3D-visualisatie van een lineaire dipool

Korte Stage

Sietze van Buuren

22 juni 2004

Samenvatting

In dit verslag is een methode beschreven hoe een elektrisch, een magnetisch veld en de poynting vector van een elektrische dipool a.d.h.v. hun formules kunnen worden gevisualiseerd. Er is gekozen om deze weer te geven als een dichtheidsgrafiek op het oppervlak van een bol. Om de details van de formules weer te geven is naast een tijdsafhankelijke animatie ook gekozen voor een animatie waarbij geïntegreerd is over de tijd en de radius van de bol wordt gevariëerd. Het geheel is geïntegreerd in een graphical user interface(GUI).

Inhoudsopgave

1 Inleiding **3**

2 Theorie **4**

2.1 Visualisatie afhankelijk van de tijd 4

2.2 Visualisatie afhankelijk van de radius 6

3 Programmaopzet **7**

3.1 Databewerking 7

3.2 Dataplotting 7

3.3 Graphical User Interface(GUI) 9

4 Resultaten en discussie **11**

4.1 Visualisatie afhankelijk van de tijd 11

4.2 Visualisatie afhankelijk van de radius 12

5 Conclusie **13**

6 Dankwoord **14**

7 Referenties **15**

8 Appendices **16**

8.1 A 16

8.2 B 18

8.3 C 19

1 Inleiding

Van een antenne of een lineaire dipool wordt er in dit verslag een visualisatie gemaakt van het elektrische, het magnetische veld en de poynting vector. Hiervoor zijn de formules uit referentie [1] als aanleiding genomen. Doordat de tijdsafhankelijke visualisatie niet genoeg details van de formule blootlegde, is er tevens gekozen voor de radiusafhankelijke visualisatie waar dit wel het geval is. Hoe dit is gedaan zal blijken in hoofdstuk 2. Beide methoden worden op eenzelfde manier gevisualiseerd. De velden worden op een rooster van variërende lengte- en breedtegraden bepaald, waarna deze velden op een bepaalde vector worden geprojecteerd. Van die waarden kan dan een dichtheidsgrafiek worden gemaakt die op een boloppervlak wordt geprojecteerd corresponderend aan de eerder genoemde lengte- en breedtegraden.

2 Theorie

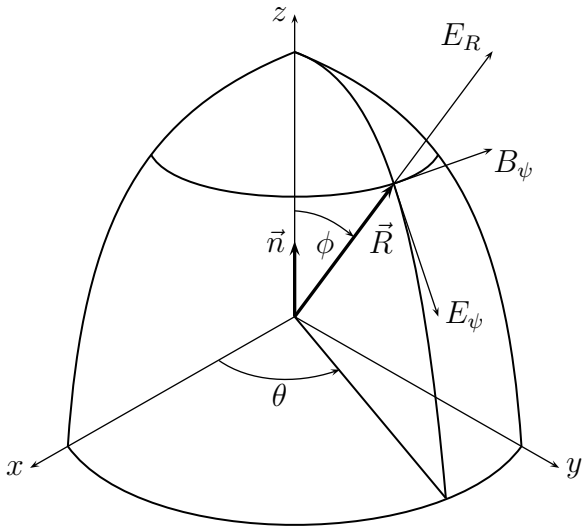
De afleiding voor de visualisatie afhankelijk van de radius volgt uit die van de tijd. Deze wordt wordt in 2.1 besproken, waarna in 2.2 de resterende afleidingen voor de radius aan bod komen.

2.1 Visualisatie afhankelijk van de tijd

Het dipool wordt in een carthesisch coördinaten in de oorsprong geplaatst. Voor de dipool geldt

$$P(\vec{r}, t) = p(t)\ \delta(\vec{r})\ \vec{n} \tag{1}$$

waarbij voor $p(t)$ in dit geval gekozen is voor $p(t) = p_0 \cos \Omega t$ en $\vec{n}$ de richting van de dipool is. De vector $\vec{r}$ is het punt van observatie.



Figuur 1: Het elektrische en magnetische veld op een afstand R van de dipool.

De formules[1] corresponderend met figuur 1 voor respectievelijk het elektrische en het magnetische veld zijn als volgt

$$\vec{E} = \left\{3\frac{P(t-R/c)}{R^5} + 3\frac{\frac{d}{dt}P(t-R/c)}{cR^4} + \frac{\frac{d^2}{dt^2}P(t-R/c)}{c^2R^3}\right\}(\vec{n}.\vec{R})\vec{R} - \\ - \left\{\frac{P(t-R/c)}{R^3} + \frac{\frac{d}{dt}P(t-R/c)}{cR^2} + \frac{\frac{d^2}{dt^2}P(t-R/c)}{c^2R}\right\}\vec{n} \tag{2}$$

$$\vec{B} = \left\{\frac{\frac{d}{dt}P(t-R/c)}{cR^3} + \frac{\frac{d^2}{dt^2}P(t-R/c)}{c^2R^2}\right\}\vec{n} \times \vec{R} \tag{3}$$

