Numerické metódy v EM
Pokus o vyrobenie knižky z Jupyter-Notebookov

Peter Fabo

17. februára 2019

Obsah

Kapitola 1

Úvod

Notebook je založený na dokumente John B. Schneider: Understanding the Finite-Difference Time-Domain Method, z ktorého je prevzaté členenie notebooku a odvodenie fundamenetálnych vzťahov. Implementácia príkladov je v Pythone a celý koncept je doplnený o príklady použitia simulátora MEEP.

1.1 Sofware

- jupyter notebook s LaTex_envs rozšírením
- python 3.x
- matplotlib
- numpy, scipy
- meep
- paraview

1.2 Použité symboly

- a, b, ... - skalárne premenné a konštanty
- f, $f()$, $f(x,y)$, F_x, ... - skalárne funkcie
- $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ - jednotkové vektory v karteziánskej súradnicovej sústave
- $\mathbf{F}$, $\mathbf{F}(\mathbf{r})$, $\mathbf{F}(x,y)$ ... - vektorové veličiny

1.3 Skratky

```
FDTD
PMC
PEC
ABC - Absorbing Boundary Conditions
```

1.4 Literatúra

https://github.com/john-b-schneider/uFDTD

Kapitola 2

Prehľad pojmov vektorovej analýzy

2.1 Skalárne pole

V matematike a fyzike je skalárne pole funkciou priraďujúce skalárnu hodnotu (t.j. určenú jediným číselným údajom) každému bodu priestoru, napríklad teplota, hustota nebo vlhkost vzduchu. Príklad definície skalárnej funkcia v 2D

$$f(x, y) = \sin(x) * y^2 + \cos(y)$$

2.2 Vektorové pole

Vektorové pole je funkcia priraďujúca každému bodu priestoru vektor (t.j. veličinu, ktorá má veľkosť a smer). Vo fyzike sa vektorové pole používa na popis toho, ako sa daná vektorová veličina mení v priestore od bodu k bodu, napríklad rýchlosť kvapaliny, elektrické alebo magnetické pole. Príklad definície vektorovej funkcie v 2D (zložky vektorov sú definované pomocou skalárnych funkcií)

$$\mathbf{F}(x, y) = F_x(x, y)\,\mathbf{i} + F_y(x, y)\,\mathbf{j} = (x^2 - y)\,\mathbf{i} + (\sin(x) + y^2)\,\mathbf{j}$$

2.3 Operátor ∇

Nabla operátor je vektorový operátor, ktorý zavedol v roku 1837 írsky matematik a fyzik William Rowan Hamilton. V karteziánskych súradniciach je operátor formálne definovaný ako vektor

$$\nabla \equiv \frac{\partial}{\partial x}\,\mathbf{i} + \frac{\partial}{\partial y}\,\mathbf{j} + \frac{\partial}{\partial z}\,\mathbf{k}$$

Transformáciou súradníc je možné získať operátor ∇ v iných súradnicových sústavách (cylindrickej, sférickej).

2.4 Gradient

Uplatnením operátora ∇ na skalárne pole $f(x, y)$ dostaneme vektorové pole, ktoré popisuje zmeny strmosti alebo poklesu skalárnej funkcie.

$$\text{grad } f(x, y, z) = \nabla f(x, y, z) = \frac{\partial f}{\partial x}\,\mathbf{i} + \frac{\partial f}{\partial y}\,\mathbf{j} + \frac{\partial f}{\partial z}\,\mathbf{k}$$

2.5 Divergencia

Skalárnym súčin operátora ∇ s vektorovým poľom dostaneme skalárne pole, ktoré popisuje žriedlovosť vektorového poľa. Ak v danom bode priestoru pole vzniká, divergencia má kladnú hodnotu, ak zaniká má hodnotu zápornú.

$$div\,\mathbf{F}(x, y, z) = \nabla \cdot \mathbf{F}(x, y, z) = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}$$

2.6 Rotácia

Vektorovým súčinom operátora ∇ s vektorovým poľom dostaneme vektorové pole, ktoré popisuje mieru vírovosti vektorového poľa, t.j. do akej miery sa v danom bode priestoru pole otáča alebo rotuje. Vektor rotácie v danom bode je v smere osi otáčania, jeho orientácia vzhľadom k rovine otáčania určíme pomocou pravidla pravej ruky.

$$curl\,\mathbf{F}(x,y,z) = \nabla \times \mathbf{F}(x,y,z) = \left(\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right)\mathbf{i} + \left(\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}\right)\mathbf{j} + \left(\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right)\mathbf{k}$$

2.7 Gaussova veta

Gaussova veta (veta o divergencii) dáva do súvislosti tok vektorového poľa $\mathbf{F}$ uzatvorenou hladkou plochou S tvoriacou povrch nejakého objemu V s integrálom cez divergenciu pola v tomto objeme

$$\oint_S \mathbf{F}\,\mathrm{d}\mathbf{S} = \int_V \nabla \cdot \mathbf{F}\,\mathrm{d}V$$

Ak sa v objeme V, ktorý je ohraničený plochou S vyskytuje zdroj (source) vytvárajúci vektorové pole $\mathbf{F}$, potom integrál toku cez polochu je kladný, bez zdroja nulový a pri vtoku (sinks) je záporný.

2.8 Stokesova veta

Stokesova veta dáva do súvisu integrál po uzatvorenej krivke C vo vektorovom poli $\mathbf{F}$ s integrálom rotácie poľa cez plochu, ktorá je touto krivkou ohraničená

$$\oint_C \mathbf{F} \cdot \mathrm{d}\mathbf{l} = \int_S (\nabla \times \mathbf{F}) \cdot \mathrm{d}\mathbf{s}$$

Stokesova veta v dvojrozmernom priestore sa nazýva **Greenova veta**.

2.9 Laplacián

2.10 Vlnová rovnica

Kapitola 3

Skalárne pole

3.1 Vlastnosti

V matematike a fyzike je skalárne pole funkciou $f(x, y, z)$ priraďujúce skalárnu hodnotu (t.j. určenú jediným číselným údajom) v každom bode priestoru (x, y, z). Hodnoty skalárneho poľa môžu byť reálne alebo komplexné. Príkladom skalárneho poľa je napríklad teplota, hustota nebo vlhkost vzduchu, v elektrotechnike potenciál.

3.2 Numerický výpočet

Pre vytvorenie skalárneho poľa musíme v *numpy* vytvoriť pole (2D alebo 3D), nad ktorým skalárne pole zadefinujeme.

```
dx = np.linspace(-1.0, 1.0, 30)
dy = np.linspace(-1.0, 1.0, 30)
x,y = np.meshgrid(dx, dy)
```

Funkcia *meshgrid()* vygeneruje polia x,y, tieto obsahujú hodnoty, ktoré reprezentujú mriežku hodnôt $(x_0, y_0), (x_0, y_1) \ldots$. V nasledujúcom príklade vytvorme a zobrazme 2D skalárne pole

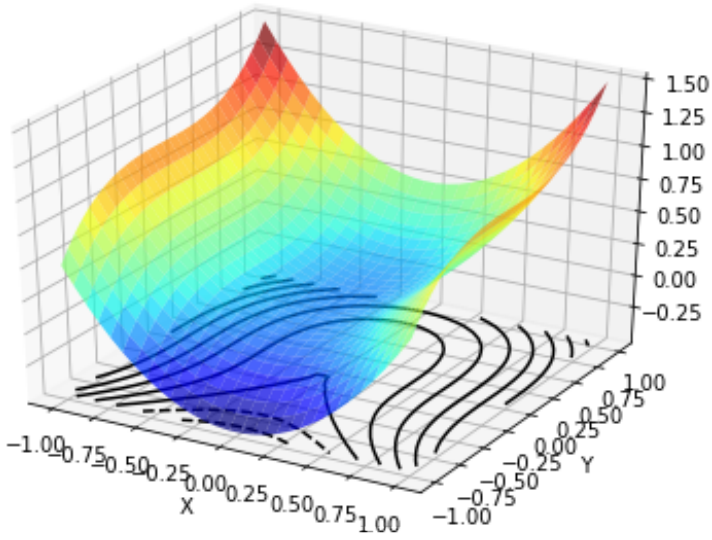
$$f(x, y) = x^2 + \frac{y^3}{2}$$

```
%reset -f
%matplotlib inline
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

dx = np.linspace(-1.0, 1.0, 30)      # rozsahy suradnic
dy = np.linspace(-1.0, 1.0, 30)
x,y = np.meshgrid(dx, dy)            # polia suradnic (xi, yi) ...

def f(x,y):                          # defincia funkcie 2D skalarne pole
    return (x**2 + y**3/2)
```

Pre vizualizáciu skalárneho poľa môžeme použiť štandardné metódy z knižnice *matplotlib*.

```
fig = plt.figure(figsize=(7, 5))           # zobrazenie skalarneho pola
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, f(x,y), alpha=0.7, cmap='jet')
ax.contour(x, y, f(x,y), 10, colors="k", offset=-0.50)
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

Poznámka: 3D vizualizácia zobrazuje hodnoty poľa nad súradnicami v 2D.


1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
X
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Y

Kapitola 4

Vektorové pole

4.1 Vlastnosti

Vektorové pole je v matematike a fyzike funkcia priraďujúca každému bodu priestoru vektor (t.j. veličinu, ktorá má veľkosť a smer). Vo fyzike sa vektorové pole používa na popis toho, ako sa daná vektorová veličina mení v priestore od bodu k bodu, napr. rýchlosť kvapaliny, elektrické alebo magnetické pole.
V 3D priestore je vektorové pole definované ako

$$\mathbf{F}(x,y,z) = F_x(x,y,z)\,\mathbf{i} + F_y(x,y,z)\,\mathbf{j} + F_y(x,y,z)\,\mathbf{k} \tag{4.1}$$

4.2 Numerický výpočet

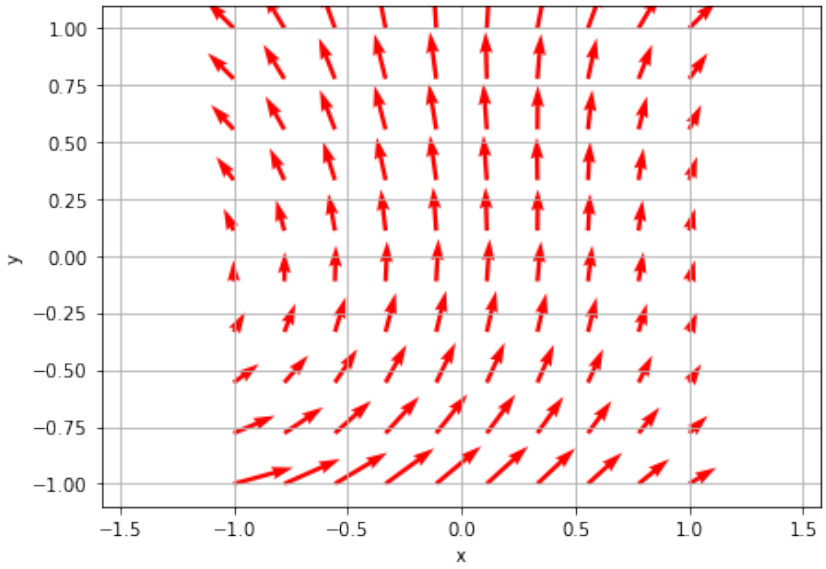
Vektorové pole $\mathbf{F}(x,y)$ v 2D môžeme zadefinovať pomocou skalárnych funkcií $F_x(x,y), F_y(x,y)$ pre jednotlivé zložky vektorov poľa

$$\mathbf{F}(x,y) = F_x(x,y)\,\mathbf{i} + F_y(x,y)\,\mathbf{j} = \left(-y\,(1-x-y) + \frac{x}{4}\right)\,\mathbf{i} + \left(2 - x^2 + \frac{y}{4}\right)\,\mathbf{j}$$

```
%reset -f
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt

dx = np.linspace(-1.0, 1.0, 10)              # rozsahy suradnic
dy = np.linspace(-1.0, 1.0, 10)
x, y = np.meshgrid(dx, dy)                   # definicny obor vektoroveho pola

[Fx, Fy] = [ -y*(1-x-y)+ x/4, 2-x**2+y/4 ]   # zlozky vektoroveho pola
```

Pre zobrazenie vektorového poľa môžeme použiť štandardné metódy z knižnice *matplotlib*

```
fig = plt.figure(figsize=(7, 5))
ax = fig.add_subplot(111)
ax.quiver(dx, dy, Fx, Fy, color='r')
plt.xlabel('x')
plt.ylabel('y')
plt.axis('equal')
plt.grid()
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x

Kapitola 5

Gradient

Gradient skalárneho pola $f(x, y, z)$ vo zvolenom bode (x, y, z) je definovaný ako vektor, udávajúci smer najväčšej zmeny skalárneho poľa a jeho veľkosť je úmerná tejto zmene. Všeobecný vzťah pre výpočet gradientu v kartézskych súradniciach je definovaný ako

$$\nabla f = \frac{\partial f}{\partial x}\mathbf{i} + \frac{\partial f}{\partial y}\mathbf{j} + \frac{\partial f}{\partial z}\mathbf{k} \tag{5.1}$$

5.1 Gradient v cylindrických súradniciach

$$\nabla f = \frac{\partial f}{\partial r}\boldsymbol{r} + \frac{1}{r}\frac{\partial f}{\partial \varphi}\boldsymbol{\varphi} + \frac{\partial f}{\partial z}\boldsymbol{z} \tag{5.2}$$

5.2 Gradient v sférických súradniciach

$$\nabla f = \frac{\partial f}{\partial r}\boldsymbol{r} + \frac{1}{r}\frac{\partial f}{\partial \theta}\boldsymbol{\theta} + \frac{1}{r\sin\theta}\frac{\partial f}{\partial \varphi}\boldsymbol{\varphi} \tag{5.3}$$

5.3 Vlastnosti gradientu

Ak $\mathbf{F}, \mathbf{G}$ sú vektorové polia, f, g sú skalárne polia, a, b sú reálne čísla, potom operátor gradientu spĺňa nasledujúce identity:

- je lineárny voči reálnym číslam

$$\nabla\left(af + bg\right) = a\nabla f + b\nabla g$$

- spĺňa Leibnizovo pravidlo (derivácia súčinu)

$$\nabla\left(fg\right) = \left(\nabla f\right)g + f\nabla g$$

- gradient skalárneho súčinu vektorov

$$\nabla\left(\mathbf{F}\cdot\mathbf{G}\right) = \nabla\mathbf{F}\cdot\mathbf{G} + \nabla\mathbf{G}\cdot\mathbf{F}$$

kde pod $\nabla\mathbf{F}$ rozumieme maticu a výsledok ako stĺpcový vektor (**TODO** dokázať)

5.4 Symbolický výpočet

Pomocou knižnice *sympy* určime gradient skalárnej funkcie

$$f(x,y) = (x^2 + y)\,\sin(y)$$

a jeho hodnotu v bode $(x,y) = (2,4)$.

```
%reset -f
from utils.utils import *
from sympy import *
from sympy.vector import CoordSys3D, gradient

r = CoordSys3D('r')
f = (r.x**2 + r.y) * sin(r.y)        # skalarna funkcia f(x,y)
G = gradient(f)                      # vypocet gradientu

ltxprint(r'\nabla f(x,y)', G)
```

$$\nabla f(x,y) = (2\mathbf{x_r}\sin(\mathbf{y_r}))\hat{\mathbf{i}}_\mathbf{r} + ((\mathbf{x_r}^2 + \mathbf{y_r})\cos(\mathbf{y_r}) + \sin(\mathbf{y_r}))\hat{\mathbf{j}}_\mathbf{r}$$

Výpočet hodnôt zložiek vektora gradientu v bode $(x,y) = (2,4)$

```
gxy = G.subs([(r.x, 2), (r.y, 4)])
print('Grad f(2,4) = {0} {1}'.format(gxy, gxy.evalf(3)))
```

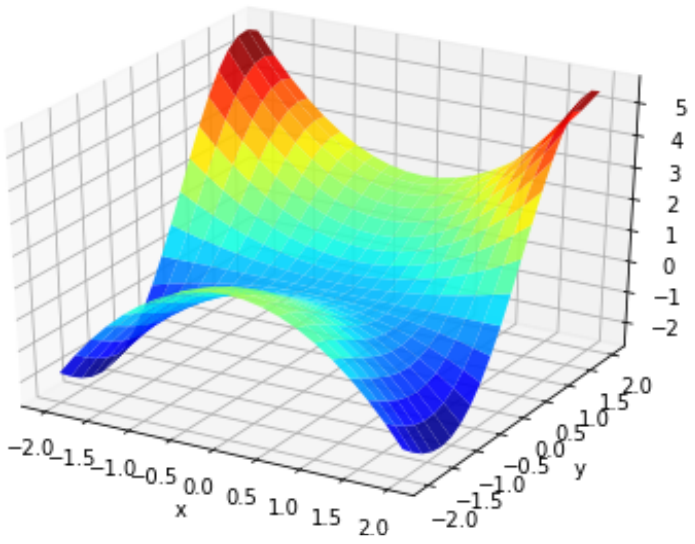
```
Grad f(2,4) = (4*sin(4))*r.i + (8*cos(4) + sin(4))*r.j (-3.03)*r.i + (-5.99)*r.j
```

