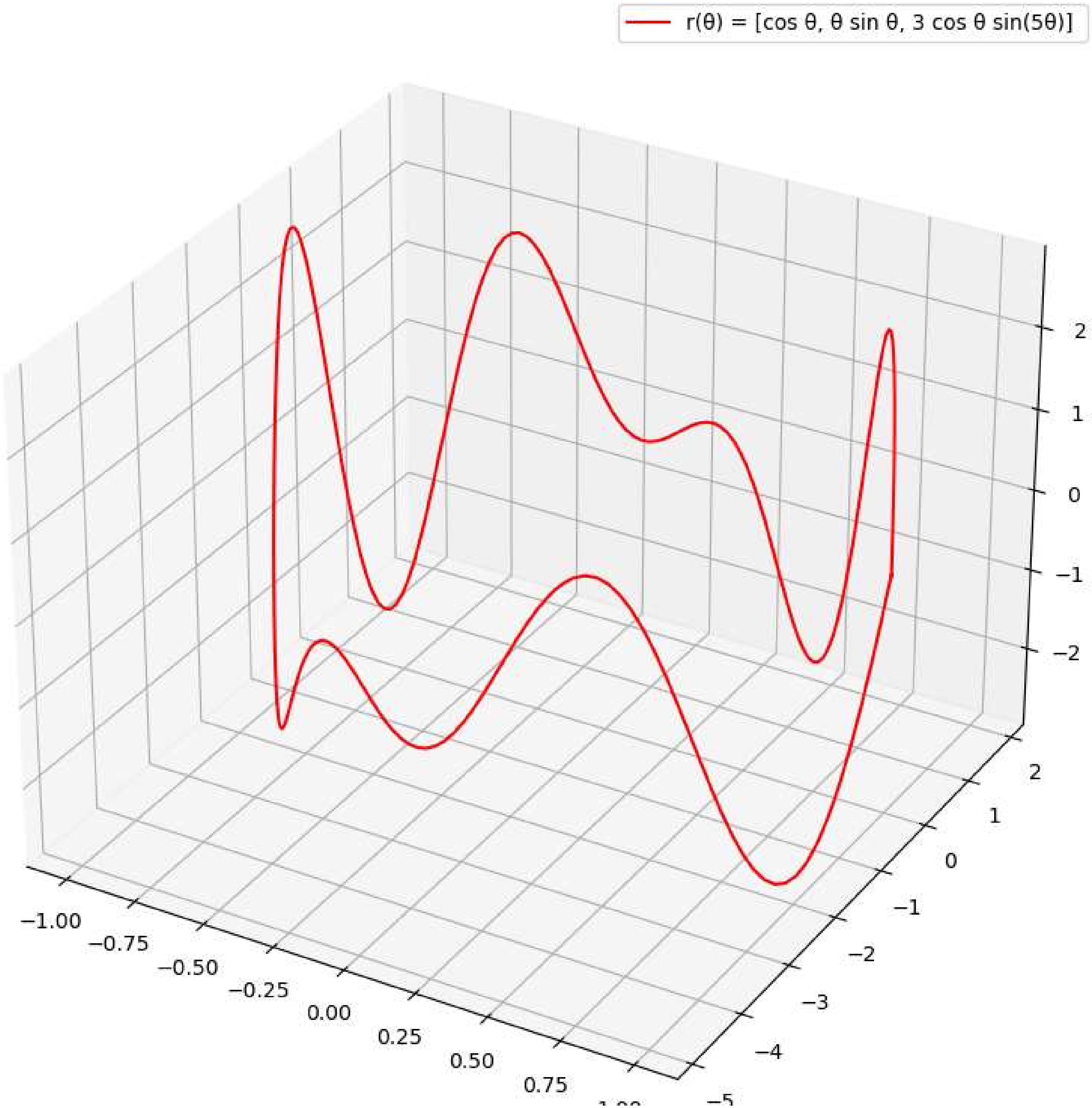
# Matematikk 3 Oblig 4

Anthony Andreassen | 258127

## Oppgave 1d

Plotting av parametrisert kurve

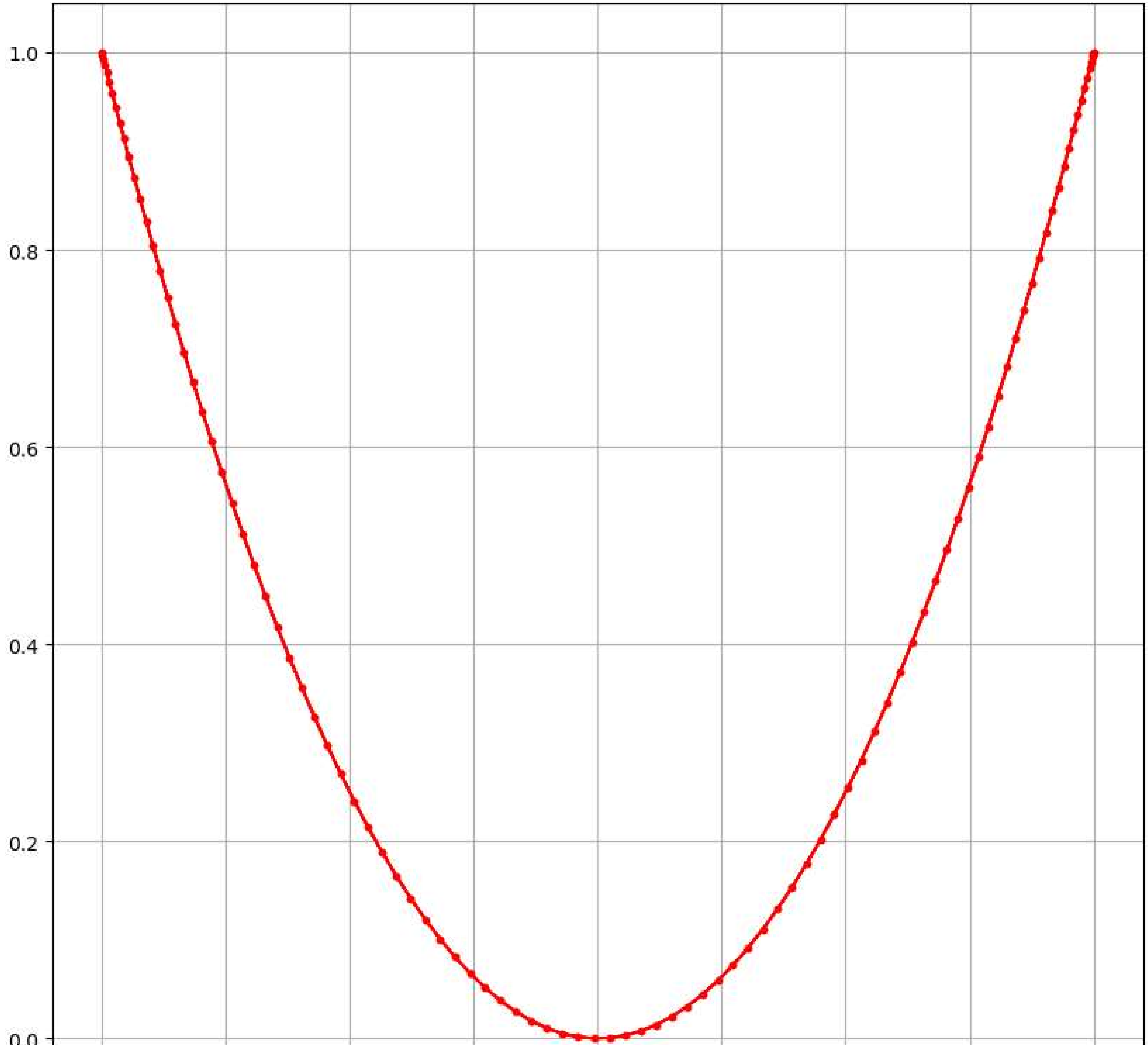
In [8]:
```
# modules
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure(figsize =(10, 10))
ax = fig.add_subplot(projection='3d')

t = np.linspace(0, 2*np.pi, 200)
x = np.cos(t)
y = t*np.sin(t)
z = 3*np.cos(t)*np.sin(5*t)

ax.plot(x, y, z, "r-", label='r(θ) = [cos θ, θ sin θ, 3 cos θ sin(5θ)] ')

ax.legend()