Hierbij is staat R voor de lengte van de vector $\vec{R}$. Om dit te kunnen gebruiken in een model moeten de grootheden worden uitgedrukt in dimensieloze grootheden. Hiervoor wordt eerst de frequentie vastgelegd op $\Omega = 2\pi f$. Nu kunnen de tijd en de afstand dimensieloos worden uit gedrukt op deze wijze

$$R = \widehat{R}\ \lambda = \widehat{R}\ \frac{c}{f} \tag{4}$$

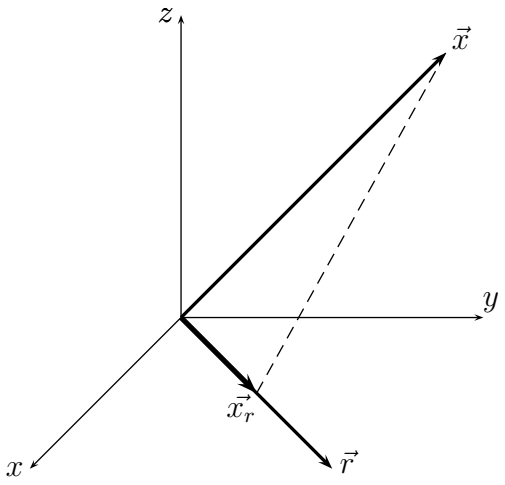
$$t = \hat{t}\ \frac{1}{f} \tag{5}$$

Als betrekking (4) en (5) worden gesubstitueerd in betrekking (2) en (3) worden deze respectievelijk

$$\vec{E} = \left\{3\frac{P((\hat{t}-\widehat{R})/f)}{\widehat{R}^5} + 3\frac{\frac{d}{d\hat{t}}P((\hat{t}-\widehat{R})/f)}{\widehat{R}^4} + \frac{\frac{d^2}{d\hat{t}^2}P((\hat{t}-\widehat{R})/f)}{\widehat{R}^3}\right\}\frac{(\vec{n}.\vec{\widehat{R}})\vec{\widehat{R}}}{\lambda^3} -$$
$$-\left\{\frac{P((\hat{t}-\widehat{R})/f)}{\widehat{R}^3} + \frac{\frac{d}{d\hat{t}}P((\hat{t}-\widehat{R})/f)}{\widehat{R}^2} + \frac{\frac{d^2}{d\hat{t}^2}P((\hat{t}-\widehat{R})/f)}{\widehat{R}}\right\}\frac{\vec{n}}{\lambda^3} \tag{6}$$

$$\vec{B} = \left\{\frac{\frac{d}{d\hat{t}}P((\hat{t}-\widehat{R})/f)}{\widehat{R}^3} + \frac{\frac{d^2}{d\hat{t}^2}P((\hat{t}-\widehat{R})/f)}{\widehat{R}^2}\right\}\frac{\vec{n}\times\vec{\widehat{R}}}{\lambda^3} \tag{7}$$

Om een goede dichtheidsgrafiek te maken op het boloppervlak moet de desbetreffende vector worden geprojecteerd op de normaal van het boloppervlak. Zie hiervoor de figuur hieronder



Figuur 2: Projectie van een $\vec{x}$ op $\vec{r}$.

Uit figuur 2 valt op te maken dat geldt

$$\vec{x_r} = \frac{(\vec{x}.\vec{r})\vec{r}}{|\vec{r}|^2} \tag{8}$$

Omdat het magnetische veld altijd loodrecht op de normaal van het boloppervlak staat, heeft het geen zin om betrekking (8) daarop toe te passen. Daarom wordt het magnetische veld geprojecteerd op de richtingsvector van het boloppervlak, waarbij als voorwaarde is genomen dat deze vector evenwijdig moet zijn aan het xy-vlak. Het is voldoende om in plaats van $\vec{r}$ nu $\vec{x}_n = \vec{n} \times \vec{r}$ te nemen in betrekking (8).

2.2 Visualisatie afhankelijk van de radius

Het voorgaande gedeelte van de theorie maakt het mogelijk om een visualisatie te maken met variërende tijd. Wil men echter de wat gedetailleerdere eigenschappen van betrekking (6) en (7) onderzoeken dan is het handiger om het kwadraat van deze te integreren over één periode op een willekeurig tijdstip. Hierdoor vervalt de tijdsafhankelijkheid en wordt tevens het dominerende gedeelte van de functie - het gedeelte met de kleinste deler - constant. Als dit met de zojuist genoemde betrekkingen wordt gedaan met een dipool richting van $n = (0, 0, 1)$ krijgt men

$$\vec{E} = \frac{1}{\lambda^3} \begin{pmatrix} C_1(R_x R_z)^2 \\ C_1(R_y R_z)^2 \\ C_1 R_z^4 + C_2 + 2C_3 R_z^2 \end{pmatrix} \tag{9}$$

$$\vec{B} = \frac{C_4}{\lambda^3} \begin{pmatrix} -R_y \\ R_x \\ 0 \end{pmatrix} \tag{10}$$

waarbij

$$C_1 = \frac{\frac{9}{2} + 6\pi^2 \widehat{R}^2 + 8\pi^4 \widehat{R}^4}{\widehat{R}^{10}} \tag{11}$$

$$C_2 = \frac{\frac{1}{2} - 2\pi \widehat{R}^2 + 8\pi^4 \widehat{R}^4}{\widehat{R}^6} \tag{12}$$

$$C_3 = \frac{3 - 4\pi^2 \widehat{R}^2 + 16\pi^4 \widehat{R}^4}{2\widehat{R}^8} \tag{13}$$

$$C_4 = 2\frac{\pi^2 + 2\pi^2 \widehat{R}^2}{\widehat{R}^6} \tag{14}$$

$$\vec{\widehat{R}} = (R_x, R_y, R_z) \tag{15}$$

3 Programmaopzet

Met de gegevens uit hoofdstuk 2 kan nu het programma worden geschreven. In dit hoofdstuk zal het programma kort worden beschreven, in de appendix is de volledige code terug te vinden.

