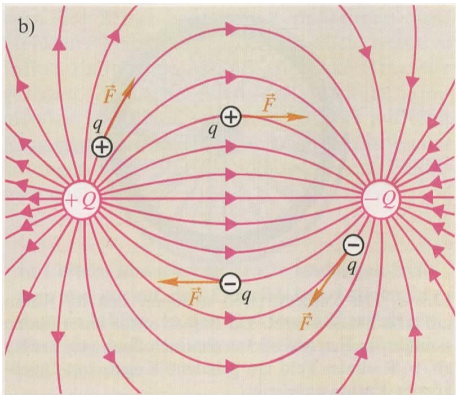
Feldlinienbilder

Einleitung

In gängigen Lehrbüchern findet man Feldlinienbilder wie das nebenstehende Beispiel. Kann man diese auch selber erstellen, nicht nur qualitativ, sondern quantitativ so, dass die Krümmung der Linien stimmt und die Linienabstände in verschiedenen Bereichen die relative Stärke des Feldes richtig wiedergeben?

Zum Zeichnen eignet sich *gnuplot* (freeware). Zum Erstellen der Datenfiles benutze ich *Fortran* (ebenfalls freeware); als Editor eignet sich hier *Notepad*.



elektrische Feldlinien zwischen entgegengesetzten Ladungen [Dorn/Bader Physik Oberstufe Gesamtband 12/13]

Elektrisches Feld

Das elektrische Feld an einem Ort bei Anwesenheit einer Reihe von Punktladungen wird wie im Folgenden beschrieben berechnet. Wir bleiben in dieser Arbeit in einer Ebene. Es befinden sich die Ladung Q_1 bei (x_1, y_1) und die Ladung Q_2 bei (x_2, y_2). Das Feld $\vec{E}$ sei am Punkt (x, y) zu ermitteln. Als Hilfsgrößen nützlich sind die Abstände $r_1 = \sqrt{(x-x_1)^2 + (y-y_1)^2}$ und $r_2 = \sqrt{(x-x_2)^2 + (y-y_2)^2}$.

Q_1 erzeugt am Punkt (x, y) das elektrische Feld mit den Komponenten $E_{x1} = \dfrac{Q_1}{4\pi\varepsilon_0} \cdot \dfrac{x-x_1}{r_1^3}$

und $E_{y1} = \dfrac{Q_1}{4\pi\varepsilon_0} \cdot \dfrac{y-y_1}{r_1^3}$ und Q_2 erzeugt dort $E_{x2} = \dfrac{Q_2}{4\pi\varepsilon_0} \cdot \dfrac{x-x_2}{r_2^3}$ und $E_{y2} = \dfrac{Q_2}{4\pi\varepsilon_0} \cdot \dfrac{y-y_2}{r_2^3}$

Das Gesamtfeld ist einfach die Summe. $\vec{E} = (E_x, E_y) = (E_{x1} + E_{x2}, E_{y1} + E_{y2})$.
Ist nur eine Ladung vorhanden, erübrigt sich die Addition, und bei mehr als zwei Ladungen sind entsprechend mehr Felder aufzusummieren.

Das Vektorfeld mit Vektoren an äquidistanten Punkten ist recht einfach darzustellen. Im folgenden Beispiel befindet sich eine positive Ladung bei (-8;0) und eine gleich große negative bei (8;0). Das *Fortran*-Programm ermittelt die Feldvektoren auf einem Gitter mit Abstand 2. Die Vektoren werden willkürlich skaliert und zu lange sehr nahe bei den Punktladungen weggelassen, damit sie sich im Diagramm nicht unübersichtlich überlappen. In das Ausgabefile werden pro Punkt vier Zahlen geschrieben: x-Koordinate des Vektoranfangspunkts, y-Koordinate des Vektoranfangspunkts, Länge des Vektors in x-Richtung, Länge des Vektors in y-Richtung.
gnuplot kann solche Daten mit dem Befehl *plot ... with vectors* darstellen. Mit *set object circle* und *set label* kann man den plot verschönern, indem die Ladungen angezeigt werden.

Um unhandliche Zahlen zu vermeiden, wird der Faktor $4\pi\varepsilon_0$ im Programm nicht mitgenommen. Was die Eingabe 1.0 für ein Q bedeutet, hängt davon ab, welche Längeneinheiten man sich hinter den Skalierungen der x- und y-Achse denkt sowie der Einheit, in der man das elektrische Feld angegeben annimmt. Sind z.B. die Ladungen in Mikrocoulomb (μC) und die Längeneinheiten sind Zentimeter (cm), so werden die E-Felder in Einheiten von 90090090 V/m angegeben.

C:\projekte\feldlinien\feldvektoren.f90 - Notepad++

Datei Bearbeiten Suchen Ansicht Codierung Sprache Einstellungen Werkzeuge Makros Ausführen Plugins Fenster ?

feldvektoren.f90

```
1  program feldvektoren
2  implicit none
3  ! 4*Pi*epsilon_0 taken as 1, OBdA liegen die beiden Ladungen auf der x-Achse
4
5  integer io_error, xzaehl, yzaehl
6  real x,y,xcomp,ycomp,q1,q2,x1,x2,r1,r2,ex,ey
7
8  x1=-8.0
9  x2=8.0
10 q1=25.0
11 q2=-25.0
12
13 open(unit=11,file='feldvektoren.dat',status='unknown',action='write',iostat=io_error)
14
15 do xzaehl=-19,19,2
16 do yzaehl=-9,9,2
17 x=real(xzaehl)
18 y=real(yzaehl)
19 r1=((x-x1)**2+y**2)**0.5
20 r2=((x-x2)**2+y**2)**0.5
21 ex=q1/r1**3*(x-x1)+q2/r2**3*(x-x2)
22 ey=q1/r1**3*y+q2/r2**3*y
23 if ((r1.gt.1.5).and.(r2.gt.1.5)) then
24  write(11,*) x-0.75*ex,y-0.75*ey,1.5*ex,1.5*ey
25 endif
26 enddo
27 enddo
28
29 close(unit=11)
30 end program feldvektoren
```

Fortra length : 661 lines : 30 Ln : 3 Col : 77 Pos : 114 Windows (CR LF) UTF-8 INS

Fortran-Programm für Feldvektoren

```
C:\projekte\feldlinien>gfortran feldvektoren.f90

C:\projekte\feldlinien>a
```

Kompilieren und Ausführen des Fortran-Programms in der Eingabeaufforderung. Das exe-file heißt zunächst a.exe; man könnte es noch umbenennen.

feldvektoren - Editor

Datei Bearbeiten Format Ansicht Hilfe

```
 -18.950121       -8.9485416      -9.97596681E-02 -0.10291635
 -18.930279       -6.9468341      -0.13944387     -0.10633191
 -18.907537       -4.9513855      -0.18492372     -9.72288400E-02
```

