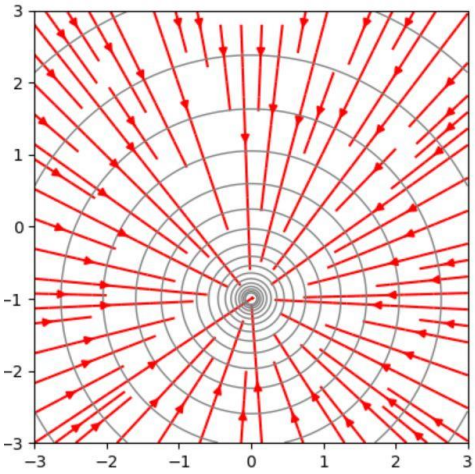
שאלה 8:

א. שדות שלושת המטענים בנפרד:

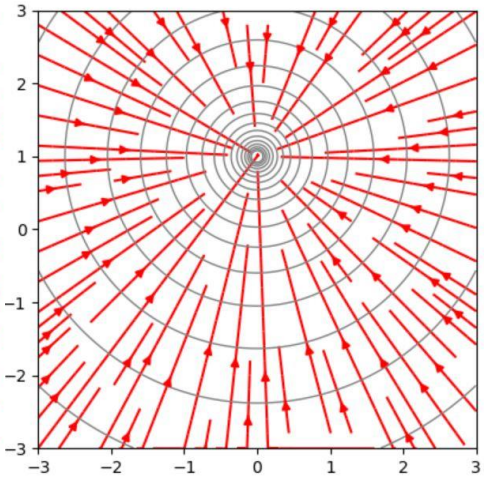
$q = -1$

Positioned at (-1,0)



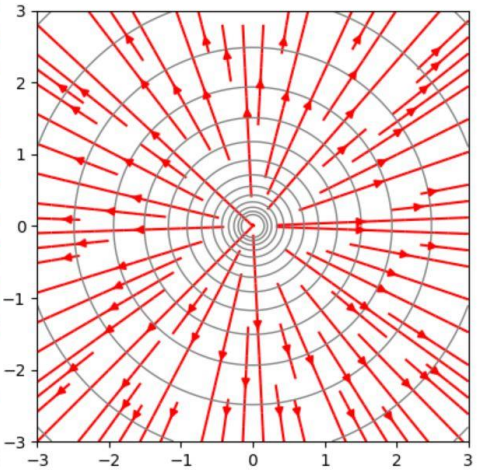
$q = -1$

Positioned at (1,0)

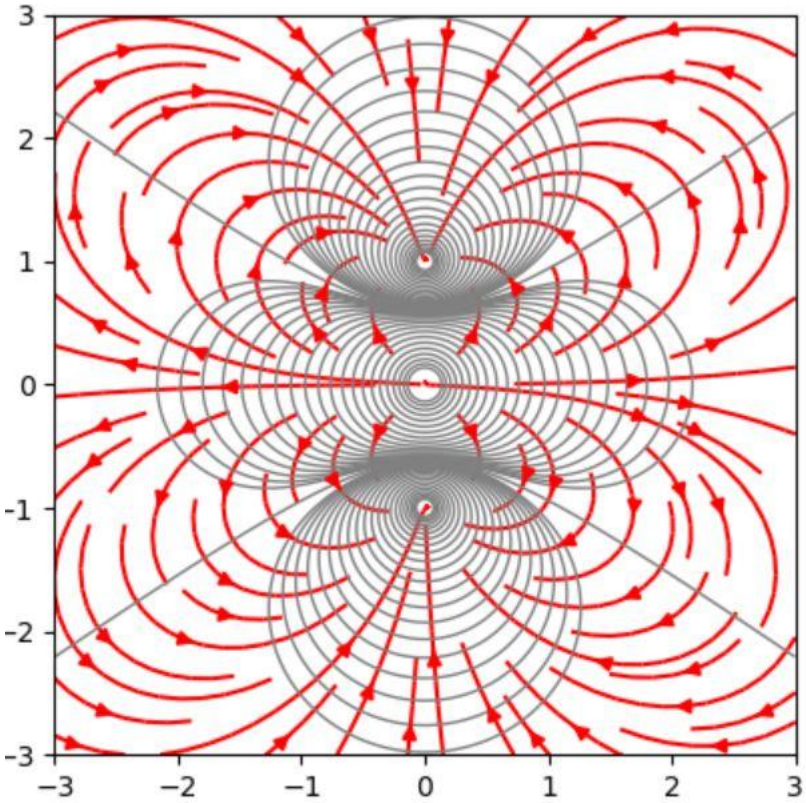


$Q = 2$

Positioned at (0,0)