3.1 Databewerking

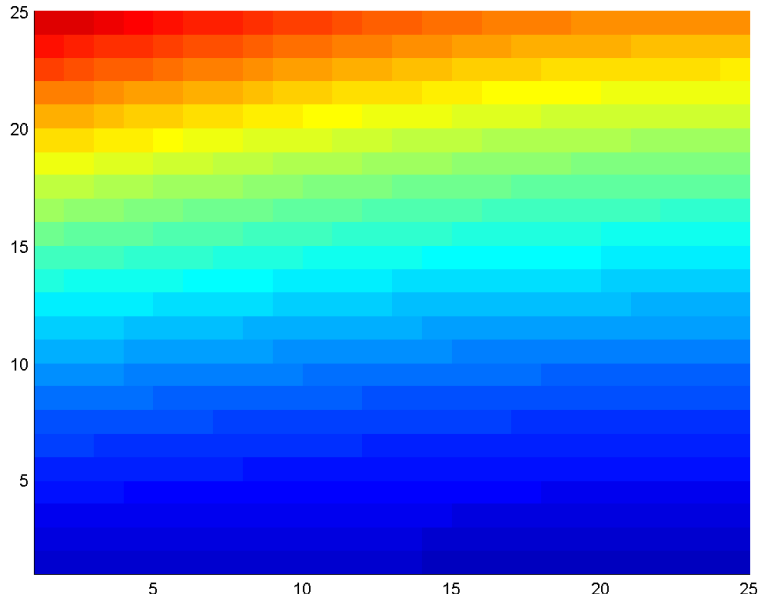
Het veld kan op elk willekeurig punt op de bol worden uitgerekend, voor alle punten van de bol kan dit echter nooit tegelijk worden weergegeven. Het is handiger om een bepaald gebied te nemen van deze bol. Dit kan makkelijk worden afgebakend d.m.v. twee intervallen van de hoeken, θ en ϕ. Respectievelijk de hoek die in het het xy-vlak ligt en de hoek die in het yz- of het xz-vlak ligt. Zie ook figuur 1 in 2.1.
Deze intervallen worden gediscretiseerd met een gekozen waarde, zodat er met behulp van de parametrische functie

$$f(x, y, z) = R(\sin\phi\cos\theta,\ \sin\phi\sin\theta,\ \cos\phi) \tag{16}$$

een rooster van punten op het boloppervlak wordt bepaald. Deze coördinaten vormen tevens stuk voor stuk de vector $\vec{r}$ uit betrekking (8) in 2.1 waarop de uiteindelijke waarden van het veld moeten worden geprojecteerd.
Deze coördinaten leveren dan, door ze bij betrekking (6), (7), (9) of (10) in te voeren, de vectoren die worden geprojecteerd op de normaal of de richtingsvector zoals beschreven is aan het einde van 2.1.
Voor de poynting vector $\vec{S}$ worden eerst het de twee andere velden uitgerekend, waarna $\vec{S} = \vec{E} \times \vec{B}$ kan worden uitgerekend. Pas hierna wordt ook de poynting vector geprojecteerd op de normaal van het boloppervlak(op eenzelfde wijze als bij het elektrische veld).

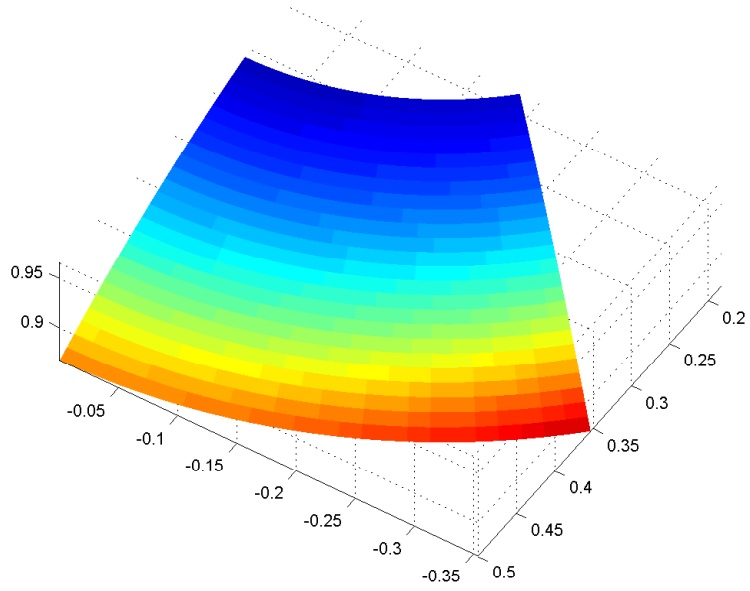
3.2 Dataplotting

Nu de data is bewerkt kan deze worden gevisualiseerd. Van de coördinaten die al zijn bewerkt kan een simpele dichtheidgrafiek worden gemaakt. Dit is gewoon een standaardfunctie in het softwarepakket waarmee het programma is geschreven. Een voorbeeld hiervan is het te zien in figuur 3.