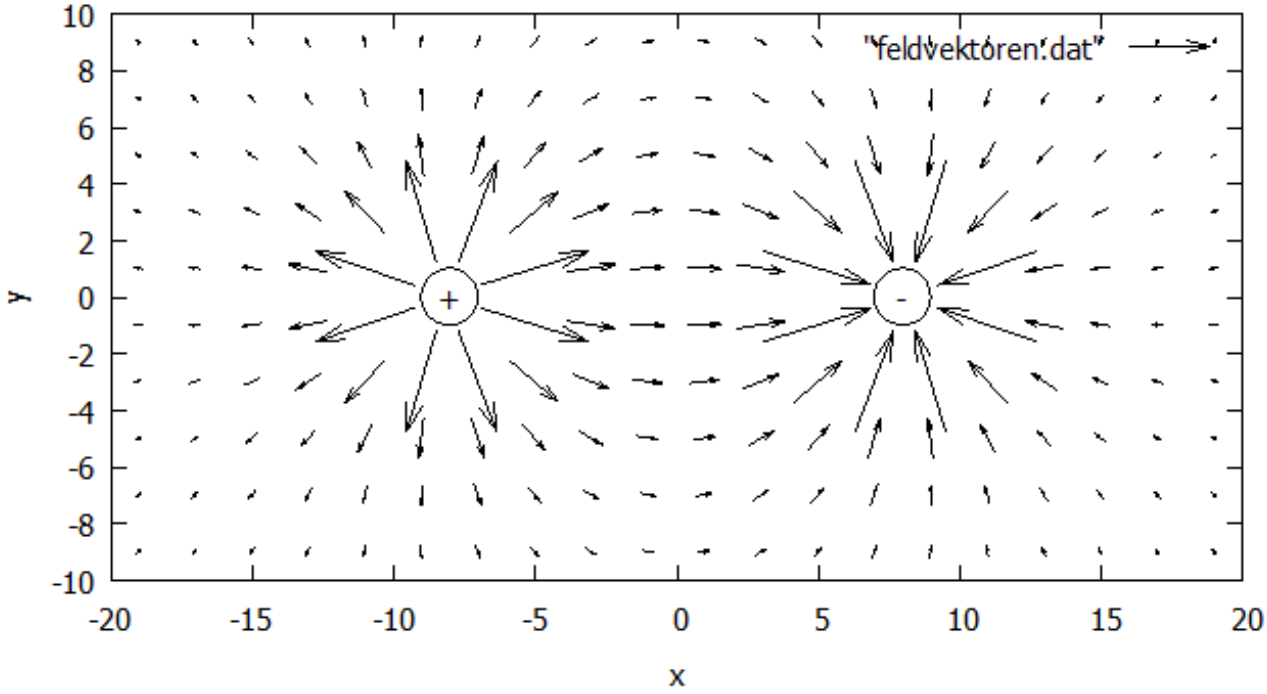
Anfang der Ausgabedatei

gnuplot

File Plot Expressions Functions General Axes Chart Styles 3D Help

```
gnuplot> cd 'C:\projekte\feldlinien'
gnuplot> set size ratio 0.5
gnuplot> set object circle at -8,0 size 1
gnuplot> set object circle at 8,0 size 1
gnuplot> set label "+" at -8.35,0
gnuplot> set label "-" at 7.8,0
gnuplot> set xlabel "x" offset 0,0
gnuplot> set ylabel "y" offset 0,0
gnuplot> plot "feldvektoren.dat" with vectors lc -1
gnuplot> _
```

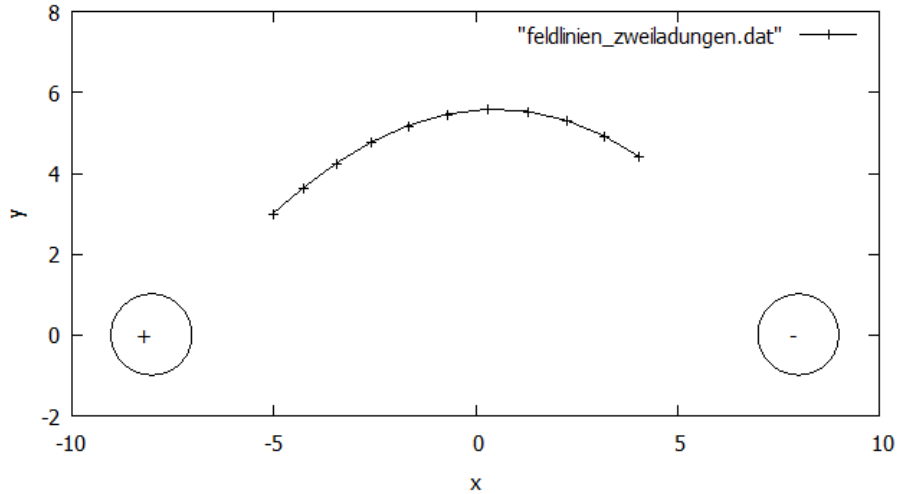
Befehle zum Zeichnen in gnuplot



Feldvektoren für zwei entgegengesetzt gleich große Ladungen

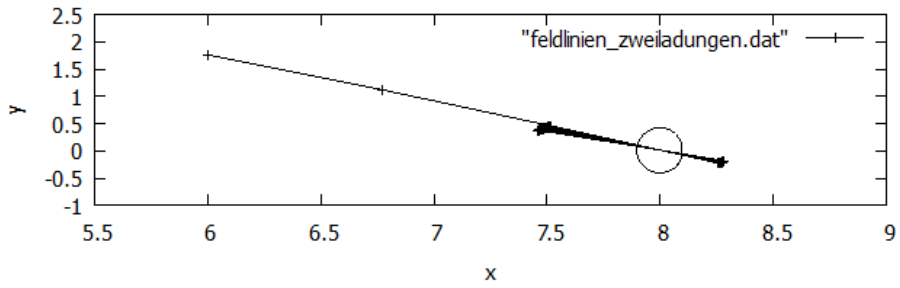
Feldlinien

Feldlinien liegen in manchen Bereichen dichter und in anderen weniger dicht, durchlaufen also nicht regelmäßig Punkte eines Koordinatengitters. Aber Feldlinien verlaufen in jedem Punkt entlang des Feldvektors und das werden wir uns zunutze machen. Von dem Punkt, an dem wir uns gerade befinden, gehen wir einen Einheitsvektor in Richtung des elektrischen Feldes, um zu einem neuen Punkt zu gelangen (ähnlich wie beim numerischen Integrieren). Wenn es genauer sein soll, können wir auch in kleineren Schritten gehen. Die durchlaufenen Punkte werden zur Feldlinie verbunden. (Das *Fortran*-Programm schreibt die Koordinaten der Punkte als Wertetabelle in eine Datei; *gnuplot* kann sie *with lines* mit Verbindungen zeichnen.) Von der Rechenweise ist die Linie zwar im Grunde eckig, aber bei hinreichend kleiner Schrittweite spielt das bei der Darstellung der Feldlinie keine Rolle mehr.



Hier ist der Startpunkt (-5;3) und es sind zehn Schritte der Feldlinie durch diesen Punkt berechnet. Sie geht von der positiven Ladung weg und läuft auf die negative Ladung zu. (Die berechneten Punkte sind noch als Kreuze gezeigt; *plot with linespoints*).

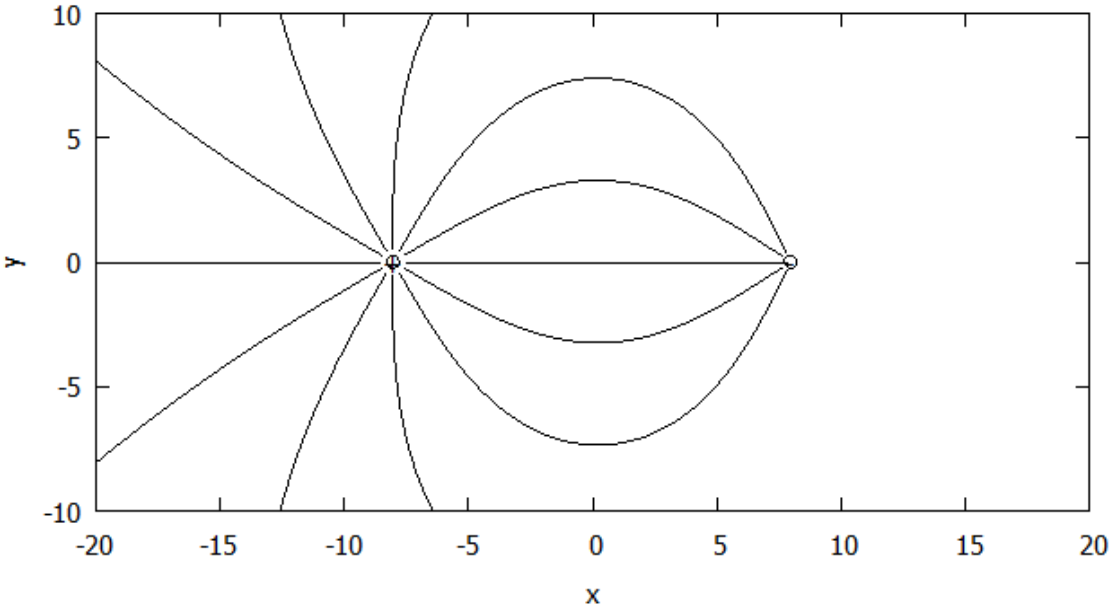
Feldlinien führen von positiven Ladungen weg und laufen auf negative Ladungen zu. Wann und wo müssen wir mit der Berechnung von einem Punkt zum nächsten aufhören? Sicherlich möchten wir das Feldlinienbild in einem begrenzten Bereich darstellen. Sobald ein Punkt außerhalb dieses Bereichs liegt, beenden wir die Berechnung der Feldlinie. Aber auch an einer negativen Ladung braucht das Programm eine Stoppbedingung.



Vom Punkt (6;1.75) aus läuft die Linie auf die Punktladung bei (8;0) zu und würde ohne Abbruch endlos um diesen Punkt hin- und heroszillieren.

