EBFieldSolver Dokumentation

Johannes Tormöhlen

16. August 2021

Die Skripte im `EBFieldSolver` sind zur einfachen Erstellung eigener Darstellungen von statischen und dynamischen Feldern konzipiert. Dazu werden vordefinierte felderzeugende Objekte instanziiert, sodass ein Feld durch Superposition der Teilfelder berechnet und dargestellt werden kann. Alle verfügbaren Skripte sind erläutert in Tab. 1. Die relevanten Skripte sind durch eine Textbox hervorgehoben.

Skript	Erläuterung
`FieldObject.py`	Definition von felderzeugenden Objekten: Punktladung q mit Position in Ruhe oder Bewegung $v \geq 0$. Antenne mit Frequenz f_0 und Strahlungsleistung P für ein Punktdipol mit Länge $L = 0$ oder Stabdipol mit nicht-ganzzahliger Länge $L \neq l\lambda_0$ mit $l \in \mathbb{Q}_+$.
`FieldOperator.py`	Numerische Berechnung des Gradienten, der Divergenz und der Rotation einer Feldfunktion f mit dem Nabla-Operator für einen Punkt (x_0, y_0, z_0). Transformation von sphärischen (r, ϑ, φ) in kartesische Koordinaten (x, y, z).
`FieldCalculator.py`	Berechnung von statischen oder dynamischen Gesamtfeldern für alle felderzeugenden Objekte für alle Punkte im Mesh von $-xyz$ bis xyz mit Skalierung n_{xyz}. Normierung und Anpassung von Pfeildarstellungen.
`FieldPlot.py`	Darstellung von statischen oder dynamischen Feldern f mit verschiedenen Varianten: Höhenlinien, Stromlinien, Vektorpfeile und Intensitäten $\lvert f\rvert^2$.
`FieldAnimation.py`	Initialisierung von Achsen mit Beschriftung und Rendering einzelner Bilder oder Animationen.
`StaticField.py`	Instanziierung von Ladungen q_i mit Position (x_i, y_i, z_i) und evtl. Geschwindigkeit $(v_{i_x}, v_{i_y}, v_{i_z})$. Aufrufen der Berechnungs- und Darstellunngs-funktion für einen Flächen- oder Raumbereich.
`DynamicField.py`	Instanziierung von Antenne mit Frequenz f_0, Strahlungsleistung P und Längenfaktor $l > 0$ für einen Stabdipol und Punktdipol sonst. Aufrufen der Berechnungs- und Darstellungs-funktion für einen Flächenbereich.

Tabelle 1: Übersicht der Skripte [3]

Anleitung zur Erstellung eines E-B-Feldbildes

1 Deklaration einer elektrischen Ladung

Damit ein Feldlinienbild geplottet werden kann, braucht es ein Klassenobjekt welches ein Feld erzeugen kann. Eine elektrische Punktladung q erzeugt im Vakuum [2] das elektrische Feld $\vec{E}$ mit zugehörigem Potential ϕ:

$$\vec{E}(\vec{r}) = \frac{1}{4\pi\varepsilon_0}\frac{q(\vec{r}-\vec{r}_0)}{|\vec{r}-\vec{r}_0|^3}$$
$$\phi(\vec{r}) = \frac{1}{4\pi\varepsilon_0}\frac{q}{|\vec{r}-\vec{r}_0|}$$

mit elektrischer Feldkonstante ε_0, Betrachtungspunkt $\vec{r}$ und Ladungsposition $\vec{r}_0$. In `FieldObject.py` ist eine Klasse unter der Bezeichnung `Charge` definiert. Für eine Punktladung genügt es die Ladung `q` und die Komponenten `x`, `y` und `z` der Position an den Konstruktor zu übergeben. Standardmäßig wird die z-Komponente und die Geschwindigkeitskomponenten `v_x`, `v_y` und `v_z` gleich Null gesetzt, sodass von einer 2D-Darstellung und einer ruhenden Ladung ausgegangen wird. Ein Objekt vom Typ `Charge` mit dem Namen `charge` kann mit folgender Anweisung erzeugt werden

```
charge = Charge(q, x, y, z)
```

Das elektrische Feld und Potential werden über die Klassenfunktionen `E()` und `phi()` berechnet. Die x-, y- und z-Positionen werden als Betrachtungspunkte übergeben und in einem Array gespeichert, welches wie ein Vektor verwendet wird.

