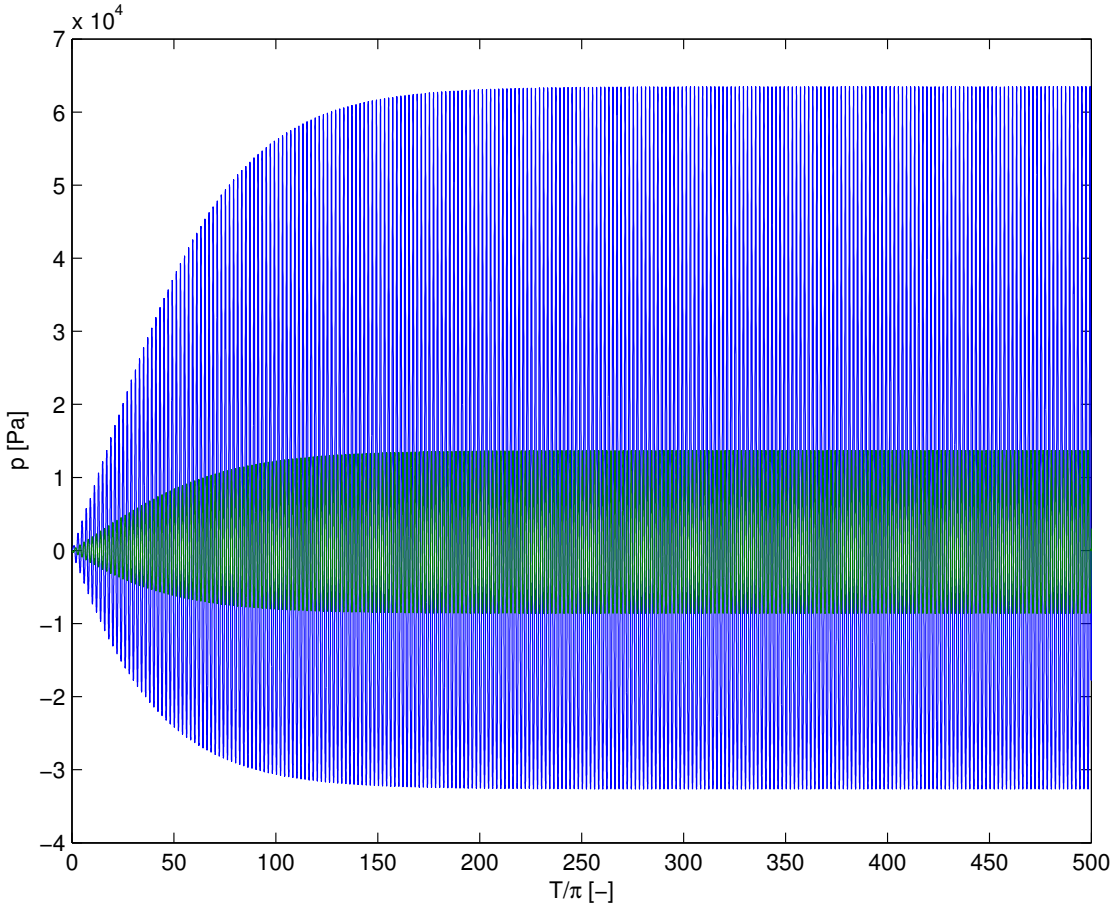
Obrázek 23. Časový vývoj akustické rychlosti uprostřed rezonátoru - ustálený stav.