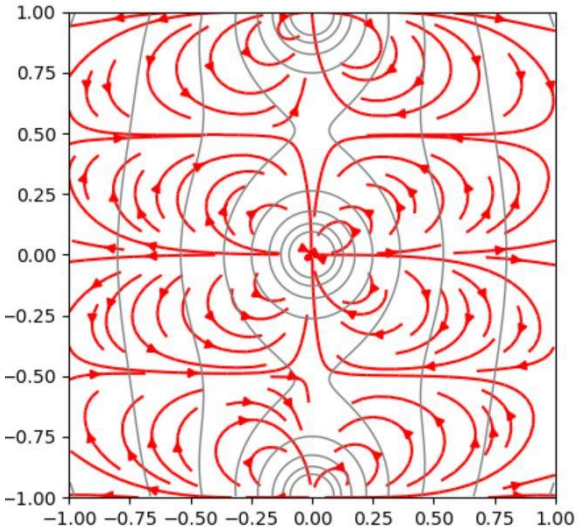


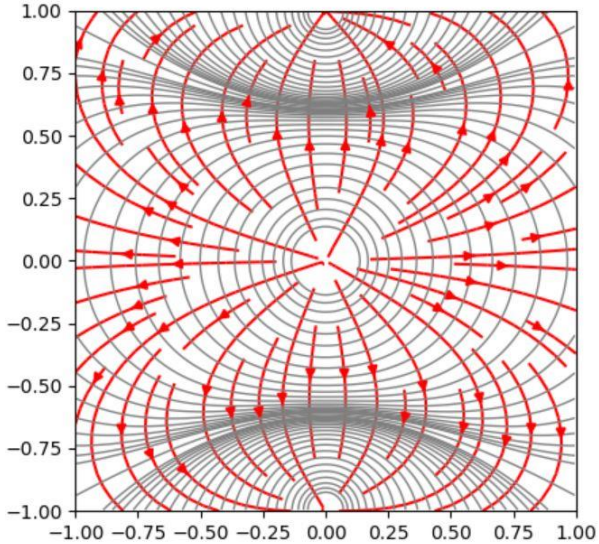
השדה של שלושת המטענים יחד:



ב.

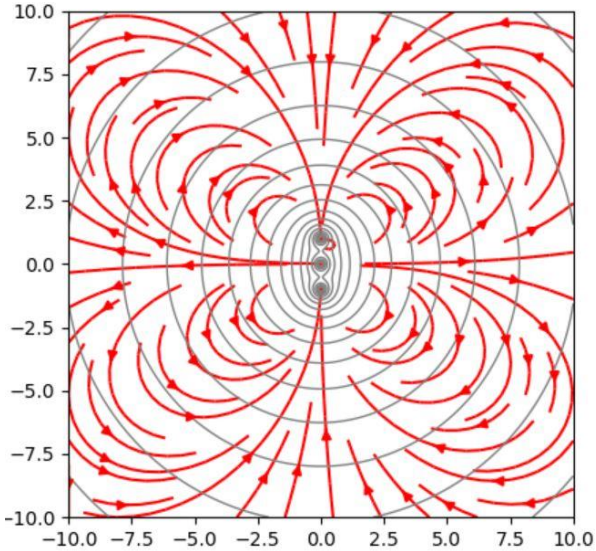
תיאוריה(ימין) וקירוב(שמאל) לפי סקאלה בטווח קטן