2 Berechnung und Darstellung eines Feldes

Die Ladung kann im Skript `StaticField.py` instanziiert werden. Zuerst wird eine leere Liste unter dem Objektnamen `charges` deklariert. Diese Liste wird mit der Funktion `append()` um Instanzen von `Charge` erweitert. Das elektrische Feld und Potential werden durch Superposition der einzelnen Felder für beliebig viele Ladungen innerhalb eines x-y-Wertebereichs von `-xy_max` bis `xy_max` und einer Skalierung `n_xy` in `FieldCalculator.py` berechnet. Letztere bestimmt wie viele Punkte im Wertebereich zur Berechnung einbezogen werden. Die Erzeugung eines elektrischen Quadrupols mit der Ausdehnung $[-1,1]_x \times [-1,1]_y$ erfolgt durch den nachfolgenden Code.

```
# ++++++++++++++++++++++
# ++++StaticField.py++++
# ++++++++++++++++++++++
q = x = y = 1.0

quadrupole = []
quadrupole.append(Charge(-q, -x, y))
quadrupole.append(Charge(q, x, y))
quadrupole.append(Charge(q, -x, -y))
quadrupole.append(Charge(-q, x, -y))
```

Für die grafische Darstellung von Feldern gibt es in `FieldPlot.py` drei Optionen zur Auswahl. Voreingestellt sind je nach Feldtyp einer der folgenden Darstellungen:

- `potential_lines()`: Höhenlinien in grau eignen sich zur Darstellung von Potentialen. Negative Potentiale werden gestrichelt und positive durchgezogen dargstellt.
- `field_lines()`: Stromlinien in Farbe eignen sich zur Darstellung von statischen Vektorfeldern. Abhängig von der Feldstärke werden Linien im Farbspektrum magenta bis cyan eingefärbt.
- `arrow_field()`: Vektorpfeile eignen sich zur Darstellung von dynamischen Vektorfeldern. Die Farbe der Vektorpfeile hängt vom Feldttyp ab und kann selbst definiert werden. Die Länge der Vektorpfeile kann normiert oder individuell angepasst werden.

Die Berechnung und Darstellung des E-Feldes erfolgt durch Aufruf der Funktion `static_field_2d()` für einen Wertebereich $[-5,5]_x \times [-5,5]_y$ wie nachfolgend gezeigt.

```
xy_max = 5.
static_field_2d(xy_max, quadrupole, function='E')
```

Neben dem Wertebereich muss die Liste von Objekten unter `charges` und die Funktion `function='E'` aus der Klasse `Charge` übergeben werden. Die zu berechnende Funktion muss immer als String übergeben werden. Das elektrische Potential wird analog zum elektrischen Feld berechnet, wobei dort `function='phi'` übergeben wird. Für die Potentiallinien geht nur das skalare Feld aus der x-Komponente und für die Feldlinien des elektrischen Feldes die x- und y-Komponenten ein.

∇-Operator

Falls vom definierten Objekt nur das Potential bekannt ist, kann mit Hilfe des ∇-Operator das korrespondierende Feld berechnet werden. Aus den MAXWELL-Gleichungen folgt für das elektrische Feld einer ruhenden Punktladung und das Magnetfeld einer bewegten Punktladung

$$\vec{E}(\vec{r}) = -\vec{\nabla} \cdot \phi(\vec{r})$$
$$\vec{B}(\vec{r}) = \vec{\nabla} \times \vec{A}(\vec{r})$$

wobei $\vec{\nabla} = \partial_x \vec{e}_x + \partial_y \vec{e}_y + \partial_z \vec{e}_z$ mit $\partial_f := \partial/\partial f$ definiert ist. Im folgenden Auschnitt ist ein Beispiel für die Berechnung des elektrischen Feldes aus dem Potential.

```
static_field_2d(xy_max, quadrupole, function='phi', nabla='gradient')
```

Das Argument `nabla` in der Funktion `static_field_2d()` kann mit den Schlüsselwörtern `'gradient'` für den Gradienten, `'divergence'` für die Divergenz und `'curl'` für die Rotation eines Feldes in `function` übergeben werden.