plt.show()
```

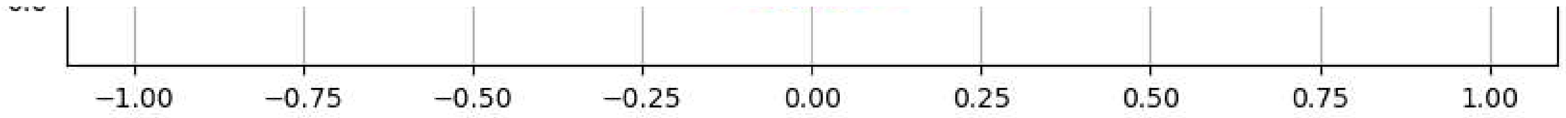

r(θ) = [cos θ, θ sin θ, 3 cos θ sin(5θ)]
2
1
0
−1
−2
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
2
1
0
−1
−2
−3
−4


## Oppgave 2b

Parametrisering for parabola i xy-plan som går gjennom punktene $(-1, 1)$,$(0, 0)$ og $(1, 1)$

In [3]:
```
fig = plt.figure(figsize =(10, 10))
ax = fig.add_subplot()

r = 1
t = np.linspace(0, 2*np.pi, 200)
x = r*np.cos(t)
y = -r**2 * (np.sin(t))**2 + 1

ax.plot(x, y, "r.-")

plt.grid()
plt.show()
```