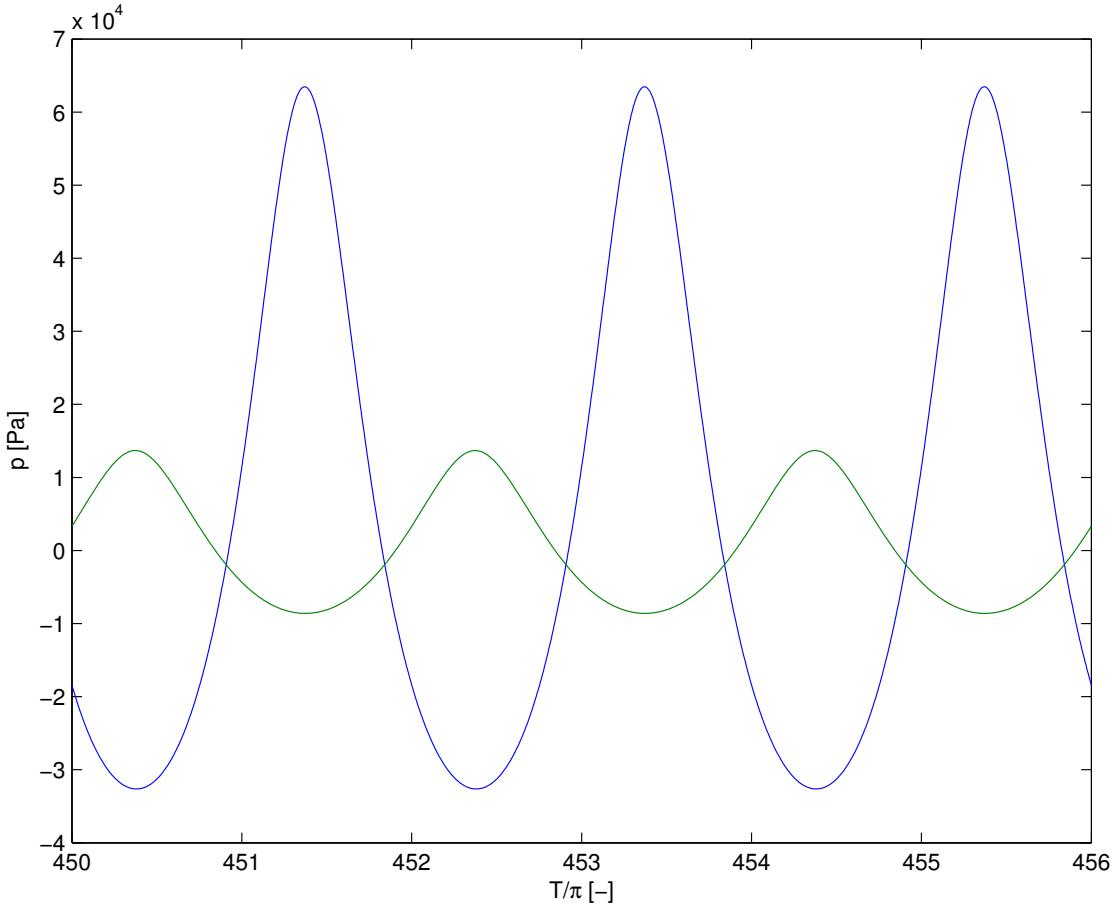
Obrázek 24. Rozložení akustické rychlosti podél rezonátoru během jedné periody.