3 Magnetfeld einer bewegten Ladung

Eine bewegte Punktladung q mit der Geschwindigkeit $\vec{v}$ induziert im Vakuum [2] das Magnetfeld $\vec{B}$ mit zugehörigem Vektorpotential $\vec{A}$

$$\vec{B}(\vec{r}) = \frac{\mu_0}{4\pi} \frac{q \cdot \vec{v} \times (\vec{r} - \vec{r}_0)}{|\vec{r} - \vec{r}_0|^3}$$
$$\vec{A}(\vec{r}) = \frac{\mu_0}{4\pi} \frac{q \cdot \vec{v}}{|\vec{r} - \vec{r}_0|}$$

mit magnetischer Feldkonstante μ_0. Die Klasse `Charge` wird um den Parameter der Geschwindigkeit erweitert. Im Konstruktor werden die Komponenten `v_x`, `v_y` und `v_z` zusätzlich übergeben und die Funktionen zur Berechnung der magnetischen Flussdichte `B()` und des Vektorpotentials `A()` definiert. Die z-Komponente des Ortes kann nicht mehr weggelassen werden, da ansonsten eine Geschwindigkeitskomponente fehlt. Wegen der Geometrie zwischen Magnetfeld und Ladung, müssen bei der Instanziierung drei Dimensionen berücksichtigt werden. Im nächsten Skriptausschnitt ist das Magnetfeld durch eine stromdurchflossene Leiterschleife in der y-z-Ebene simuliert. Der Radial- und Tangential- bzw. Orts- und Geschwindigkeitsvektor werden über die Beziehungen

$$\vec{r}(t) = \cos t \vec{e}_y + \sin t \vec{e}_z$$
$$\vec{v}(t) = \dot{\vec{r}}(t)$$
$$= -\sin t \vec{e}_y + \cos t \vec{e}_z$$

komponentenweise bestimmt, wobei jedem Zeitpunkt ein Winkel $\varphi = t$ zugeordnet wird. Die Ladungen haben jeweils den Wert `q=1` und werden auf einem Einheitskreis mit Mittelpunkt im Ursprung verteilt.

```
# ++++++++++++++++++++++
# ++++StaticField.py++++
# ++++++++++++++++++++++
q = 1.0
r_x = v_x = 0.
```

```
loop = []
for angle in np.linspace(0, 2. * np.pi, 32, endpoint=False):
    r_y = np.cos(angle)
    r_z = np.sin(angle)
    v_y = -np.sin(angle)
    v_z = np.cos(angle)
    loop.append(Charge(1., r_x, r_y, r_z, v_x, v_y, v_z))
```

Die Winkelpositionen werden durch Iteration auf einem Array mit `n` Zahlen zwischen `[0., 2.*np.pi]` bestimmt, wobei `n` das dritte Argument in der Funktion `np.linspace()` ist. Der Endpunkt wird nicht berücksichtigt da er auf dem Startpunkt liegt und somit nur einmal verwendet werden soll. Die `r0`- und `v`-Tupel werden dem Konstruktor der Klasse `Charge` übergeben und die erzeugten Objekte an die Liste `loop` angehängt. Die Berechnung und die Darstellung des Feldes im Skript erfolgen mit

```
static_field_2d(xy_max, loop, function='B')
```

Neben dem Wertebereich in der x-y-Ebene `xy_max` werden die Ladungen auf dem Ring in `loop` übergeben und das B-Feld mit dem Argument `function='B'` berechnet und dargestellt.

3D-Felder

Um die vorherige Projektion des Magnetfeldes der Leiterschleife auf die x-y-Ebene zu überprüfen gibt es die Möglichkeit eines 3D-Plots. Durch geringfügige Modifikationen folgt

```
static_field_3d(xyz_max, loop, function='B')
```

Zur Berechnung des 3D-Feldes steht die Funktion `static_field_3d()` zur Verfügung. Die Darstellung des 3D-Plots funktioniert nur mit normierten Vektorpfeilen über die Funktion `arrow_field3d()`. Wieder werden der Funktion Wertebereich inkl. der z-Achse `xyz_max` und Leiterschleife in `loop` übergeben. Zum Plotten werden alle drei Komponenten des Feldes übergeben. Aus Gründen der Erkennbarkeit der einzelnen Vektorpfeile wird die Skalierung möglichst klein gewählt, sodass nur wenige Vektorpfeile dargestellt werden.

