

Nalezení minimální plochy pomocí Jacobiho metody

Adam Pavlát

13. 5. 2016

1 Teoretický úvod

Jak je napsáno v zadání, máme zadanou parametrizovanou křivku l se souřadnicemi $\mathbf{r} = \mathbf{r}(t)$, kde $t \in (0, T)$. Tato křivka je okrajovou podmínkou pro Laplaceovu rovnici

$$\frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2} = 0,$$

která řeší problematiku nalezení minimální plochy. Tato rovnice je typu parciální diferenciální a obecně neexistuje její analytické řešení, a tak přistoupíme k numerickým metodám - Jacobiho iterativní metoda.

V této metodě se nahradí derivace konečnými diferenciemi

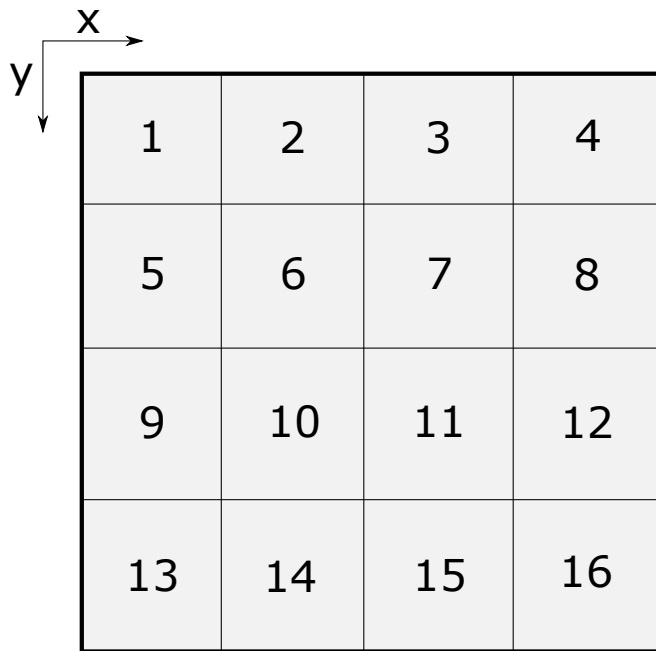
$$z(m, n, i + 1) = \frac{1}{4}(z(m + 1, n, i) + z(m - 1, n, i) + z(m, n + 1, i) + z(m, n - 1, i)) \quad (1)$$

Tento algoritmus je zastaven testem kontrolního bodu, kde měříme změnu před a po provedení iterace.

Ve 4. bodě této zprávy jsou ukázky řešení pro různé křivky.

2 Řešení serializací pracovního prostoru

Jedno z možných výpočetních řešení je metoda, při které se problém převede na soustavu lineárních rovnic. To se provede tak, že vzestupně označíme prvky diskrétního pracovního prostoru, který je velikosti $M \times N$ viz. Obrázek 1. Hodnoty jsou určeny zaokrouhlením zobrazení křivky do plochy. Je to tedy ekvidistantní mřížka. Tak vznikne matice resp. sloupcový vektor S .



1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

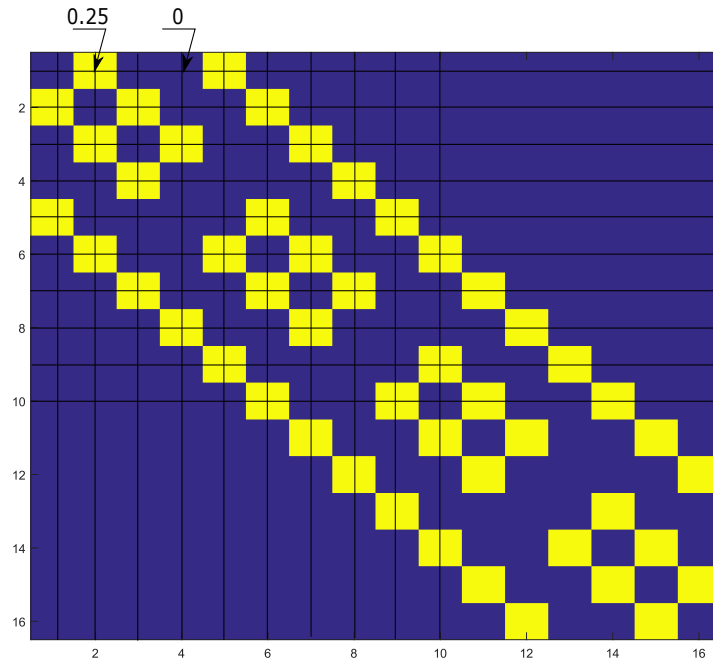
Obrázek 1: Indexování prvků pracovního prostoru (rastr 4x4)

Následně budeme potřebovat matici A, která v podstatě každému bodu vybere jeho čtyři sousedy. Vynásobením matic

$$AS_i = S_{i+1}$$

získáme další iteraci S.

Ukažme si tvar matice A pro rastr 4x4 na Obrázku 2



Obrázek 2: Matice A pro rastr 4x4

Na obrázku 2 můžeme dále vidět, že v jednotlivých řádcích nejsou vždy 4 čísla, to proto, protože se nachází u kraje pracovního prostoru a nemá tedy více sousedů (šikmí směr neuvažujeme) než dva. Pro ukázkou první řádek matice A má odpovídat po přenásobení vektorem S sousedům prvního prvku v pracovním prostoru na obrázku 1 (prvek vlevo nahoře). V prvním řádku tedy vidíme první prvek ne (je 0), druhý ano, třetí a čtvrtý ne, pátý ano. A skutečně, s prvkem s indexem 1 sousedí prvky 2 a 5. A takto bychom mohli pokračovat pro všechny prvky vektoru S.

Úlohu jsme tímto převedli na řešení soustavy lineárních rovnic, kterých je mnoho, resp. $M \times N$.

Asi největším problémem tohoto řešení je potřeba generování mnohdy velké matice A, jejíž počet prvků je dán vztahem $M \times N \times M \times N$. Pro ukázkou s rastrem, kde $N = M$ a $M \in \{50, 100, 150, 200, 250, 300\}$ roste počet prvků 4. mocninou, tedy počet řádků čtvercové matice A bude $\{2500, 10000, 22500, 40000, 62500, 90000\}$ a počty prvků budou druhé mocniny těchto čísel. To představuje opravdu hodně paměti. V prostředí matlab tuto situaci vylepší funkce sparse, která si neukládá nuly této řídké matice.

Řešení pracovní oblasti bylo řešeno tak, že byli zobrazeny body parametrizované křivky do roviny a následně se pro řádky resp. sloupce této matice cykli tato oblast ohraničila. Toto je výsledkem funkce getBound, která vrací zobrazené body a především vytyčí vnitřek a vnějšek křivky.

Do programu je možné vložit i křivky, které jsou v xy větší než 1. Tyto jsou zmenšeny tak, aby se vešly do pracovního prostoru, při koncovém vykreslení jsou ale zase vráceny zpět na původní velikost. Toto můžeme pozorovat například u nephroidu (bod 4.4). Program by tak měl respektovat zadanou velikost křivky.

3 Řešení pomocí konvoluce

Konvoluce ve 2D rovině znamená součet prvků na typicky čtvercové doméně, jejich váhování dané hodnotami kernelu a uložení výsledné hodnoty do prostředního prvku.

Pro ukázkou si princip ukažme. Máme matici L a matici K , matici K nazýváme konvoluční matice neboli kernel a matice L je náš pracovní prostor.

Označme $*$ jako znak konvoluce

$$L * K = L_1$$

Pokud např. $L = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{pmatrix}$ a $K = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$, pak

$$L * K = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & \mathbf{7} & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & \mathbf{19} & 20 \\ 21 & 22 & 23 & 24 & 25 \end{pmatrix} * \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & \mathbf{7} & 8 & 9 & 10 \\ 11 & 12 & \mathbf{26} & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{pmatrix}$$

protože (teď bude $*$ normální násobení) $7 * 1 + 19 * 1 + \text{ostatní} * 0 = 26$

V matlabu je pro 2D konvoluci funkce `conv2`, která je silně optimalizovaná. Aby řešila naší úlohu resp. rovnici (1), měli bychom zvolit kernel

$$K = \frac{1}{4} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Na tomto řešení je zajímavé to, že program sestavený v rámci této práce (serializace, bod 2) nebyl nikdy rychlejší než konvoluce.

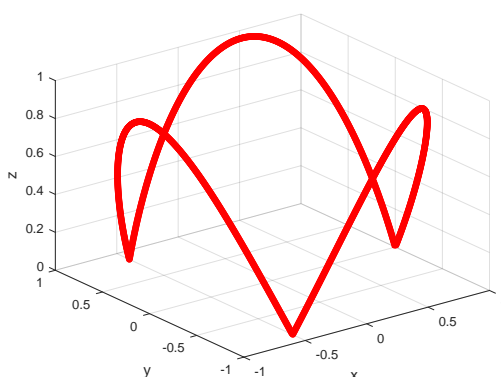
4 Výsledky

Výpočty byli prováděny na notebooku Lenovo Y700, Intel Core i7-6700HQ (2.6GHz, TB 3.5GHz, HyperThreading), 16GB RAM DDR4, GeForce GTX 960M 4GB GDDR5, SSD disk. Byl použit MATLAB 2015b pod ČVUT FEL licenci.

4.1 Testovací křivky

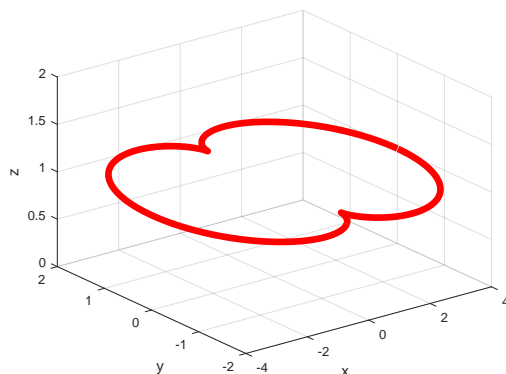
Křivka ze zadání má jako jedinou proměnnou z závislou na parametru t , ostatní funkce jí mají nastavenou na konstantu. To by mělo znamenat, že budou křivky 2 až 4 skončit jako ploché plochy. Nicméně díky konci našeho algoritmu, kde při malém progresu na dané souřadnici výpočet končí, uvidíme, že plochy budou stejně trochu zakřivené.

- křivka ze zadání: $x = \cos(t)$, $y = \sin(t)$, $z = \left| \sin\left(\frac{3}{2}t\right) \right|$



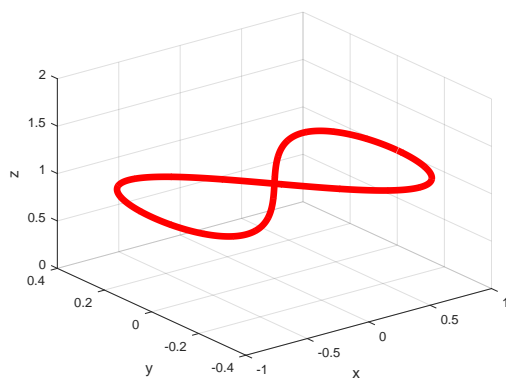
Obrázek 3: Křivka ze zadání

- nephroid: $x = 3\cos(t) + \cos(3t)$, $y = 3\sin(t) + \sin(3t)$, $z = 1$



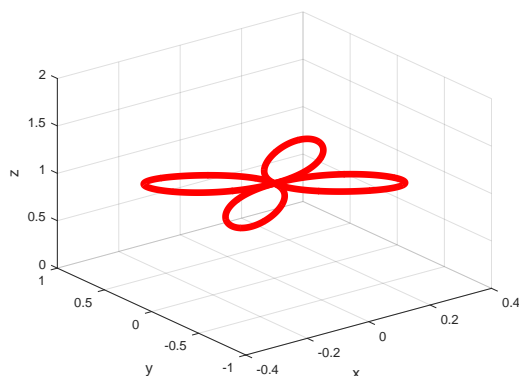
Obrázek 4: Křivka s názvem nephroid

- symbol nekonečna: $x = \cos(t)$, $y = \sin(t)\cos(t)$, $z = 1$



Obrázek 5: Symbol nekonečna

- další křivka: $x = \cos(t) - \cos^3(t)$, $y = \sin(t) - \sin^5(t)$, $z = 1$



Obrázek 6: Jedna z testovacích křivek

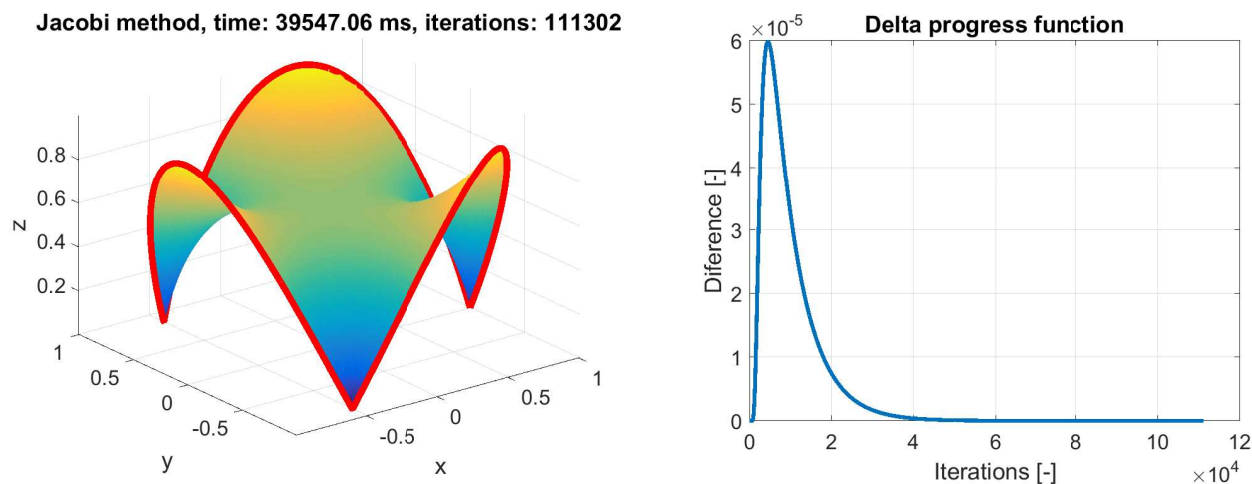
4.2 Testovací parametry

Abychom mohli srovnat jednotlivé metody a časové náročnosti, zvolíme si následující parametry:

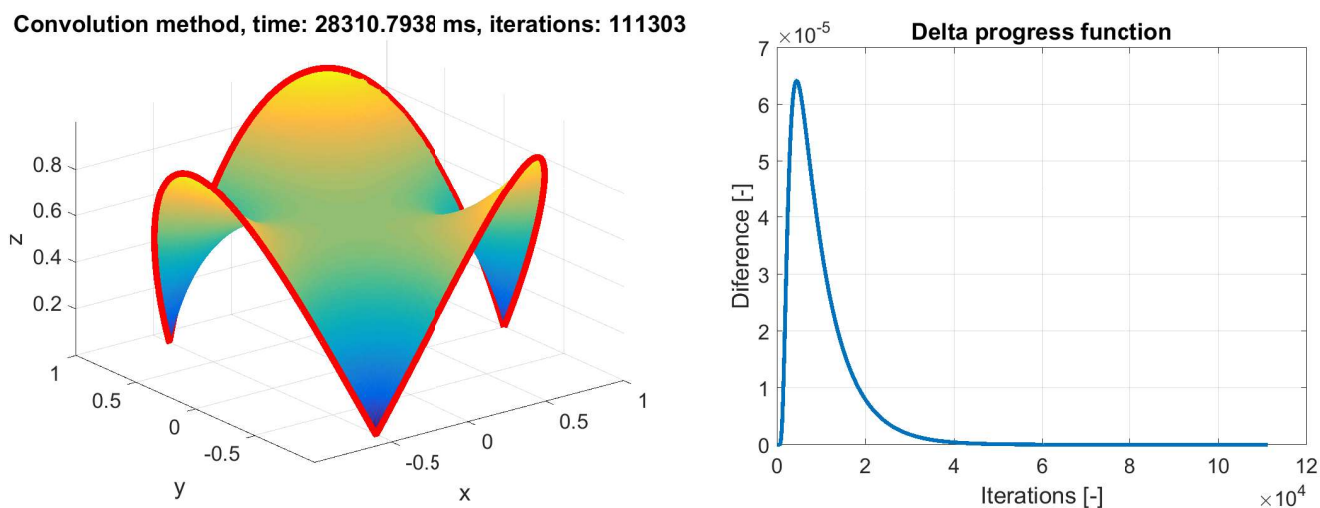
1. velikost rastru 200x200 bodů
2. $t\epsilon(0, 2\pi)$ a počet hodnot v simulaci je 2000
3. relativní přesnost měření $\varepsilon_{max} < 10^{-11}$ pro aktuální a následnou iteraci, kontrolní body jsou voleny individualně

4.3 Křivka ze zadání

Kontrolní bod pro určování delta progressu je v bodě $[0, 0]$.

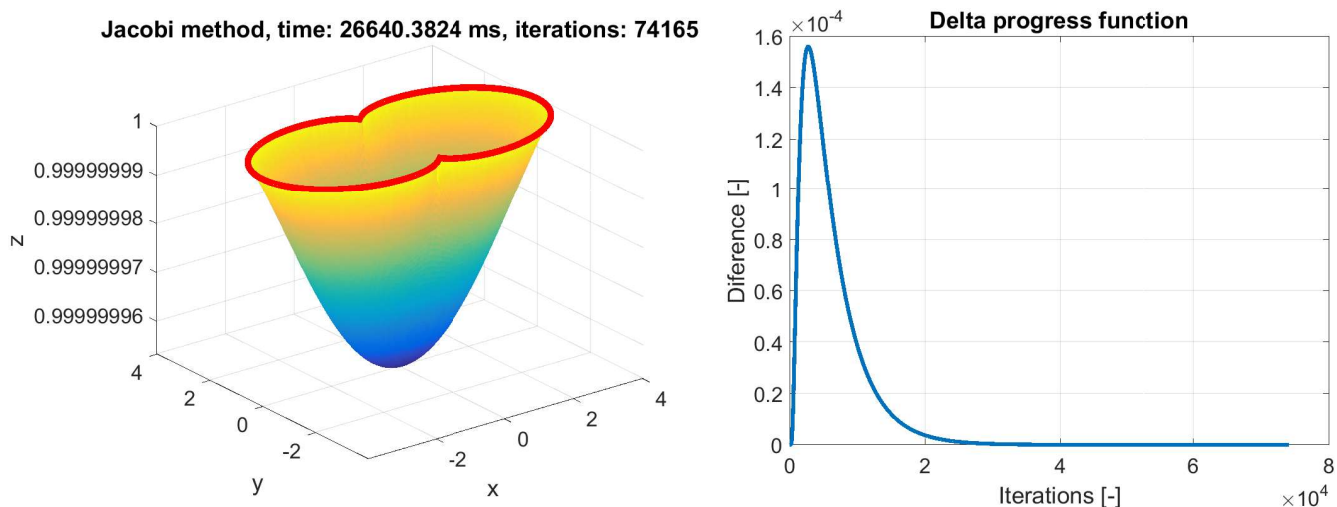


Obrázek 7: Jacobiho metoda - výsledná plocha na levém a průběh progressu během iterací na pravém obrázku

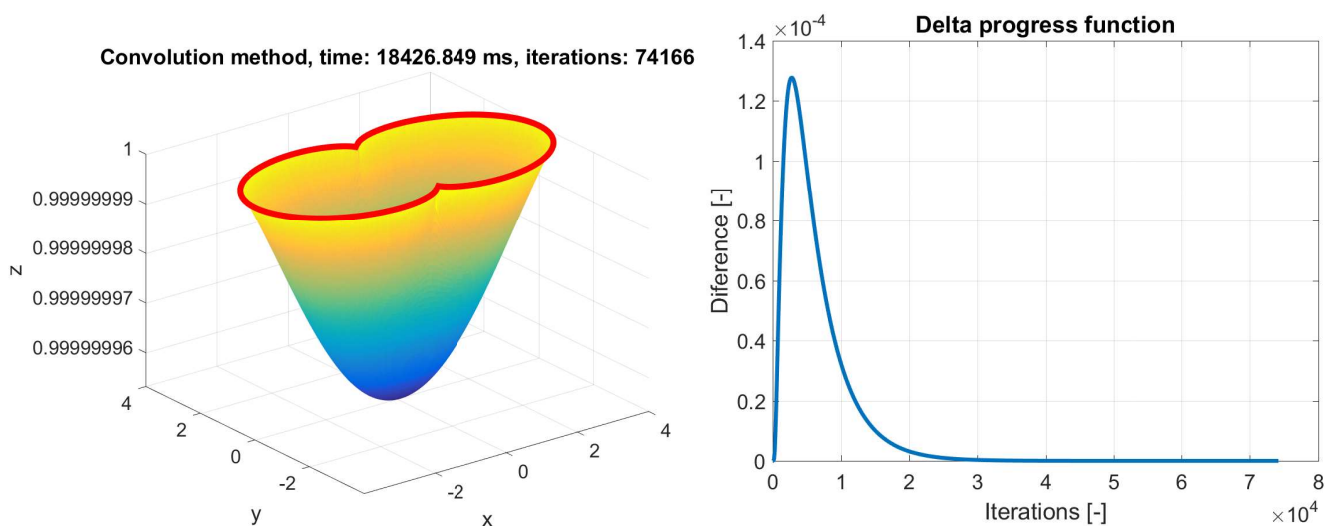


Obrázek 8: konvoluční metoda - výsledná plocha na levém a průběh progressu během iterací na pravém obrázku

4.4 Nephroid

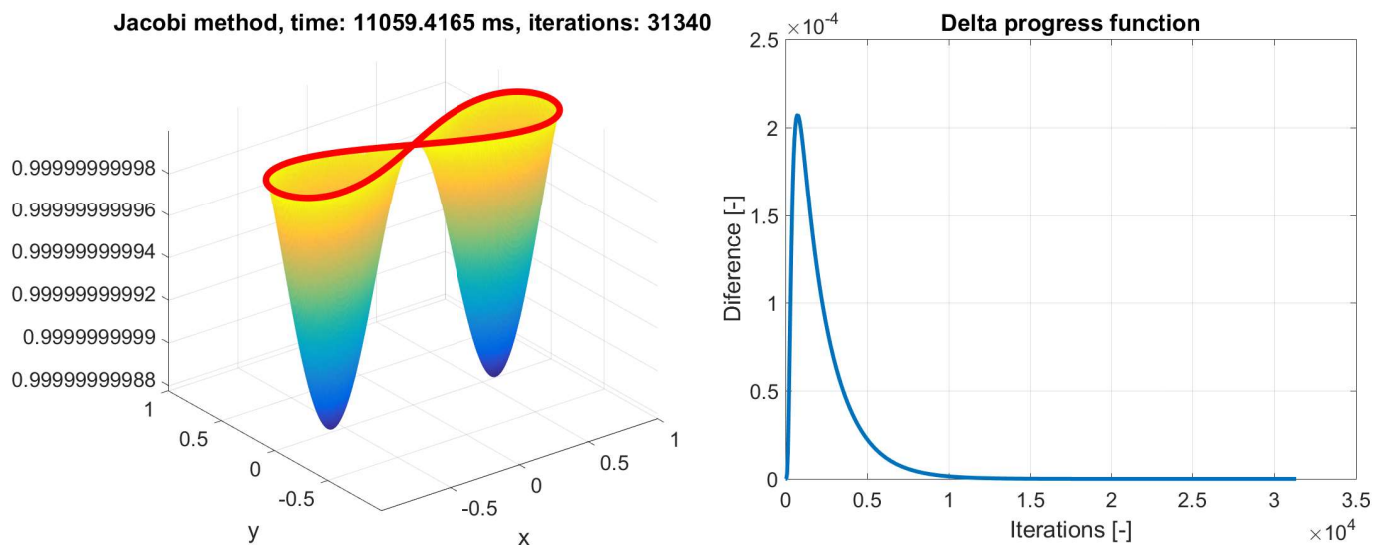


Obrázek 9: Jacobiho metoda - výsledná plocha na levém a průběh progressu během iterací na pravém obrázku

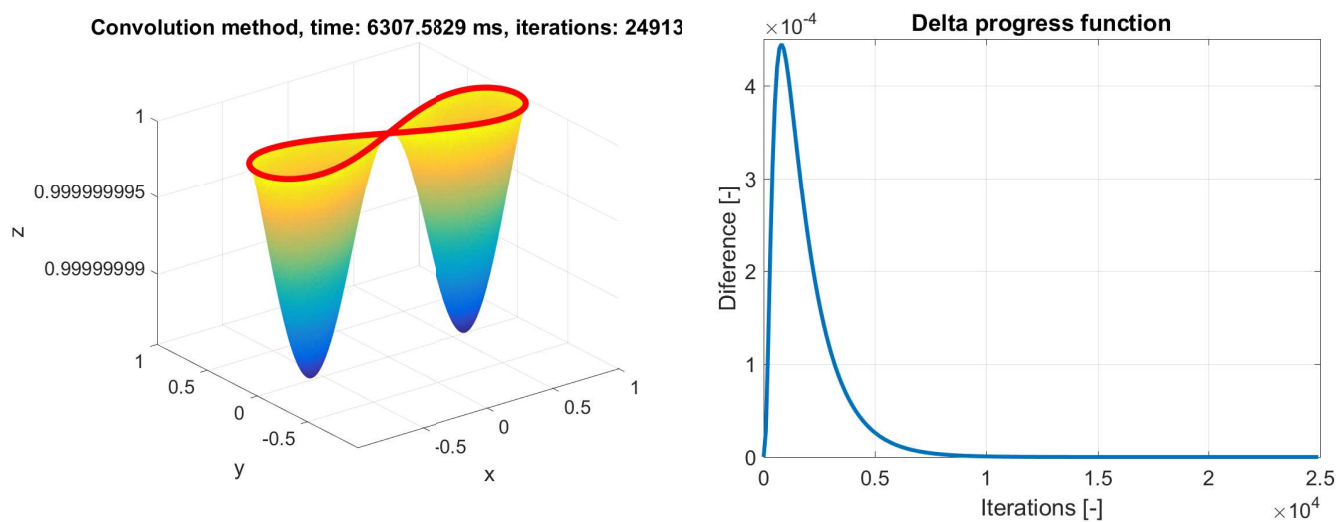


Obrázek 10: konvoluční metoda - výsledná plocha na levém a průběh progressu během iterací na pravém obrázku

4.5 Symbol nekonečna

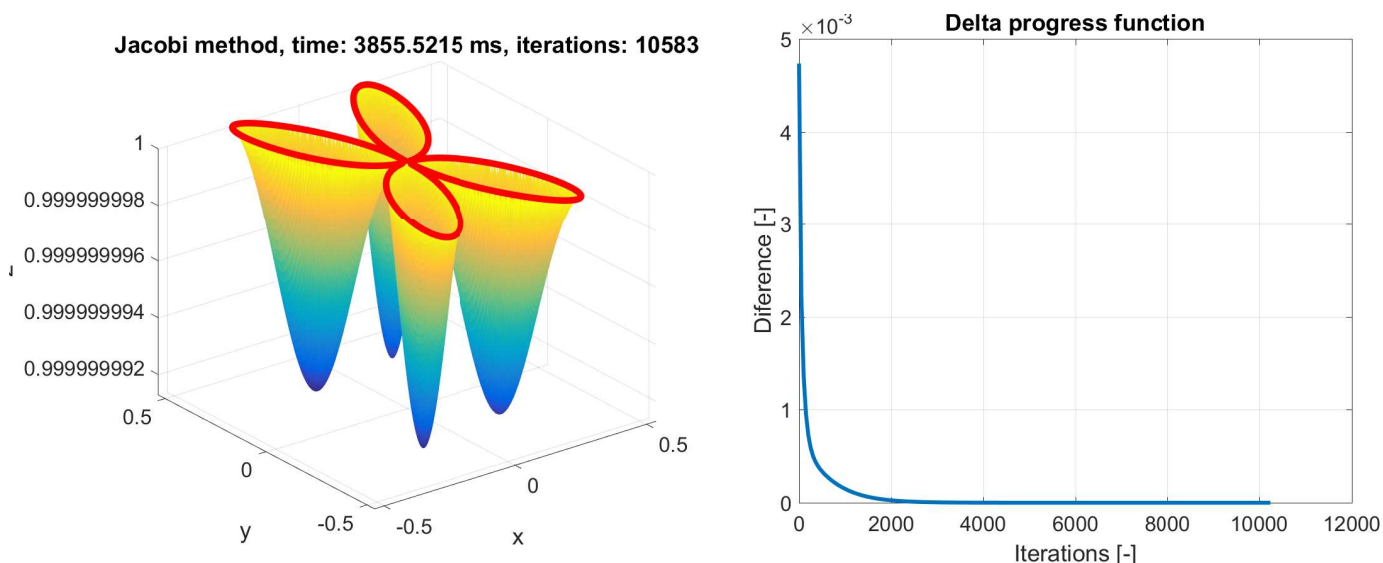


Obrázek 11: Jacobiho metoda - výsledná plocha na levém a průběh progressu během iterací na pravém obrázku

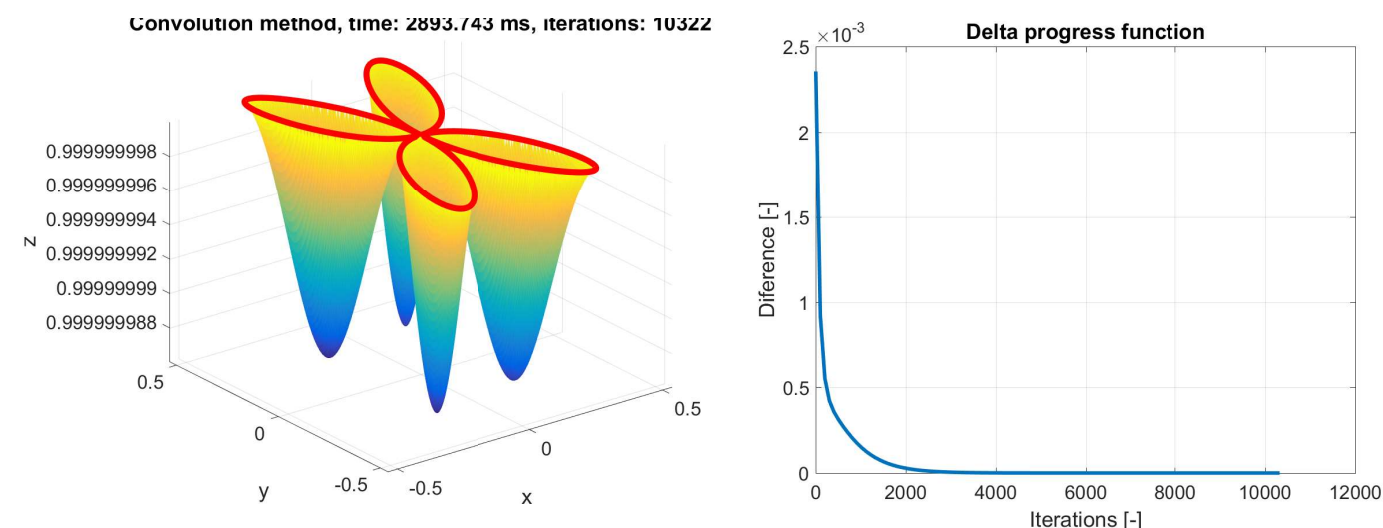


Obrázek 12: konvoluční metoda - výsledná plocha na levém a průběh progressu během iterací na pravém obrázku

4.6 Další křivka



Obrázek 13: Jacobiho metoda - výsledná plocha na levém a průběh progressu během iterací na pravém obrázku



Obrázek 14: konvoluční metoda - výsledná plocha na levém a průběh progressu během iterací na pravém obrázku

5 Závěr

Naše aplikace Jacobiho metody řeší laplaceovu rovnici pomocí jejího přepisu konečnými diferencemi. Serializací pracovního prostoru jsme problém převedli na řešení soustavy lineárních rovnic, která je vždy vyřešena (s právě jedním řešením) během každé iterace. Pro tento postup je nutné generování velké matice A, což je v mnoha ohledech nevýhodné.

Necméně není pravděpodobné, že by naprogramovaný algoritmus mohl být použit v neakademickém prostředí, protože je na výpočet pořád velmi časově náročný a při velkém rastru nepoužitelný.

Na YOUTUBE bylo nahráno [video](#), kde je vidět průběh (s kroky po 100 iteracích) vývoje zadané křivky až do relativní přesnosti $\varepsilon < 10^{-5}$ v bodě $[0, 0]$.