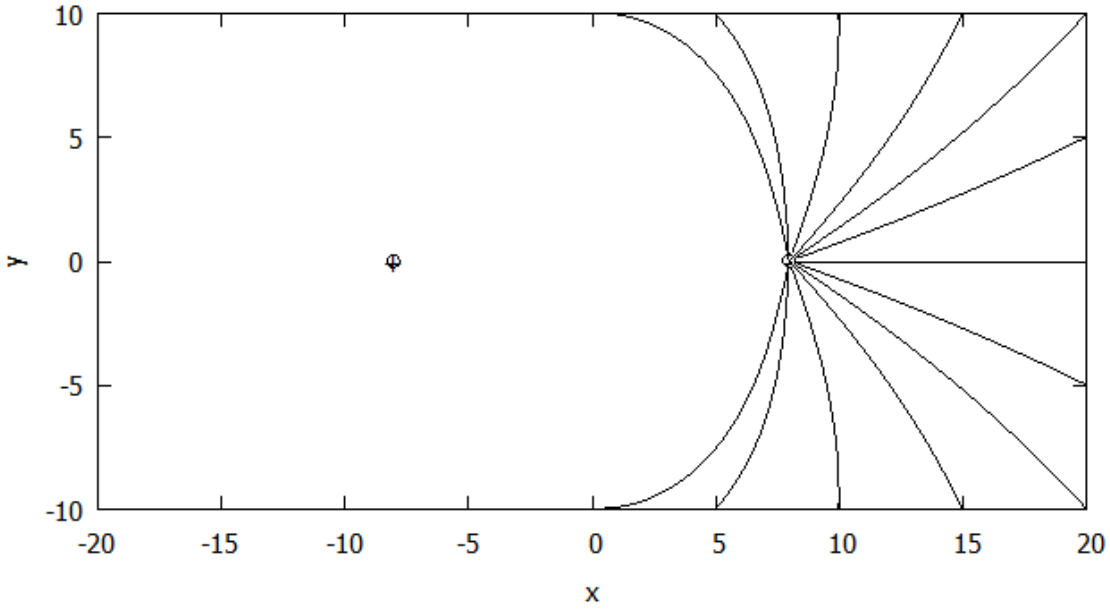
Wir beenden deshalb eine Feldlinie, wenn wir uns weniger als eine Schrittweite von einer negativen Punktladung entfernt befinden. (Eine kleine völlig von Punkten bedeckte Fläche am Ort einer Punktladung würde in der Darstellung nicht unbedingt stören, aber es muss verhindert werden, dass das Programm in einer Endlosschleife gefangen bleibt.)

Die bis jetzt gegebene Beschreibung klingt, als ob wir irgendwo anfangen und von diesem Punkt aus eine Feldlinie berechnen. Wir möchten natürlich ein systematisches Bild aus vielen Feldlinien. Das Programm soll mehrere Linien berechnen. Für die Ausgabe zum Zeichnen ist das kein Problem. Es ist nur eine Leerzeile in die Ausgabedatei einzufügen. Durch eine Leerzeile getrennte Punkte verbindet *gnuplot* auch bei *plot with lines* nicht. So lassen sich mehrere Feldlinien auflisten. Aber welche Startpunkte führen zu einem guten und vollständigen Ergebnis?
Ich habe mich entschieden, die Startpunkte für Feldlinien als Eingabedatei zu geben. So kann man die Startpunkte für erste Versuche sogar manuell zusammenstellen bzw. später aus mehreren Teilen zusammenfügen.



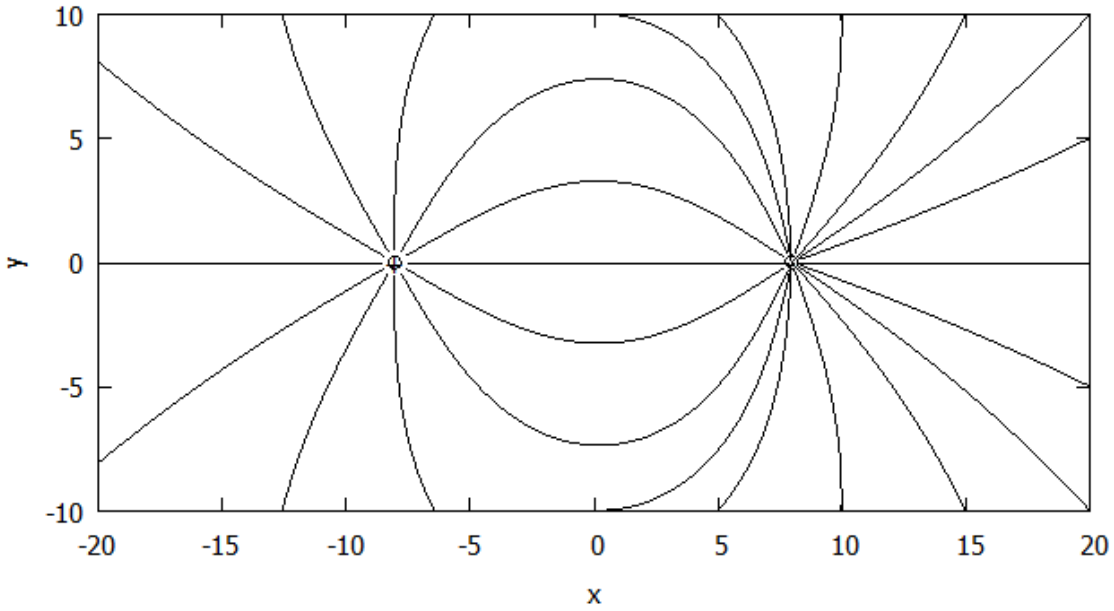
Feldlinien von der positiven Ladung aus

Im Bild hier sind die Startpunkte in einem kleinen Kreis um die positive Ladung herum angeordnet. Die Feldlinien laufen richtig auf die negative Ladung zu oder aus dem Bild heraus. Jedoch fehlen Feldlinien, die von außerhalb des Bildes auf die negative zulaufen.



Feldlinien von Punkten auf dem Rand aus

Im Bild hier oben sind Startpunkte auf dem Rand des gezeigten Bereichs gesetzt. Die Feldlinien laufen richtig auf die negative Ladung zu. Aber es fehlen sämliche Feldlinien bzw. Feldlinienteile, die von der positiven Ladung ausgehen.



die beiden vorherigen Feldlinienbilder zusammen

Und wenn wir beide Bilder einfach zusammenfügen? Dann sind in der linken und der rechten Bildhälfte nicht entsprechende Feldlinien gezeichnet, die das Bild verlassen. Noch wird der Fall mit zwei entgegengesetzt gleich großen Ladungen behandelt. Das Bild sollte also symmetrisch sein. So stimmt zwar die Richtung und die Krümmung jeder einzelnen Feldlinie, aber die Liniendichte gibt kein Maß für die Feldstärke.

Auf der folgenden Seite werden zunächst das Programm und die Startpunktedatei gezeigt.

C:\projekte\feldlinien\feldlinien_zweiladungen.f90 - Notepad++

Datei Bearbeiten Suchen Ansicht Codierung Sprache Einstellungen Werkzeuge Makros Ausführen Plugins Fenster ?

feldlinien_zweiladungen.f90

```
1  program feldlinien_zweiladungen
2  implicit none
3  ! 4*Pi*epsilon_Null taken as 1,OBdA beide Ladungen auf der x-Achse
4  integer io_error,io_stat
5  real x1,x2,q1,q2,x,y,xalt,yalt,r1,r2,ex,ey,e,efact
6  x1=-8.0
7  x2=8.0
8  q1=25.0
9  q2=-25.0
10 efact=0.45
11 io_stat=0
12 open(unit=12,file='startpunkte_zweiladungen.dat',status='old',action='read',iostat=io_error)
13 open(unit=11,file='feldlinien_zweiladungen.dat',status='unknown',action='write',iostat=io_error)
14
15 do while(io_stat.eq.0)
16 read(12,*,IOSTAT=io_stat) x,y
17 if (io_stat.eq.0) then
18   write(11,*) x,y
19   xalt=x
20   yalt=y
21   r1=((x-x1)**2+y**2)**0.5
22   r2=((x-x2)**2+y**2)**0.5
23    do while ((r1.gt.efact).and.(r2.gt.efact).and.(abs(x).le.20.0).and.(abs(y).le.10.0))
24    ex=q1/r1**3*(x-x1)+q2/r2**3*(x-x2)
25    ey=q1/r1**3*y+q2/r2**3*y
26    e=(ex**2+ey**2)**0.5
27    x=xalt+ex/e*efact
28    y=yalt+ey/e*efact
29    write(11,*) x,y
30    xalt=x
31    yalt=y
32    r1=((x-x1)**2+y**2)**0.5
33    r2=((x-x2)**2+y**2)**0.5
34  enddo
35  write(11,*)
36  endif
37 enddo
38
39 close(unit=12)
40 close(unit=11)
41
42 end program feldlinien_zweiladungen
```