4 Dynamische Felder

4.1 Punktdipol

Die Funktion zur Berechnung eines Feldes ist auch in der Lage zeitabhängige Felder zu berechnen. Das elektromagnetische Feld, welches durch einen Hertzschen Dipol [1] erzeugt wird, ist ein oszillierendes Feld. Das Dipolmoment des punktförmigen Dipols schwingt entlang der z-Achse und wird berechnet aus

$$\vec{p}(t) = \vec{p}_0 \mathrm{e}^{-\mathrm{j}\omega t}$$
$$\vec{p}_0 = \frac{12\pi c P}{\mu_0 \omega^4} \vec{e}_z$$

mit Kreisfrequenz ω, Vakuumlichtgeschwindigkeit c, Strahlungsleistung P, Zeitpunkt t und $j^2 := -1$. Die vollständige mathematische Beschreibung des elektrischen und magnetischen Feldes $\vec{E}$ und $\vec{H}$ lauten

$$\vec{E}_\omega(\vec{r}, t) = \frac{\omega^3}{4\pi\varepsilon_0 c^3} \left((\vec{n} \times \vec{p}) \times \vec{n} \frac{1}{\rho} + (3\vec{n}(\vec{n} \cdot \vec{p}) - \vec{p}) \left(\frac{1}{\rho^3} - \frac{\mathrm{j}}{\rho^2} \right) \right) \mathrm{e}^{\mathrm{j}(\rho - \omega t)}$$
$$\vec{H}_\omega(\vec{r}, t) = \frac{\omega^3}{4\pi c^2} (\vec{n} \times \vec{p}) \left(\frac{1}{\rho} + \frac{\mathrm{j}}{\rho^2} \right) \mathrm{e}^{\mathrm{j}(\rho - \omega t)}$$

mit normiertem Betrachtungsvektor $\vec{n} = \vec{r}/|\vec{r}|$ und $\rho = (\omega r)/c$. Analog zu den stationären Objekten, wird auch hier ein neues Objekt unter dem Klassennamen `Antenna` definiert. Der schwingende Dipol ist durch seine Frequenz und abgestrahlten Leistung charakterisiert. Alle anderen Parameter, wie Wellenlänge, Periodendauer, Wellenzahl, Betrag des Dipolmoments etc. werden im Konstruktor berechnet und können über das Antennen-Objekt ausgegeben werden. Neben Funktionen zur Feldberechnung wird eine

Funktion `p()` zur Berechnung des Dipolmoments definiert. Aus Gründen der Lesbarkeit sind die Feldberechnungen in Teilterme zerlegt. Für dynamische Darstellungen werden immer Zeitfenster benötigt, die in Form eines Arrays angelegt werden. Das Zeitfenster enthält Einträge vom Startzeitpunkt `t_0=0` bis `t_max`, wobei Letzterer wie folgt selbst festgelegt wird.

```
# ++++++++++++++++++++++
# ++++DynamicField.py+++
# ++++++++++++++++++++++
frequency = 500.e6
power = 1.
antenna = Antenna(frequency, power)

xyz_max = 2 * antenna.lambda_0
t_max = antenna.T
```

Für ein periodisch strahlenden Dipol mit der Frequenz $f = 500\text{MHz}$ und einer Strahlungsleistung von $P = 1\text{W}$ wird eine Periodendauer T betrachtet. Über die Zeitpunkte wird dann iteriert und für jeden Zeitpunkt das Feld berechnet. Der Wertebereich enthält mindestens einen Wellenzug und wird daher mit mindestens einer Wellenlänge λ_0 gewählt. Die Berechnung und Darstellung der E- und H-Felder erfolgen mit der Funktion `dynamic_field()` und den Klassenfunktionen `E` und `H` in `Antenna`

```
dynamic_field(xyz_max, t_max, antenna, function='E')
dynamic_field(xyz_max, t_max, antenna, function='H')
dynamic_field(xyz_max, t_max, antenna, function='S')
```