1.0
0.8
0.6
0.4
0.2
0.0

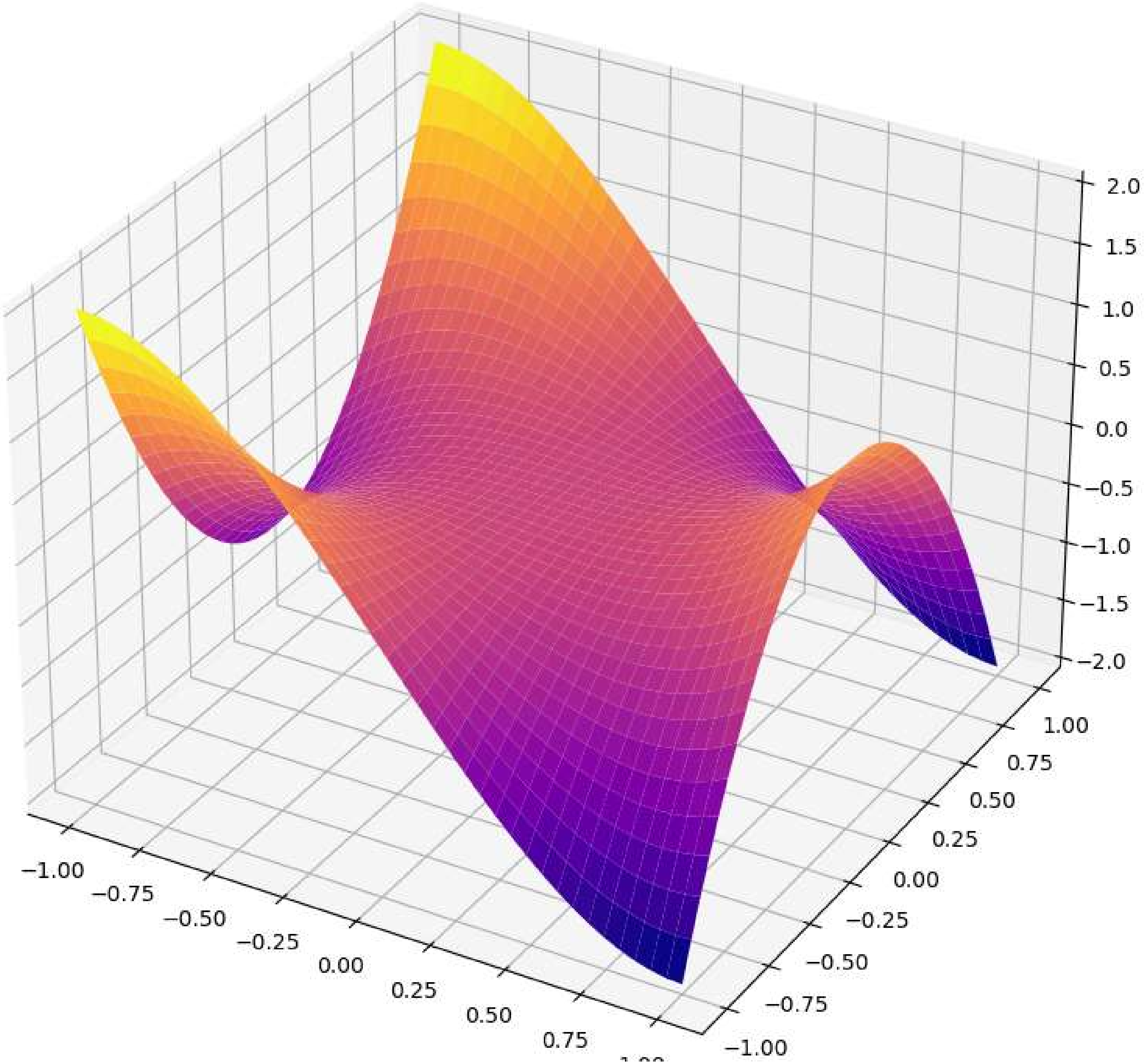



## Oppgave 3b

Plotting av parametrisert flate

$$\vec{r}(x, y) = [x, y, x^3 - 3xy^2] \text{ med } (x, y) \in [-1, 1] \times [-1, 1].$$

In [4]:
```
def f(x, y):
    return x**3 - 3*x*y**2

x = np.linspace(-1, 1, 40)
y = np.linspace(-1, 1, 40)
x, y = np.meshgrid(x, y)
X = x
Y = y
Z = f(X, Y)

fig = plt.figure(figsize = (10, 10))
ax = plt.axes(projection = '3d')
ax.plot_surface(X, Y, Z, cmap = 'plasma', edgecolor = 'none')
plt.show()
```