Figuur 3: Een dichtheidsgrafiek van de Poynting vector genomen over een bepaald gedeelte van het boloppervlak.

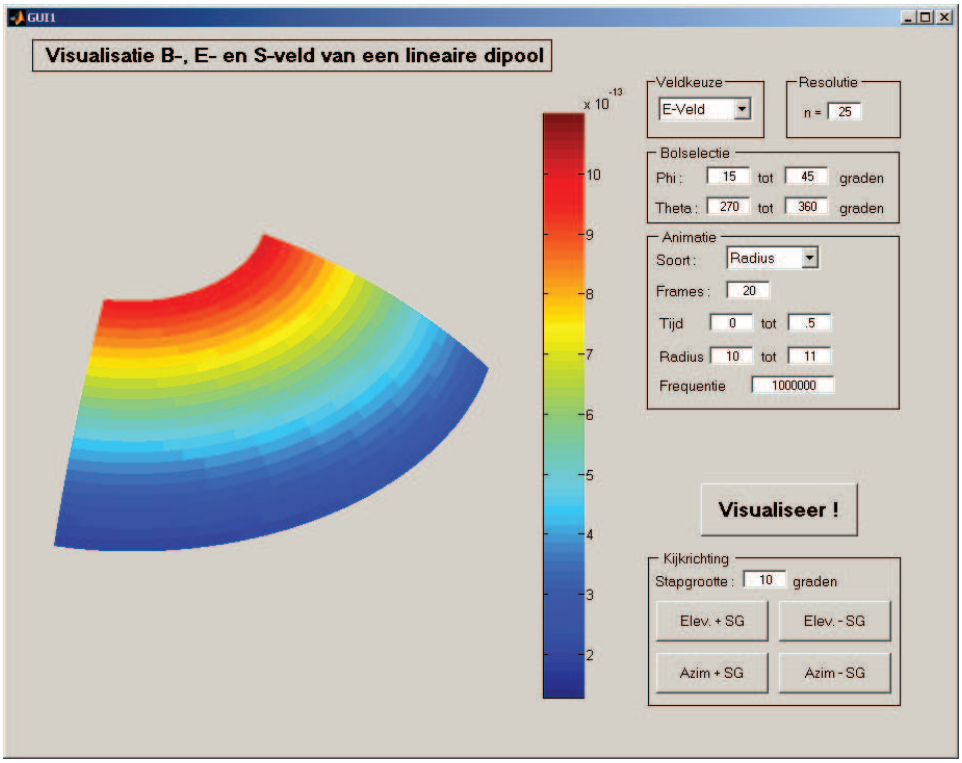
Met de coördinaten uit betrekking (16) kan met een andere functie een boloppervlak worden gecreëerd. Het aantal roosterpunten dat deze functie krijgt aangeleverd, bepaalt de nauwkeurigheid van de het oppervlak, het aantal polygonen. Mits het aantal vlakjes in bijvoorbeeld figuur 3 gelijk is aan het aantal polygonen op het boloppervlak, kan deze dichtheidsgrafiek hierover heen worden gelegd. Een voorbeeld hiervan is te zien in figuur 4 op pagina 9.
Er kan ook nog een geïnterpoleerde kleurschakering worden toegepast over het rooster van vlakken met discontinue overgangen. Dit is uiteindelijk alleen gedaan bij de tijdsafhankelijke visualisatie, omdat dit bij de andere visualisatie als gevolg had dat de nuances minder duidelijk zichtbaar werden.



Figuur 4: De dichtheidsgrafiek uit figuur 3 geplaatst op het boloppervlak.

3.3 Graphical User Interface(GUI)

Bij het bepalen van een stuk boloppervlak en het kijken in een 3-dimensionale ruimte is een functie waar alleen maar data in valt te voeren lastig. Alle functies zijn daarom aan te sturen via een GUI(NL: grafische werkomgeving). Hierdoor kunnen gemakkelijk andere instellingen voor de visualisatie worden ingevoerd, de effecten van deze veranderingen zijn vrijwel direct zichtbaar en er kunnen daarna dus ook weer gemakkelijk instellingen worden aangepast of bijgesteld. Een voorbeeld van de GUI staat op pagina 10 in figuur 5.



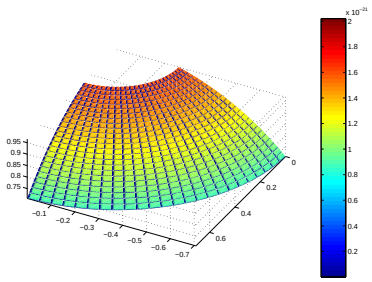
Figuur 5: De Graphical User Interface.