5.5 Numerický výpočet

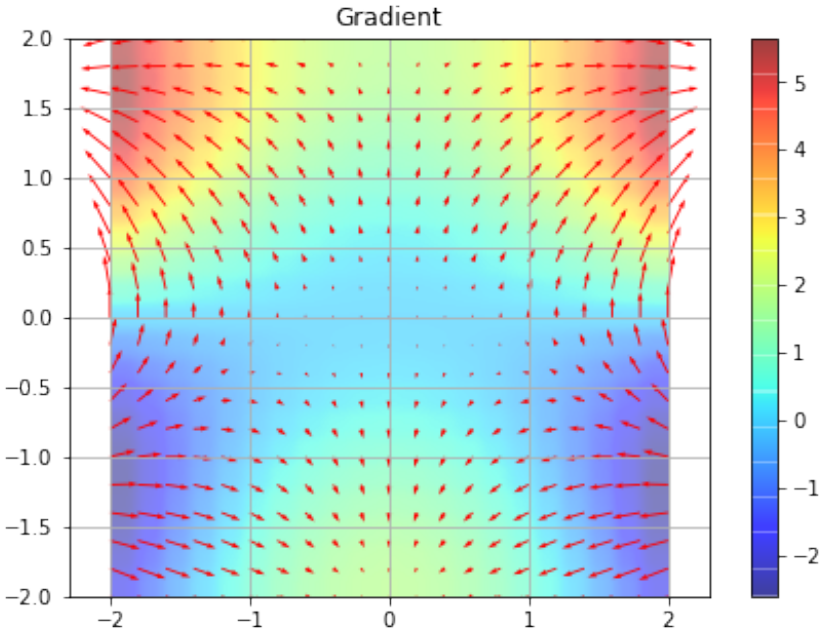
V *numpy* je možné pre výpočet gradientu n-rozmernej skalárnej funkcie použiť funkciu *gradient()*. Funkcia vráti vektorové pole, ktoré môžeme v 2D zobraziť pomocou funkcie *quiver()*, šipky smerujú v smere stúpajúcej hodnoty.

```
%reset -f
%matplotlib inline
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
dx = np.arange(-2, 2.2, .2)
dy = np.arange(-2, 2.2, .2)
x,y = np.meshgrid(dx, dy)

fn = (x**2 + y) * np.sin(y)
gy, gx = np.gradient(fn)

fig = plt.figure(figsize=(7,5))          ax = fig.add_subplot(111, projection='3d
    ')
ax.plot_surface(x, y, fn, alpha=0.9, cmap='jet')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

```
fig = plt.figure(figsize=(7,5))
ax = fig.add_subplot(111)
ax.quiver(x, y, gx, gy, color='r')
img = ax.imshow(fn, extent=[-2, 2, -2, 2], origin='lower', interpolation='kaiser
    ',
                    clip_on=True, alpha=0.5, cmap='jet')
fig.colorbar(img)
```




```
plt.axis('equal')
plt.title('Gradient')
plt.grid()
plt.show()
```



Kapitola 6

Divergencia

Divergencia je skalárna veličina, ktorá popisuje žriedlovosť vektorového pola, t.j. či do zvoleného bodu pole vteká (sinks) alebo z neho vyteká (source).
Divergencia vektorového poľa $\mathbf{F}$ definujeme ako

$$div\,\mathbf{F} = \nabla.\mathbf{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z} \tag{6.1}$$

6.1 Divergencia v cylindrických súradniciach

$$\operatorname{div}\mathbf{F} = \nabla\cdot\mathbf{F} = \frac{1}{r}\frac{\partial}{\partial r}\left(rF_r\right) + \frac{1}{r}\frac{\partial F_\theta}{\partial\theta} + \frac{\partial F_z}{\partial z}$$

6.2 Divergencia vo sférických súradniciach

$$\operatorname{div}\mathbf{F} = \nabla\cdot\mathbf{F} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2F_r\right) + \frac{1}{r\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta\,F_\theta) + \frac{1}{r\sin\theta}\frac{\partial F_\varphi}{\partial\varphi}$$

6.3 Vlastnosti divergencie

Ak $\mathbf{F}, \mathbf{G}$ sú vektorové polia, φ je skalárne pole, a, b sú reálne čísla, potom operátor divergencie spĺňa nasledujúce identity:

$$\operatorname{div}(a\mathbf{F} + b\mathbf{G}) = a\operatorname{div}\mathbf{F} + b\operatorname{div}\mathbf{G}$$

- divergencia súčiny skalárneho a vektorvého poľa

$$\operatorname{div}(\varphi\mathbf{F}) = \operatorname{grad}\varphi\cdot\mathbf{F} + \varphi\operatorname{div}\mathbf{F}$$

$$\nabla\cdot(\varphi\mathbf{F}) = (\nabla\varphi)\cdot\mathbf{F} + \varphi(\nabla\cdot\mathbf{F})$$

- divergencia vektorového súčinu dvoch vektorvých polá

$$\operatorname{div}(\mathbf{F}\times\mathbf{G}) = \operatorname{curl}\mathbf{F}\cdot\mathbf{G} - \mathbf{F}\cdot\operatorname{curl}\mathbf{G}$$

$$\nabla\cdot(\mathbf{F}\times\mathbf{G}) = (\nabla\times\mathbf{F})\cdot\mathbf{G} - \mathbf{F}\cdot(\nabla\times\mathbf{G})$$

- divergencia gradientu je Laplacián

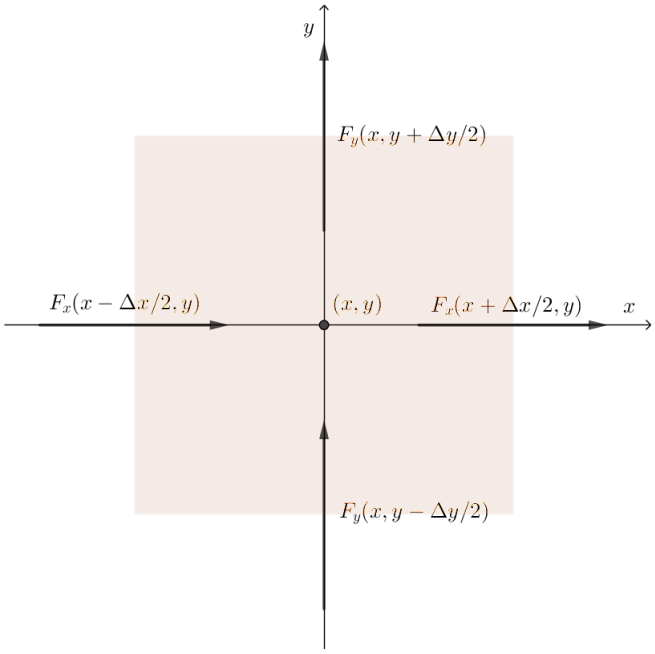
$$\operatorname{div}(\nabla\varphi) = \Delta\varphi$$

- divergencia rotácie vektorového poľa je nulová

$$\nabla \cdot (\nabla \times \mathbf{F}) = 0$$

6.4 Diskrétna aproximácia divergencie v 2D

Diskrétnu aproximáciu divergencie v rovine (x, y) môžeme odvodiť zo zložiek vektorového poľa na hranách malej plošky o rozmere $dx \times dy$. Predpokladáme, že hodnoty zložiek poľa sú konštantné pozdĺž hrán plošky.



Obr. 6.1: Divergencia v x-y rovine

Pre divergenciu potom môžeme približne písať (pre $\Delta x, \Delta y \to 0$)

$$\frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} \approx \frac{F_x(x + \frac{\Delta x}{2}, y) - F_x(x - \frac{\Delta x}{2}, y)}{\Delta x} + \frac{F_y(x, y + \frac{\Delta y}{2}) - F_y(x, y - \frac{\Delta y}{2})}{\Delta y}$$

Pre $\Delta x = \Delta y = \delta$ môžeme vzťah upraviť

$$\frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} \approx \frac{1}{\delta}\big(F_x(x + \frac{\delta}{2}, y) - F_x(x - \frac{\delta}{2}, y) + F_y(x, y + \frac{\delta}{2}) - F_y(x, y - \frac{\delta}{2})\big) \tag{6.2}$$

6.5 Symbolický výpočet

Pomocou symbolických manipulácií *sympy* určime divergenciu vektorového poľa v 2D rovine

$$\mathbf{F}(x, y) = \mathbf{i}\left(-y\,(1 - x - y) + \frac{x}{4}\right) + \mathbf{j}\left(2 - x^2 + \frac{y}{4}\right)$$

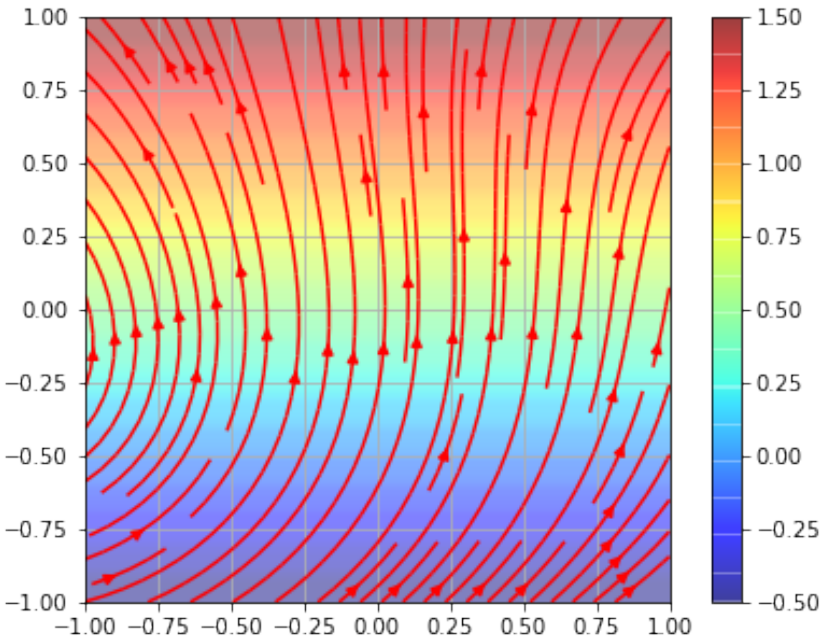
```
%reset -f
from utils.utils import *
from sympy import *
from sympy.vector import CoordSys3D, divergence

r = CoordSys3D('r')                    # suradnicova sustava
F = (-r.y*(1 - r.x - r.y) + r.x/4)*r.i + (2 - r.x**2 + r.y/4)*r.j
D = divergence(F)

ltxprint(r'\nabla \cdot \mathbf{F}', D)
```

$$\nabla \cdot \mathbf{F} = \mathbf{y_r} + \frac{1}{2}$$

6.6 Numerický výpočet

S divergenciou môžeme pracovať ako so štandardným skalárnym poľom. V našom prípade výsledok symbolického výpočtu divergencie v *sympy* zobrazíme pomocou knižníc *matplotlib* a *numpu* formálnou substitúciou premenných a vyhodnotením reťazcov.

```
# formalna substitucia premennych, x,y - nie su premenne v sympy, ale retazce
Fx = F.components[r.i].subs([(r.x, 'x'), (r.y, 'y')])              Fy = F.
    components[r.j].subs([(r.x, 'x'), (r.y, 'y')])
Ds = D.subs([(r.x, 'x'), (r.y, 'y')])

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

dx = np.linspace(-1.0, 1.0, 20)     # rozsahy suradnic
dy = np.linspace(-1.0, 1.0, 20)
x, y = np.meshgrid(dx, dy)

a = eval(str(Fx))                   # numericky vypocet zloziek
b = eval(str(Fy))                   # vyhodnotenie retazcov - konverzia zo sympy
d = eval(str(Ds))

fig = plt.figure(figsize=(7, 5))
ax = fig.add_subplot(111)
ax.streamplot(x, y, a, b, color='r')
img = ax.imshow(d, extent=[-1, 1, -1, 1], origin='lower', interpolation='kaiser'
    ,
                  clip_on=True, alpha=0.5, cmap='jet')
fig.colorbar(img)
plt.grid()
plt.show()
```



Kapitola 7

Rotácia

Vektorovým súčinom operátora ∇ s vektorovým poľom dostaneme vektorové pole, ktoré popisuje mieru vírovosti vektorového poľa, t.j. do akej miery sa v danom bode priestoru pole otáča alebo rotuje. Vektor rotácie v danom bode je v smere osi otáčania, jeho orientácia vzhľadom k rovine otáčania určíme pomocou pravidla pravej ruky.

$$\operatorname{curl}\mathbf{F} = \nabla \times \mathbf{F} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \dfrac{\partial}{\partial x} & \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial z} \\ F_x & F_y & F_z \end{vmatrix} = \left(\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right)\mathbf{i} + \left(\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}\right)\mathbf{j} + \left(\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right)\mathbf{k} \tag{7.1}$$

7.1 Rotácia v cylindrických súradniciach

$$\nabla \times \mathbf{F} = \left(\frac{1}{r}\frac{\partial F_z}{\partial \varphi} - \frac{\partial F_\varphi}{\partial z}\right)\mathbf{r} + \left(\frac{\partial F_r}{\partial z} - \frac{\partial F_z}{\partial r}\right)\boldsymbol{\varphi} + \frac{1}{r}\left(\frac{\partial (rF_\varphi)}{\partial r} - \frac{\partial F_r}{\partial \varphi}\right)\mathbf{z} \tag{7.2}$$

7.2 Rotácia v sférických súradniciach

$$\nabla \times \mathbf{F} = \frac{1}{r\sin\theta}\left(\frac{\partial}{\partial \theta}(F_\varphi \sin\theta) - \frac{\partial F_\theta}{\partial \varphi}\right)\mathbf{r} + \frac{1}{r}\left(\frac{1}{\sin\theta}\frac{\partial F_r}{\partial \varphi} - \frac{\partial}{\partial r}(rF_\varphi)\right)\boldsymbol{\theta} + \frac{1}{r}\left(\frac{\partial}{\partial r}(rF_\theta) - \frac{\partial F_r}{\partial \theta}\right)\boldsymbol{\varphi} \tag{7.3}$$

7.3 Vlastnosti rotácie

Ak $\mathbf{F}, \mathbf{G}$ sú vektorové polia, f je skalárne pole, a, b sú reálne čísla, potom operátor rotácie spĺňa nasledujúce identity:

- lineárita rotácie voči reálnym čislam

$$\nabla \times (a\,\mathbf{F} + b\,\mathbf{G}) = a\nabla \times \mathbf{F} + b\nabla \times \mathbf{G}$$

- rotácia gradientu je nulový vektor

$$\nabla \times \nabla f = \operatorname{curl}\operatorname{grad} f = \mathbf{0}$$

- rotácia vektorvého poľa násobeného skalárnou funkciou

$$\nabla \times (f\mathbf{F}) = \nabla f \times \mathbf{F} + f\nabla \times \mathbf{F}$$

- rotácia z vektorvého súčinu dvoch vektorových polí

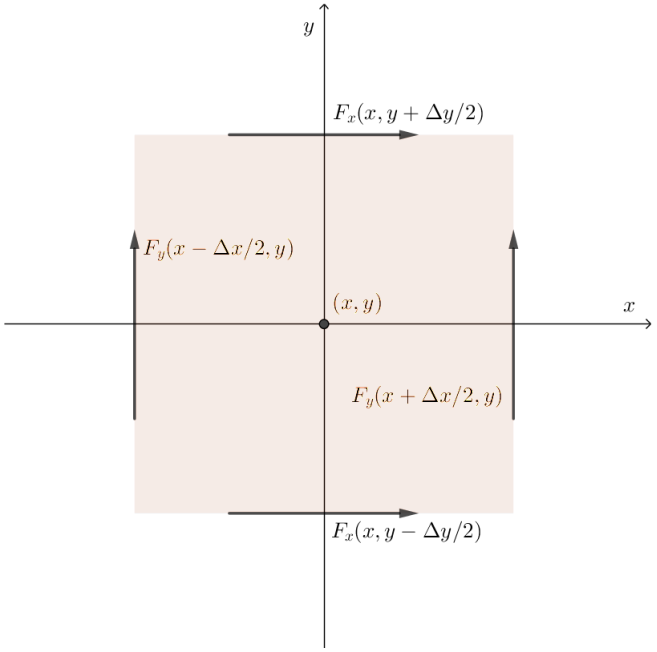
$$\nabla \times (\mathbf{F} \times \mathbf{G}) = (\mathbf{G} \cdot \nabla)\,\mathbf{F} - (\mathbf{F} \cdot \nabla)\,\mathbf{G} + \mathbf{F}\,(\nabla \cdot \mathbf{G}) - \mathbf{G}\,(\nabla \cdot \mathbf{F})$$

- rotácia z rotácie vektorvého poľa

$$\nabla \times (\nabla \times \mathbf{F}) = \nabla\,(\nabla \cdot \mathbf{F}) - \Delta \mathbf{F}$$

7.4 Diskrétna aproximácia rotácie v 2D

Rotácia v (x, y) rovine má len zložku v smere osi z.



Obr. 7.1: Rotácia v x-y rovine

$$\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y} \approx \frac{F_y(x + \frac{\Delta x}{2}, y) - F_y(x - \frac{\Delta x}{2}, y)}{\Delta x} - \frac{F_x(x, y + \frac{\Delta y}{2}) - F_x(x, y - \frac{\Delta y}{2})}{\Delta y}$$

Pre $\Delta x = \Delta y = \delta$ môžeme vzťah upraviť

$$\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y} \approx \frac{1}{\delta}\big(F_y(x + \frac{\delta}{2}, y) - F_y(x - \frac{\delta}{2}, y) - F_x(x, y + \frac{\delta}{2}) - F_x(x, y - \frac{\delta}{2})\big) \tag{7.4}$$

7.5 Symbolický výpočet

Pomocou symbolických manipulácií *sympy* určime rotáciu vektorového poľa v 2D rovine

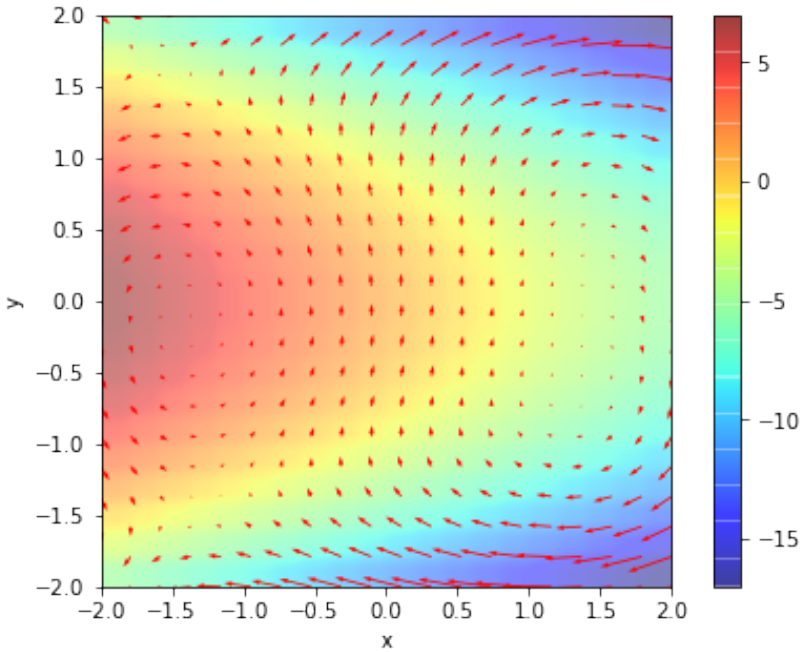
$$\mathbf{F}(x, y) = \left(-y\,(1 - x - y) + \frac{x}{4}\right)\,\mathbf{i} + \left(2 - x^2 + \frac{y}{4}\right)\,\mathbf{j}$$