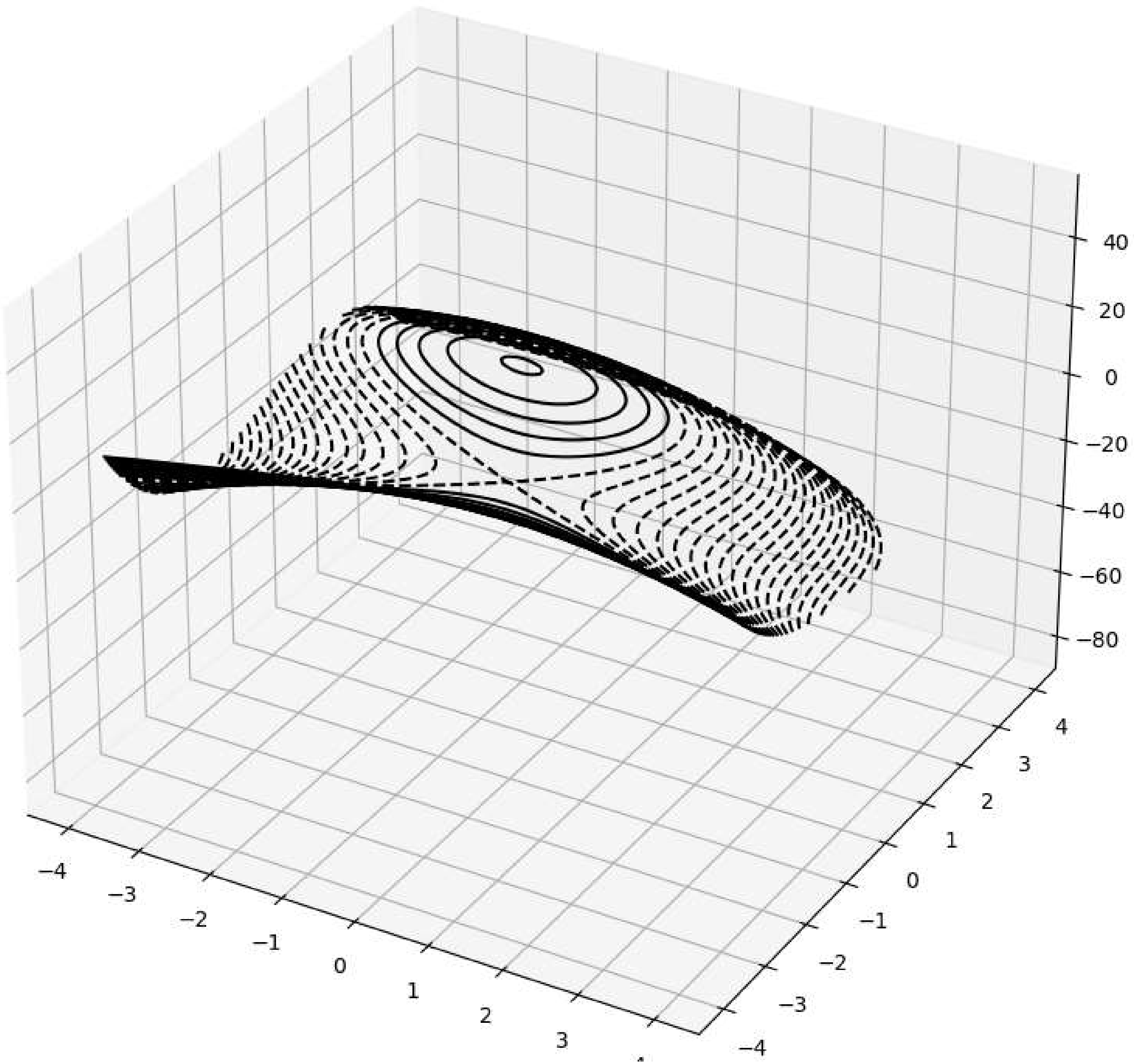

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

1.00

## Oppgave 5a

Nivåkurver av paramertrisert flate $\vec{r}(x, y) = [x, y, 1 - x^2 - y^3 - x + 3y - xy]$ med $(x, y) \in [-4, 4] \times [-4, 4]$.

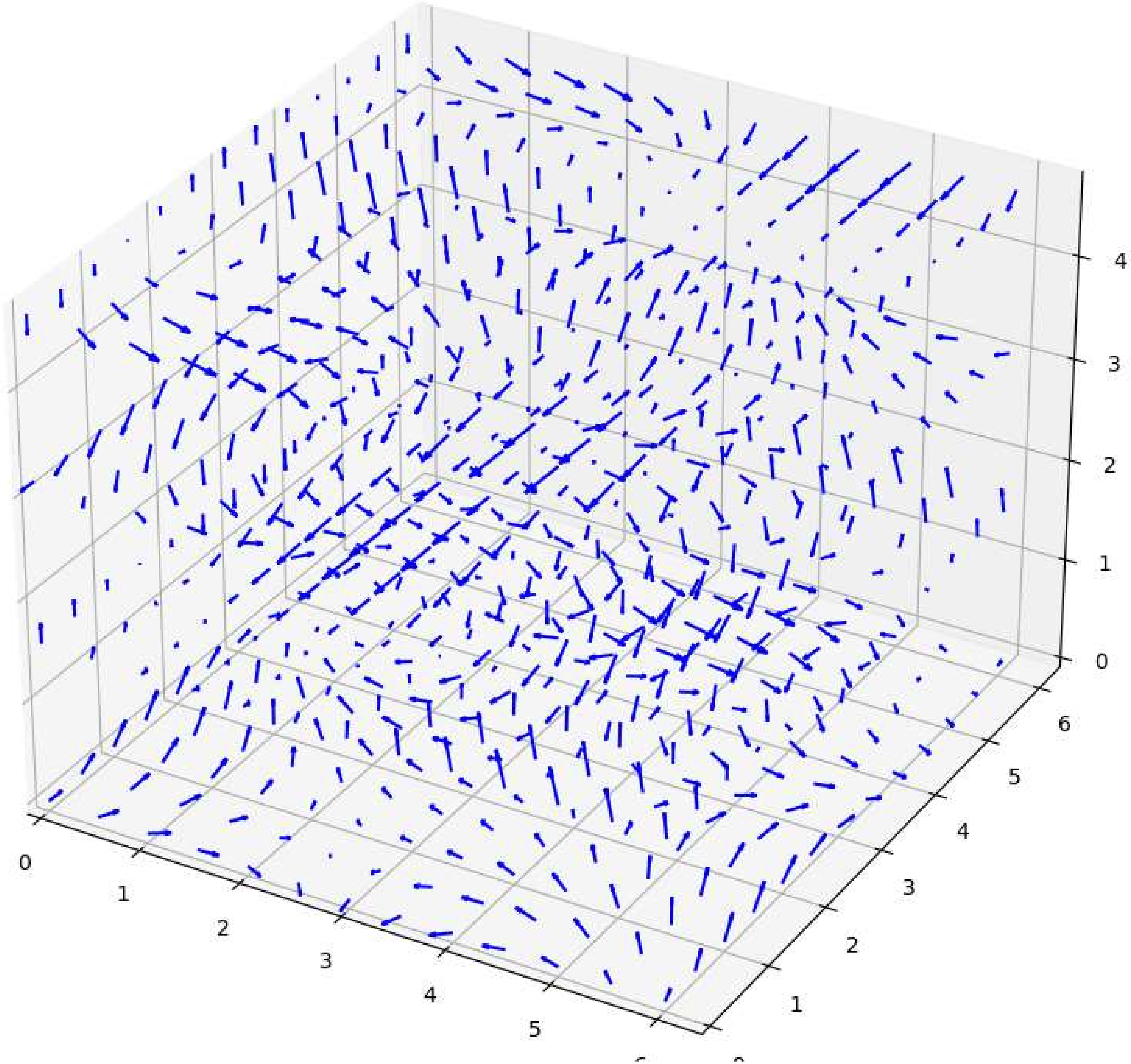
In [5]:
```
x = np.linspace(-4.0, 4.0, 100)
y = np.linspace(-4.0, 4.0, 100)
u, v = np.meshgrid(x, y)
w = 1 - u**2 - v**3 - u + 3*v - u*v

fig = plt.figure(figsize =(10, 10))
ax = fig.add_subplot(projection='3d')
levels = np.arange(-20, 10, 1)
cont = ax.contour(u, v, w, levels, colors = 'black')
plt.show()
```