4 Resultaten en discussie

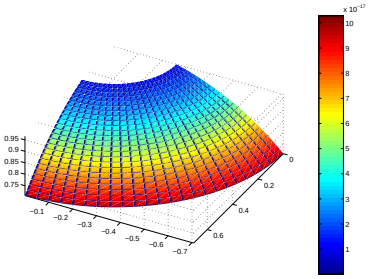
In dit hoofdstuk zullen de visualisatie nader worden besproken. Omdat de uiteindelijke uitvoer van het programma een animatie is, is dit verslag vergezeld met een CD waarop het programma zelf is terug te vinden. Om het programma uit te voeren is het bezit van het softwarepakket Matlab echter wel een vereiste.

4.1 Visualisatie afhankelijk van de tijd

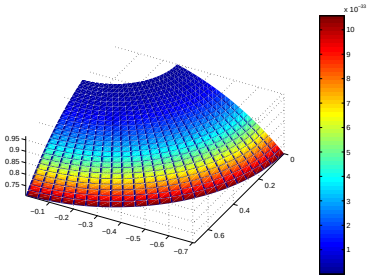
In de figuren 6 tot en met 7 staan respectievelijk shots van de visualisaties van het elektrische veld, het magnetische veld en de poynting vector.



Figuur 6: Het elektrische veld.

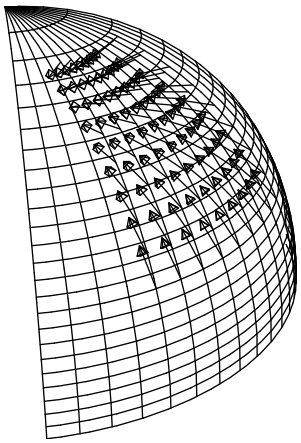


Figuur 7: Het magnetische veld.

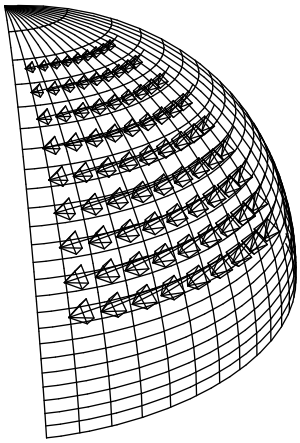


Figuur 8: De poynting vector.

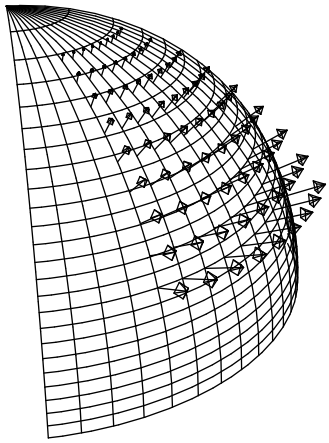
Op de figuren hierboven zijn geen details in het veld te zien, het veld variëert alleen maar in de z-richting. Dit laatste komt door de symmetrische situatie, de dipool is onafhankelijk van rotaties om de z-as. De details worden - zoals in hoofdstuk 2 al is besproken - overheerst door het gedeelte in de formule met kleinste deler. De code van dit gedeelte van het programma is te vinden in Appendix A.
Om aan te tonen dat figuren 6 t/m 8 correct zijn, is er met behulp van andere software(Mathematica) ook een voorstelling van de velden gemaakt als vectoren. Zie hiervoor de figuren 9 t/m 11 op pagina 12. In deze figuren zijn in plaats van polygonen met een bepaalde kleur, vectoren met een bepaalde lengte(grootte) te zien. Des te groter en langer de pijl des te groter is het veld daar. Bij de figuren 6 t/m 8 staat rood voor een grotere waarde, blauw juist voor een kleinere waarde. Ook van de figuren 9 t/m 11 is er een broncode aanwezig in dit verslag, namelijk in appendix B.



Figuur 9: Het elektrische veld.



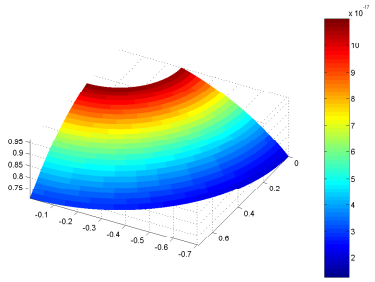
Figuur 10: Het magnetische veld.



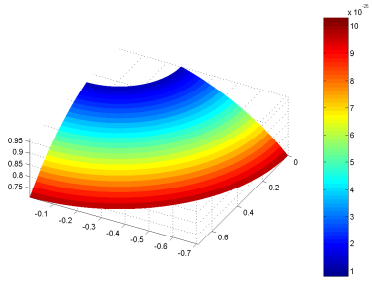
Figuur 11: De poynting vector.

4.2 Visualisatie afhankelijk van de radius

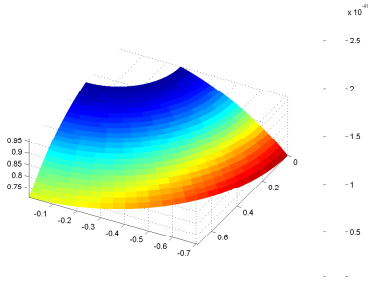
In de figuren 12 t/m 14 zijn de shot te zien van de visualisatie waarbij de radius wordt gevariëerd.