*startpunkte_zweiladungen - Editor

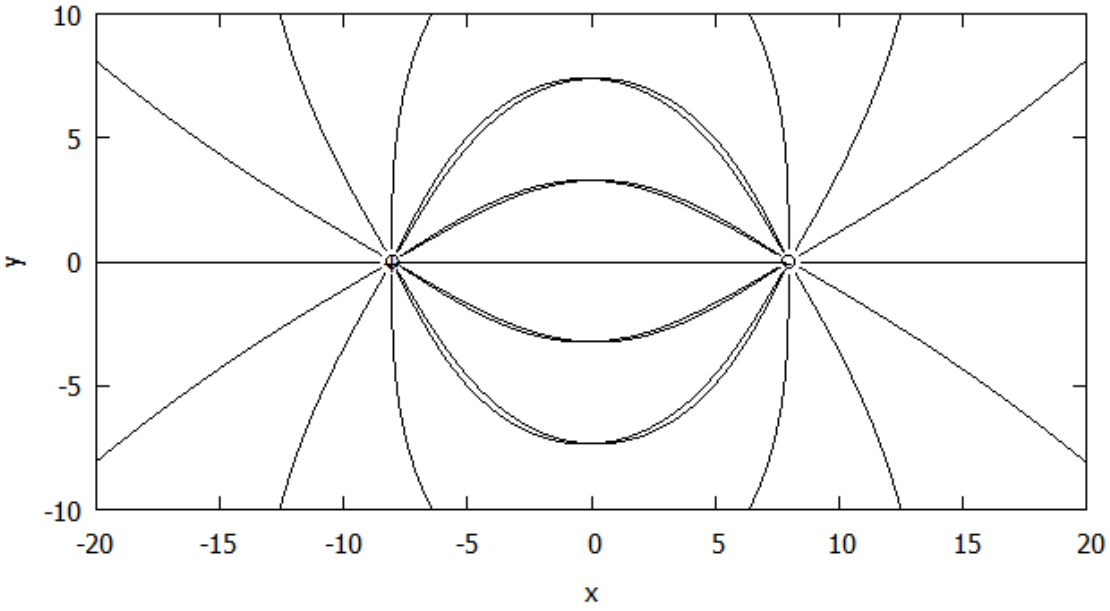
Datei Bearbeiten Format Ansicht Hilfe

```
-7.5 0.0
-7.57 0.25
-7.75 0.43
-8.0 0.5
-8.25 0.43
-8.43 0.25
-8.5 0.0
-8.43 -0.25
-8.25 -0.43
-8.0 -0.5
-7.75 -0.43
-7.57 -0.25
0.0 10.0
5.0 10.0
10.0 10.0
15.0 10.0
20.0 10.0
20.0 5.0
20.0 0.0
20.0 -5.0
20.0 -10.0
15.0 -10.0
10.0 -10.0
5.0 -10.0
0.0 -10.0
```

Fortran-Programm für Feldlinien und Startpunkte-Datei.
efact ist die Schrittweite, hier bereits kleiner als 1.
Die Ausgabedatei wird in gnuplot mit *plot with lines* dargestellt.

Wir sollten Startpunkte äquidistant in einem Bereich ansetzen, von dem wir wissen, dass die Feldstärke dort überall gleich ist. Auf einem eckigen Rand ist sie dies bei der gegebenen Ladungsanordnung nicht. Es ist zwar die negative Ladung vorhanden, aber wenn sich diese recht weit weg von der positiven befindet, ist eine gute Annahme, dass auf einem engen Kreis um die positive Ladung herum die Feldstärke in alle Richtungen gleich ist. Desgleichen auf einem engen Kreis um die negative Ladung. Jetzt wird's pragmatisch. Neben unserem Startkreis um die positive Ladung kontruieren wir einen ebensolchen um die negative Ladung. Und da Feldlinien nun einmal auf negative Ladungen zulaufen, durchlaufen wir diese Feldlinien eben

rückwärts. Dazu wird nach einer kleinen Änderung das Programm ein zweites Mal laufen gelassen. Ändern kann man entweder die Vorzeichen der Ladungen oder die des elektrischen Feldes. Nicht vergessen, zwischen beiden Programmläufen, ebenso wie die Startpunktedatei zu ändern, die Ausgabedatei umzubenennen, damit sie nicht überschrieben wird. Wenn die beiden Dateien "feldlinien_vonplus.dat" und "feldlinien_vonminus.dat" heißen, kann man sie zusammen zeichnen mit *plot ''feldlinien_vonplus.dat" with lines, "feldlinien_vonminus.dat" with lines.* Das Ergebnis ist wie erwartet das folgende Bild.



Feldlinien von der positiven und von der negativen Ladung aus berechnet

In der Mitte wurden noch die von beiden Ladungen aus berechneten Feldlinien in den Daten belassen. Sie liegen nicht exakt aufeinander. Wegen der endlichen Schrittweite enden Linien von der positiven Ladung nicht genau auf dem kleinen Kreis um die negative Ladung, auf dem Linien von dort starten, und umgekehrt. Solche Duplikate sind im folgenden Bild durch Wahl der Startpunkte herausgenommen und die Startpunkte enger gesetzt, um wie im anfangs gezeigten Bild aus einem Lehrbuch mehr Feldlinien zu erhalten.

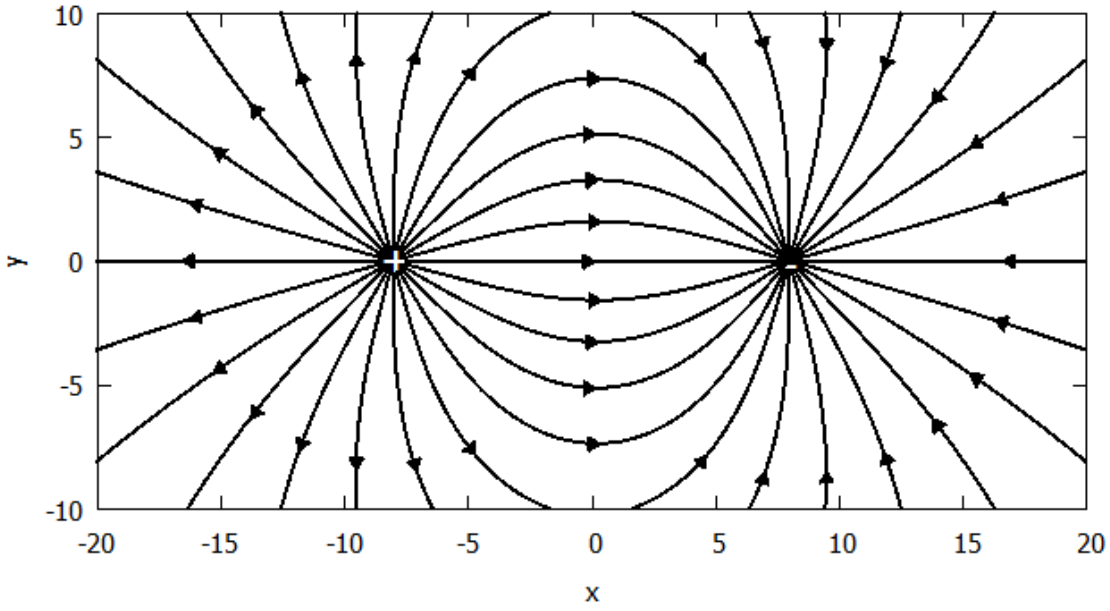


Bild mit mehr gezeichneten Feldlinien

Da die vollständig im Bild enthaltenen Feldlinien von der positiven zur negativen Ladung nicht im selben Abstand von dieser enden (in einen kleinen Kreis unterschiedlich weit hineinragen), wurden hier die Orte der Ladungen mit kleinen schwarz gefüllten Kreisen überdeckt und mit weißer Schrift versehen. Der Optik wegen wurden die Linien breiter gezeichnet. Und die kleinen Pfeile auf den Feldlinien, wie man sie aus Lehrbüchern kennt, möchten wir auch nicht missen. Sie sind in *gnuplot* als Polygone (Dreiecke) gemacht. Aber wir möchten natürlich nicht jede Position und Richtung eines Pfeils im Bild von Hand bestimmen, sondern vom Programm mit berechnen lassen. Dazu wird das Programm um eine Ausgabedatei erweitert:

```
open(unit=13,file='pfeile.p',status='unknown',action='write',iostat=io_error)
```

In *gnuplot* setzt man dann *load "pfeile.p"* in einer Zeile vor den plot-Befehl für die Feldlinien.