Neben dem Werte- und Zeitfenster geht das Antennen-Objekt, dass im Ursprung liegt in die Funktion ein. Zusätzlich wird das Feld von Poynting-Vektoren mit dem Argument `function='S'` über die Beziehung $\vec{S} = \vec{E} \times \vec{H}$ berechnet und dargestellt. Physikalisch ist der Betrag des Feldes ein Maß für die Energiestromdichte.

4.2 Stabdipol

Die Linearantenne [1] gehört zur gebräuchlichsten Strahlerform. Ein dünner zylindrischer Leiter der Länge L paralell zur z-Achse wird symmetrisch in der Mitte elektrisch gespeist. Der Strom I_0 am Speisepunkt ist konstant während er an den offenen Enden verschwindet. Es wird angenommen, dass die symmetrische Stromverteilung auf dem Stab durch eine stehende Sinuswelle angenähert werden kann. Das zeitabhängige Fernfeld einer dünnen Linearantenne in sphärischen Koordinaten $\vec{F}(r, \vartheta, \varphi) = F_r\vec{e}_r + F_\vartheta\vec{e}_\vartheta + F_\varphi\vec{e}_\varphi$ wird berechnet aus

$$
\begin{aligned}
H_r &= H_\vartheta = 0 \\
H_\varphi &= jI_0 \frac{\mathrm{e}^{j(k_0 r - \omega t)}}{2\pi r} \frac{\cos(k_0 h \cos\vartheta) - \cos(k_0 h)}{\sin\vartheta} \\
E_r &= E_\varphi = 0 \\
E_\vartheta &= Z_0 H_\varphi
\end{aligned}
$$

mit dem reellen Feldwellenwiderstand des freien Raumes $Z_0 = \sqrt{\mu_0/\varepsilon_0} \approx 377\Omega$, Wellenzahl $k_0 = \omega/c$ und $h = L/2$. Multiplikation des Feldes mit einer Drehmatrix, zusammengesetzt aus dem Orthogonalsystem der Einheitsvektoren in Kugelkoordinaten, transformiert das Feld in kartesische Koordinaten.

$$
\begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} = \begin{pmatrix} \sin\vartheta\cos\varphi & \cos\vartheta\cos\varphi & -\sin\varphi \\ \sin\vartheta\sin\varphi & \cos\vartheta\sin\varphi & \cos\varphi \\ \cos\vartheta & -\sin\vartheta & 0 \end{pmatrix} \cdot \begin{pmatrix} F_r \\ F_\vartheta \\ F_\varphi \end{pmatrix}
$$

Der Punktdipol wird für den Fall einer Ausdehnung $L > 0$ erweitert, sodass für die Feldfunktionen `E`, `H` und `S` eine Fallunterscheidung nach Antennenlänge erfolgt. Standardmäßig ist der Längenfaktor des Antennen-Objekt $l = 0$ und wird in der Antennen-Klasse mit der Wellenlänge λ_0 multipliziert. Eine $\lambda/2$-Stabantenne wird also wie folgt instanziiert:

```
l = 0.5
antenna = Antenna(frequency, power, l)
```

wobei die Frequenz und Leistung des Punktdipols übernommen werden.

A Referenzen

[1] Klaus W. Kark. *Antennen und Strahlungsfelder: Elektromagnetische Wellen auf Leitungen, im Freiraum und ihre Abstrahlung*. 7., überarbeitete und erweiterte Auflage. Lehrbuch. Wiesbaden, Germany: Springer Vieweg, 2018. ISBN: 3658223189.

[2] Wolfgang Nolting. *Elektrodynamik*. 10. Aufl. Bd. 3, Ed. 10. Springer-Lehrbuch. Berlin: Springer Spektrum, 2013. ISBN: 3642379044.

[3] Johannes Tormöhlen. *EBFieldSolver*. 2021. URL: `https://github.com/jtormoehlen/EBFieldSolver`.

B Flussdiagramm