Figuur 12: Het elektrische veld.



Figuur 13: Het magnetische veld.



Figuur 14: De poynting vector.

Gezegd moet worden dat het raakvlak met werkelijkheid van deze animaties/shots kleiner is dan die uit paragraaf 4.1. Een integratie over het kwadraat van een vector zorgt ervoor dat bepaalde eigenschappen de formule naar voren komen. Andere eigenschappen komen sterker op de achtergrond te liggen of verdwijnen helemaal. Dit geldt sterk voor het shot uit de animatie van het S-veld(figuur 14).
Men moet er tevens rekening mee houden dat de situatie van de figuren hierboven een significant verschillende is in vergelijking met de figuren uit de vorige paragraaf. Hier wordt namelijk de radius vergroot, met andere woorden de bol wordt eigenlijk geleidelijk 'opgeblazen'.
Voor meer animaties wordt verwezen naar het programma dat is bijgevoegd bij dit verslag.

5 Conclusie

De visualisaties leverden het gewenste resultaat op, maar de snelheid waarmee deze werden berekend kon beter.
Het systeem in de GUI waarmee een bepaald gedeelte van de bol nader kon worden bekeken, bood hier uitkomst. De radiusafhankelijke simulatie geeft een beter beeld van details van betrekkingen die als uitgangspunt zijn gekozen. Deze visualisatie geeft echter veel minder goed de werkelijke fysische situatie weer. De resultaten hiervan moeten dus met enige oplettendheid worden geïnterpreteerd.

6 Dankwoord

Graag zou ik Prof.dr. H. De Raedt willen bedanken voor de assistentie bij deze korte stage.

7 Referenties

[1] M. Born and E. Wolf e.a., *Principles of Optics*, Electromagnetic theory of propagation interference and diffraction of light, 82, Cambridge(1980)[6].

[2] J.E. Marsden and A.J. Tromba, *Vector Calculus*, W.H. Freeman and Company(1996)[4].

8 Appendices

8.1 A

Code van de tijdsafhankelijke visualisatie, geschreven voor matlab.

Plotzooi.m

```
% Script file

clear

n0 = 24; n = 25; afdruk = 2;

theta = [270 360]; phi = [15 45];

a(:, :, 1) = Field(0, theta, phi, n, 100, [0 0 1], afdruk) Cmin =
min(min(a(:,:,1))); Cmax = max(max(a(:, :, 1)))

j = 2; for t = 1/(n0 - 1):1/(n0 - 1):1;
    a(:, :, j) = Field(t, theta, phi, n, 100, [0 0 1], afdruk);
    if Cmin > min(min(a(:, :, j)))
        Cmin = min(min(a(:, :, j)));
    end
    if Cmax < max(max(a(:, :, j)))
        Cmax = max(max(a(:, :, j)));
    end
    j = j + 1;
end

[x, y, z] = PlotData(theta, phi, n);

for i = 1:1
    surf(x,y,z,a(:,:,i));
    caxis([Cmin Cmax]);
    colormap(jet);
    view(-118,-32)
    axis equal;
    axis vis3d;
    shading interp;
    colorbar;
    P(i) = getframe;
end

for i = 1:j - 1
    s = ['Test' sprintf('%03d',i) '.png'];
    [X, jet] = frame2im(P(i));
    imwrite(X, s, 'png');
end
```

Field.m

```
% Functie van tijd van E-, B- of S-veld
% t          :tijd
% theta      :[beginhoek eindhoek] in graden
% phi        :[beginhoek eindhoek] in graden
% resolution :aantal pixels per lengte of breedte
% radius     :straal van de bol waarop wordt geprojecteerd
% n          :richting lineaire dipool
% afdruk     :vul 0, 1 of 2 voor repsectievelijk een uitvoer van het E-, het B-
%             of het S-veld.
```

```
function F = Field(t, theta, phi, resolution, radius, n, afdruk,
f)

% Declaratie van constanten
f = 10^6; c = 2.998*10^8; labda = c/f; Dist = radius; points =
resolution;

% R(phi, ... , theta);
L = 1; for i =
theta(1)*pi/180:pi/180*(theta(2)-theta(1))/points:theta(2)*pi/180
    K = 1;
    for j = phi(1)*pi/180:pi/180*(phi(2)-phi(1))/points:phi(2)*pi/180
        Rphi(K, :) = Dist*[sin(j).*cos(i)  sin(j).*sin(i)  cos(j)];
        K = K + 1;
    end
    R(:, :, L) = Rphi;
    L = L + 1;
end

% Berekening van de daadwerkelijke veldsterkten
if afdruk == 0
    for i = 1:points
        for j = 1:points
            Fi = Eveld(t, n, R(i, :, j), labda);
            F(i,j) = norm(dot(Fi, R(i, :, j))*R(i, :, j)/(norm(R(i, :, j)))^2)^2;
        end
    end
elseif afdruk == 1
    for i = 1:points
        for j = 1:points
            Fi = Bveld(t, n, R(i, :, j), labda);
            r = cross(n, R(i, :, j));
            F(i,j) = norm(dot(Fi, r)*r/(norm(r))^2)^2;
        end
    end
elseif afdruk == 2
    for i = 1:points
        for j = 1:points
            Fi = cross(Bveld(t, n, R(i, :, j), labda), Eveld(t, n, R(i, :, j), labda));
            F(i,j) = norm(dot(Fi, R(i, :, j))*R(i, :, j)/(norm(R(i, :, j)))^2)^2;
        end
    end
end
```