Für das vorherige Bild wurden drei Pfeile-Dateien produziert. Auf den Feldlinien, die von der positiven zur negativen Ladung vollständig im Bild sind, wurden die Pfeile an dem Punkt produziert, der die x=0 Linie überschreitet:

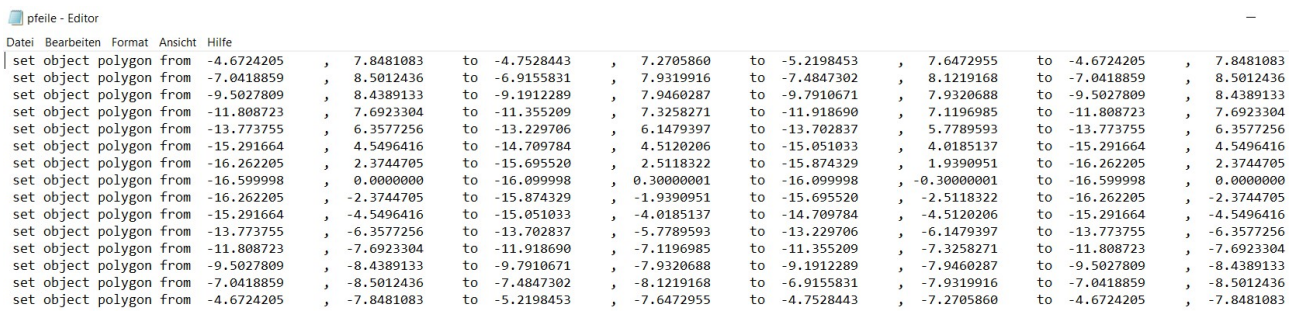
```
if ((xalt.lt.0.0).and.(x.ge.0.0)) then
  write(13,*) 'set object polygon from',x,',',y,'to',x-0.5,',',y+0.3,'to',
  x-0.5,',',y-0.3,'to',x,',',y
endif
```

Für die von der positiven Ladung aus dem Bild laufenden Feldlinien wurden die Pfeile in dem Schritt gesetzt, wo der Abstand zur Ladung 8 überschreitet.

```
if ((r1.gt.8.0).and.(r1.lt.8.0+efact)) then
  ecke2x=x-0.5*ex/e+0.3*ey/e
  ecke2y=y-0.5*ey/e-0.3*ex/e
  ecke3x=x-0.5*ex/e-0.3*ey/e
  ecke3y=y-0.5*ey/e+0.3*ex/e
  write(13,*) 'set object polygon from',x,',',y,'to',ecke2x,',',ecke2y,'to',
  ecke3x,',',ecke3y,'to',x,',',y
endif
```

(Im Programm passen die write-Befehle auf eine Zeile.)

Als Beispiel sei diese Pfeil-Datei hier gezeigt:


pfeile - Editor

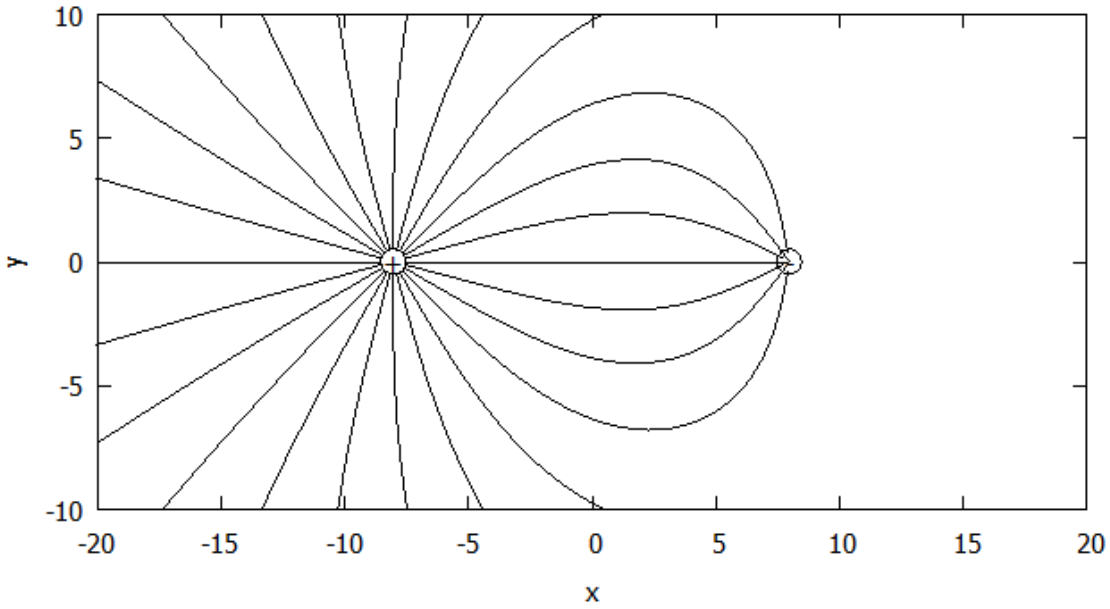
Datei Bearbeiten Format Ansicht Hilfe

```
set object polygon from  -4.6724205     ,   7.8481083     to  -4.7528443     ,   7.2705860     to  -5.2198453     ,   7.6472955     to  -4.6724205     ,   7.8481083
set object polygon from  -7.0418859     ,   8.5012436     to  -6.9155831     ,   7.9319916     to  -7.4847302     ,   8.1219168     to  -7.0418859     ,   8.5012436
set object polygon from  -9.5027809     ,   8.4389133     to  -9.1912289     ,   7.9460287     to  -9.7910671     ,   7.9320688     to  -9.5027809     ,   8.4389133
set object polygon from  -11.808723     ,   7.6923304     to  -11.355209     ,   7.3258271     to  -11.918690     ,   7.1196985     to  -11.808723     ,   7.6923304
set object polygon from  -13.773755     ,   6.3577256     to  -13.229706     ,   6.1479397     to  -13.702837     ,   5.7789593     to  -13.773755     ,   6.3577256
set object polygon from  -15.291664     ,   4.5496416     to  -14.709784     ,   4.5120206     to  -15.051033     ,   4.0185137     to  -15.291664     ,   4.5496416
set object polygon from  -16.262205     ,   2.3744705     to  -15.695520     ,   2.5118322     to  -15.874329     ,   1.9390951     to  -16.262205     ,   2.3744705
set object polygon from  -16.599998     ,   0.0000000     to  -16.099998     ,  0.30000001     to  -16.099998     , -0.30000001     to  -16.599998     ,   0.0000000
set object polygon from  -16.262205     ,  -2.3744705     to  -15.874329     ,  -1.9390951     to  -15.695520     ,  -2.5118322     to  -16.262205     ,  -2.3744705
set object polygon from  -15.291664     ,  -4.5496416     to  -15.051033     ,  -4.0185137     to  -14.709784     ,  -4.5120206     to  -15.291664     ,  -4.5496416
set object polygon from  -13.773755     ,  -6.3577256     to  -13.702837     ,  -5.7789593     to  -13.229706     ,  -6.1479397     to  -13.773755     ,  -6.3577256
set object polygon from  -11.808723     ,  -7.6923304     to  -11.918690     ,  -7.1196985     to  -11.355209     ,  -7.3258271     to  -11.808723     ,  -7.6923304
set object polygon from  -9.5027809     ,  -8.4389133     to  -9.7910671     ,  -7.9320688     to  -9.1912289     ,  -7.9460287     to  -9.5027809     ,  -8.4389133
set object polygon from  -7.0418859     ,  -8.5012436     to  -7.4847302     ,  -8.1219168     to  -6.9155831     ,  -7.9319916     to  -7.0418859     ,  -8.5012436
set object polygon from  -4.6724205     ,  -7.8481083     to  -5.2198453     ,  -7.6472955     to  -4.7528443     ,  -7.2705860     to  -4.6724205     ,  -7.8481083
```