6.2. Rezonátor proměnného průřezu

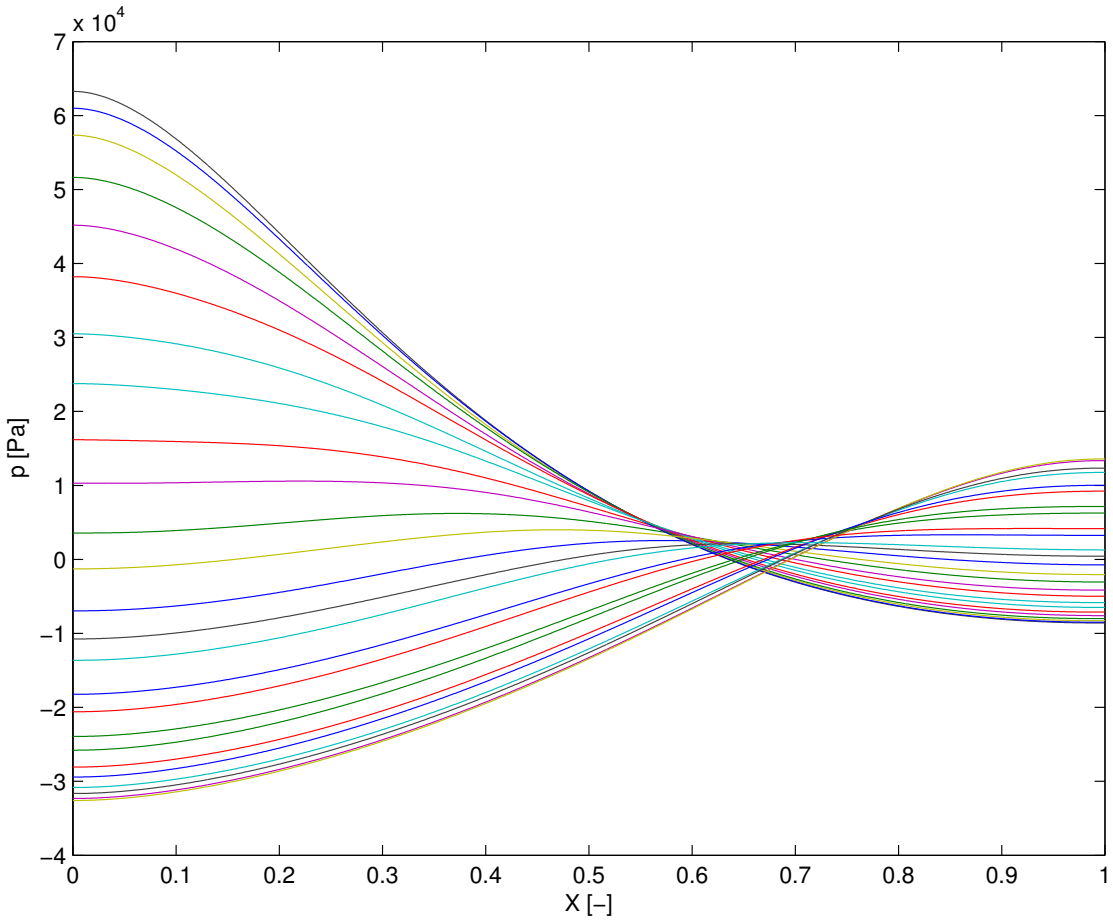
V této sekci jsou uvedeny výsledky běhu programu s nastaveným proměnným průřezem rezonátoru $R(X) = 0{,}268 \cdot X + 0{,}0352941$. Nastavení parametrů bylo stejné jako v minulém případě, s vyjímkou parametru $\Omega = 1{,}2930439285$. Tato frekvence odpovídá prvnímu vlastnímu kmitočtu pro tento tvar (komolý kužel). Na obrázku 25 je zobrazen průběh akustického tlaku na užším konci (X = 0) a na širším konci (X = 1) rezonátoru. Tlak na užsím konci je výrazně vyšší než na širším konci (podstavě kuželu), což je rozdílné oproti rezonátoru s konstantním průřezem (obrázek 20). Ve srovnání s rezonátorem konstantního průřezu je také patrné, že pro kužel je delší přechodový jev. Obrázek 26 zachycuje detail průběhu akustického tlaku na počátku a konci rezonátoru v ustáleném stavu. Oproti rezonátoru konstantního průřezu (obrázek 21) lze vidět, že dosažený tlak je znatelně vyšší (přibližně $6 \cdot 10^4$ Pa proti $1 \cdot 10^4$ Pa), vidíme že kuželový rezonátor je vhodnější než válcový. Dalším důležitým rozdílem je, že tento průběh netrpí rázovými vlnami, ve kterých se disipuje energie (nábežné i sestupné hrany nejsou tak strmé). Obrázek 27 zachycuje v pravidelných časových intervalech pohyb vlny rezonátorem během jedné periody. Vlna se na konci odráží v protifázi. Z tvaru křivek je možné usoudit, že není přítomna rázová vlna. Celkově tedy můžeme říci, že tvar komolého kuželu je vhodnější než válce – je dosaženo vyšších tlaků při absenci rázové vlny.



Obrázek 25. Časový vývoj akustického tlaku na koncích rezonátoru.



Obrázek 26. Časový vývoj akustického tlaku na koncích rezonátoru - ustálený stav.



Obrázek 27. Rozložení akustického tlaku podél rezonátoru během jedné periody.

7. Závěr

V rámci této práce byl implementován řešič Kuzněcovovy rovnice pro GPGPU a byl prověřen z hlediska numerické stability a konvergence výpočtu. Byly implementovány dvě verze – *wave_solver.cu* pro metodu konečných diferencí a *wave_solver_fft.cu* pro metodu Fourierovu. Obě tyto verze umožňují integraci v čase metodami Runge–Kutta 4. řádu a prediktor–korektor (prediktor 4. řádu, korektor 5. řádu). Urychlení výpočtu bylo dosaženo použitím bufferu pro výpis (čímž se snížil počet volání funkcí pro výpis a zvětšil se objem dat na jedno volání), výpisem do binárního formátu HDF a paralelizaci výpisu (dvouvláknová implementace). Při numerických testech bylo zjištěno, že metoda RK4 je oproti PK45 pro tento případ méně časově náročná i přes vyšší aritmetickou náročnost – důvodem je její lepší stabilita, která umožňuje nastavit větší krok. Pro srovnání byla rovněž implementována jednovláknová CPU verze (metody RK4 a PK45, výpočet derivace podle prostorové souřadnice metodou konečných diferencí). Bylo ověřeno, že GPGPU verze přináší zlepšení časové závislosti (pro N do přibližně 8000 odpovídající přibližně $N^{1,4}$) oproti CPU verzi (jejíž závislost je úměrná N^2) a je výrazně rychlejší.