Eveld.m

```
% Functie E-veld.

function E = Eveld(t, n, R, labda)

Rn = norm(R); E = dot(n,R)*R/(labda^3)*(3*cos(2*pi*(t -
Rn))/Rn^5-6*pi*sin(2*pi*(t - Rn))/Rn^4-4*pi^2*cos(2*pi*(t -
Rn))/Rn^3 )-n/(labda^3)*( cos(2*pi*(t - Rn))/Rn^3 -
2*pi*sin(2*pi*(t - Rn))/Rn^2-4*pi^2*cos(2*pi*(t - Rn))/Rn);
```

Bveld.m

```
% Functie B-veld.

function B = Bveld(t, n, R, labda)

Rn = norm(R); B = cross(n,R)/(labda^3)*(-2*pi*sin(2*pi*(t -
Rn))/Rn^3 - 4*pi^2*cos(2*pi*(t - Rn))/Rn^2);
```

8.2 B

Code van het programma voor mathematica waarmee een animatie van het vectorveld kan worden gecreëerd.

Rekenwerk.nb

```
Clear[t, n, R, t, t1]

Off[General::spell1]

f = 10^6; c = 2.998*10^8;
\[Lambda] = c/f;
p0 = 1; Scale = 3500*\[Lambda]^3;

p1[t1_] = p0*Cos[2*Pi*f*t1]; P[t_, R_] = p1[(t - R)/f];

Eveld[R_,
      n_] = (3*P[t, Norm[R]]/Norm[R]^5 +
            3*D[P[t, Norm[R]], {t, 1}]/Norm[R]^4 +
            D[P[t, Norm[R]], {t, 2}]/Norm[R]^3)*(n.R)*
        R/\[Lambda]^3 - (P[t, Norm[R]]/Norm[R]^3 +
            D[P[t, Norm[R]], {t, 1}]/Norm[R]^2 +
            D[P[t, Norm[R]], {t, 2}]/Norm[R])*n/\[Lambda]^3;
Bveld[R_,
      n_] = (D[P[t, Norm[R]], {t, 1}]/Norm[R]^3 +
          D[P[t, Norm[R]], {t, 2}]/Norm[R]^2)*Cross[n, R]/\[Lambda]^3 ;
Sveld[R_, n_] = Cross[Eveld[R, n], Bveld[R, n]];

n = {0, 0, 1};

Coordinates =
    Table[{Sin[\[Phi]]*Cos[\[Theta]], Sin[\[Phi]]*Sin[\[Theta]],
        Cos[\[Phi]]}, {\[Theta], 52*Pi/32, 60*Pi/32, Pi/32}, {\[Phi], 4*Pi/32,
         12*Pi/32, Pi/32}];

For[i = 1, i < 10, i++,
  For[j = 1, j < 10, j++,
    a[i, j] = Sveld[Extract[Coordinates, {i, j}], n];
    ]
  ]

n0 = 0; For[i = 1, i < 10, i++,
  For[j = 1, j < 10, j++,
    b[n0] = {Extract[Coordinates, {i, j}], Scale*a[i, j]};
    n0++;
    ]
  ]

P1 = ParametricPlot3D[{Sin[\[Phi]]*Cos[\[Theta]],
Sin[\[Phi]]*Sin[\[Theta]],
        Cos[\[Phi]]}, {\[Theta], 52*Pi/32, 60*Pi/32}, {\[Phi], 4*Pi/32,
        12*Pi/32}, ViewPoint -> {-0.578, -2.988, 1.479}, PlotPoints -> 9,
      Shading -> False, DisplayFunction -> Identity];
P3 = ParametricPlot3D[
      Evaluate[Table[{Sqrt[1 - i^2]*Cos[t], -Sqrt[1 - i^2]*Sin[t], i}, {i, 0,
            1, 1/23.5}]], {t, 0, 2*Pi}, Boxed -> False, Axes -> False,
      PlotRange -> {{0, 1}, {0, -1}, {0, 1.1}},
      ViewPoint -> {-0.578, -2.988, 1.479}, DisplayFunction -> Identity,
      ImageSize -> 250];
P4 = ParametricPlot3D[
      Evaluate[Table[{-Cos[i*Pi/2]*Cos[t], Sin[i*Pi/2]*Cos[t], Sin[t]}, {i, 0,
             1, 1/16}]], {t, 0, 2*Pi}, Boxed -> False, Axes -> False,
      PlotRange -> {{0, 1}, {0, -1}, {0, 1.1}},
      ViewPoint -> {-0.578, -2.988, 1.479}, DisplayFunction -> Identity,
```

```
      ImageSize -> 250];

Func[t_] = Table[b[i], {i, 0, 80}];

ns = 23; Ts = 1; For[i = 1, i < ns + 1, i++,
  P2 = ListPlotVectorField3D[Func[i*Ts/ns], VectorHeads -> True,
      DisplayFunction -> Identity, ImageSize -> 250];
  l = StringLength[ToString[i]];
  Export[StringReplacePart["g0000.eps", ToString[i], {5 - l, 5}],
    Show[{P3, P4, P2}], "EPS"];
  Clear[t]
  ]
```

8.3 C

Code van de radiusafhankelijke visualisatie, geschreven voor matlab.