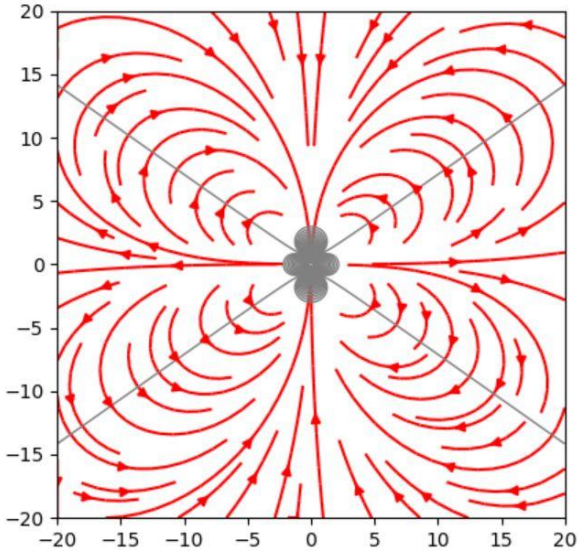




ניתן לראות שההתאמה לא מדויקת בלשון המעטה.

תיאוריה(ימין) וקירוב(שמאל) לפי סקאלה בטווח גדול (פי 20)





ניתן לראות שבמקרה זה הקירוב הרבה יותר טוב ואכן מייצג בצורה נאמנה את התיאוריה.

התייחסנו לרחוק באמצעות הקשר $r \gg a = 1$ (עד כמה שהפייתון מאפשר)

```python
#!/usr/bin/python3
#
# Plot electric potential
#
import numpy as np
from scipy.optimize    import curve_fit
from matplotlib import pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
import matplotlib.cm as cm
import matplotlib.mlab as mlab
#
#==============================================================================
epsilon_0=1
pai = np.pi
#==============================================================================
#
# simple example: plot the field of a single charge.
#                 To make the plot units simple i just set epsilon_0 to be 1.
#
# general case  : To plot a dipole or a more complicated plannar charge settings,
#                 just write down the electric field components
#                     Ex(x,y,0),Ey(x,y,0)
#                 and the electric potential
#                     V(x,y,0).
#

def electric_field(x,y,q):
    k = 1./(4*pai*epsilon_0)
    r = np.sqrt(x**2+y**2)
    V =  k*q*(1./r)
    Ex = k*q*(x/r**3)
    Ey = k*q*(y/r**3)
    return V,Ex,Ey


def approx_electric_field(x,y):
    qq=-1
    a=1
    k = 1. / (4 * pai * epsilon_0)
    r = np.sqrt(x ** 2 + y ** 2)
    V = k * qq * (1. / r)
    cos_t = x/r
    sin_t = y/r
    Ex = 3*(5*sin_t**2-1)*(k*qq*a**2*cos_t)/(r**4)
    Ey = (15*sin_t**2-9)*(k*qq*a**2*sin_t)/(r**4)
    return V, Ex, Ey
#==============================================================================
# set grid
x_range = 10
delta_x = 0.01
delta_y = delta_x
y_range = x_range
xvec = np.arange(-x_range,x_range+delta_x,delta_x)
yvec = np.arange(-y_range,y_range+delta_y,delta_y)
Xgrid, Ygrid = np.meshgrid(xvec,yvec)
#==============================================================================
"""
this part is shown as an example to the way the functions are being called
the same had been applied to the actual electric field (but is not shown here
because it was implemented in the same way)
"""
V1,Ex1,Ey1 = approx_electric_field(Xgrid,Ygrid-1)
V2,Ex2,Ey2 = approx_electric_field(Xgrid,Ygrid+1)
V3,Ex3,Ey3 = approx_electric_field(Xgrid,Ygrid)
Vf = V1+V2+V3
Exf = Ex1+Ex2+Ex3
Eyf = Ey1+Ey2+Ey3
#==============================================================================
plt.figure()
xp = np.exp(np.arange(-5., 0.01, 0.25))
xm = [-x for x in xp[::-1]]
zz = [0.]
levels = np.concatenate((xm,zz,xp))
plt.streamplot(Xgrid, Ygrid, Exf, Eyf,
               color='r',linewidth=1.5,density=[0.75,1.]
               ,maxlength=10.)
CS = plt.contour(Vf, levels,
                 origin='lower',colors='grey',
                 linestyles='solid',
                 linewidths=1.,
                 extent=(-x_range, x_range, -y_range, y_range))
plt.axis([-x_range,x_range,-y_range,y_range])
plt.axes().set_aspect('equal','box')
plt.show()
```