40
20
0
−20
−40
−60
−80
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

## Oppgave 7c

Plotting av vektorfelt $\vec{F}(x, y, z) = [\sin x \cos y, \cos(x+z), \sin(y+z)]$ med $(x, y, z) \in [0, 2\pi] \times [0, 2\pi] \times [0, 2\pi]$.

In [6]:
```
delta_z = 2*np.pi/4 # step length in z-direction
x = np.arange(0, 2*np.pi, 0.5)
y = np.arange(0, 2*np.pi, 0.5)
z = np.arange(0, 2*np.pi, delta_z)

# 3d point mesh
x, y, z = np.meshgrid(x, y, z)

# the vector field
u = np.sin(x)*np.cos(y) # array with x-coord.
v = np.cos(x+z) # array with y-coord.
w = np.sin(y+z) # array with z-coord.

# set up coordinate system
fig = plt.figure(figsize =(10, 10))
ax = fig.add_subplot(projection='3d')

# plot vector field
ax.quiver(x, y, z, u, v, w, color = 'blue', length = 0.2)

# determine interval for plot
ax.axis([0, 2*np.pi, 0, 2*np.pi])
plt.show()
```


0
1
2
3
4
5
0
1
2
3
4
5
6
0
1
2
3
4

## Oppgave 10c

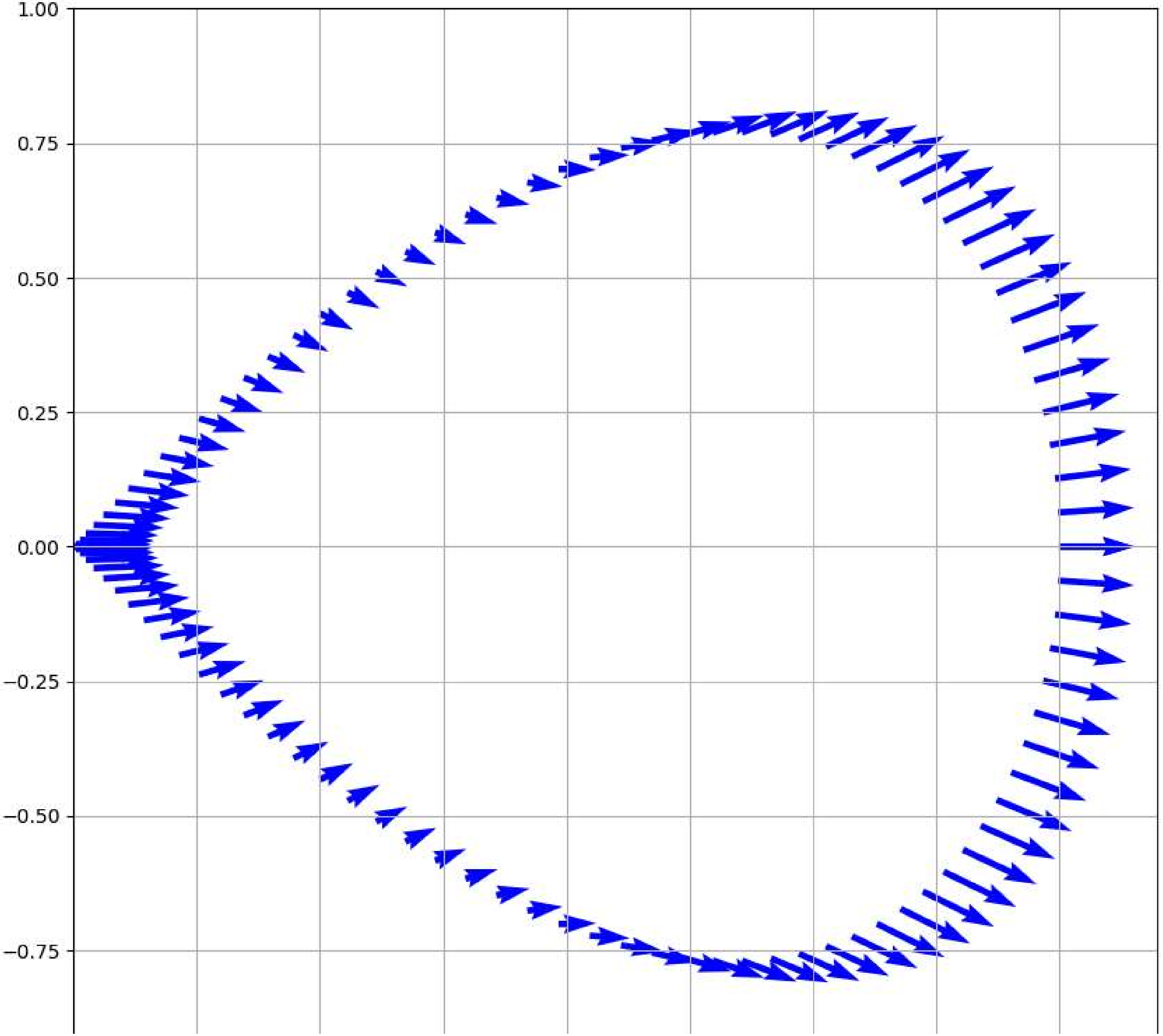
$$\vec{F}(x,y) = [x^2 + y^2, xy]$$