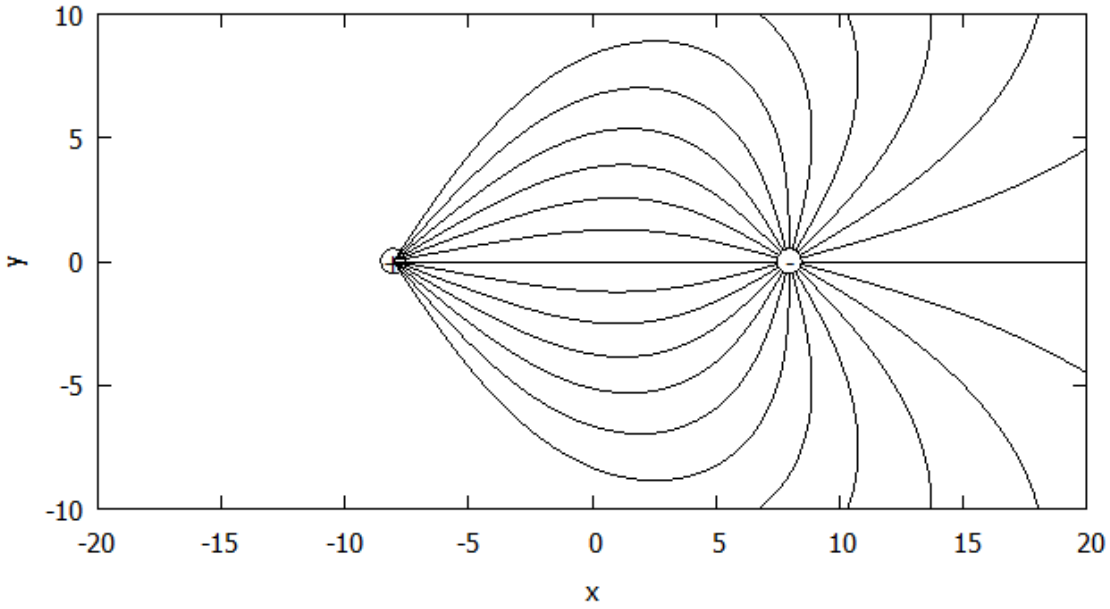
Für die von der negativen Ladung aus rückwärts berechneten Feldlinien wurden die Pfeile ebenso in dem Schritt erzeugt, bei dem der Abstand 8 zur Ladung überschritten wird; man muss die Vorzeichen anpassen, damit die Pfeile in die richtige Richtung zeigen.

Ungleich große Ladungen

Was ändert sich bei ungleich großen Ladungen und wie müssen wir eventuell unser Vorgehen abändern, um ein konsistentes Feldlinienbild zu erhalten? In den folgenden Bildern ist die positive Ladung links 2,5mal größer als die negative rechts. Im oberen Bild wurden 24 Startpunkte auf einem kleinen Kreis um die positive Ladung gewählt, im unteren entsprechende 24 Startpunkte auf einem kleinen Kreis um die negative Ladung.



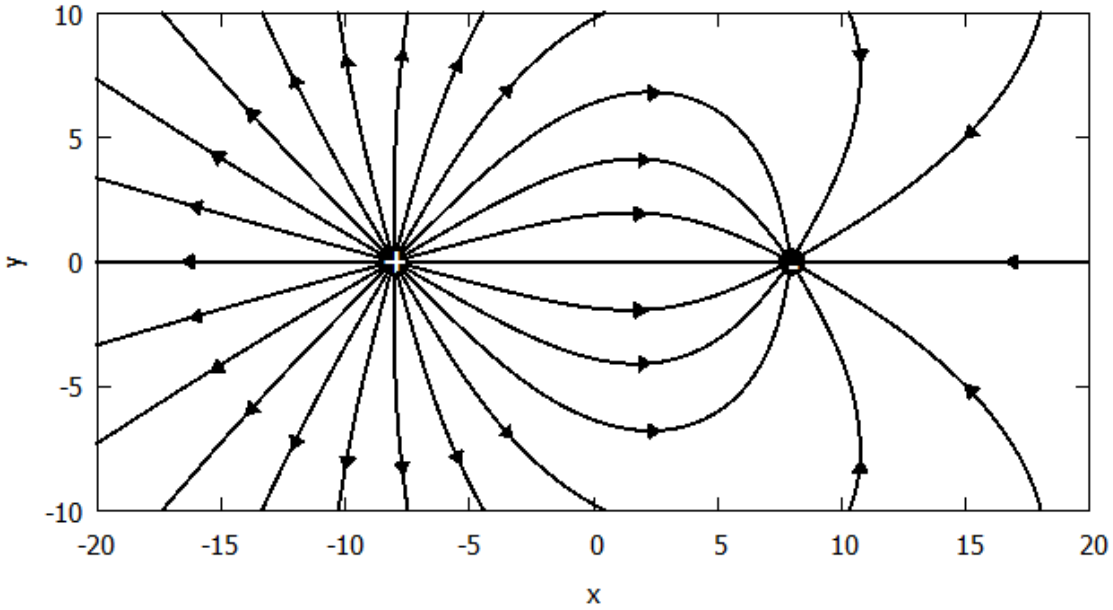
Linien von der größeren positiven Ladung aus berechnet



Linien von der kleineren negativen Ladung aus berechnet

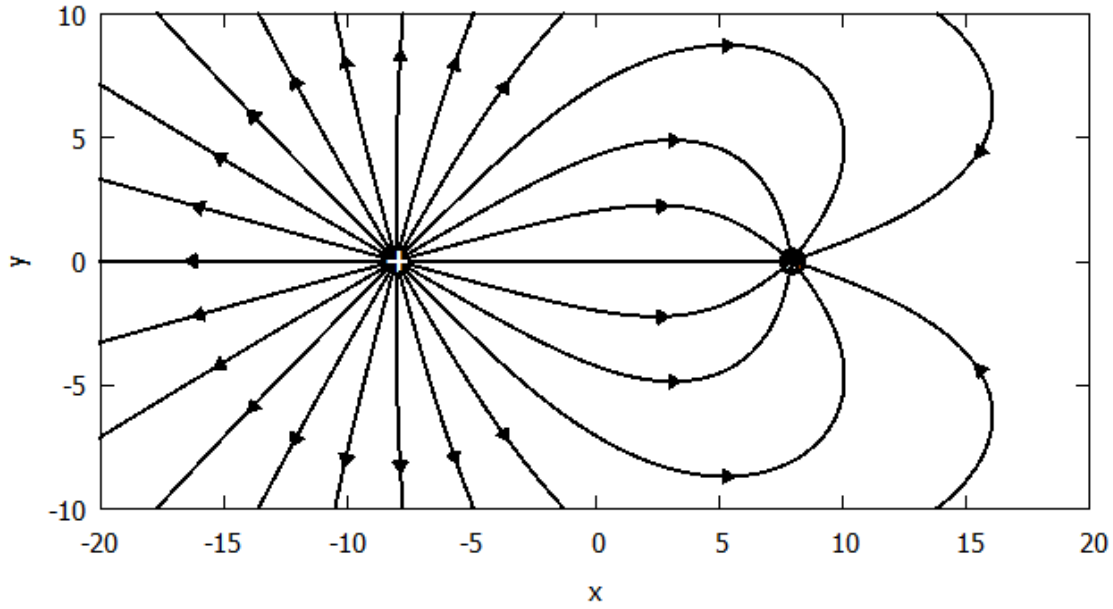
Die Linien, die von der positiven zur negativen Ladung laufen, gehen bei der großen positiven Ladung in einem engeren Winkel zusammen als bei der kleinen negativen. So können wir die beiden Bilder nicht übereinanderlegen. Das gäbe in der Mitte viel zu viele Linien und die Linienabstände in den Außenbereichen links und rechts würden im Verhältnis kein angemessenes Verhältnis der Stärke des Feldes wiederspiegeln. Wir ergänzen das obere Bild, indem wir von der negativen Ladung aus rückwärts nur solche Linien berechnen, die das Bild verlassen, und legen auch weniger Startpunkte auf den kleinen Kreis um die negative Ladung, mit dem geschätzten

Winkelabstand der Linien, die von der positiven Ladung ankommen. Die kleinen Pfeile auf den Linien in der Mitte wurden hier auf den Vorzeichenwechsel von E_y gesetzt.



vollständiges Feldlinienbild für Ladungsverhältnis 2,5 : 1

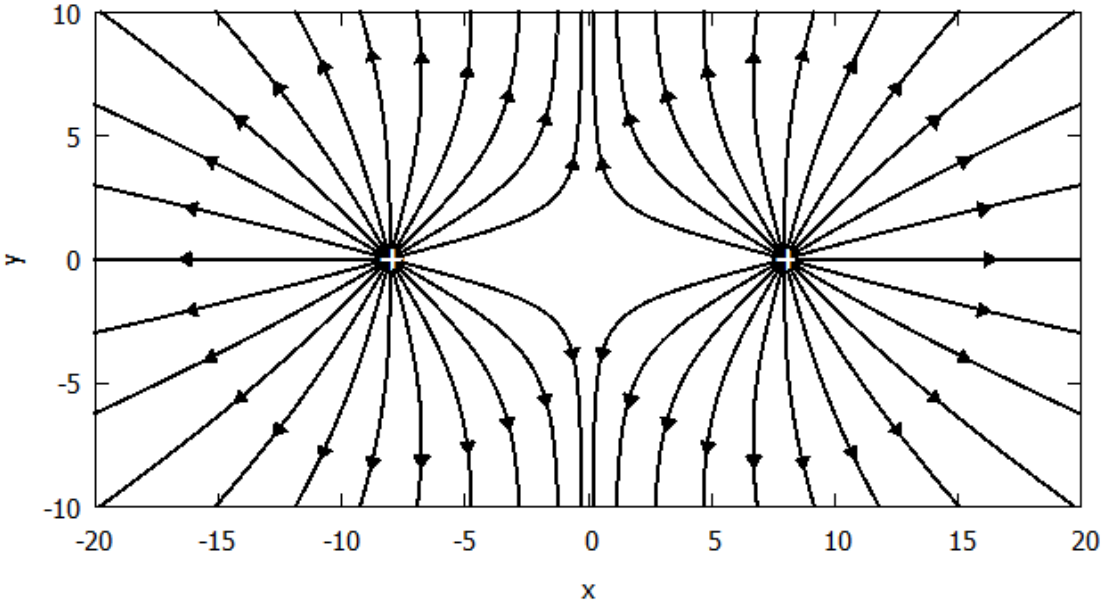
Und es sei noch eine extremere Situation mit Ladungsverhältnis 5:1 gezeigt:



vollständiges Feldlinienbild für Ladungsverhältnis 5 : 1

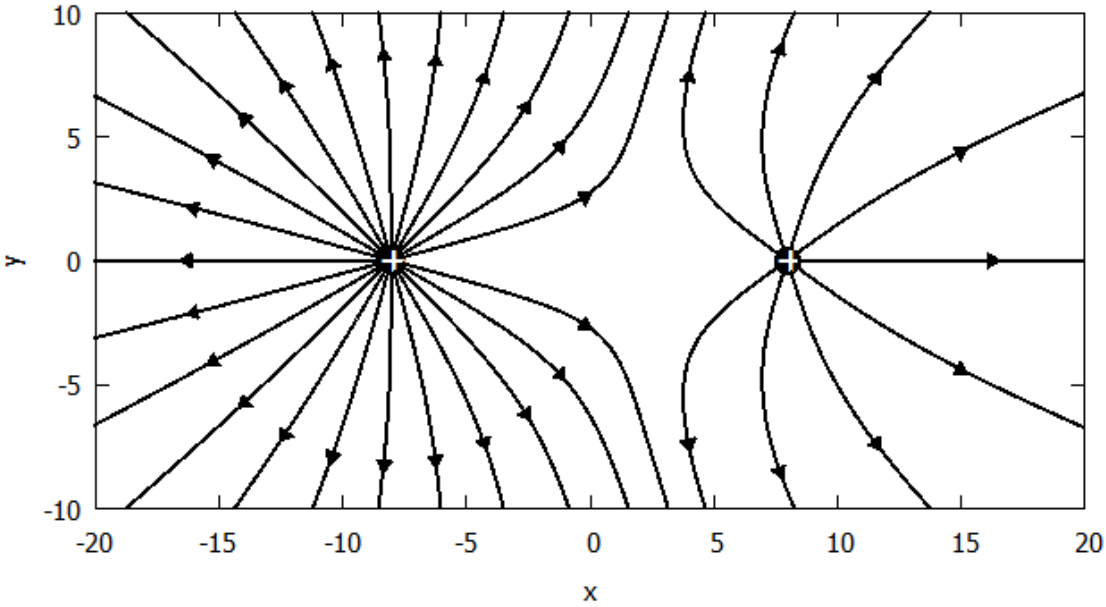
Gleichnamige Ladungen

Feldlinienbilder für gleichnamige Ladungen zu erstellen, ist einfacher als für ungleichnamige. Man setzt die beiden Ladungen als positiv und kann in einem Programmlauf Feldlinien von Startpunkten um beide Ladungen berechnen, die alle von den Ladungen wegführen. Einzig Startpunkte auf der Verbindungslinie der beiden Ladungen sind zu vermeiden. Auf dieser Linie bleibt das Programm in einer Endlosschleife hängen, wo das Feld die Richtung wechselt bzw. den Punkt überschreitet, wo der Feldvektor exakt der Nullvektor ist.



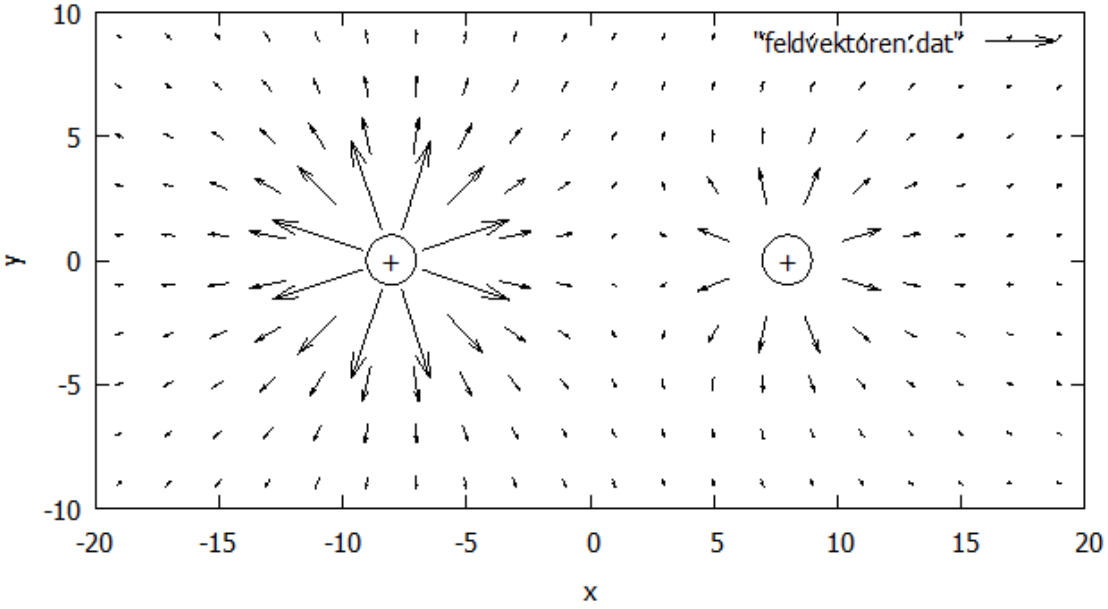
Feldlinienbild für zwei gleichnamige gleich große Ladungen

Für zwei gleich große positive Ladungen (siehe oben) ist das Feldlinienbild auch aus Lehrbüchern bekannt. Für gleichnamige, aber ungleich große Ladungen drängen die Feldlinien der größeren Ladung die der kleineren weg (siehe unten). Sollen wir von beiden Ladungen gleich viele Feldlinien ausgehen lassen? Das würde eine zu große Feldstärke nahe der kleineren Ladung suggerieren. Das Verhältnis der Feldlinienanzahl von jeder Ladung aus ist etwa dem Verhältnis der Ladungen entsprechend gewählt. Außer dem Punkt auf der Verbindungslinie beider Ladungen sind die Startpunkte äquidistant auf Kreisen um die Ladungen gesetzt.