IntPlotzooi.m

```
% Script file - Geintregreeerde Gekwadrateerde Velden

clear

% Randvoorwaarden
R = [100 110]; n0 = 48; n = 25; afdruk = 2;

theta = [270 360]; phi = [15 45];

% Berekening veldwaarden, Bepaling colormap
a(:, :, 1) = intField(theta, phi, n, R(1), afdruk); Cmin =
min(min(a(:,:,1))); Cmax = max(max(a(:, :, 1)));

j = 2; for r = (R(1) + (R(2) - R(1))/n0):(R(2) - R(1))/n0:R(2)
    a(:, :, j) = intField(theta, phi, n, r, afdruk);
    if Cmin > min(min(a(:, :, j)))
        Cmin = min(min(a(:, :, j)));
    end
    if Cmax < max(max(a(:, :, j)))
        Cmax = max(max(a(:, :, j)));
    end
    j = j + 1;
end

% Dataplotting and vlakberekening
[x, y, z] = PlotData(theta, phi, n);

for i = 1:3
    surf(x,y,z,a(:,:,i));
    set(gcf,'Renderer','OpenGL')
    caxis([Cmin Cmax]);
    colormap(jet);
    view(-118,-32)
    axis equal;
    axis vis3d;
    shading flat;
    colorbar;
    P(i) = getframe(gcf);
end

% Uitvoer naar .png bestanden
for i = 1:j - 1
    s = ['Test' sprintf('%03d',i) '.png'];
    [X, jet] = frame2im(P(i));
    imwrite(X, s, 'PNG');
end
```

IntField.m

```
% Functie van radius van E-, B- of S-veld
% theta      :[beginhoek eindhoek] in graden
% phi        :[beginhoek eindhoek] in graden
% resolution :aantal pixels per lengte of breedte
% radius     :straal van de bol waarop wordt geprojecteerd
% afdruk     :vul 0, 1 of 2 voor repsectievelijk een uitvoer van het E-, het B-
%             of het S-veld.

function F = intField(theta, phi, resolution, radius, afdruk)

% Declaratie van constanten
f = 10^6; c = 2.998*10^8; labda = c/f; n = [0 0 1];

% R(phi, ... , theta);
L = 1; for i =
theta(1)*pi/180:pi/180*(theta(2)-theta(1))/resolution:theta(2)*pi/180
    K = 1;
    for j = phi(1)*pi/180:pi/180*(phi(2)-phi(1))/resolution:phi(2)*pi/180
        Rphi(K, :) = radius*[sin(j).*cos(i)  sin(j).*sin(i)  cos(j)];
        K = K + 1;
    end
    R(:, :, L) = Rphi;
    L = L + 1;
end

% Berekening van de daadwerkelijke veldsterkten
if afdruk == 0
    for i = 1:resolution
        for j = 1:resolution
            Fi = intEveld(R(i, :, j), labda);
            F(i,j) = norm(dot(Fi, R(i, :, j))*R(i, :, j)/(norm(R(i, :, j)))^2)^2;
        end
    end
elseif afdruk == 1
    for i = 1:resolution
        for j = 1:resolution
            Fi = intBveld(R(i, :, j), labda);
            r = cross(n, R(i, :, j));
            F(i,j) = norm(dot(Fi, r)*r/(norm(r))^2)^2;
        end
    end
elseif afdruk == 2
    for i = 1:resolution
        for j = 1:resolution
            Fi = cross(intBveld(R(i, :, j), labda), intEveld(R(i, :, j), labda));
            F(i,j) = norm(dot(Fi, R(i, :, j))*R(i, :, j)/(norm(R(i, :, j)))^2)^2;
        end
    end
end
```

IntEveld.m

```
% Functie E^2-veld geintegreerd over een periode(op een willekeurig tijdstip).

function E = intEveld(R, labda)

Rn = norm(R); C1 = (9/2 + 6*pi^2*Rn^2 + 8*pi^4*Rn^4)/Rn^10; C2 =
(1/2 - 2*pi^2*Rn^2 + 8*pi^4*Rn^4)/Rn^6; C12 = (3 - 4*pi^2*Rn^2 +
16*pi^4*Rn^4)/(2*Rn^8);

E = [C1*(R(1)*R(3))^2 C1*(R(2)*R(3))^2 C1*R(3)^4 + C2 +
2*C12*R(3)^2]./labda^3;
```

IntBveld.m

```
% Functie B^2-veld geintegreerd over een periode(op een willekeurig tijdstip).

function B = intBveld(R, labda)

Rn = norm(R); C1 = 2*(pi^2 + 2*pi^2*Rn^2)/Rn^6;

B = [-R(2) R(1) 0]./labda^3*C1;
```