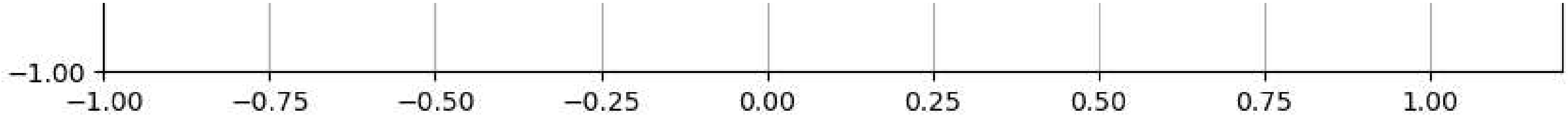
på kurven gitt ved parametriseringen:

$$\vec{r}(t) = [\cos(2t), \cos t \sin(2t)] \text{ med } t \in [0, 2\pi].$$

Plotting av vektorfelt på kurve

In [7]:
```
# curve
t = np.linspace(0, 2*np.pi, 100)
x = np.cos(2*t)
y = np.cos(t)*np.sin(2*t)

# the vector field
u = x**2 + y**2 # array with x-coord.
v = x*y # array with y-coord.

# set up coordinate system
fig = plt.figure(figsize =(10, 10))
ax = fig.add_subplot()

# plot vector field
ax.quiver(x, y, u, v, color = 'blue')

# determine interval for plot
ax.axis([-1, 1.2, -1, 1])
plt.grid()
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75




## Oppgave 11a

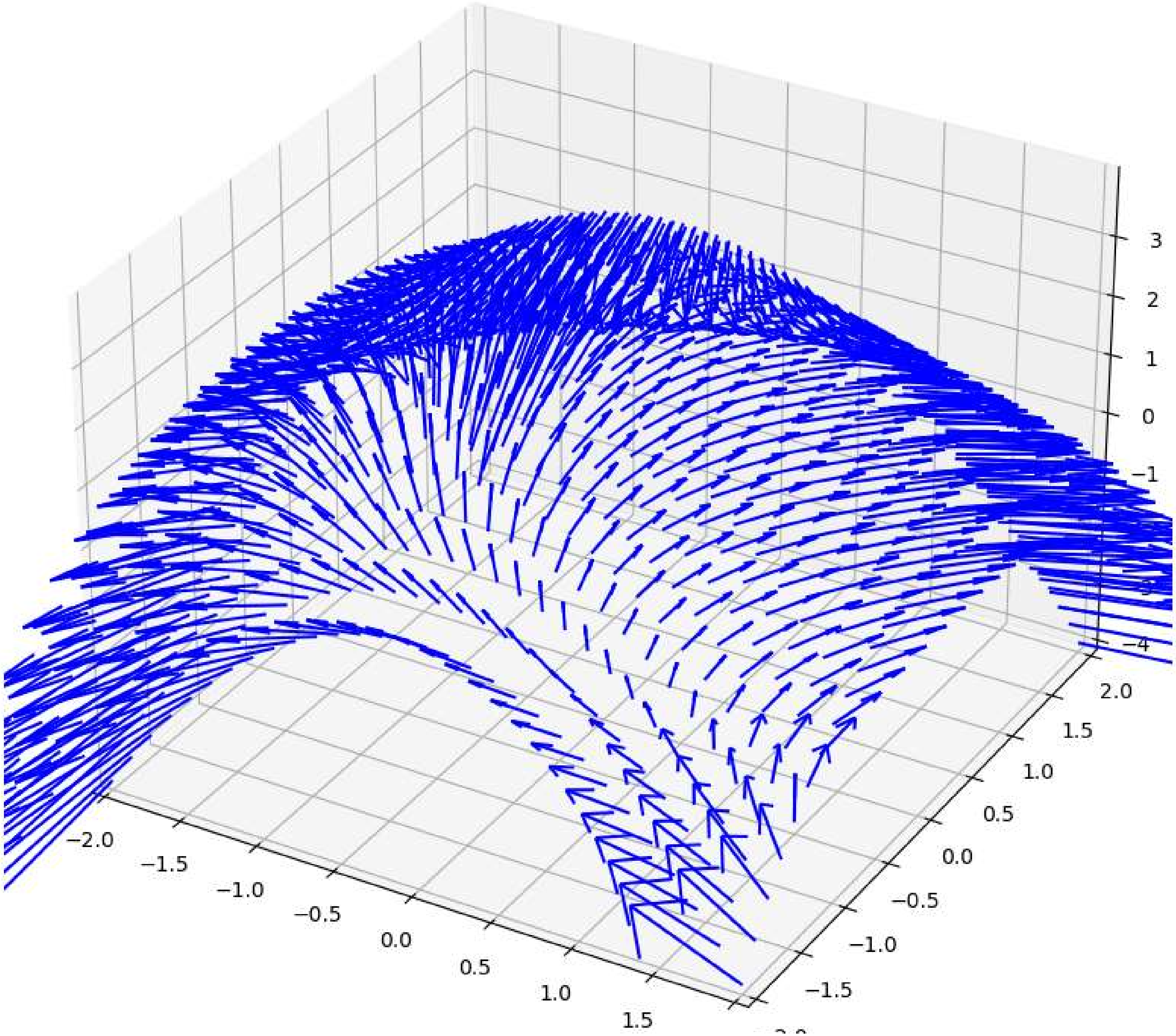
$$\vec{F}(x, y) = [x + y, x - yz, xyz]$$

på flaten gitt ved parametriseringen:

$$\vec{r}(x, y) = [x, y, 4 - x^2 - y^2] \text{ med } (x, y) \in [-2, 2] \times [-2, 2].$$

Plotting av vektorfelt på flate

In [9]:
```
def f(x, y):
    return 4 - x**2 - y**2