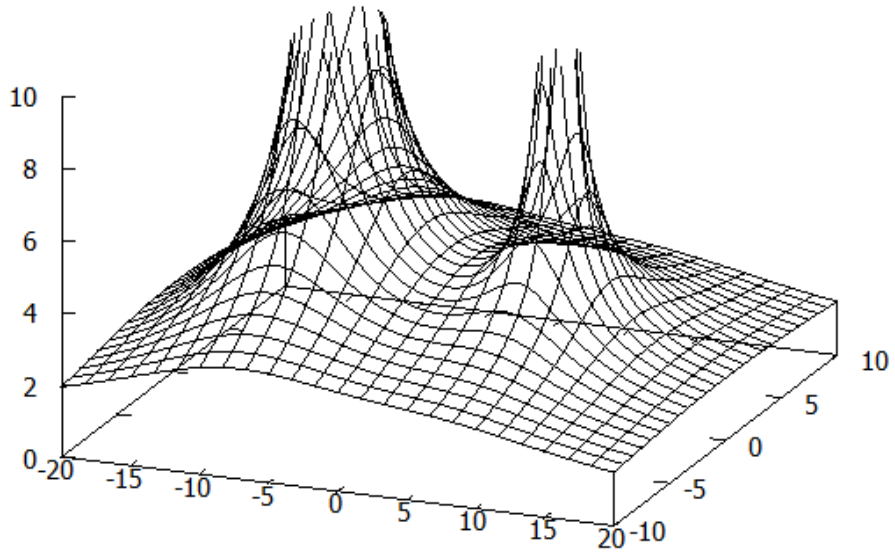


Feldlinienbild für zwei gleichnamige Ladungen mit Verhältnis 2,5 : 1

Am oberen und am unteren Rand, wo die Feldlinien von beiden Ladungen zusammenkommen, scheint es einen zu abrupten Wechsel in der Feldliniendichte zu geben. Um der Frage nachzugehen, wie realistisch das ist, sind auf der nächsten Seite für das vorherige Beispiel mit Ladungsverhältnis 2,5 : 1 das Feldvektorenbild sowie die Potentiallandschaft gezeigt.



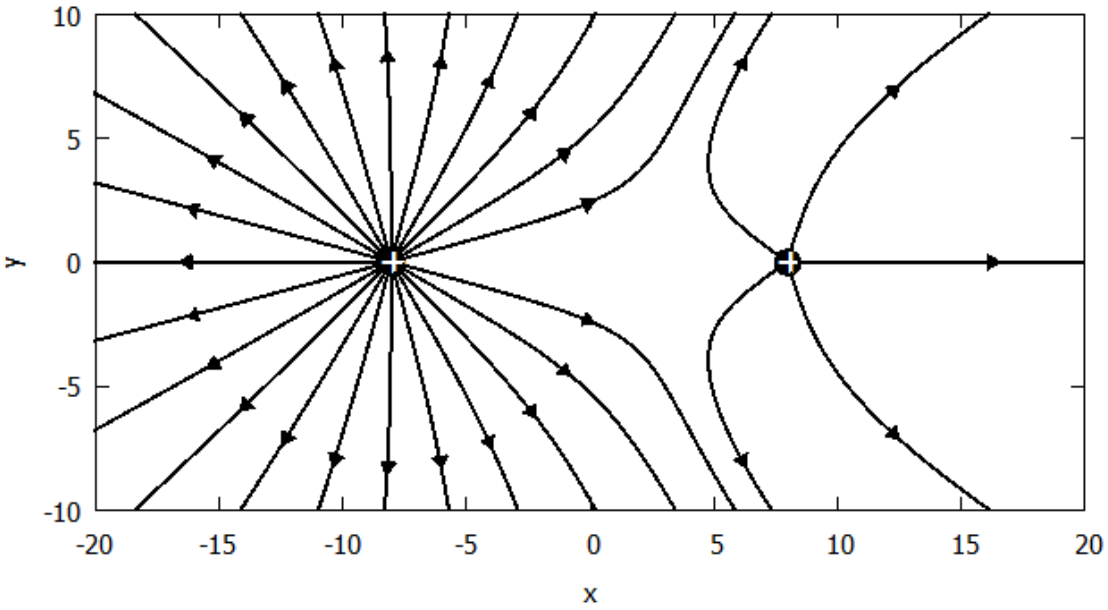
Feldvektoren für zwei gleichnamige Ladungen mit Verhältnis 2,5 : 1



Potentiallandschaft für zwei gleichnamige Ladungen mit Verhältnis 2,5 : 1

Dem Vektorenbild ist schon zu entnehmen, dass das Feld bei x-Koordinate -8 stärker ausfällt als bei x-Koordinate 8. Auch die Potentiallandschaft zeigt, dass bei y-Koordinate -10 wir noch so nahe an den Ladungen sind, dass es bei der linken großen Ladung steiler ist als bei der rechten kleinen, allerdings mit einem allmählichen Übergang. Hier kann das Feldlinienbild nicht überall die richtige Liniendichte produzieren; die bei den Ladungen gestarteten Linien müssen ja irgendwo hin! Da sollte man sich die mittleren Linien von beiden Ladungen vielleicht als zu einer zusammenlaufend denken, was wohl auch passiert wäre, wären wir mit noch flacheren Winkeln gegen die Horizontale gestartet.

Und der Vollständigkeit halber machen wir es auch mit den gleichnamigen Ladungen extremer und wählen noch das Verhältnis 5:1. Ist bei den gleichnamigen Ladungen nicht so spektakulär wie bei den ungleichnamigen.



Feldlinienbild für zwei gleichnamige Ladungen mit Verhältnis 5 : 1

Schlusswort

Ich hoffe gezeigt zu haben, dass man mit einem Graphikprogramm, das die hier gezeigten Optionen von *gnuplot* hat, und einer einfachen Programmiersprache, die es erlaubt, die benötigten Datensätze zu produzieren, berechnete Feldlinienbilder erstellen kann. Freunde der objektorientierten Programmierung mögen mir verzeihen, aus Gewohnheit Spaghetti-Code in *Fortran* zu schreiben und die Methode zur Erstellung von Feldlinienbildern in ihre Welt übersetzen. Ladungen, Abstände usw. hätte man natürlich als Tastatur-Eingabewerte programmieren können. Da meist nur ein Wert zu ändern war, stellte sich dies direkt im Programm und mit Neu-Kompilieren durch Befehlswiederholung in der Eingabe-Aufforderung als praktikabler heraus. Zudem haben wir gesehen, dass man Feldlinienbilder sinnigerweise aus den enthaltenen Gruppen von Feldlinien "zusammenbastelt". Statt mühsamer Fallunterscheidungen tun es ein paar Vorzeichenänderungen von Programmlauf zu Programmlauf auch.

Mit der Erläuterung sinniger Startpunkte in Bezug auf ungleichnamige sowie verschieden große Ladungen ebenso wie mit der Diskussion der Liniendichte wurden die Schwierigkeiten beim Erstellen von Feldlinienbildern aufgezeigt.

Ursprünglich waren allgemeine Ladungskonstellationen angedacht. Dieser lange Artikel ist dann doch bei der Situation zweier Punktladungen geblieben. Wirklich self-made würde ich auch für jede Konstellation ein eigenes kleines Programm schreiben. Die vorgestellte Methode setzt voraus, dass das elektrische Feld an jedem Ort als analytische Funktion aufstellbar ist und nicht selber noch numerisch ermittelt werden muss. Mehr als zwei Punktladungen wären kein Problem. Auch eine Punktladung vor einer metallischen Wand kann über die Spiegelladung behandelbar sein. Metallische Objekte in einem Kondensatorfeld stellen ein anderes Kaliber dar. In eine andere Richtung ist die Ausweitung der Methode auf magnetische Felder (aus Dipolfeldern zusammengesetzt) denkbar.

OK, OK, Simulationen, die solche Feldlinienbilder erstellen, gibt es heutzutage natürlich einfach zu finden online. Z.B. https://academo.org/demos/electric-field-line-simulator/ oder https://www.leifiphysik.de/elektrizitaetslehre/ladungen-elektrisches-feld/versuche/elektrisches-feld-und-potential-simulation (was nichts anderes ist als didaktikonline.physik.uni-muenchen.de/programme/e_feld/E_Feld_leifi.html).
Um Schüler die Physik erkunden zu lassen, ist das Fortran-Programmen vorzuziehen.

Ich fand es eine wertvolle Herausforderung zu erforschen, mit welcher Logik und Systematik solche Liniendiagramme zu erzeugen sind. Selber-Machen schult ein kritisches Auge für die kleinen Schwierigkeiten der Feldliniendarstellungen. Ziel war in diesem Fall auch nicht Material, mit dem Schüler die Bilder produzieren sollen, sondern die Erstellung korrekter Abbildungen zum Thema wie in Lehrbüchern. Mit einem eigenen Programm stehen in der Darstellung viel mehr Anpassungmöglichkeiten zur Verfügung als in einer fertigen Simulationsumgebung.