Nevýhodou verze používající metodu konečných diferencí je neefektivní využití grafické karty při výpočtu krajních bodů. Verze používající metodu Fourierovu tímto nedostatkem netrpí – splnění okrajových podmínek (12) je vynuceno použitou metodou a není proto nutné speciálním způsobem ošetřovat krajní body. Tato metoda je ale pomalejší, nicméně protože pracuje ve spektrální oblasti, je možné implementovat filtraci dat či složitější mechanismy absorpce a disperze (popsaných v kmitočtové oblasti). Byla provedena rozsáhlá testování algoritmu na dvou zařízeních a základní numerické experimenty s rezonátory konstantního a proměnného průřezu a bylo ukázáno, že v dutinovém rezonátoru tvaru komolého kužele lze generovat stojaté akustické vlnění výrazně vyšších amplitud než v rezonátoru tvaru válce.

Příloha A.

Konečné diference

V následujících přílohách jsou ukázány postupy vedoucí k získání vztahů pro aproximaci derivací a extrapolaci okrajových bodů metodou konečných diferencí.

A.1. Aproximace první a druhé derivace metodou centrálních konečných diferencí

Funkční hodnoty $f(x \pm h)$, $f(x \pm 2h)$ mohou být spočteny rozvojem do Taylorovy řady jako:

$$\begin{aligned} f(x+h) &= f(x) + f^{(1)}(x)h + \frac{1}{2}f^{(2)}(x)h^2 + \frac{1}{6}f^{(3)}(x)h^3 + \\ &\quad + \frac{1}{24}f^{(4)}(x)h^4 + \frac{1}{120}f^{(5)}(x)h^5 + \frac{1}{720}f^{(6)}(x)h^6 + \dots, \qquad (45) \\ f(x+2h) &= f(x) + 2f^{(1)}(x)h + 2f^{(2)}(x)h^2 + \frac{4}{3}f^{(3)}(x)h^3 + \\ &\quad + \frac{2}{3}f^{(4)}(x)h^4 + \frac{4}{15}f^{(5)}(x)h^5 + \frac{4}{45}f^{(6)}(x)h^6 + \dots, \qquad (46) \\ f(x-h) &= f(x) - f^{(1)}(x)h + \frac{1}{2}f^{(2)}(x)h^2 - \frac{1}{6}f^{(3)}(x)h^3 + \\ &\quad + \frac{1}{24}f^{(4)}(x)h^4 - \frac{1}{120}f^{(5)}(x)h^5 + \frac{1}{720}f^{(6)}(x)h^6 - \dots, \qquad (47) \\ f(x-2h) &= f(x) - 2f^{(1)}(x)h + 2f^{(2)}(x)h^2 - \frac{4}{3}f^{(3)}(x)h^3 + \\ &\quad + \frac{2}{3}f^{(4)}(x)h^4 - \frac{4}{15}f^{(5)}(x)h^5 + \frac{4}{45}f^{(6)}(x)h^6 - \dots, \qquad (48) \end{aligned}$$

kde $f^{(i)}(x)$ představuje i-tou derivaci funkce f podle x.

Nejdřív eliminujeme třetí derivaci z rovnic (45)-(48):

$$\begin{aligned} 3f(x+2h) - 24f(x+h) &= -21f(x) - 18f^{(1)}(x)h - 6f^{(2)}(x)h^2 + \\ &\quad + f^{(4)}(x)h^4 + \frac{3}{5}f^{(5)}(x)h^5 + \frac{7}{30}f^{(6)}(x)h^6 + \dots, \\ 3f(x-2h) - 24f(x-h) &= -21f(x) + 18f^{(1)}(x)h - 6f^{(2)}(x)h^2 + \\ &\quad + f^{(4)}(x)h^4 - \frac{3}{5}f^{(5)}(x)h^5 + \frac{7}{30}f^{(6)}(x)h^6 + \dots. \end{aligned}$$

Odečtením těchto výsledků získáme:

$$3f(x+2h) - 24f(x+h) + 24f(x-h) - 3f(x-2h) = -36f^{(1)}(x)h + \frac{6}{5}f^{(5)}(x)h^5 + \dots,$$

což může být přepsáno jako

$$f^{(1)}(x) = \frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12h} + \mathcal{O}(h^4). \qquad (49)$$

Vztah (49) může být použit pro aproximaci první derivace metodou konečných diferencí. Chyba aproximace klesá se čtvrtou mocninou velikosti diference h.

Dále eliminujeme čtvrtou derivaci z rovnic (45)-(48):

$$\begin{aligned} 3f(x+2h)-48f(x+h) &= -45f(x)-42f^{(1)}(x)h-18f^{(2)}(x)h^2- \\ &\quad -4f^{(3)}(x)h^3+\frac{2}{5}f^{(5)}(x)h^5+\frac{1}{5}f^{(6)}(x)h^6+\ldots, \\ 3f(x-2h)-48f(x-h) &= -45f(x)+42f^{(1)}(x)h-18f^{(2)}(x)h^2+ \\ &\quad +4f^{(3)}(x)h^3-\frac{2}{5}f^{(5)}(x)h^5+\frac{1}{5}f^{(6)}(x)h^6+\ldots. \end{aligned} \tag{50}$$

Sečtením těchto výsledků získáme

$$3f(x+2h)-48f(x+h)-48f(x-h)+3f(x-2h)=-90f(x)-36f^{(2)}(x)h^2+\frac{2}{5}f^{(6)}(x)h^6+\ldots,$$

což může být přepsáno jako

$$f^{(2)}(x)=\frac{-f(x-2h)+16f(x-h)-30f(x)+16f(x+h)-f(x+2h)}{12h^2}+\mathcal{O}(h^4). \tag{51}$$

Vztah (51) může být použit pro aproximaci druhé derivace metodou konečných diferencí. Chyba aproximace klesá se čtvrtou mocninou velikosti diference h.

A.2. Extrapolace metodou konečných diferencí

Vycházíme z předpokladu, že $f^{(1)}(x)$, $f(x+h)$, $f(x+2h)$, a $f(x+3h)$ jsou známé a $f(x)$ a $f(x-h)$ budou aproximovány. Rozvoj funkce $f(x)$ Taylorovou řadou je:

$$f(x-h) = f(x)-f^{(1)}(x)h+\frac{1}{2}f^{(2)}(x)h^2-\frac{1}{6}f^{(3)}(x)h^3+\frac{1}{24}f^{(4)}(x)h^4+\ldots, \tag{52}$$

$$f(x+h) = f(x)+f^{(1)}(x)h+\frac{1}{2}f^{(2)}(x)h^2+\frac{1}{6}f^{(3)}(x)h^3+\frac{1}{24}f^{(4)}(x)h^4+\ldots, \tag{53}$$

$$f(x+2h) = f(x)+2f^{(1)}(x)h+2f^{(2)}(x)h^2+\frac{4}{3}f^{(3)}(x)h^3+\frac{2}{3}f^{(4)}(x)h^4+\ldots, \tag{54}$$

$$f(x+3h) = f(x)+3f^{(1)}(x)h+\frac{9}{2}f^{(2)}(x)h^2+\frac{9}{2}f^{(3)}(x)h^3+\frac{27}{8}f^{(4)}(x)h^4+\ldots, \tag{55}$$

Druhá derivace může být odstraněna použitím rovnic (53) a (54):

$$4f(x+h)-f(x+2h)=3f(x)+2f^{(1)}(x)h-\frac{2}{3}f^{(3)}(x)h^3-\frac{1}{2}f^{(4)}(x)h^4+\ldots \tag{56}$$

a podobně rovnicemi (53) a (55):

$$18f(x+h)-f(x+3h)=16f(x)+12f^{(1)}(x)h-6f^{(3)}(x)h^3-6f^{(4)}(x)h^4+\ldots. \tag{57}$$

Rovnice (56) a (57) mohou být použity pro eliminaci třetí derivace:

$$36f(x+h)-18f(x+2h)+4f(x+3h)=22f(x)+12f^{(1)}(x)h+3f^{(4)}(x)h^4+\ldots. \tag{58}$$

Odtud můžeme psát

$$f(x)=\frac{18f(x+h)-9f(x+2h)+2f(x+3h)-6f^{(1)}(x)h}{11}+\mathcal{O}(h^4), \tag{59}$$

odtud vidíme, že chyba odhadu hodnoty $f(x)$ klesá se čtvrtou mocninou h.

Podobně je možné eliminovat druhé derivace použitím vztahů (52) a (54)

$$4f(x-h)-f(x+2h)=3f(x)-6f^{(1)}(x)h-2f^{(3)}(x)h^3-\frac{1}{2}f^{(4)}(x)h^4+\ldots \tag{60}$$

a vztahů (52) a (55) jako

$$18f(x-h) - 2f(x+3h) = 16f(x) - 24f^{(1)}(x)h - 12f^{(3)}(x)h^3 - 6f^{(4)}(x)h^4 + \dots. \tag{61}$$

Rovnice (60) a (61) mohou být použity pro eliminaci třetí derivace:

$$6f(x-h) - 6f(x+2h) + 2f(x+3h) = 2f(x) - 12f^{(1)}(x)h + 3f^{(4)}(x)h^4 + \dots. \tag{62}$$

Konečně, rovnice (58) a (62) jsou použity pro eliminaci $f(x)$, tedy

$$6f(x+h) + 8f(x+2h) - 3f(x+3h) - 11f(x-h) = 24f^{(1)}(x)h - 5f^{(4)}(x)h^4 + \dots. \tag{63}$$

Odtud můžeme psát

$$f(x-h) = \frac{6f(x+h) + 8f(x+2h) - 3f(x+3h) - 24f^{(1)}(x)h}{11} + \mathcal{O}(h^4), \tag{64}$$

odtud vidíme, že chyba odhadu hodnoty $f(x-h)$ klesá se čtvrtou mocninou h.

Literatura

[1] Christopher C. Lawrenson et al. "Measurements of macrosonic standing waves in oscillating closed cavities". In: *Acoustical Society of America* (1998).

[2] Milan Červenka, Martin Šoltés a Michal Bednařík. "Optimal shaping of acoustic resonators for the generation of high-amplitude standing waves". In: *Acoustical Society of America* (2014).

[3] Yurii A. Ilinskii et al. "Nonlinear standing waves in an acoustical resonator". In: *Acoustical Society of America* (1998).

[4] Steven G. Johnson. *Notes on FFT–based differentiation*. Massachusetts Institute of Technology. 2015. URL: `http://math.mit.edu/~stevenj/fft-deriv.pdf`.

[5] *cuFFT Documentation*. NVIDIA Corporation. 2015. URL: `http://docs.nvidia.com/cuda/cufft/`.

[6] William H. Press et al. *Numerical Recipes*. 3. vyd. Cambridge University Press, 2007. ISBN: 9780521880688.

[7] Daniel Zwillinger. *Handbook of Differential Equations*. 3. vyd. Academic Press, 1997. ISBN: 9780127843964.

[8] Mohamed Zahran. *History of GPU computing*. New York University. 2015. URL: `http://cs.nyu.edu/courses/spring12/CSCI-GA.3033-012/lecture2.pdf`.

[9] *Computer Systems Architecture: GPU*. University of Maryland. 2015. URL: `http://www.cs.umd.edu/class/fall2013/cmsc411-0201/lectures/lec23.pdf`.

[10] *HDF Projects*. HDF Group. 2015. URL: `https://www.hdfgroup.org/projects/`.

[11] Dave McCracken. *POSIX Threads and the Linux Kernel*. IBM Linux Technology Center. 2015. URL: `https://www.kernel.org/doc/ols/2002/ols2002-pages-330-337.pdf`.

[12] *NVIDIA GeForce 580 GTX Specifications*. NVIDIA Corporation. 2015. URL: `http://www.geforce.co.uk/hardware/desktop-gpus/geforce-gtx-580/specifications`.

[13] *NVIDIA GeForce Series Compute Capability*. NVIDIA Corporation. 2015. URL: `https://developer.nvidia.com/cuda-gpus`.

[14] *NVIDIA GeForce 750 GTX Specifications*. NVIDIA Corporation. 2015. URL: `http://www.geforce.co.uk/hardware/desktop-gpus/geforce-gtx-750/specifications`.

[15] Eric W. Weisstein. *Least Squares Fitting–Polynomial*. From MathWorld–A Wolfram Web Resource. 2015. URL: `http://mathworld.wolfram.com/LeastSquaresFittingPolynomial.html`.