# surface
x = np.linspace(-2, 2, 30)
y = np.linspace(-2, 2, 30)
x, y = np.meshgrid(x, y)
X = x
Y = y
Z = f(X, Y)

# the vector field
u = X + Y # array with x-coord.
v = X - Y*Z # array with y-coord.
w = X*Y*Z # array with z-coord.

# set up coordinate system
fig = plt.figure(figsize =(10, 10))
ax = fig.add_subplot(projection='3d')

# plot vector field
ax.quiver(X, Y, Z, u, v, w, color = 'blue', length = 0.2)

# determine interval for plot
ax.axis([-2, 2, -2, 2])
plt.show()
```


3
2
1
0
−1
−4
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

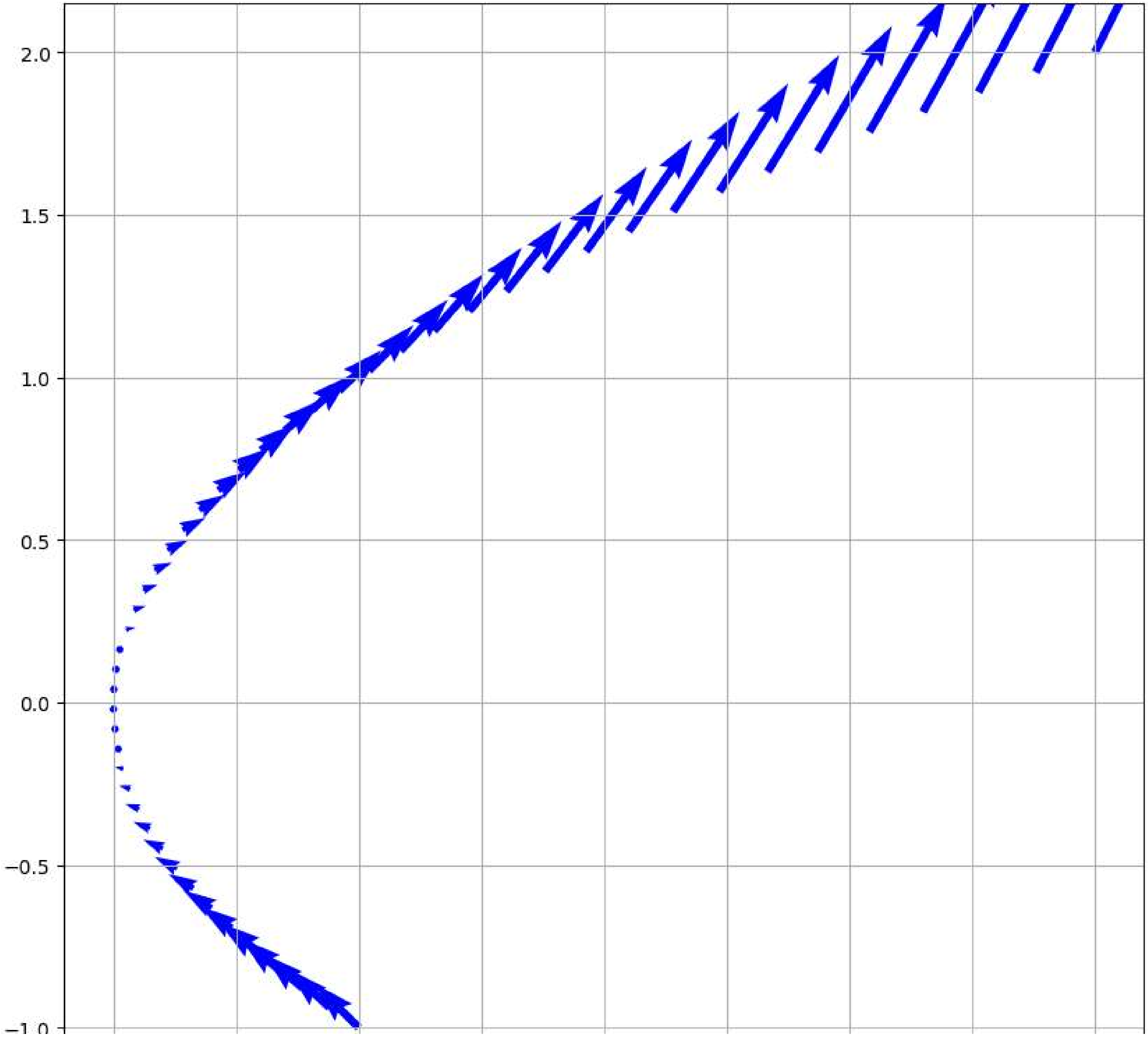

## Oppgave 13a

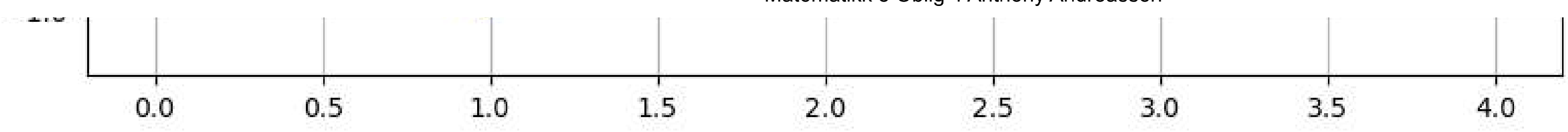
Numerisk integrasjon av arbeidsintegral av parametrisert kurve i vektorfelt

Kurven og vektorer plottet, integral utregnet til slutt

In [10]:
```python
# curve
t = np.linspace(-1, 2, 50)
x = t**2
y = t