```
%reset -f
from utils.utils import *
from sympy.vector import CoordSys3D, curl
r = CoordSys3D('R')
F = (-r.y*(1 - r.x - r.y**2) + r.x/4)*r.i + (2 - r.x**2 + r.y/4)*r.j
C = curl(F)

ltxprint(r'\nabla \times \mathbf{F}', C)
```

$$\nabla \times \mathbf{F} = (-3\mathbf{x_R} - 3\mathbf{y_R}^2 + 1)\hat{\mathbf{k}}_\mathbf{R}$$

7.6 Numerický výpočet

```
%matplotlib inline
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

Fx = F.components[r.i].subs([(r.x, 'x'), (r.y, 'y')])
Fy = F.components[r.j].subs([(r.x, 'x'), (r.y, 'y')])
Cs = C.subs([(r.x, 'x'), (r.y, 'y'), (r.k, 1)])

dx = np.linspace(-2.0, 2.0, 20)
dy = np.linspace(-2.0, 2.0, 20)
x,y = np.meshgrid(dx, dy)

c = eval(str(Cs))
fx = eval(str(Fx))
fy = eval(str(Fy))

fig = plt.figure(figsize=(7, 5))
ax = fig.add_subplot(111)
img = ax.imshow(c, extent=[-2, 2, -2, 2], origin='lower', interpolation='kaiser'
    ,
                clip_on=True, alpha=0.5, cmap='jet')
fig.colorbar(img)
ax.quiver(x, y, fx, fy, color='r')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Kapitola 8

Gaussova veta

Gaussova veta (Gauss-Ostrogradskeho veta alebo veta o divergencii) dáva do súvislosti tok vektorového poľa $\mathbf{F}$ plochou S tvoriacou povrch nejakého objemu V so zdrojom poľa uzatvoreným v tomto objeme.

$$\int_V \nabla \cdot \mathbf{F}\, dV = \oint_S \mathbf{F}\, d\mathbf{S} \tag{8.1}$$

kde

$$\mathbf{F}(x,y,z) = F_x(x,y,z)\,\mathbf{i} + F_x(x,y,z)\,\mathbf{j} + F_x(x,y,z)\,\mathbf{k}$$

8.1 Gaussova veta v 3D

Objem V rozdelíme na kocky s rozmermi hrán $\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z$. Orientovaný element plochy $\mathrm{d}\mathbf{S}$ má smer kolmý na plochu a veľkosť rovnú veľkosti tejto plochy

$$\mathrm{d}\mathbf{S} = \mathbf{n}\,\mathrm{d}S, \quad \mathbf{n} \perp S$$

$$\begin{aligned}
\mathrm{d}\mathbf{S_x} &= \mathbf{i}\,S_x = \mathbf{i}\,\mathrm{d}y\,\mathrm{d}z \\
\mathrm{d}\mathbf{S_y} &= \mathbf{j}\,S_y = \mathbf{j}\,\mathrm{d}x\,\mathrm{d}z \\
\mathrm{d}\mathbf{S_z} &= \mathbf{k}\,S_z = \mathbf{k}\,\mathrm{d}x\,\mathrm{d}y
\end{aligned}$$

TODO Obrazok
Divergenciu vektora môžeme potom rozpísať ako

$$\begin{aligned}
\int_V \nabla.\mathbf{F}\,\mathrm{d}V = &\sum_{i,j,k}\left(\frac{F_x(x_{i+1},y_j,z_k) - F_x(x_i,y_j,z_k)}{\mathrm{d}x}\right)\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z + \\
&\sum_{i,j,k}\left(\frac{F_y(x_i,y_{j+1},z_k) - F_y(x_i,y_j,z_k)}{\mathrm{d}y}\right)\mathrm{d}x\,\mathrm{d}y,\mathrm{d}z + \\
&\sum_{i,j,k}\left(\frac{F_z(x_i,y_j,z_{k+1}) - F_z(x_i,y_j,z_k)}{\mathrm{d}z}\right)\mathrm{d}x\,\mathrm{d}y,\mathrm{d}z
\end{aligned}$$

Ak zoberieme prvú zložku vektorového poľa F_x, pre fixované indexy j,k môžeme písať

$$\sum_i \left(\frac{F_x(x_{i+1},y_j,z_k) - F_x(x_i,y_j,z_k)}{\mathrm{d}x}\right)\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z = \sum_i \big(F_x(x_{i+1},y_j,z_k) - F_x(x_i,y_j,z_k)\big)\,\mathrm{d}y\,\mathrm{d}z$$

Ak prevedieme sumáciu, hodnoty poľa na susedných kockách budú mať opačné hodnoty a vo výslednej sume nám zostane len tok cez vonkajšie steny krajných kociek

$$\sum_i \big(F_x(x_{i+1},y_j,z_k) - F_x(x_i,y_j,z_k)\big)\,\mathrm{d}y\,\mathrm{d}z = \big(F_x(x_n,y_j,z_k) - F_x(x_1,y_j,z_k)\big)\,\mathrm{d}y\,\mathrm{d}z$$

Po úpravách potom môžeme vzťah pre divergenciu prepísať

$$\begin{aligned}\int_V \nabla.\mathbf{F}\,\mathrm{d}V = &\sum_{j,k}\big(F_x(x_n,y_j,z_k) - F_x(x_1,y_j,z_k)\big)\,\mathrm{d}y\,\mathrm{d}z + \\ &\sum_{i,k}\big(F_y(x_i,y_n,z_k) - F_y(x_i,y_1,z_k)\big)\,\mathrm{d}x\,\mathrm{d}z + \\ &\sum_{i,j}\big(F_z(x_i,y_j,z_n) - F_z(x_i,y_j,z_1)\big)\,\mathrm{d}x\,\mathrm{d}y\end{aligned}$$

čo je súčet tokov vektorového poľa cez všetky vonkajšie steny kociek tvoriace objem V. Z tohoto vyplýva

$$\int_V \nabla.\mathbf{F}\,\mathrm{d}V = \int_{S_x}(F_{x_n} - F_{x_1})\,\mathrm{d}S_x + \int_{S_y}(F_{y_n} - F_{y_1})\,\mathrm{d}S_y + \int_{S_z}(F_{z_n} - F_{z_1})\,\mathrm{d}S_z = \oint_S \mathbf{F}\,\mathrm{d}\mathbf{S}$$

8.2 Gaussova veta v 2D

Do 2D geometrie $(x, y, 0)$ môžeme Gaussovu vetu previesť pomocou jednoduchého priemetu do roviny, v tomto prípade sa objemový integrál redukuje na integrál cez plochu a integrál cez plochu na integrál po krivke tvoriacej hranicu plochy

$$\iint_\Omega (\nabla\cdot\mathbf{F})\;\mathrm{d}x\,\mathrm{d}y = = \int_{\partial\,\Omega}\mathbf{F}\,\mathrm{d}\mathbf{l}_\perp \tag{8.2}$$

kde Ω je plocha, $\partial\,\Omega$ je jej krivka tvoriaca jej hranicu a $\mathrm{d}\mathbf{l}_\perp$ je normálový vektor k elementu krivky.
Obrazok
Ak je vektor tangenciálneho elementu krivky $d\mathbf{l} = (dx, dy)$, zložky vektora kolmého určíme z podmienky kolmosti vektorov pri ich skalárnom súčine

$$\mathbf{x}\cdot\mathbf{y} = (x_1,x_2)\cdot(y_1,y_2) = x_1y_1 + x_2\,y_2$$

zároveň pre $\mathbf{x}\perp\mathbf{y}$ platí

$$\mathbf{x}\cdot\mathbf{y} = |\mathbf{x}|\,|\mathbf{y}|\,cos(\phi) = 0$$

z čoho

$$x_1y_1 + x_2\,y_2 = 0,\quad \text{resp.}\quad y_1 = x_2 \text{ a } y_2 = -x_1$$

Po dosadení a rozpísaní má potom Gaussova veta pre 2D konfiguráciu tvar

$$\iint_\Omega \left(\frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y}\right)\mathrm{d}x\,\mathrm{d}y = = \int_{\partial\,\Omega} F_x\,\mathrm{d}y - F_y\,\mathrm{d}x$$

8.3 Symbolický výpočet

Symbolický výpočet toku vektora $\mathbf{F}(x, y)$

$$\mathbf{F}(x,y) = \left(-y\,(1-x-y) + \frac{x}{4}\right)\,\mathbf{i} + \left(2 - x^2 + \frac{y}{4}\right)\,\mathbf{j}$$

v 2D rovine cez krivku zadanú parametrickou rovnicou

$$\begin{aligned}x &= \cos(t)\\ y &= \sin(t)\end{aligned}$$

Po úprave má vzťah pre výpočet toku cez hranicu oblasti v 2D tvar

$$\Phi = \int\limits_{\partial\,\Omega} F_x(x,y)\,\mathrm{d}y - F_y(x,y)\,\mathrm{d}x = \int\limits_{C} (F_x(t)\,\cos(t) - F_y(t)(-\sin(t))\,\mathrm{d}t$$

```
%reset -f
from utils.utils import *
from sympy import *
from sympy.vector import CoordSys3D, divergence
q = CoordSys3D('q')                    # suradnicova sustava
t, r = symbols("t,r")                  # parameter pre vypocet integralu
                                       # vektorove pole a substitucia parametrov
F = (-q.y*(1 - q.x - q.y) + q.x/4)*q.i + (2 - q.x**2 + q.y/4)*q.j
Fs = F.subs([(q.x, cos(t)), (q.y, sin(t))])
                                  ns = (cos(t)*q.i) - (-sin(t))*q.j    # normalovy
    vektor dl
Ph = integrate(Fs.dot(ns), (t, 0, 2*pi) )

ltxprint('\mathbf{F}(x,y)', F)
ltxprint('\Phi', Ph)
```

$$\mathbf{F}(x,y) = (\frac{\mathbf{x_q}}{4} - \mathbf{y_q}\,(-\mathbf{x_q} - \mathbf{y_q} + 1))\hat{\mathbf{i}}_\mathbf{q} + (-\mathbf{x_q}^2 + \frac{\mathbf{y_q}}{4} + 2)\hat{\mathbf{j}}_\mathbf{q}$$

$$\Phi = \frac{\pi}{2}$$

8.4 Numerický výpočet

```
from scipy.integrate import quad
import numpy as np

def Fp(t):
    x = np.cos(t)
    y = np.sin(t)
    return (-y*(1-x-y)+ x/4)*(-np.sin(t)) + (2-x**2+y/4)*(np.cos(t))

Flux, err = quad(Fp, 0, 2*np.pi)
print(Flux)
```

```
3.141592653589793
```

Pre porovnanie a ilustráciu výpočet zopakujeme pre ľavú stranu výrazu s výpočtom toku cez plochu kružnice o polomere r. Ak označíme

$$D(x,y) = \nabla \cdot \mathbf{F}(x,y)$$

potom po substitúcii

$$\Phi = \iint\limits_{\Omega} \nabla.\mathbf{F}(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int\limits_0^{2\pi}\int\limits_0^{1} D(\,r\,\cos(\psi), r\,\sin(\psi)\,)\,r\,\mathrm{d}r\,\mathrm{d}\psi$$

```
D = divergence(F)
Ds = D.subs([(q.x, r*cos(t)), (q.y, r*sin(t))])
Ph = integrate(  integrate(Ds*r, (r, 0, 1)), (t, 0, 2*pi) )

ltxprint('\Phi', Ph)
```

$$\Phi = \frac{\pi}{2}$$

Kapitola 9

Stokesova veta

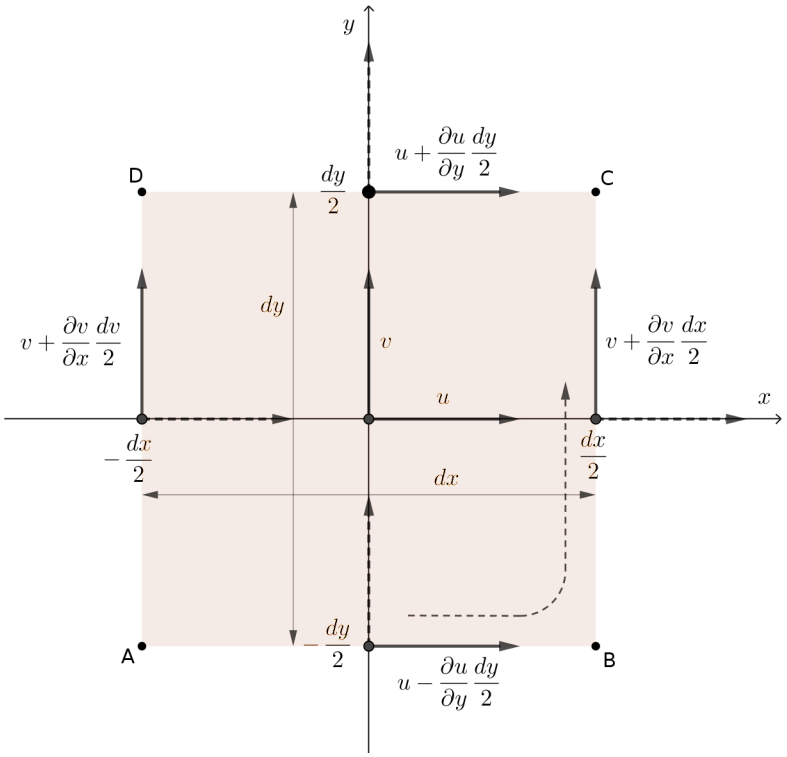
Stokesova veta dáva do súvisu integrál po uzatvorenej krivke C vo vektorovom poli $\mathbf{F}$ s integrálom rotácie poľa cez plochu, ktorá je touto krivkou ohraničená

$$\oint_C \mathbf{F} \cdot \mathrm{d}\mathbf{l} = \int_S (\nabla \times \mathbf{F}) \cdot \mathrm{d}\mathbf{s} \tag{9.1}$$

Stokesova veta v 2D priestore sa nazýva **Greenova veta**.

9.1 Stokesova veta v 2D

Predpokladajme, že vo vektorovom poli $\mathbf{F}$ sa nachádza krivkou C ohraničená plocha S, ktorú rozdelíme na veľa malých plošiek $ABCD$ o rozmeroch $\mathrm{d}x \times \mathrm{d}y$.



Obr. 9.1: Vektorové pole na ploche

Vektorové pole $\mathbf{F}$ môžeme formálne v strede plošky, ktorá sa nachádza na pozícii (x, y) rozpísať ako superpozíciu dvoch kolmých zložiek poľa

$$\mathbf{F} = F_x(x, y)\,\mathbf{i} + F_y(x, y)\,\mathbf{j} = u(x, y)\,\mathbf{i} + v(x, y)\,\mathbf{j}$$

Pre spehľadnenie a zjednodušenie zápisu zaveďme označenie

$$u = u(x, y)$$
$$v = v(x, y)$$

Zaujíma nás hodnota toku poľa pozdĺž krivky, ktorou je plôška ohraničená - v našom prípade štvorec. Výraz môžeme rozdeliť na samostatné časti pre jednotlivé strany plošky

$$\begin{aligned}\mathbf{F}\,\mathrm{d}\mathbf{l} = u\,(x, y - \frac{\mathrm{d}y}{2})\,\mathrm{d}x + v\,(x + \frac{\mathrm{d}x}{2})\,\mathrm{d}y \\ - u\,(x, y + \frac{\mathrm{d}y}{2})\,\mathrm{d}x - \,(x - \frac{\mathrm{d}x}{2})\,\mathrm{d}y\end{aligned}$$

Pri rozklade sme využili smer integrácia a kolmosť medzi zložkami poľa a orientovaným dĺžkovým elementom, v skalárnom súčine sa uplatnia len paralelné zložky

$$\mathbf{F}(x, y)\,\mathrm{d}\mathbf{l} = \big(u(x, y)\,\mathbf{i} + v(x, y)\,\mathbf{j}\big) \cdot \big(\mathbf{i}\,\mathrm{d}x + \mathbf{j}\,\mathrm{d}y\big)$$

Približnú hodnotu zložky poľa v strede hrany plošky získame rozvojom do Taylorovho radu, pre hranu AB

$$u(x, y - \frac{\mathrm{d}y}{2}) \approx u(x, y) - \frac{\partial u(x, y)}{\partial y}\frac{\mathrm{d}y}{2} = u - \frac{\partial u}{\partial y}\frac{\mathrm{d}y}{2}$$

podobne aj pre ostatné hrany plošky.
Dosadením a úpravou dostaneme

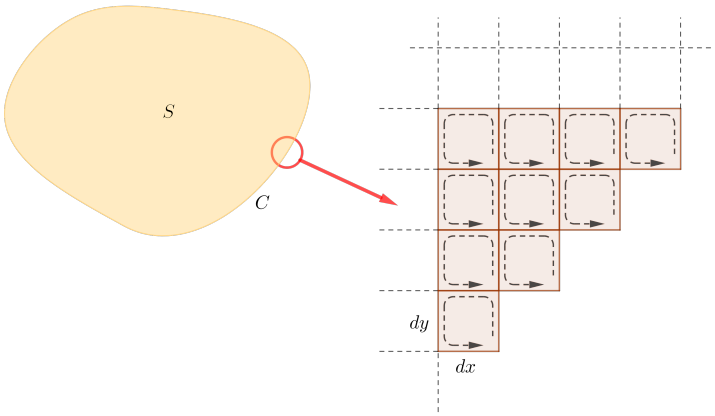
$$\begin{aligned}\mathbf{F}\,d\,\mathbf{l} &= \big(u - \frac{\partial u}{\partial y}\frac{\mathrm{d}y}{2}\big)\,\mathrm{d}x + \big(v + \frac{\partial v}{\partial x}\frac{\mathrm{d}x}{2}\big)\,dy \\ &- \big(u + \frac{\partial u}{\partial y}\frac{\mathrm{d}y}{2}\big)\,\mathrm{d}x - \big(v - \frac{\partial v}{\partial x}\frac{\mathrm{d}x}{2}\big)\,\mathrm{d}y \\ &= \big(-\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\big)\,\mathrm{d}x\,\mathrm{d}y\end{aligned}$$

Výraz $\mathrm{d}x\,\mathrm{d}y$ zodpovedá ploche elemetárnej plošky $\mathrm{d}s$. Ak teraz 'vytapetujeme' ploškami celú plochu S, hodnoty toku na hranách susedných plošiek sa vyrušia a zostanú len hodnoty na vonkajšom obvode plochy, čo môžeme zapísať ako

$$\oint_C \mathbf{F}\,\mathrm{d}\,\mathbf{l} = \int_C F_x\,\mathrm{d}x + F_y\,\mathrm{d}y = \int_S \big(-\frac{\partial Fx}{\partial y} + \frac{\partial Fy}{\partial x}\big)\,\mathrm{d}s$$

Prehodením znamienka pri F_x a úpravou dostaneme známy tvar **Greenovej vety**

$$\int_C F_y\,\mathrm{d}y - F_x\,\mathrm{d}x = \int_S \big(\frac{\partial Fx}{\partial y} + \frac{\partial Fy}{\partial x}\big)\,\mathrm{d}s \tag{9.2}$$



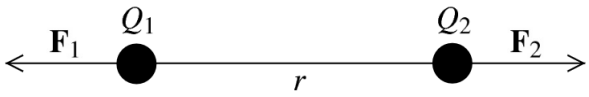
Obr. 9.2: Tok na okraji plochy

Kapitola 10

Elektrostatické pole

10.1 Coulombov zákon

Coulombov zákon popisuje silové pôsobenie medzi nábojmi.



Obr. 10.1: Silové pôsobenie dvoch nábojov

Náboj Q_2 pôsobí vo vákuu na náboj Q_1 vo vzdialenosti r silou $\mathbf{F}_1$ ktorú môžeme vyjadriť vo vektorovej forme

$$\mathbf{F}_1 = \frac{1}{4\pi\epsilon_0}\frac{Q_1\,Q_2}{r^2}\,\mathbf{r}_{21} \tag{10.1}$$

kde $\mathbf{r}_{21}$ je jednotkový vektor smerujúci od Q_2 ku Q_1. Vektorová forma zohľadňuje polaritu nábojov, pri rovnakej polarite sa náboje odpodzujú, pri opačnej priťahujú. Podľa tretieho Newtonovho zákona platí $\mathbf{F}_1 = -\mathbf{F}_2$
V skalárnej forme

$$|\mathbf{F}| = \frac{1}{4\pi\epsilon_0}\frac{|Q_1\,Q_2|}{r^2} \tag{10.2}$$

10.2 Intenzita elektrického poľa

Silové pôsobenie na náboj Q_1 vieme vyjadriť pomocou elektrického poľa $\mathbf{E}$ náboja Q_2, ktoré tento vo svojom okolí vytvára

$$\mathbf{F}_1 = Q_1\left(\frac{1}{4\pi\epsilon_0}\frac{Q_2}{r^2}\mathbf{r}_{21}\right) = Q_1\,\mathbf{E} \tag{10.3}$$

Každý statický náboj vo svojom okolí vytvára elektrické pole

$$\mathbf{E} = \frac{1}{4\pi\epsilon_0}\frac{Q}{r^2}\,\frac{\mathbf{r}}{r} = \frac{1}{4\pi\epsilon_0}\frac{Q}{r^2}\,\mathbf{r}_0 \tag{10.4}$$

kde $\mathbf{r}_0$ je jednotkový vektor. Vektor elektrického poľa smeruje od kladného náboja k zápornému, táto konvencia je určená historickými dôvodmi.

10.3 Potenciál

Pre presun náboja q medzi dvoma bodmi a, b v elektrickom poli iného náboja Q musíme vykonať prácu, ktorá je definovaná ako

$$A = \int_a^b \mathbf{F}\,\mathrm{d}\mathbf{r} = \int_a^b q\,\mathbf{E}\,\mathrm{d}\mathbf{r}$$

Predpokladajme, že q je jednotkový náboj, potom

$$A = 1 \int_a^b \mathbf{E}\,\mathrm{d}\mathbf{r} = \frac{1}{4\pi\epsilon_0} \int_a^b \frac{Q}{r^2}\mathrm{d}r = \frac{Q}{4\pi\epsilon_0}\left[-\frac{1}{r}\right]_a^b = \frac{Q}{4\pi\epsilon_0}\left(-\frac{1}{b}+\frac{1}{a}\right)$$

Práca pre presunutie jednotkového náboja z bodu a do bodu b je potom rovná rozdielu prác, ktoré sme vynaložili pre presun náboja do bodu a a potom do bodu b. Je ale zrejmé, že počiatočnú hodnotu v bode a nepoznáme, preto by bolo vhodné zadefinovať nejaký referenčný bod, z ktorého by sme pri určení práce vychádzali s počiatočnou nulovou hodnotou. Pretože práca na presun náboja v elektrickom poli iného náboja je nepriamo úmerná ich vzdialenosti, zvoľme ako referenčný bod s nulovým potenciálom v nekonečne.
Určime, akú prácu preto treba vykonať pri presune jednotkového náboja z nekonečna do vzdialenosti r od náboja Q. Substitúciou za $a \to \infty$, $b \to r$ dostaneme

$$A = -\frac{Q}{4\pi\epsilon_0}\frac{1}{r} = -\frac{Q}{4\pi\epsilon_0}\frac{1}{r^2}\,\mathbf{r}_0\,\mathbf{r} = -\mathbf{E}\,\mathbf{r}$$

Potenciál v bode vzdialenom r od bodového náboja, ktorý vytvára vo svojom okolí pole $E(e)$ je potom definovaný ako

$$V(r) = -A = \mathbf{E}\,\mathbf{r} \tag{10.5}$$

TODO - OBRAZOK
Potenciál medzi dvoma bodmi a, b potom môžeme určiť ako

$$V_{ab} = \int_a^b \mathbf{E}\,\mathrm{d}\mathbf{r} = V_b - V_a \tag{10.6}$$

pretože elektrické pole je konzervatívne, hodnota krivkového integrálu závisí len od jeho koncových bodov. Z tohoto vyplýva, že integrál po uzatvorenej krivke bude zrejme rovný nule (počuatočný a koncový bod krivky sú zhodné)

$$\oint_C \mathbf{E}\,\mathrm{d}\mathbf{r} = 0 \tag{10.7}$$

Uplatnením Stokesovej vety môžeme vzťah prepísať do tvaru

$$\oint_C \mathbf{E}\,\mathrm{d}\mathbf{r} = \int_S \left(\nabla \times \mathbf{E}\right)\cdot \mathrm{d}\mathbf{s} = 0 \tag{10.8}$$

alebo v diferenciálnom tvare

$$\nabla \times \mathbf{E} = 0 \tag{10.9}$$

ktorý hovorí o tom, že elektrostatické pole nevytvára slučky.

10.4 Gaussov zákon

Spočítajme, aký je tok elektrického poľa uzatvorenou spojitou hladkou plochou, ktorá obklopuje bodový náboj. Pre jednoduchosť zvoľme gulovú plochu a náboj umiestnime do jej stredu. V sférických súradniciach potom pre tok poľa dostaneme

$$\oint_S \mathbf{E}\,\mathrm{d}\mathbf{s} = \int_{\phi=0}^{\pi}\int_{\zeta=0}^{2\pi} \frac{Q}{4\pi\epsilon_0}\frac{1}{r^2}\,r^2\sin(\phi)\,\mathrm{d}\phi\,\mathrm{d}\zeta = \frac{Q}{\epsilon_0}$$

Vo všeobecnosti nezávisí na tvare plochy (musí byť len hladká a spojitá) a ani na tom, či je náboj bodový, umiestnený v strede alebo je rozložený v objeme uzatvorenom plochou. Pre objemovo rozložený náboj s hustotou ρ v objeme V potom platí vzťah označovaný aj ako Gaussov zákon elektrostatiky v integrálnom tvare

$$\oint_S \mathbf{E}\,\mathrm{d}\mathbf{s} = \frac{1}{\epsilon_0}\int_V \rho(r)\,\mathrm{d}v \tag{10.10}$$

Použitím Gaussovej vety dostaneme

$$\oint_S \mathbf{E}\,\mathrm{d}\mathbf{s} = \int_V \nabla\cdot\mathbf{E}\,\mathrm{d}v = \frac{1}{\epsilon_0}\int_V \rho(r)\,\mathrm{d}v$$

Porovnaním podintegrálnych funkcií dostaneme Gaussov zákon v diferenciálnom tvare, ktorý definuje divergenciu elektrického poľa vo zvolenom bode priestoru

$$\nabla\cdot\mathbf{E} = \frac{\rho(r)}{\epsilon_0} \tag{10.11}$$

Príklad - Elektrické pole nabitej gule
Predpokladajme, že máme vodivú guľu s polomerom r, ktorá je nabitá nábojom Q. Z Gaussovho zákona vyplýva

$$\oint_S \mathbf{E}\,d\mathbf{s} = \frac{Q}{\epsilon_0}$$

Ak predpokladáme, že absolútna hodnota poľa na povrchu gule je konštantná, môžeme vzťah upraviť s pomocou vzťahu pre plochu gule

$$E(r)\,4\pi r^2 = \frac{Q}{\epsilon_0}$$

Pre pole na povrchu nabitej gule potom platí

$$\mathbf{E} = \frac{Q}{4\pi\epsilon_0 r^2}\mathbf{r}_0$$

Porovnaním so vzťahom pre pole bodového náboja vyplýva, na povrchu vodivej gule také pole, ako keby bol všetok náboj z povrchu gule sústredený do jej stredu. Potenciál nabitej gule je potom

$$V(r) = \mathbf{E}\cdot\mathbf{r} = \frac{Q}{4\pi\epsilon_0 r^2}\mathbf{r}_0\mathbf{r} = \frac{Q}{4\pi\epsilon_0 r}$$

10.5 Kapacita

Kapacita určuje mieru schopnosti telesa zhromažďovať náboj. Definovaná je ako pomer veľkosti náboja na telese voči potenciálu telesa, tento môže byť voči referenčnému nulovému potenciálu v nekonečne, vtedy hovoríme o vlastnej kapacite, alebo voči inému telesu, vtedy hovoríme o vzájomnej kapacite.

$$C = \frac{Q}{V} \tag{10.12}$$

Príklad - Vlastná kapacita zemegule
Predpokladajme gulu v neohraničenom priestore, do vzťahu pre kapacitu dosadíme potenciál gule

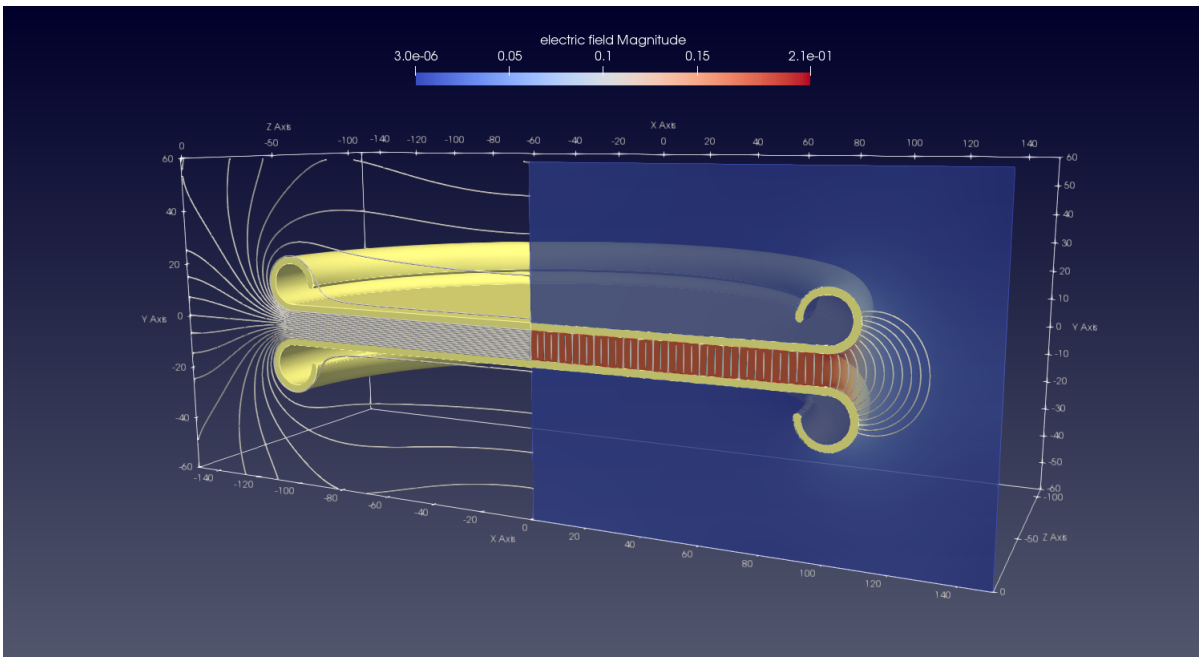
$$C = \frac{Q}{\frac{Q}{4\pi\epsilon_0 r}} = 4\pi\epsilon_0\, r$$

```
# Kapacita zemegule
from scipy import pi
from scipy.constants import epsilon_0    # import konstant z databazy

r = 6370e3                                # polomer zeme [m]
Cap = 4*pi*epsilon_0*r
print('C= {0:.3f} [uF]'.format(Cap*1e6) )
```

```
C= 708.758 [uF]
```

Priklad - Kapacita kondenzátora
Kondenzátor pozostáva z dvoch platní o ploche S vzdialených od seba d, medzi platňami je potenciálový rozdiel V. Medzi doskami kondenzátora je homogénne elektrické pole E. Použitím Gaussovej vety a vzťahu pre výpočet potenciálu dostaneme
TODO - OBRAZOK

$$C = \frac{Q}{V} = \frac{\epsilon_0 ES}{Ed} = \frac{\epsilon_0 S}{d}$$



Obr. 10.2: Simulácia elektrického poľa medzi doskami kondenzátora

Kapitola 11

Magnetostatické pole

11.1 Magnetická indukcia

Magnetické pole vytvára uzatvorené slučky, ktoré nekončia v nejakom bode priestoru, pretože magnetické náboje neexistujú. Magnetické pole je popísané magnetickou indukciou $\mathbf{B}$, na pohybujúci sa náboj pôsobí silou (Lorentzova sila)

$$\mathbf{F} = Q\left(\mathbf{v} \times \mathbf{B}\right)$$

Zdrojom magnetického poľa sú pohybujúce sa náboje - elektrický prúd. Prúd I pretekajúci úsekom vodiča o dĺžke $\mathrm{d}\mathbf{l}$ vytvára vo svojom okolí magnetické pole (Biot-Savartov zákon)

$$\mathrm{d}\mathbf{B} = \frac{\mu_0}{4\pi}\frac{I\,\mathrm{d}\mathbf{l} \times \mathbf{r}}{r^3}$$

kde $\mathbf{r}$ je polohový vektor medzi úsekom vodiča a bodom pozorovania. Pretože z dôvodu princípu zachovania náboja musí elektrický prúd tiecť v uzatvorenom obvode, pre magnetické pole slučky s prúdom potom platí

$$\mathbf{B} = \frac{\mu_0}{4\pi}\oint_L \frac{I\,\mathrm{d}\mathbf{l} \times \mathbf{r}}{r^3}$$

11.2 Ampérov zákon

Vzťah medzi elektrickým prúdom a magnetickou indukciou vyjadruje Amperov zákon

$$\oint_L \mathbf{B} \cdot \mathrm{d}\mathbf{l} = \mu_0 I_c$$

kde I_c je celkový prúd pretekajúci plochou obopnutou krivkou L.
Použitím Stokesovej vety dostaneme

$$\oint_L \mathbf{B} \cdot \mathrm{d}\mathbf{l} = \int_S \nabla \times \mathbf{B}\,\mathrm{d}\mathbf{s} = \mu_0 I_c = \mu_0 \int_S \mathbf{J}\,\mathrm{d}\mathbf{s}$$

Porovnaním podintegrálnych výrazov dostaneme Ampérov zákon v tvare

$$\nabla \times \mathbf{B} = \mu_0\,\mathbf{J}$$

Kapitola 12

Yee Algoritmus

Yee algoritnmus je pomenovaný po matematikovi Kane Yee, ktorý ho predstavil v roku 1966. Všeobecný postup výpočtu elektromagnetických polí je možné defnovať nasledujúcimi krokmi

- vo zvolenom priestore definujeme vybrané uzly, v ktorých budeme určovať hodnoty polí E a H
- aproximujeme časové a priestorové derivácie v Ampérovom a Faradayovom zákone konečnými diferenciami
- vyriešime sústavu diferenčných rovníc tak, aby sme získali *aktualizačné* rovnice, ktoré popisujú budúce hodnoty na základe predchádzajúcich hodnôt v priestore a čase
- v jednotlivých časových krokoch spočítame hodnoty polí E,H pre všetky uzly priestoru
- opakujeme predchádzajúci krok, kým nedosiahneme stanovený čas ukončenia simulácie

Vyššie uvedený postup ukážeme na odvodení algoritmu pre jednorozmerný prípad.

12.1 Yee Algoritmus v 1D

Predpokladajme jednorozmenrný priestor v ktorom sa môže šíriť elektromagnetická vlna v smere osi x. Zložky elektromagnetického poľa sú na seba kolmé, predpokladajme, že elektrická zložka je orintovaná v smere osi z a magnetická v smere osi y.
Podľa Faradayovho zákona potom môžeme písať

$$-\mu\frac{\partial \mathbf{H}}{\partial t} = \nabla \times \mathbf{E} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \frac{\partial}{\partial x} & 0 & 0 \\ 0 & 0 & E_z \end{vmatrix} = -\mathbf{j}\frac{\partial E_z}{\partial x} \tag{12.1}$$

a z Ampérovho zákona vyplýva

$$-\epsilon\frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{H} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \frac{\partial}{\partial x} & 0 & 0 \\ 0 & H_y & 0 \end{vmatrix} = -\mathbf{k}\frac{\partial H_y}{\partial x} \tag{12.2}$$

po rozpísaní na zložky dostaneme sústavu skalárnych rovníc

$$\mu\frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\partial x} \tag{12.3}$$

$$\epsilon\frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x} \tag{12.4}$$

Z rovníc je zrejmé, že časová derivácia jedného poľa závisí od priestorovej derivácia druhého poľa.

12.2 Diskretizácia rovníc

Pre disktertizáciu rovníc zavedieme označenie, ktoré popisuje časovo-priestorovú lokalizáciu polí, index q nie je mocnina, ale časový krok

$$E_z(x,t) = E_z(m\,\Delta x, q\,\Delta t) = E_z^q[m]$$
$$H_y(x,t) = H_y(m\,\Delta x, q\,\Delta t) = H_y^q[m]$$

12.2.1 Diskretizácia Faradayovho zákona

V priestorovo-časovej súradnici $[(m+1/2)\Delta x, q\Delta t]$ môžeme pre Faradayov zákon písať

$$\mu \frac{\partial H_y}{\partial t}\bigg|_{(m+1/2)\Delta x,\, q\Delta t} = \frac{\partial E_z}{\partial x}\bigg|_{(m+1/2)\Delta x,\, q\Delta t}$$

Deriváciu H_y v čase $q\Delta t$ aproximovať pomocou dvoch po sebe nasledujúcich hodnôt

$$\mu \frac{\partial H_y}{\partial t}\bigg|_{(m+1/2)\Delta x,\, q\Delta t} \approx \mu \frac{H_y^{q+1/2}[m+1/2] - H_y^{q-1/2}[m+1/2]}{\Delta t}$$

Deriváciu E_z na pozícii $(m+1/2)\Delta x$ aproximovať pomocou dvoch susedných hodnôt

$$\frac{\partial E_z}{\partial x}\bigg|_{(m+1/2)\Delta x,\, q\Delta t} \approx \frac{E_z^q[m+1] - E_z^q[m]}{\Delta x}$$

Z vyššie uvedeného potom vyplýva

$$\mu \frac{H_y^{q+1/2}[m+1/2] - H_y^{q-1/2}[m+1/2]}{\Delta t} = \frac{E_z^q[m+1] - E_z^q[m]}{\Delta x}$$

po úprave dostaneme pre hodnotu $H_y^{q+1/2}[m+1/2]$ *aktualizačnú* rovnicu v tvare

$$H_y^{q+1/2}[m+1/2] = H_y^{q-1/2}[m+1/2] + \frac{\Delta t}{\mu\,\Delta x}\big(E_z^q[m+1] - E_z^q[m]\big) \tag{12.5}$$

Priestorovo-časové rozloženie zložiek poľa E_z a H_y potom môžeme znázorniť na nasledujúcom diagrame, $\otimes$ označuje bod $[(m+1/2)\Delta x, q\Delta t]$.
Je zrejmé, že budúca hodnota magnetického poľa závisí od predchádzajúcej hodnoty a susedných hodnôt elektrického poľa.

12.2.2 Diskretizácia Ampérovho zákona

V priestorovo-časovej súradnici $[(m\Delta x, (q+1/2)\Delta t]$ môžeme pre Ampérov zákon písať

$$\epsilon \frac{\partial E_z}{\partial t}\bigg|_{m\Delta x,\, (q+1/2)\Delta t} = \frac{\partial H_y}{\partial x}\bigg|_{m\Delta x,\, (q+1/2)\Delta t}$$

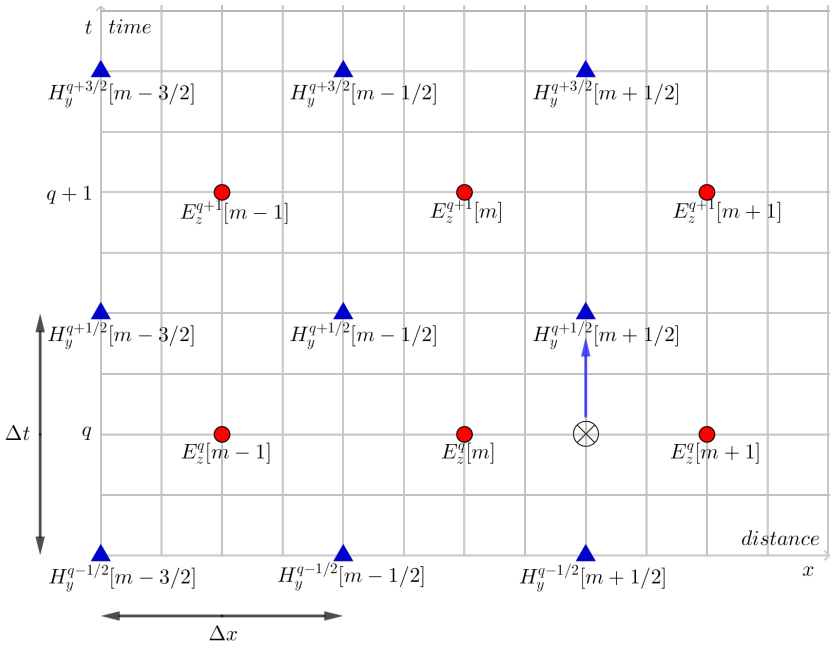
Z vyššie uvedeného potom vyplýva

$$\epsilon \frac{E_z^{q+1}[m] - E_z^q[m]}{\Delta t} = \frac{H_y^{q+1/2}[m+1/2] - H_y^{q+1/2}[m-1/2]}{\Delta x}$$

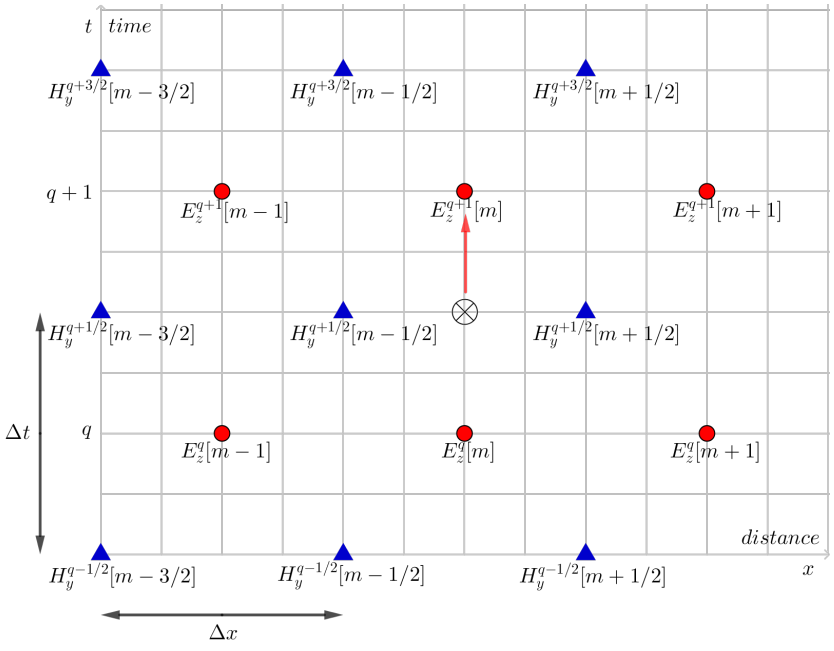
po úprave dostaneme pre hodnotu $E_z^{q+1}[m]$ aktualizačnú rovnicu v tvare

$$E_z^{q+1}[m] = E_z^q[m] + \frac{\Delta t}{\epsilon\,\Delta x}\big(H_y^{q+1/2}[m+1/2] - H_y^{q+1/2}[m-1/2]\big) \tag{12.6}$$

Podobne ako pri diskretizovanom Faradayovom zákone budúca hodnota elektrického poľa závisí od predchádzajúcej hodnoty a susedných hodnôt magnetického poľa.



Obr. 12.1: Aktualizačná rovnica pre Faradayov zákon



Obr. 12.2: Aktualizačná rovnica pre Ampérov zákon

12.3 Courantovo číslo

Koeficienty v aktualizačných rovniciach $\Delta t \,/\, \mu \Delta x$ a $\Delta t \,/\, \epsilon \Delta x$ určujú, ako ďaleko (v čase a priestore) sa môže šíriť elektromagnetická energia. Ich veľkosť má vplyv na stabiliu riešenia a zároveň určujú rozmerové paramere simulačného modelu. Pri platnosti vzťahov

$$c = \frac{1}{\sqrt{\epsilon_0 \mu_0}}$$
$$z_0 = \sqrt{\frac{\mu_0}{\epsilon_0}}$$
$$\mu = \mu_0 \mu_r$$
$$\epsilon = \epsilon_0 \epsilon_r$$

môžeme koeficienty prepísať do tvaru

$$\frac{\Delta t}{\epsilon \Delta x} = \frac{1}{\epsilon_0 \epsilon_r} \frac{\sqrt{\epsilon_0 \mu_0}}{\sqrt{\epsilon_0 \mu_0}} \frac{\Delta t}{\Delta x} = \frac{\sqrt{\epsilon_0 \mu_0}}{\epsilon_0 \epsilon_r} \frac{c \Delta t}{\Delta x} = \frac{1}{\epsilon_r} \sqrt{\frac{\mu_0}{\epsilon_0}} \frac{c \Delta t}{\Delta x} = \frac{z_0}{\epsilon_r} S_c$$
$$\frac{\Delta t}{\mu \Delta x} = \frac{1}{\mu_0 \mu_r} \frac{\sqrt{\epsilon_0 \mu_0}}{\sqrt{\epsilon_0 \mu_0}} \frac{\Delta t}{\Delta x} = \frac{\sqrt{\epsilon_0 \mu_0}}{\mu_0 \mu_r} \frac{c \Delta t}{\Delta x} = \frac{1}{\mu_r} \sqrt{\frac{\epsilon_0}{\mu_0}} \frac{c \Delta t}{\Delta x} = \frac{1}{\mu_r \, z_0} S_c$$

kde S_c je veličina nazývaná Courantovo číslo, definované ako

$$S_c = \frac{c \Delta t}{\Delta x}$$

Výber hodnoty S_c
Elektromagnetická vlna sa vo voľnom priestore nemôže šíriť rýchlosťou väčšou ako je rýchlosť svetla c. Čas potrebný na prechod vlny o veľkosť jednej bunky vyžaduje minimálny čas $\Delta t = \Delta x \,/\, c$. V dvojrozmernom prípade, ak sa vlna šíri po diagonále, bude požiadavka na hodnotu časového kroku rovná

$$\Delta t = \frac{\Delta x}{\sqrt{2}\, c}$$

Pre trojrozmerný prípad pri šírení sa po priestorovej diagonále podobne platí

$$\Delta t = \frac{\Delta x}{\sqrt{3}\, c}$$

Zovšeobecnením dostaneme *Courantovu podmienku*

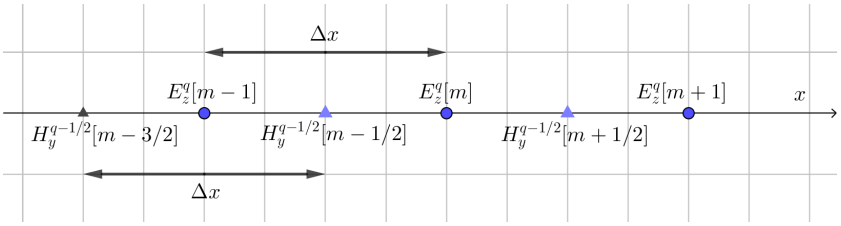
$$\Delta t \leq \frac{\Delta x}{\sqrt{n}\, c}$$

ktorá prakticky hovorí o tom, že (numericky) by mal byť krok v čase rovný alebo menší ako je priestorový krok. Menšia hodnota Δt zodpovedá väčšiemu počtu krokov na zvolený interval simulácie, jemnejšie časové rozlíšenie a väčšiu stabilitu riešenia za cenu dlhšieho výpočtu.
V metóde FDTD z praktických dôvodov pokladáme hodnotu $c = 1$ a ako východziu hodnotu pri simulácii zvyčajne volíme $S_c = 1$.

Kapitola 13

Implementácia v 1D

Obsahom tejto kapitoly je ukážka implementácie Yee algoritmu v jednorozmernom prípade. V takomto prípade budeme predpokladať priestorové usporiadanie zložiek polí E a H podľa nasledujúceho diagramu



Obr. 13.1: Jednorozmerná konfigurácia problému

Simulačné prostredie pre šírenie sa EM vlny v 1D priestore bude o dĺžke 200 uzlov v časovom intervale 300 krokov. E_z a H_y budú uložené v samostatných poliach. Fyzická reprezentácia simulačného prostredia závisí od našej voľby jednotiek, ak pri Courantovom čísle $S_c = 1$ zvolíme priestorový krok $\Delta x = 1\,mm$, potom časovému kroku bude zodpovedať doba

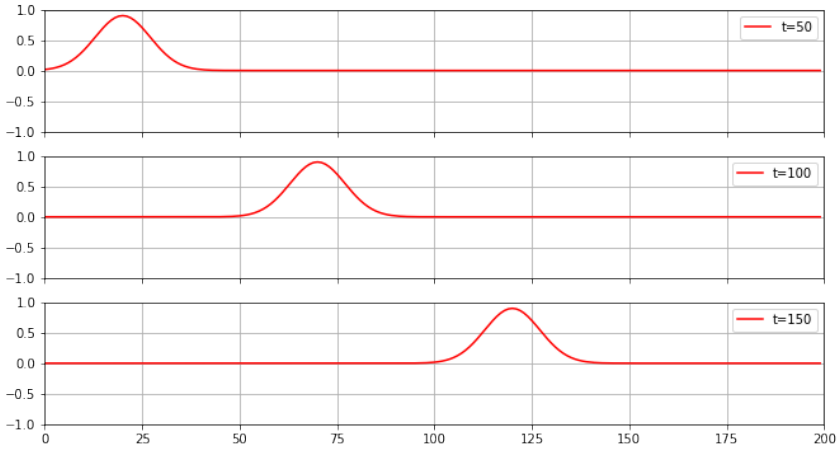
$$\Delta t = S_c \frac{\Delta x}{c} = \frac{1.10^{-3}}{3.10^8} = 3.3.10^{-12} = 3.3\,ps$$

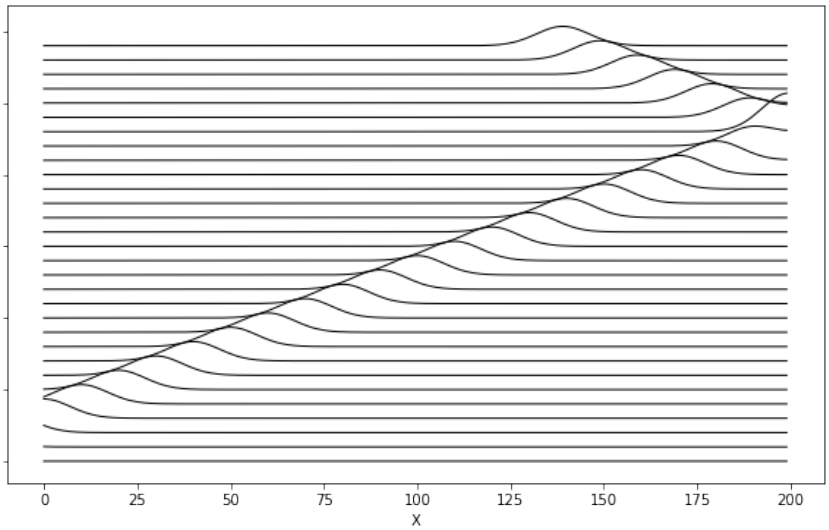
Elektromagnetickú vlnu vybudíme eletrickou zložkou v tvare exponenciálneho impulzu posunutom v čase voči začiatku simulácia

$$E_z[0] = \exp\left(-\frac{(t-30)^2}{100}\right)\,[V\,m^{-1}]$$

```
%reset -f
%matplotlib inline
from scipy import *
from utils.utils import *
import matplotlib.pyplot as plt

nodes = 200
time = 300
ez = zeros(nodes)
hy = zeros(nodes)
z0 = 377.0
Sc = 1.0

r = zeros((time, nodes))                              # pole vysledkov

for t in range(time):
    ez[0] = exp(-(t - 30.)**2 / 100.) *0.9               # zdroj E vlny
    for m in range(1,nodes):                          # 1...199
        ez[m] = ez[m] + (hy[m] - hy[m - 1]) * z0 * Sc     # aktualizacia Ez
    for m in range(nodes-1):                          # 0...198
```

```
        hy[m] = hy[m] + (ez[m + 1] - ez[m]) / z0 * Sc     # aktualizacia Hy
        r[t,:] += ez

tm_plot([50,100, 150], r)
wv_plot(time, nodes, r)
```





Úloha
Vyskúšajte výpočet s rôznymi hodnotami Courantovho čísla (0.5, 1.0, 1.1 ...)

13.0.1 Okrajové podmienky

V metóde FDTD dôležitú rolu hrajú okrajové podmienky. Z predchádzajúceho príkladu je zrejmé, že na kraji simulačného priestoru dochádza k odrazu vlnenia. Pre analýzu okrajových podmienok predchádzajúci príklad zjednodušíme, veľkosť prostredia obmedzíme na 5 uzlov, ako signál použijeme jednotkový impulz a impedanciu vákua pokladáme za rovnú 1.

```
%reset -f
from scipy import *

def tprint(res_e, res_h, ts, te):
    # formatovana tlac tabulky
    print('         +' +'------+'*5)
```

```
    print('        |' +' E  H |'*5)
    print('--------+' +'------+'*5)
    for q in range(ts,te):
        s1 = 'q = {0:2.0f}  '.format(q)
        s2 = ''
        s3 = '        |   '
        for m in range(nodes):
            s2 += '| {0: 1.0f}   '.format(res_e[q,m])
            s3 += '{0: 1.0f} |   '.format(res_h[q,m])
        print(s1 + s2 + '|\n' + s3)
        print('--------+' +'------+'*5)

nodes = 5                                   # pocet uzlov
time = 15                                   # pocet krokov
ez = zeros(nodes)                       hy = zeros(nodes)

res_e = zeros((time, nodes))
res_h = zeros((time, nodes))

for q in range(time):
    ez[0] = (1 if q==0 else 0)
    for m in range(1,nodes):                        ez[m] = ez[m] + (hy[m] - hy[m
   - 1])      # aktualizacia Ez
    for m in range(nodes-1):
        hy[m] = hy[m] + (ez[m + 1] - ez[m])      # aktualizacia Hy
        res_h[q,:] += hy
    res_e[q,:] += ez
```

13.0.2 Generovanie impulzu

Pri generovaní impulzu v prvom kroku nadobudne zdroj EM vlnenia - uzol Ez[0] hodnotu 1, v ďalších krokoch bude mať tento uzol nulovú hodnotu bez ohľadu na stav vedľajších uzlov. Simulácia prebieha cyklicky pre celé simulačné prostredie opakovane pre hodnoty m=0...4

```
Ez[0] = 1                                                   q = 0
Ez[1] = 0                                                   q = 0

Hy[0] = Hy[0] + (Ez[1] - Ez[0]) =  0 + (0 - 1) = -1         q = 1/2
Hy[1] = Hy[1] + (Ez[2] - Ez[1]) =  0 + (0 - 0) =  0         q = 1/2
...
                                                            nasledujuci cyklus
Ez[0] = 0                                                   q = 1
Ez[1] = Ez[1] + (Hy[1] - Hy[0]) =  0 + (0 --1) =  1         q = 1

Hy[0] = Hy[0] + (Ez[1] - Ez[0]) = -1 + (1 - 0) =  0         q = 1 + 1/2
Hy[1] = Hy[1] + (Ez[2] - Ez[1]) =  0 + (0 - 1) = -1         q = 1 + 1/2
...
```

Z priebehu výpočtu a nasledujúcej tabuľky je zrejmé, ako sa vlna postupne po každom cykle aktualizácie presúva od zdroja k pravému okraju.

```
tprint(res_e, res_h, 0, 4)
```

```
        +------+------+------+------+------+
        | E  H | E  H | E  H | E  H | E  H |
--------+------+------+------+------+------+
q =  0  |  1   |  0   |  0   |  0   |  0   |
        |   -1 |    0 |    0 |    0 |    0 |
--------+------+------+------+------+------+
```

```
q =  1  |  0   |  1   |  0   |  0   |  0   |
        |    0 |   -1 |    0 |    0 |    0 |
--------+------+------+------+------+------+
q =  2  |  0   |  0   |  1   |  0   |  0   |
        |    0 |    0 |   -1 |    0 |    0 |
--------+------+------+------+------+------+
q =  3  |  0   |  0   |  0   |  1   |  0   |
        |    0 |    0 |    0 |   -1 |    0 |
--------+------+------+------+------+------+
```

13.0.3 PMC Odraz na pravej strane prostredia

Cyklickým prepočítavaním postupne prejdeme k vyhodnoteniu stavu na pravom okraji. Pretože cyklus pre Hy končí pri hodnote m-1, bude hodnota Hy[m] rovná vždy 0.

```
Ez[3] = Ez[3] + (Hy[3] - Hy[2]) = 0 + (0 - -1) =  1       q = 3
Ez[4] = 0                                                 q = 3

Hy[3] = Hy[3] + (Ez[4] - Ez[3]) = 0 + (0 -  1) = -1       q = 3 + 1/2
Hy[4] = 0                                                 q = 3 + 1/2
...

Ez[3] = 0                                                 q = 4
Ez[4] = Ez[4] + (Hy[4] - Hy[3]) = 0 + (0 - -1) =  1       q = 4

Hy[3] = 0                                                 q = 4 + 1/2
Hy[4] = 0                                                 q = 4 + 1/2
```

Pri nasledujúcom cykle preto nedôjde k zmene hodnoty Ez, pretože hodnota Hy v poslednom uzle je rovná 0.

```
Ez[3] = Ez[3] + (Hy[3] - Hy[2]) = 0 + (0 -  0) =  0       q = 5
Ez[4] = Hy[3] + (Ez[4] - Ez[3]) = 0 + (1 -  0) =  1       q = 5

Hy[3] = Hy[3] + (Ez[4] - Ez[3]) = 0 + (1 -  0) =  1       q = 5 + 1/2
Hy[4] = 0                                                 q = 5 + 1/2
...
Ez[3] = Ez[3] + (Hy[3] - Hy[2]) = 0 + (1 -  0) =  1       q = 6
Ez[4] = Ez[4] + (Hy[4] - Hy[3]) = 1 + (0 -  1) =  0       q = 6

Hy[3] = Hy[3] + (Ez[4] - Ez[3]) = 0 + (1 -  1) =  0       q = 6 + 1/2
Hy[4] = 0                                                 q = 6 + 1/2
```

Pri odraze sa elektrická zložka zachovala, magnetická otočila, z technického hľadiska takýto stav zodpovedá odrazu vlnenia na voľnom konci vedenia. V terminológii FDTD sa takyto uzol označuje ako PMC (Perfect Magnetic Conductor).

```
tprint(res_e, res_h, 3, 7)
```

```
        +------+------+------+------+------+
        | E  H | E  H | E  H | E  H | E  H |
--------+------+------+------+------+------+
q =  3  |  0   |  0   |  0   |  1   |  0   |
        |    0 |    0 |    0 |   -1 |    0 |
--------+------+------+------+------+------+
q =  4  |  0   |  0   |  0   |  0   |  1   |
        |    0 |    0 |    0 |    0 |    0 |
--------+------+------+------+------+------+
q =  5  |  0   |  0   |  0   |  0   |  1   |
```

```
        |    0 |    0 |    0 |    1 |    0 |
--------+------+------+------+------+------+
q =  6  |  0   |  0   |  0   |  1   |  0   |
        |    0 |    0 |    1 |    0 |    0 |
--------+------+------+------+------+------+
```

13.0.4 PEC Odraz na ľavej strane vedenia

Po vygenerovaní impulzu má elektrická zložka v uzle [0] nulovú hodnotu, čo zodpovedá pripojenému zdroju napätia s nulovým odporom, z vlnového hladiska je to ekvivalent skratovaného vedenia. Podobným rozborom ako bolo uvedené vyššie sa zistíme, že elektická zložka sa otočila, magnetická zostala zachovaná, toto zodpovedá odrazu vlnenia na pevnom resp. skratovanom konci. V terminológii FDTD sa takyto uzol označuje ako PEC (Perfect Electric Conductor).

```
tprint(res_e, res_h, 7, 11)
```

```
        +------+------+------+------+------+
        | E  H | E  H | E  H | E  H | E  H |
--------+------+------+------+------+------+
q =  7  |  0   |  0   |  1   |  0   |  0   |
        |    0 |    1 |    0 |    0 |    0 |
--------+------+------+------+------+------+
q =  8  |  0   |  1   |  0   |  0   |  0   |
        |    1 |    0 |    0 |    0 |    0 |
--------+------+------+------+------+------+
q =  9  |  0   |  0   |  0   |  0   |  0   |
        |    1 |    0 |    0 |    0 |    0 |
--------+------+------+------+------+------+
q = 10  |  0   | -1   |  0   |  0   |  0   |
        |    0 |    1 |    0 |    0 |    0 |
--------+------+------+------+------+------+
```

Pre simuláciu EM vlnenia vo voľnom priestore je potrebné definovať okrajové podmienky prostredia tak, aby nedochádzalo k odrazom a následným interferenciám. Vytvorenie bezodrazových alebo impedančne prispôsobených okrajových podmienok bude diskutované v nasledujúcich kapitolách.

13.1 Implementácia aditívneho zdroja

Implementácia zdroja pomocou striktne definovanej zložku poľa nie je optimálna, pretože uzol so zdrojom má charakter skratu a elektromagnetická energia sa cez takýto zdroj nešíri. Problém je možné riešiť aditívnym zdrojom elektromagnetického poľa, ktorý nebráni šíreniu sa energie.
Pri odvodení môžeme vychádzať z Ampérovho zákona s prúdovou hustotou, ktorý má tvar

$$\nabla \times \mathbf{H} = \mathbf{J} + \epsilon \frac{\partial \mathbf{E}}{\partial t}$$

Prúdová hustota reprezentuje pohyb nosičov náboja pod vplyvom elektrického poľa ako aj prúdy ktoré sú zviazané s inými zdrojmi v prostredí. Úpravou dostaneme

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\epsilon} \nabla \times \mathbf{H} - \frac{1}{\epsilon} \mathbf{J}$$

Prevedením na do diferenčného tvaru a úpravou na aktualizačnú rovnicu dostaneme

$$E_z^{q+1}[m] = E_z^q[m] + \frac{\Delta t}{\epsilon \, \Delta x} \left(H_y^{q+1/2}[m + 1/2] - H_y^{q+1/2}[m - 1/2] \right) - \frac{\Delta t}{\epsilon} J_z^{q+1/2}[m] \tag{13.1}$$

Aby sme nemuseli meniť implementáciu príkladu z predchádzajúcej kapitoly, rozložíme aktualizáciu na dva kroky

$$E_z^{q+1}[m] = E_z^q[m] + \frac{\Delta t}{\epsilon\,\Delta x}\left(H_y^{q+1/2}[m+1/2] - H_y^{q+1/2}[m-1/2]\right) \tag{13.2}$$

$$E_z^{q+1}[m] = E_z^q[m] - \frac{\Delta t}{\epsilon} J_z^{q+1/2}[m] \tag{13.3}$$

13.2 Okrajové podmienky pre simuláciu neohraničeného prostredia

V predchádzajúcich príkladoch reprezentovali okrajové podmienky fiktívny ideálny elektrický (PEC) alebo magnetický (PMC) vodič, kde na hranici prostredia dochádzalo k odrazom. Pre simuláciu šírenia sa EM energie v priestore je vhodné pri konečnom počte uzlov podmienky zadefinovať tak, aby sme mohli pokladať prostredie za neohraničené, t.j. na hranici oblasti nedochádza k odrazom. V jednorozmernom prípade pri Courantovom čísle $S_c = 1$ je možné takéto podmienky definovať exaktne (ABC - Absorbing Boundary Condition), pre ostatné prípady je potrebná aproximácia.
ABC pre ľavý okraj odvodíme jednoduchou úvahou. Pretože vyžadujeme, aby sa energia šírila výhradne zprava doľava a zároveň predpokladáme, že zvonka sa nebude do prostredia šíriť žiadna energia, musí sa po aktualizácii hodnota $E_z[0]$ rovnať hodnote $E_z[1]$ pred aktualizáciou, t.j

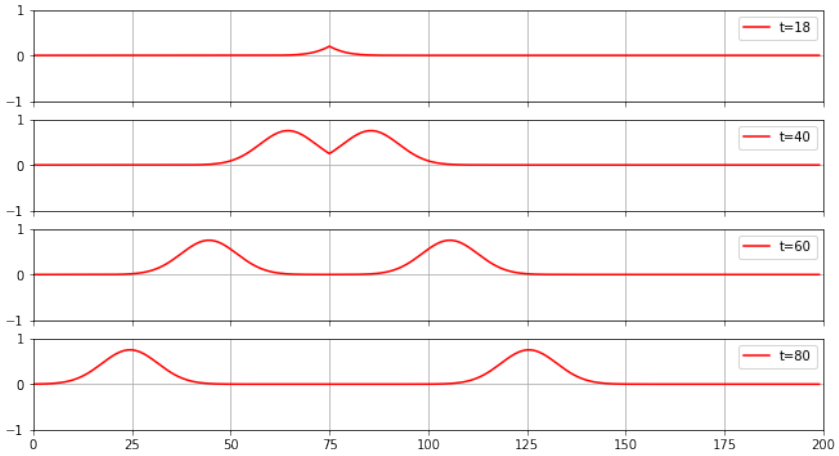
$$E_z[0] = E_z[1]$$

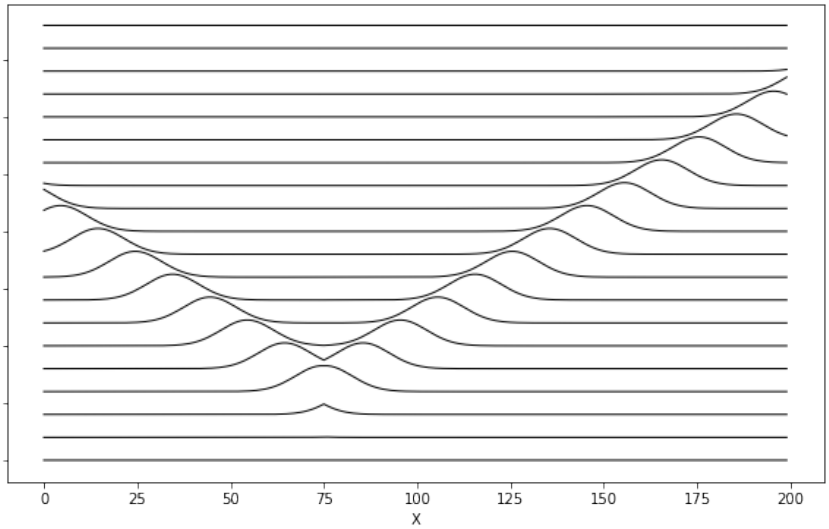
podobne pre pravý okraj

$$H_y[nodes-1] = H_y[nodes-2]$$

Uvedená forma okrajových podmienok platí len v jednoduchom jednorozmernom prípade a je zrejmé, že závisia od poradia výpočtu zložiek poľa v simulačnom kroku.

```
%reset -f
%matplotlib inline
from scipy import *
from utils.utils import *
import matplotlib.pyplot as plt

nodes = 200
time = 200
ez = zeros(nodes)
hy = zeros(nodes)
z0 = 377.0

r = zeros((time, nodes))                              # pole vysledkov

for t in range(time):
    ez[0] = ez[1]
    for m in range(1,nodes):                          # 0...199
        ez[m] = ez[m] + (hy[m] - hy[m - 1]) * z0      # aktualizacia Ez
    ez[75] += exp(-(t - 30.)**2 / 100.)*1.5             # zdroj
    hy[nodes-1] = hy[nodes-2]
    for m in range(nodes-1):                          # 0...198
        hy[m] = hy[m] + (ez[m + 1] - ez[m]) / z0      # aktualizacia Hy
        r[t,:] += ez

tm_plot([18, 40, 60, 80], r)
wv_plot(time, nodes, r)
```


t=18
t=40
t=60
t=80
1
0
−1
0
25
50
75
100
125
150
175
200


0
25
50
75
100
125
150
175
200
X

Kapitola 14

Nehomogénne dielektrické prostredie

Úpravou sústavy rovníc z odvodenia Yee algoritmu pre 1D prostredie dostaneme

$$\mu_r \mu_0 \frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\partial x}$$
$$\epsilon_r \epsilon_0 \frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x}$$

z čoho dostaneme aktualizačné rovnice v tvare

$$H_y^{q+1/2}[m+1/2] = H_y^{q-1/2}[m+1/2] + \left(E_z^q[m+1] - E_z^q[m]\right)\frac{S_c}{\mu_r}$$

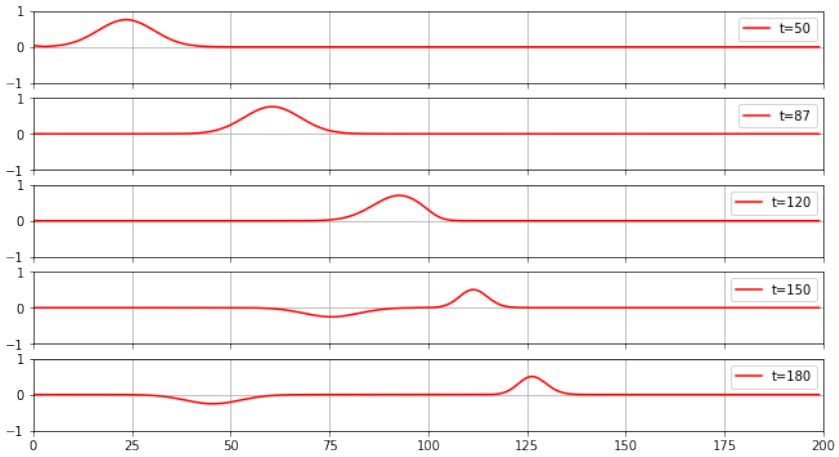
$$E_z^{q+1}[m] = E_z^q[m] + \left(H_y^{q+1/2}[m+1/2] - H_y^{q+1/2}[m-1/2]\right)\frac{S_c}{\epsilon_r}$$

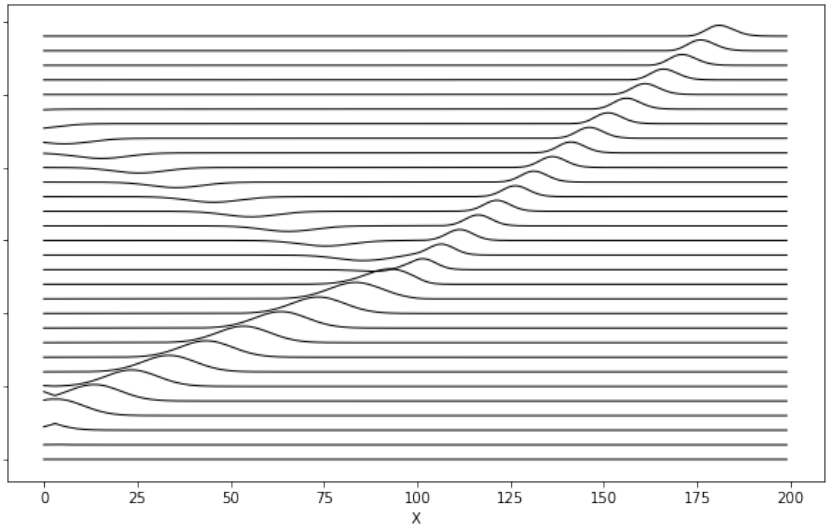
Rozšírenie simulácie o nehomogenity je jednoduché a spočíva v rozšírení o pole hodnôt ϵ_r (pre dielektrické nehomogenity). V aktualizačných rovniciach príslušnému uzlu pripadíme zodpovedajúcu hodnotu ϵ_r.

```
ez[m] = ez[m] + (hy[m] - hy[m - 1]) * z0 * Sc / epsr[m]
```

```
%reset -f
%matplotlib inline
from scipy import *
import matplotlib.pyplot as plt

nodes = 200
time = 300
ez = zeros(nodes)
hy = zeros(nodes)
epsr = ones(nodes)

for i in range(100,nodes):
    epsr[i] = 4
z0 = 377.0
Sc = 1.0

r = zeros((time, nodes))                    # pole vysledkov

for t in range(time):
    ez[0] = ez[1]
    for m in range(1,nodes):                # aktualizacia Ez
        ez[m] = ez[m] + (hy[m] - hy[m - 1]) * z0 * Sc / epsr[m]
        ez[3] += exp(-(t - 30.)**2 / 100.)*1.5
    for m in range(nodes-1):                # aktualizacia Hy
        hy[m] = hy[m] + (ez[m + 1] - ez[m]) / z0 * Sc
        r[t,:] += ez
```

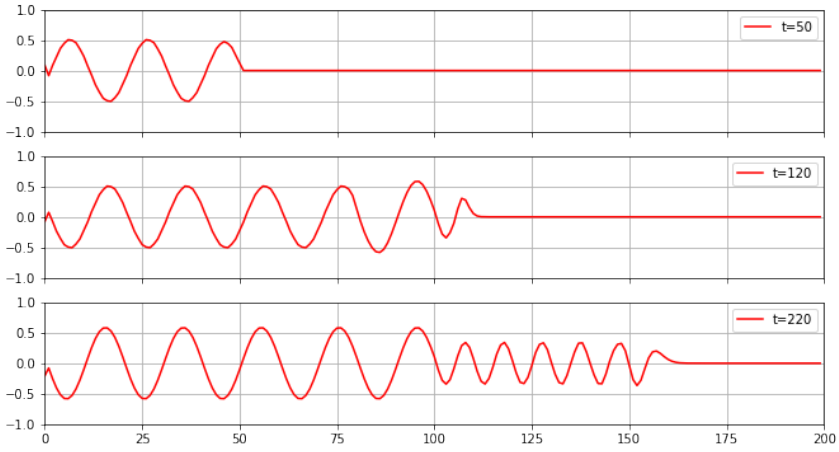
```
from utils.utils import *
tm_plot([50, 87, 120, 150, 180], r)
wv_plot(time, nodes, r)
```






```
%reset -f
%matplotlib inline
from scipy import *
import matplotlib.pyplot as plt

nodes = 200
time = 300
ez = zeros(nodes)
hy = zeros(nodes)
epsr = ones(nodes)

for i in range(100,nodes):
    epsr[i] = 4
z0 = 377.0
Sc = 1.0

r = zeros((time, nodes))                    # pole vysledkov
```

```
for t in range(time):
    ez[0] = ez[1]
    for m in range(1,nodes):                # aktualizacia Ez
        ez[m] = ez[m] + (hy[m] - hy[m - 1]) * z0 * Sc / epsr[m]
        #ez[3] += exp(-(t - 30.)**2 / 100.)*1.5
    ez[1] += sin(2*pi*0.05*t)*(1-exp(-t*0.5))
    for m in range(nodes-1):                # aktualizacia Hy
        hy[m] = hy[m] + (ez[m + 1] - ez[m]) / z0 * Sc
        r[t,:] += ez

from utils.utils import *
tm_plot([50, 120, 220], r)
#wv_plot(time, nodes, r)
```



14.1 Nejednoznačnosť hranice medzí médiami

14.2 Koeficient odrazu a prechodu

Koeficienty odrazu Γ a prechodu T pre elektromagnetickú vlnu prechádzajúcu z prostredia s charakteristickou impedanciou z_1 do prostredia s impedanciou sú definované ako

$$\Gamma = \frac{z_2 - z_1}{z_2 + z_1}$$

$$T = \frac{2\,z_2}{z_2 + z_1}$$

V našom prípade pre $z_1 = z_0$, $z_2 = z_0/2$,

$$\Gamma = \frac{\frac{z_0}{2} - z_0}{\frac{z_0}{2} + z_0} = -\frac{1}{3}$$

$$T = \frac{2\,\frac{z_0}{2}}{\frac{z_0}{2} + z_0} = \frac{2}{3}$$

Kapitola 15

Stratové dielektrické médium

Pre materiál s konečnou vodivosťou σ rozšírime Ampérov zákon o člen reprezentujúci prúd materiálom vytváraný elektrickým poľom. Tento člen je významom odlišný od prúdu aditívneho zdroja elektromagnetického poľa z predchádzajúcej kapitoly.

$$\sigma\,\mathbf{E} + \epsilon\frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{H} \tag{15.1}$$

Rozpísaním do zložiek pre jednorozmerný prípad dostaneme

$$\sigma\,E_z + \epsilon\frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x} \tag{15.2}$$

Pre odvodenie aktualizačnej rovnice potrebujeme rozvinúť vzťah okolo priestorovo-časovej súradnice $[(m\Delta x, (q + 1/2)\,\Delta t]$.Pre časové a priestorové derivácie vieme použiť aproximačné vzťahy, v rovnici sa nám ale potom vyskytuje zložka E_z v čase $(q + 1/2)\,\Delta t$, ktorú ale nepoznáme.
TODO Obrazok
Tento problém môžeme obísť vypočítaním hodnoty E_z v čase $(q + 1/2)\,\Delta t$ ako priemeru hodnôt E_z v časoch $(q + 1)\,\Delta t$ a $q\,\Delta t$

$$E_z^{q+1/2}[m] \approx \frac{E_z^{q+1}[m] + E_z^{q}[m]}{2}$$

Dosadením a rozvojom rovnice do diskrétneho tvaru dostaneme

$$\sigma\frac{E_z^{q+1}[m] + E_z^{q}[m]}{2} + \epsilon\frac{E_z^{q+1}[m] - E_z^{q}[m]}{\Delta t} = \frac{H_y^{q+1/2}[m + 1/2] - H_y^{q+1/2}[m - 1/2]}{\Delta x}$$

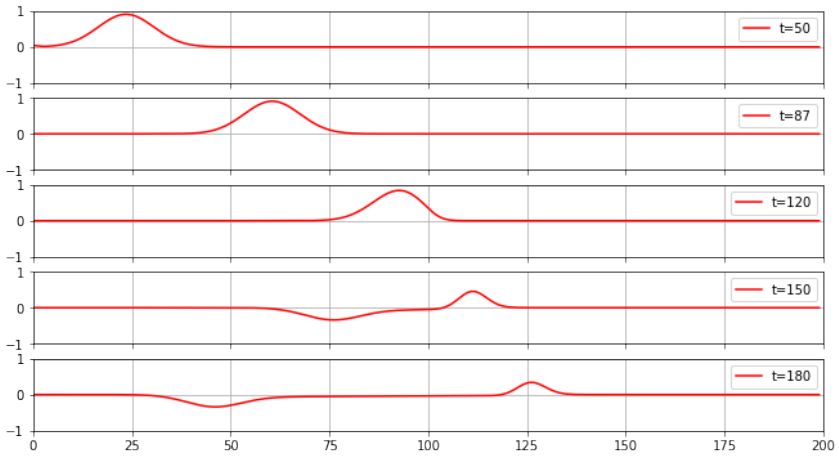
po úprave má pre hodnotu $E_z^{q+1}[m]$ aktualizačná rovnica tvar

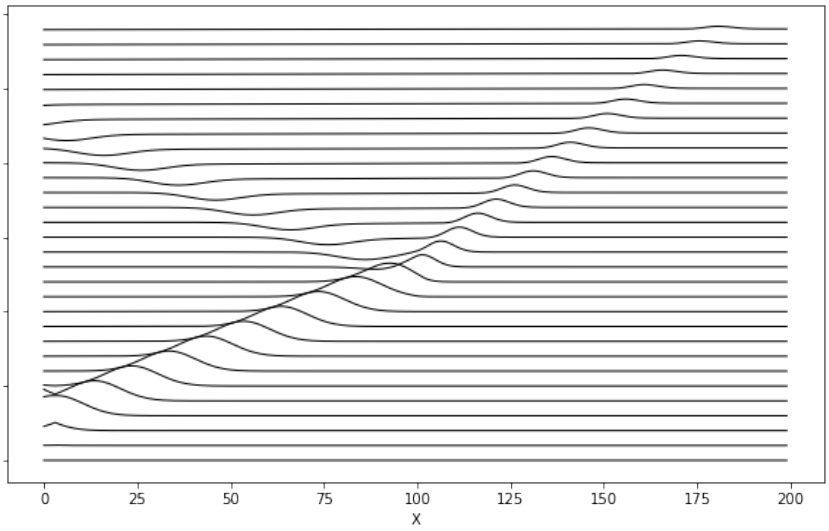
$$E_z^{q+1}[m] = \frac{1 - \frac{\sigma\Delta t}{2\epsilon}}{1 + \frac{\sigma\Delta t}{2\epsilon}}E_z^{q}[m] + \frac{\frac{\Delta t}{\epsilon\,\Delta x}}{1 + \frac{\sigma\Delta t}{2\epsilon}}\left(H_y^{q+1/2}[m + 1/2] - H_y^{q+1/2}[m - 1/2]\right) \tag{15.3}$$

V predchádzajúcich kapitolách boli priestorovo-časové vzťahy vyjadrené pomerom pomocou Courantovho čísla, v rovnici sa ale vyskytuje člen $\sigma\,\Delta t/2\,\epsilon$, ktorý vyžaduje explicitnú deklaráciu časového kroku, neskôr si ukážeme, ako je možné toto obmedzenie eliminovať s využitím niektorých materiálových parametrov.

```
%reset -f
%matplotlib inline
from scipy import *
from utils.utils import *
import matplotlib.pyplot as plt

nodes = 200
time = 300
ez = zeros(nodes)
hy = zeros(nodes)
loss = 0.01
z0 = 377.0
Sc = 1.0
```

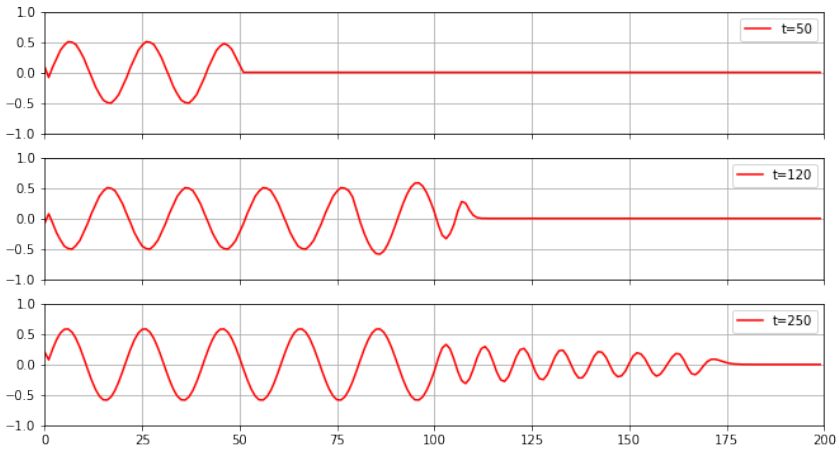
```
lse = ones(nodes)
lsh = ones(nodes)*z0

for i in range(100,nodes):
    lse[i] = (1 - loss) / (1 + loss)
    lsh[i] = z0 / (1 + loss) / 4.
r = zeros((time, nodes))                                    # pole vysledkov

for t in range(time):
    ez[0] = ez[1]
    for m in range(1,nodes):                                  ez[m] = lse[m] * ez[m] +
   lsh[m] *(hy[m] - hy[m - 1]) #* z0 # * Sc    # aktualizacia Ez
        ez[3] += exp(-(t - 30.)**2 / 100.)*1.8
    for m in range(nodes-1):                                 hy[m] = hy[m] + (ez[m + 1]
     - ez[m]) / z0 #* Sc     # aktualizacia Hy
        r[t,:] += ez

tm_plot([50, 87, 120, 150, 180], r)
wv_plot(time, nodes, r)
```






```
%reset -f
```

```
%matplotlib inline
from scipy import *
import matplotlib.pyplot as plt

nodes = 200
time = 300
ez = zeros(nodes)
hy = zeros(nodes)
loss = 0.005
z0 = 377.0
Sc = 1.0

lse = ones(nodes)
lsh = ones(nodes)*z0

for i in range(100,nodes):
    lse[i] = (1 - loss) / (1 + loss)
    lsh[i] = z0 / (1 + loss) / 4.
r = zeros((time, nodes))                                    # pole vysledkov

for t in range(time):
    ez[0] = ez[1]
    for m in range(1,nodes):                                  ez[m] = lse[m] * ez[m] +
    lsh[m] *(hy[m] - hy[m - 1]) #* z0 # * Sc    # aktualizacia Ez
        ez[1] += sin(2*pi*0.05*t)*(1-exp(-t*0.5))  #exp(-(t - 30.)**2 / 100.)
    *1.8
    for m in range(nodes-1):                                 hy[m] = hy[m] + (ez[m + 1]
     - ez[m]) / z0 #* Sc     # aktualizacia Hy
        r[t,:] += ez

from utils.utils import *
tm_plot([50, 120, 250], r)
```



Kapitola 16

MEEP

16.1 Inštalácia

Podrobný postup pre inštaláciu FDTD simulačného frameworku *meep* sa nachádza v dokumentácii MIT Electromagnetic Equation Propagation

16.2 Transformácie elementárnych veličín

16.2.1 Dĺžka

Všetky jednotky používané v *meep* sú relatívne. Základom pre konverziu veličín do reálnych hodnôt je jednotka dĺžky, v ktorej udávame rozmery oblasti, veľkosti geometrických objektov atď.

$$[l] = a$$

Napr. jedna dĺžková jednotka v *meep* môže byť reprezentovaná ako $a = 1\mu m,\ 5cm \ldots$

16.2.2 Čas

Rýchlosť svetla má v *meep* hodnotu $c = 1$, takže elektromagnetická vlna sa prejde jednu dĺžkovú jednotku za časovú jednotku

$$[t] = \frac{a}{c} = 1$$

Pretože

$$c = \frac{1}{\epsilon_0 \mu_0} \quad \longrightarrow \quad \epsilon_0 \mu_0 = 1$$

16.2.3 Prúd a hmotnosť

Pre transformáciu odvodených veličín je potrebné poznať transformačné vzťahy pre prúd a hmotnosť.
Prúd má v *meep* hodnotu $I = 1$. Transformačný vzťah pre hmotnost odvodíme z Coloumbovho zákona

$$F = \frac{1}{4\pi\epsilon_0}\frac{Q_1 Q_2}{r^2} \quad \longrightarrow \quad [kg\, m\, s^{-2}] = \frac{1}{\epsilon_0}\Big[\frac{A^2 s^2}{m^2}\Big]$$

$$M\, a\, t^{-2} = \frac{I^2\, t^2}{\epsilon_0\, a^2}$$

$$M = \frac{I^2\, t^4}{\epsilon_0\, a^3} = \frac{I^2\, a}{\epsilon_0\, c^4}$$

16.3 Transformácie odvodených veličín

16.3.1 Intenzita elektrického poľa E

Z definičného vzťahu vyplýva

$$E = \frac{F}{Q} \quad \longrightarrow \quad \frac{M\,a}{I\,t^3} = \frac{I^2\,a}{\epsilon_0\,c^4}\frac{a}{I\,t^3} = \frac{I}{\epsilon_0\,a\,c}$$

Transformačný vzťah pre hodnoty v SI sústave po dosadení numerických hodnôt

$$E_{SI} = E\,\frac{I}{a}\,3.7673 * 10^2\,[\frac{V}{m}]$$

16.3.2 Elektrická indukcia D

$$D = \epsilon_0\,E \quad \longrightarrow \quad \frac{I}{a\,c}$$

$$D_{SI} = D\,\frac{I}{a}\,3.3356 * 10^{-9}\,[\frac{C}{m^2}]$$

Kapitola 17

MEEP - Simulácia v 1D

Princíp simulácie šírenia sa elektromagnetickej vlny v prostredí pomocou simulátora *meep* ukážeme na jednoduchom príklade. Jednotlivé časti API simulátora budú podrobnejšie rozobrané v nasledujúcich kapitolách.

17.1 Simulácia odrazu elektromagnetickej vlny od ideálne vodivej prekážky.

Prostredie je tvorené lineárnou postupnosťou uzlov reprezentujúcich oblasť simulácie v smere osi Z, na krajoch je ohraničené ideálnym absorbérom PML, vo vnútri oblasti sa na jednom konci nachádza generátor impulzu so zložkou E_x a na druhom dokonalý kovový vodič M s $\epsilon_r = \infty$. Uzly prostredia označené V reprezentujú ideálne vákuum.

Obr. 17.1: Konfigurácia jednorozmerného simulačného prostredia

17.1.1 Inicializácia simulátora

```
%reset -f
%matplotlib inline

import meep as mp
import pylab as plt
from scipy import *
```

```
Using MPI version 3.0, 1 processes
```

17.1.2 Definícia oblasti simulácie

Definícia oblasti simulácie pozostáva z definovania rozmerov simulačnej oblasti a vytvorenia zoznamu objektov oblasti. Objekty presahujúce rozmery sú ignorované. Okolo simulovanej oblasti (z vnútornej strany) je možné vytvoriť ideálny absorbér s definovanou hrúbkou.

```
size = 200                          # pocet buniek v simulacii
res = 20                            # rozlisenie - rozdelenie buniek

area = mp.Vector3(0, 0, size)       # definovanie rozmerov oblasti simulacie (1D)
                                    # v smere osi Z
```

```
geometry = [                        # definicia vlastnosti prostredia pomocou 1D
    blokov
    mp.Block(mp.Vector3(0,0,size-2), center=mp.Vector3(0,0,0),          material
    =mp.vacuum),
    mp.Block(mp.Vector3(0,0,1),      center=mp.Vector3(0,0,size/2-1.5), material
    =mp.metal)
  ]

pml_layers = mp.PML(1.0)            # idealny obsorber okolo oblasti simulacie
```

17.1.3 Vytvorenie a definícia zdrojov

Okrem sady preddefinovaných typov zdrojov je možné v *meep* definovať aj zdroje vlastné, v našom prípade zdroj generujúci impulz posunutý v čase on t_0 v tvare Gaussovej krivky o šírke w.

$$s(t) = exp\,(-\frac{(t-t_0)^2}{w}) \tag{17.1}$$

Zdrojov môže byť v simulovanom prostredí niekoľko, v definícii určíme polohu zdroja, zložku elektromagnetickej vlny a amplitúdu. Zdroje zaraddíme do zoznamu, ktorý sa odovzdá v parametroch objektu simulácie.

```
def pulse(t):
    return -2*exp(-(t -20.)**2 / 20.0)

sources = [mp.Source(mp.CustomSource(src_func=pulse), component=mp.Ex,
                     center=mp.Vector3(0, 0, -size/2 + 1),
                     amplitude=1.0)
          ]
```

17.1.4 Vytvorenie objektu simulácie

Rozmery simulačnej oblasti, zoznam objektov v oblasti a zdrojov odovzdáme ako parametre triede *Simulation*.

```
sim = mp.Simulation(                # vytvorenie objektu simulacie s parametrami
    cell_size=area,                 geometry=geometry,
    sources=sources,
    boundary_layers=[pml_layers],
    resolution=res,
    dimensions=1)
```

17.1.5 Simulácia šírenia sa impulzu

Simuláciu môžeme spustiť pomocou metódy *run* s argumentom určujúcim dobu simulácie.

```
sim.run(until=70)      # doba simulacie 200 krokov, impul je oneskoreny 20
    jednotiek
data_1 = sim.get_efield_x()
```

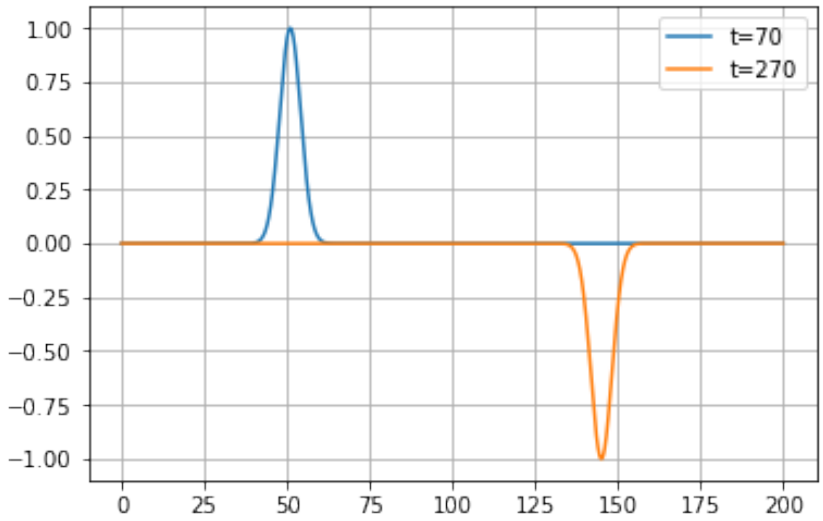
```
-----------
Initializing structure...
run 0 finished at t = 70.0 (2800 timesteps)
```

```
sim.run(until=200)      # pokracovanie simulacie o dalsich 200 krokov
data_2 = sim.get_efield_x()
```

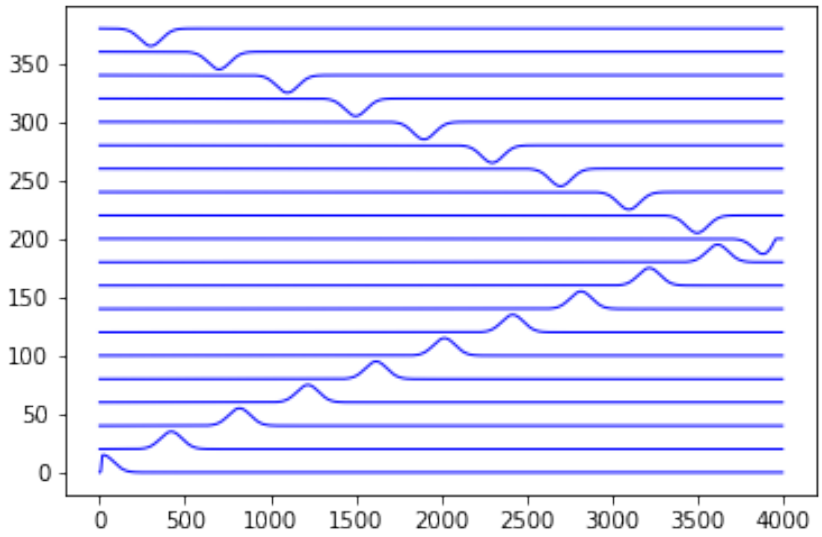
```
run 1 finished at t = 270.0 (10800 timesteps)
```

17.2 Vizualizácia výsledkov

```
x = arange(len(data_1))/res

plt.plot(x, real(data_1), label='t=70')  # impulz v case 70 jednotiek, poloha 52
    - zdroj je na pozicii [1]
plt.plot(x, real(data_2), label='t=270')  # impulz v case 270, poloha 158 -
    dlzka oblasti je 200 - 4
plt.grid()
plt.legend()
plt.show()
```




```
from IPython.utils import io

sim.reset_meep()                    # restart simulacie

fig = plt.figure() #figsize=(10, 7), facecolor='r')
ax = fig.add_subplot(111)

for i in range(20):
    with io.capture_output() as captured:
        sim.run(until=20);      # pokracovanie simulacie o dalsich 200 krokov
    data = real(sim.get_efield_x()) * 15
    ax.plot(data +i*20, 'b', lw=1)
plt.show()
```


350
300
250
200
150
100
50
0
0
500
1000
1500
2000
2500
3000
3500
4000

Kapitola 18

Zdroje

Vlastnosti zdrojov budeme skúmať na 2D oblasti o rozmere 100 x 100 uzlov, v oblasti sa okrem zdroja nebudú vyskytovať žiadne iné objekty. Okraj oblasti bude pre zamedzenie odrazov pokrytý prispôsobenou vrstvou PML o hrúbke 5 nodov. Zdroj elektrického poľa so zložkou E_z bude umiestnený v strede oblasti.

18.1 ContinuousSource

Zdroj *ContinuousSource* reprezentuje spojitý harmonický zdroj elektromagnetickej energie.

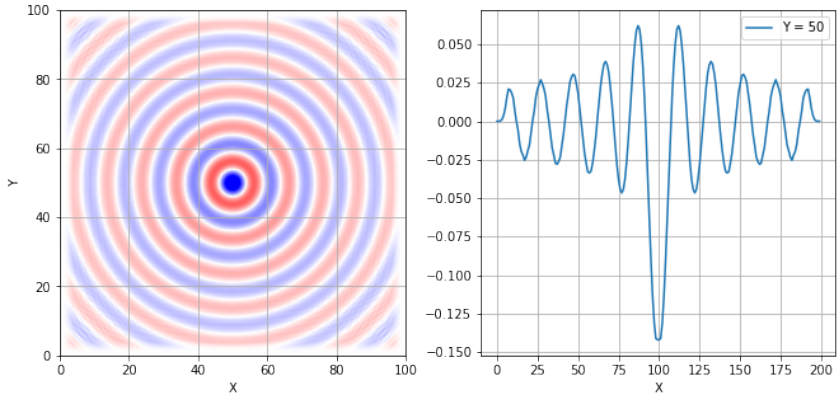
```
%reset -f
%matplotlib inline

import meep as mp
import matplotlib.pyplot as plt

dx = 100
dy = 100
area = mp.Vector3(dx, dy, 0)

sources = [ mp.Source(mp.ContinuousSource(frequency=0.1), #, width=5),
                      component=mp.Ez, center=mp.Vector3(0,0,0) ) ]
                  sim = mp.Simulation(cell_size=area, boundary_layers=[mp.PML
    (5.0)],
                      geometry=[], sources=sources, resolution=2)

sim.run(until=80)
data = sim.get_array(center=mp.Vector3(), size=area, component=mp.Ez)
```

```
-----------
Initializing structure...
run 0 finished at t = 80.0 (320 timesteps)
```

Zobrazenie zložky pola E_z v čase $t = 80$ a prierez v strede plochy.

```
fig = plt.figure(figsize=(11,5))
ax = fig.add_subplot(121)

plt.imshow(data.transpose(), interpolation='spline36', cmap='bwr', alpha=1.0,
           extent=(0, dx, 0, dy), norm=plt.Normalize(-0.1, 0.1))
plt.xlabel('X')
plt.ylabel('Y')
plt.grid()

ax = fig.add_subplot(122)
plt.plot(data[100,:], label='Y = 50')
```

```
plt.xlabel('X')
plt.grid()
plt.legend()
plt.show()
```



18.2 GaussianSource

Zdroj *GausianSource* reprezentuje impulz elektromagnetickej energie s nulovou strednou hodnotou.

```
%reset -f
%matplotlib inline

import meep as mp
import matplotlib.pyplot as plt

dx = 100
dy = 100
area = mp.Vector3(dx, dy, 0)

sources = [ mp.Source(mp.GaussianSource(frequency=0.1, width=5),
                      component=mp.Ez, center=mp.Vector3(0,0,0) ) ]
                  sim = mp.Simulation(cell_size=area, boundary_layers=[mp.PML
    (5.0)],
                      geometry=[], sources=sources, resolution=2)

sim.run(until=50)
data = sim.get_array(center=mp.Vector3(), size=area, component=mp.Ez)
```

```
Using MPI version 3.1, 1 processes
-----------
Initializing structure...
run 0 finished at t = 50.0 (200 timesteps)
```

```
fig = plt.figure(figsize=(11,5))
ax = fig.add_subplot(121)

plt.imshow(data.transpose(), interpolation='spline36', cmap='bwr', alpha=1.0,
           extent=(0, dx, 0, dy), norm=plt.Normalize(-0.1, 0.1))
plt.xlabel('X')
plt.ylabel('Y')
plt.grid()
```

```
ax = fig.add_subplot(122)
plt.plot(data[100,:], label='Y = 50')
plt.xlabel('X')
plt.grid()
plt.legend()
plt.show()
```