# the vector field
u = y # array with x-coord.
v = x # array with y-coord.

# set up coordinate system
fig = plt.figure(figsize =(10, 10))
ax = fig.add_subplot()

# plot vector field
ax.quiver(x, y, u, v, color = 'blue')

# determine interval for plot
#ax.axis([-1, 1.2, -1, 1])
plt.grid()
plt.show()

# the integral
# 3*t**2 = vec_F dot vec_T ds = vec_F dot vec_v dt
# ^utregnet for hånd

def work(t):
    return 3*t**2

def simpson(a, b, n):
    assert n % 2 == 0, 'n is not even'
    epsilon = (b-a)/n
    interval = np.linspace(a, b, n + 1)

    # defining g-vector as array
```

```
    lst = [1]
    for i in range(1, n):
        if i % 2 == 0:
            lst.append(2)
        else:
            lst.append(4)
    lst.append(1)

    gvec = np.array(lst, dtype = int)

    # computing f_0, ..., f_n
    fvec= work(interval)

    # scalar product
    integral = epsilon/3*gvec.dot(fvec.T)

    # error
    error = -epsilon**4/180
    return integral, error

print("Resultat av arbeidsintegral: (integralsum, feilestimat)")
print(simpson(-1,2,1000))
```


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0




```
Resultat av arbeidsintegral: (integralsum, feilestimat)
(9.0, -4.5e-13)
```

## Oppgave 22a

Utregning av masse i gitt område med tetthetsfunksjon

In [11]:
```python
# density function
def func(x, y):
    return np.log(1 + x**2 + y**2*np.sin(x)**4)

# calculate mass by numerical integration
def simpson2d(xx, yy, n):
    a, b = xx
    c, d = yy
    Delta_x = (b-a)/n
    Delta_y = (d-c)/n

    x = np.linspace(a, b, n + 1)
    y = np.linspace(c, d, n + 1)

    x, y = np.meshgrid(x, y)
    F = func(x, y)

    # g-vector
    lst = [1]
    for i in range(1, n):
        if i % 2 == 0:
            lst.append(2)
        else:
            lst.append(4)
    lst.append(1)
    g = np.array(lst, dtype = int)
    Int = (Delta_x*Delta_y/9)*(g.dot(F)).dot(g.T)
    return Int

print(simpson2d([0, 6], [0, 6], 100))
```

```
88.9963003989028
```

## Oppgave 22b

Utregning av masse i gitt område (i sylinderkoordinater) med tetthetsfunksjon

In [12]:
```
# density funciton
def func(r, t):
    return np.log(1 + (r*np.cos(t))**2 + (r*np.sin(t))**2*np.sin(r*np.cos(t))**4)*r

# calculate mass by numerical integration
def simpson2d(xx, yy, n):
    a, b = xx
    c, d = yy
    Delta_x = (b-a)/n
    Delta_y = (d-c)/n

    x = np.linspace(a, b, n + 1)
    y = np.linspace(c, d, n + 1)

    x, y = np.meshgrid(x, y)
    F = func(x, y)

    # g-vector
    lst = [1]
    for i in range(1, n):
        if i % 2 == 0:
            lst.append(2)
        else:
            lst.append(4)
    lst.append(1)
    g = np.array(lst, dtype = int)
    Int = (Delta_x*Delta_y/9)*(g.dot(F)).dot(g.T)
    return Int

print(simpson2d([7*np.pi/19, 9*np.pi/11], [np.exp(-3), np.exp(-1/5)], 100))
```

```
3.0750009653692967
```

In [ ]: