In [85]:
```
import matplotlib as mpl
mpl.rcParams["font.size"] = 14
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import scipy.spatial.distance as dist
import pdb
```

In [58]:
```
N1 = 5 #leader fishy
N2 = 6 # follower fishy
N3 = 10 # undecided fishy
N = N1 + N2 + N3
Nt = 5000
dt = 0.01
tlist = np.arange(Nt) * dt
prefac1 = dt/N1
prefac2 = dt/N2
prefac3 = dt/N3

save_every = 50
S = int(Nt/save_every)

Rx = 2.
rx = 2.
rw = 2.

# Morse potential
CR = 1.5
LR = 0.25
CA = 1.
LA = 0.75

alpha = 1.
beta = 0.5

# direction to the two different targets
v_r = np.array([1,0])
v_b = np.array([-1,0])
tau_r = 0.9
tau_b = 0.9
muL = 0.001 * dt
muF = 0.001 * dt

# initialise everything
z1 = np.zeros(shape=(S+1, 5, N1))
z2 = np.zeros(shape=(S+1, 5, N2))
z = np.zeros(shape=(S+1, 5, N3))
```

In [59]:
```
def euclidian_distance(y1, y2):
    diffxixj = y1[:,np.newaxis]-y2
    return [diffxixj, np.sqrt(np.sum(diffxixj**2, axis=2))]
def opinion_distance(w1, w2):
    return (w1[:, np.newaxis] - w2)
```

In [72]:
```
# leaders
preference1 = np.ones(N1)
position1 = np.random.uniform(-1,1,size=(2,N1))
velocity1 = np.random.uniform(-1,1,size=(2,N1))

# followers
preference2 = -1. * np.ones(N2)
position2 = np.random.uniform(-1,1,size=(2,N2))
velocity2 = np.random.uniform(-1,1,size=(2,N2))

# undecided
preference = np.zeros(N3)
position = np.random.uniform(-1,1,size=(2,N3))
velocity = np.random.uniform(-1,1,size=(2,N3))

mean_vel = np.zeros(shape = (Nt+1,2))
dev_vel = np.zeros(Nt+1)
mean_pref = np.zeros(Nt+1)
dev_pref = np.zeros(Nt+1)

z[0, 0:2,:] = position.copy()
z[0, 2:4,:] = velocity.copy()
z[0, 4,:] = preference.copy()

z1[0, 0:2,:] = position1.copy()
z1[0, 2:4,:] = velocity1.copy()
z1[0, 4,:] = preference1.copy()

z2[0, 0:2,:] = position2.copy()
z2[0, 2:4,:] = velocity2.copy()
z2[0, 4,:] = preference2.copy()

mean_vel[0,:] = np.sum(velocity, axis=1)/N3 + np.sum(velocity1, axi
s=1)/N1 + np.sum(velocity2, axis=1)/N2
dev_vel[0] = (np.sum((velocity[0,:] - mean_vel[0,0])**2 + (velocity
[1,:]-mean_vel[0,1])**2) +  np.sum((velocity1[0,:] - mean_vel[0,0])
**2 + (velocity1[1,:]-mean_vel[0,1])**2) +  np.sum((velocity2[0,:]
- mean_vel[0,0])**2 + (velocity2[1,:]-mean_vel[0,1])**2))/N
mean_pref[0] = (np.sum(preference) + np.sum(preference1) + np.sum(p
reference2))/N
dev_pref[0] = (np.sum((preference - mean_pref[0])**2) + np.sum((pre
ference1 - mean_pref[0])**2) + np.sum((preference2 - mean_pref[0])*
*2))/N
count = 1

for i in np.arange(1,Nt):

    vel = velocity[0,:]**2 + velocity[1,:]**2
    vel1 = velocity1[0,:]**2 + velocity1[1,:]**2
    vel2 = velocity2[0,:]**2 + velocity2[1,:]**2

    # difference in preference
    prefdiffUU = opinion_distance(preference, preference)  # undeci
ded - undecided
    prefdiffUL = opinion_distance(preference, preference1) # undeci
ded - leader
    prefdiffUF = opinion_distance(preference, preference2) # undeci
ded - followers
    prefdiffLL = opinion_distance(preference1, preference1) # leade
r - leader
```

```
    prefdiffFF = opinion_distance(preference2, preference2) # follo
wer - follower
    prefdiffLF = opinion_distance(preference1, preference2) # leade
r - follower

    # difference among all fish
    [diffUU,distUU] = euclidian_distance(position.T,position.T) # u
ndecided - undecided
    [diffUL,distUL] = euclidian_distance(position.T,position1.T) #
undecided - leader
    [diffUF,distUF] = euclidian_distance(position.T,position2.T) #
undecided - follower
    [diffLL,distLL] = euclidian_distance(position1.T,position1.T) #
leader - leader
    [diffFF,distFF] = euclidian_distance(position2.T,position2.T) #
follower - follower
    [diffLF,distLF] = euclidian_distance(position1.T,position2.T) #
leader - follower

    # interact if close enough in position and preference
    interactionUU = 1.0 * (np.abs(prefdiffUU) <= rw)* (distUU <= r
x)
    interactionUL = 1.0 * (np.abs(prefdiffUL) <= rw)* (distUL <= r
x)
    interactionUF = 1.0 * (np.abs(prefdiffUF) <= rw)* (distUF <= r
x)
    interactionLL = 1.0 * (np.abs(prefdiffLL) <= rw)* (distLL <= r
x)
    interactionFF = 1.0 * (np.abs(prefdiffFF) <= rw)* (distFF <= r
x)
    interactionLF = 1.0 * (np.abs(prefdiffLF) <= rw)* (distLF <= r
x)

    go_to_r_U = 1.0 * (preference<=0.0)
    go_to_b_U = 1.0 * (preference>0.0)
    go_to_r_L = 1.0 * (preference1<=0.0)
    go_to_b_L = 1.0 * (preference1>0.0)
    go_to_r_F = 1.0 * (preference2<=0.0)
    go_to_b_F = 1.0 * (preference2>0.0)

    # precalculations for the morse potentials - R for repulsion, A
for attraction
    prefacRUU = np.exp(-distUU/LR)/distUU
    prefacAUU = np.exp(-distUU/LA)/distUU
    prefacRUU[prefacRUU == np.inf] = 0
    prefacAUU[prefacAUU == np.inf] = 0
    prefacRUL = np.exp(-distUL/LR)/distUL
    prefacAUL = np.exp(-distUL/LA)/distUL
    prefacRUL[prefacRUL == np.inf] = 0
    prefacAUL[prefacAUL == np.inf] = 0
    prefacRUF = np.exp(-distUF/LR)/distUF
    prefacAUF = np.exp(-distUF/LA)/distUF
    prefacRUF[prefacRUF == np.inf] = 0
    prefacAUF[prefacAUF == np.inf] = 0
    prefacRLL = np.exp(-distLL/LR)/distLL
    prefacALL = np.exp(-distLL/LA)/distLL
    prefacRLL[prefacRLL == np.inf] = 0
    prefacALL[prefacALL == np.inf] = 0
    prefacRFF = np.exp(-distFF/LR)/distFF
    prefacAFF = np.exp(-distFF/LA)/distFF
    prefacRFF[prefacRFF == np.inf] = 0
```

```
    prefacAFF[prefacAFF == np.inf] = 0
    prefacRLF = np.exp(-distLF/LR)/distLF
    prefacALF = np.exp(-distLF/LA)/distLF
    prefacRLF[prefacRLF == np.inf] = 0
    prefacALF[prefacALF == np.inf] = 0

    morseUU = CR * prefacRUU[:,:,np.newaxis] * diffUU - CA * prefac
AUU[:,:,np.newaxis] * diffUU
    morseUL = CR * prefacRUL[:,:,np.newaxis] * diffUL - CA * prefac
AUL[:,:, np.newaxis] * diffUL
    morseUF = CR * prefacRUF[:,:,np.newaxis] * diffUF - CA * prefac
AUF[:,:, np.newaxis] * diffUF
    morseLL = CR * prefacRLL[:,:,np.newaxis] * diffLL - CA * prefac
ALL[:,:, np.newaxis] * diffLL
    morseFF = CR * prefacRFF[:,:,np.newaxis] * diffFF - CA * prefac
AFF[:,:, np.newaxis] * diffFF
    morseLF = CR * prefacRLF[:,:,np.newaxis] * diffLF - CA * prefac
ALF[:,:, np.newaxis] * diffLF

    # Update U
    velocity = velocity + prefac3 * np.sum(morseUU, axis=1).T + pre
fac1 *  np.sum(morseUL, axis=1).T + prefac2 * np.sum(morseUF, axis=
1).T  + dt * (alpha - beta * vel) * velocity - dt * go_to_r_U/tau_r
* (velocity-v_r[:,np.newaxis]) - dt * go_to_b_U/tau_b * (velocity-v
_b[:,np.newaxis])
    position = position + prefac3 * velocity
    preference = preference - prefac3 * np.sum(interactionUU * pref
diffUU, axis=1) - prefac2 * np.sum(interactionUL * prefdiffUL, axis
=1) - prefac3 * np.sum(interactionUF * prefdiffUF, axis=1)

    # update L
    velocity1 = velocity1 - prefac3 * np.sum(morseUL, axis=0).T + p
refac2 * np.sum(morseLF, axis=1).T + prefac1 * np.sum(morseLL, axis
=0).T  + dt * (alpha - beta * vel1) * velocity1 - dt * go_to_r_L/ta
u_r * (velocity1-v_r[:,np.newaxis]) - dt * go_to_b_L/tau_b * (veloc
ity1-v_b[:,np.newaxis])
    position1 = position1 + prefac1 * velocity1
    preference1 = preference1 + prefac3 * np.sum(interactionUL * pr
efdiffUL, axis=0) - prefac1 * np.sum(interactionLL * prefdiffLL, ax
is=1) - prefac2 * np.sum(interactionLF * prefdiffLF, axis=1) - muL
* (preference1 - 1)

    # Update F
    velocity2 = velocity2 - prefac3 * np.sum(morseUF, axis=0).T - p
refac1 * np.sum(morseLF, axis=0).T + prefac2 * np.sum(morseFF, axis
=0).T + dt * (alpha - beta * vel2) * velocity2 - dt * go_to_r_F/tau
_r * (velocity2-v_r[:,np.newaxis]) - dt * go_to_b_F/tau_b * (veloci
ty2-v_b[:,np.newaxis])
    position2 = position2 + prefac2 * velocity2
    preference2 = preference2 + prefac1 * np.sum(interactionLF * pr
efdiffLF, axis=0) - prefac2 * np.sum(interactionFF * prefdiffFF, ax
is=1) + prefac3 * np.sum(interactionUF * prefdiffUF, axis=0) - muF
* (preference2 + 1)

    mean_vel[i,:] = (np.sum(velocity, axis=1) + np.sum(velocity2, a
xis=1) + np.sum(velocity1, axis=1))/N
    dev_vel[i] = np.sum((velocity[0,:] - mean_vel[i,0])**2 + (veloc
ity[1,:]-mean_vel[i,1])**2) + np.sum((velocity1[0,:] - mean_vel[i,
0])**2 + (velocity1[1,:]-mean_vel[i,1])**2) + np.sum((velocity2
[0,:] - mean_vel[i,0])**2 + (velocity2[1,:]-mean_vel[i,1])**2)
    mean_pref[i] = (np.sum(preference) + np.sum(preference1) + np.s
```

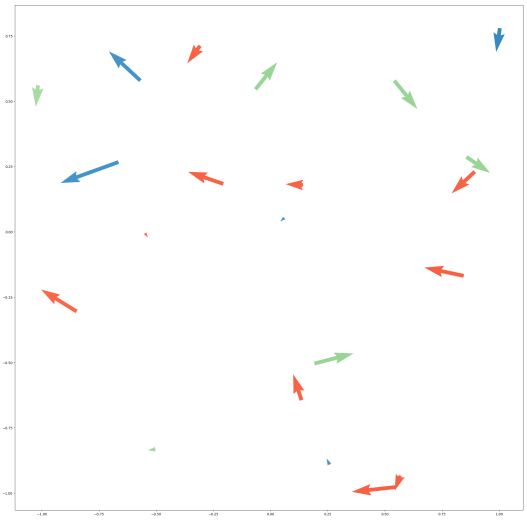
```
um(preference2))/N
    dev_pref[i] = np.sum((preference - mean_pref[i])**2)/N3 + np.su
m((preference1 - mean_pref[i])**2)/N1 + np.sum((preference2 - mean_
pref[i])**2)/N2

    if i%save_every == 0:
        z[count, 0:2,:] = position.copy()
        z[count, 2:4,:] = velocity.copy()
        z[count, 4,:] = preference.copy()

        z1[count, 0:2,:] = position1.copy()
        z1[count, 2:4,:] = velocity1.copy()
        z1[count, 4,:] = preference1.copy()

        z2[count, 0:2,:] = position2.copy()
        z2[count, 2:4,:] = velocity2.copy()
        z2[count, 4,:] = preference2.copy()
        count = count+1
```

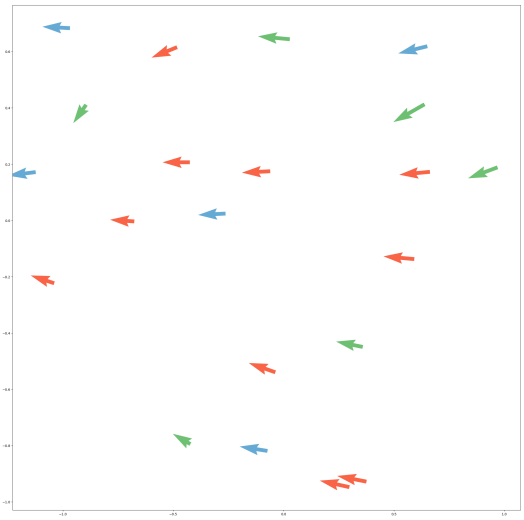
```
/tmp/ipykernel_133287/1657679613.py:77: RuntimeWarning: divide by z
ero encountered in divide
  prefacRUU = np.exp(-distUU/LR)/distUU
/tmp/ipykernel_133287/1657679613.py:78: RuntimeWarning: divide by z
ero encountered in divide
  prefacAUU = np.exp(-distUU/LA)/distUU
/tmp/ipykernel_133287/1657679613.py:89: RuntimeWarning: divide by z
ero encountered in divide
  prefacRLL = np.exp(-distLL/LR)/distLL
/tmp/ipykernel_133287/1657679613.py:90: RuntimeWarning: divide by z
ero encountered in divide
  prefacALL = np.exp(-distLL/LA)/distLL
/tmp/ipykernel_133287/1657679613.py:93: RuntimeWarning: divide by z
ero encountered in divide
  prefacRFF = np.exp(-distFF/LR)/distFF
/tmp/ipykernel_133287/1657679613.py:94: RuntimeWarning: divide by z
ero encountered in divide
  prefacAFF = np.exp(-distFF/LA)/distFF
```

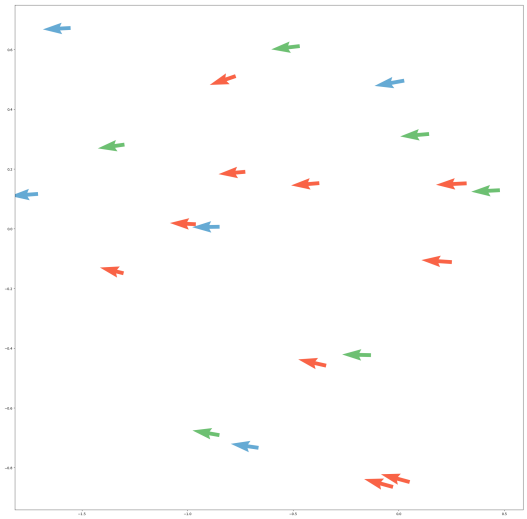
In [73]:
```
z[:,4,:]
```

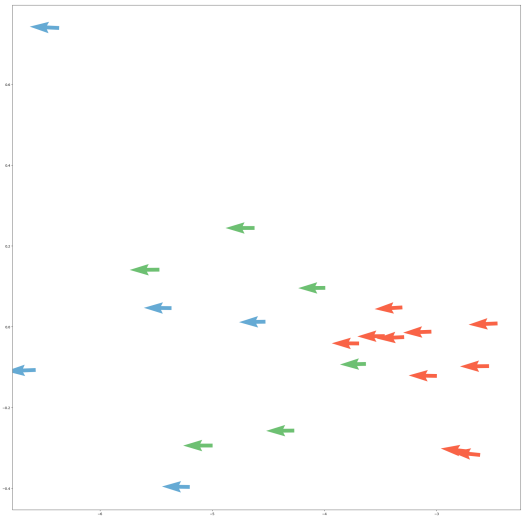
Out[73]:
```
array([[0.        , 0.        , 0.        , ..., 0.        , 0.
,
        0.        ],
       [0.04375987, 0.04375987, 0.05986024, ..., 0.04375987, 0.0188
466 ,
        0.04375987],
       [0.04083338, 0.04083338, 0.05116947, ..., 0.04083338, 0.0257
7293,
        0.04083338],
       ...,
       [0.03156704, 0.0315671 , 0.03162596, ..., 0.03161672, 0.0315
6532,
        0.03160657],
       [0.03155057, 0.03155061, 0.0316127 , ..., 0.03160712, 0.0315
4958,
        0.03160098],
       [0.        , 0.        , 0.        , ..., 0.        , 0.
,
        0.        ]])
```

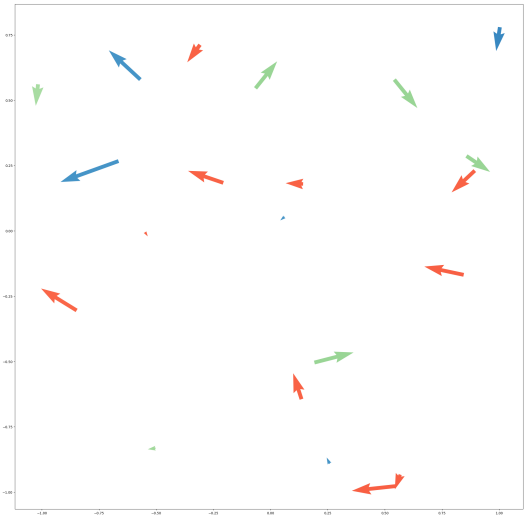
In [91]:
```
NORMZ = mpl.colors.Normalize(vmin=-1., vmax=1.)

num1 = 1
num2 = 5
num3 = 10
num4 = 50

fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(60,60))
axs[0,0].quiver(z[num1,0,:], z[num1,1,:], z[num1,2,:], z[num1,3,:],
z[num1,4,:], cmap='Reds', norm=NORMZ)
axs[0,0].quiver(z1[num1,0,:], z1[num1,1,:], z1[num1,2,:], z1[num1,
3,:], z1[num1,4,:], cmap='Blues', norm=NORMZ)
axs[0,0].quiver(z2[num1,0,:], z2[num1,1,:], z2[num1,2,:], z2[num1,
3,:], z2[num1,4,:], cmap='Greens', norm=NORMZ)
axs[0,1].quiver(z[num2,0,:], z[num2,1,:], z[num2,2,:], z[num2,3,:],
z[num2,4,:], cmap = 'Reds', norm=NORMZ)
axs[0,1].quiver(z1[num2,0,:], z1[num2,1,:], z1[num2,2,:], z1[num2,
3,:], z1[num2,4,:], cmap='Blues', norm=NORMZ)
axs[0,1].quiver(z2[num2,0,:], z2[num2,1,:], z2[num2,2,:], z2[num2,
3,:], z2[num2,4,:], cmap='Greens', norm=NORMZ)
axs[1,0].quiver(z[num3,0,:], z[num3,1,:], z[num3,2,:], z[num3,3,:],
z[num3,4,:], cmap='Reds', norm=NORMZ)
axs[1,0].quiver(z1[num3,0,:], z1[num3,1,:], z1[num3,2,:], z1[num3,
3,:], z1[num3,4,:], cmap='Blues', norm=NORMZ)
axs[1,0].quiver(z2[num3,0,:], z2[num3,1,:], z2[num3,2,:], z2[num3,
3,:], z2[num3,4,:], cmap='Greens', norm=NORMZ)
axs[1,1].quiver(z[num4,0,:], z[num4,1,:], z[num4,2,:], z[num4,3,:],
z[num4,4,:], cmap='Reds', norm=NORMZ)
axs[1,1].quiver(z1[num4,0,:], z1[num4,1,:], z1[num4,2,:], z1[num4,
3,:], z1[num4,4,:], cmap='Blues', norm=NORMZ)
axs[1,1].quiver(z2[num4,0,:], z2[num4,1,:], z2[num4,2,:], z2[num4,
3,:], z2[num4,4,:], cmap='Greens', norm=NORMZ)
plt.savefig('fishy_second_attempt.png')
plt.show()
```

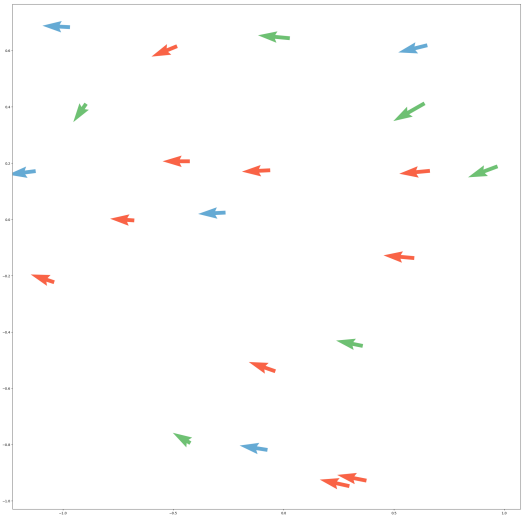


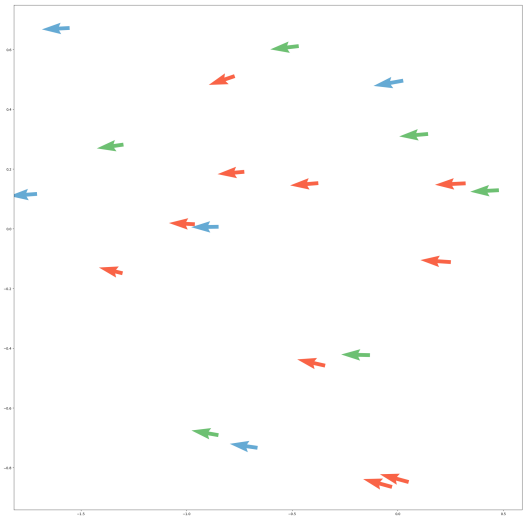


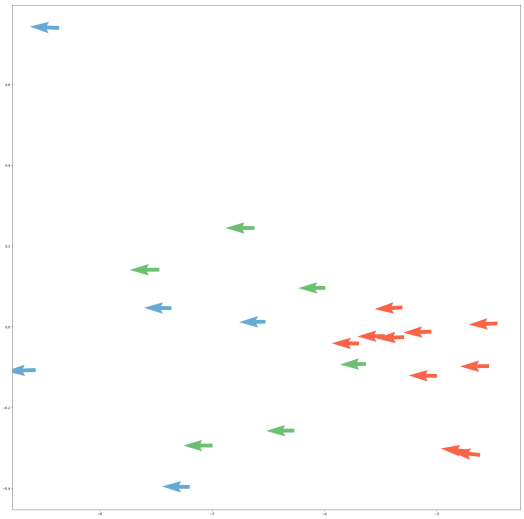




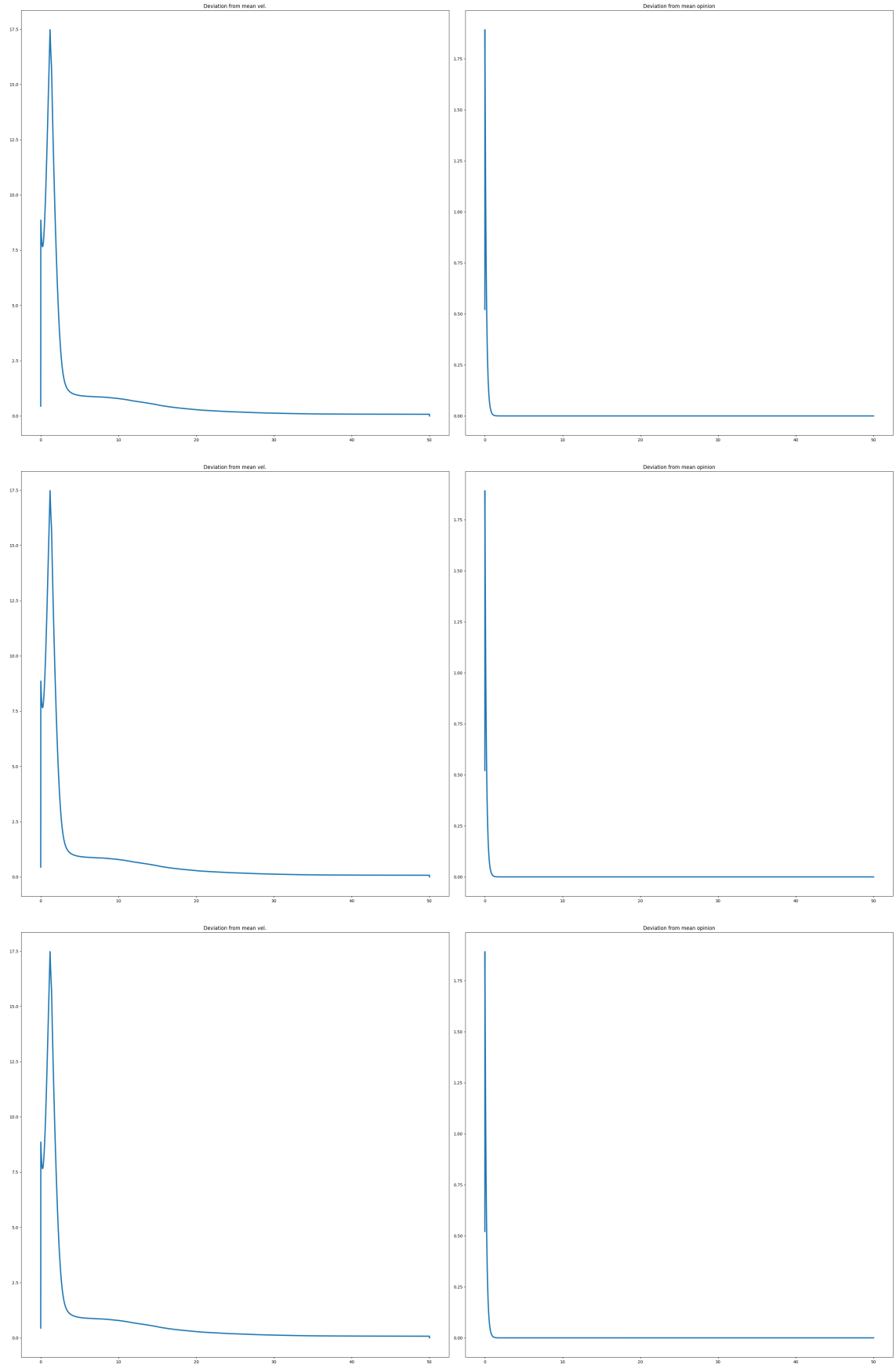








In [89]:
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(30,15))
axs[0].plot(np.arange(Nt+1)*dt, dev_vel, lw=3)
axs[0].set_title('Deviation from mean vel.')
axs[1].plot(np.arange(Nt+1) * dt, dev_pref, lw=3)
axs[1].set_title('Deviation from mean opinion')
plt.tight_layout()
plt.savefig('evolution_deviation.png')
plt.show()
```



In [69]:
```
print('Preference', preference)
print('Velocity', velocity)
```

```
Preference [0.83849106 0.83849106 0.83849106 0.83849106 0.83849106
0.83849106
 0.83849106 0.83849106 0.83849106 0.83849106 0.83849106 0.83849106
 0.83849106 0.83849106 0.83849106]
Velocity [[-1.24796784e+00 -1.24442214e+00 -1.24739635e+00 -1.24688
364e+00
  -1.25344826e+00 -1.24962735e+00 -1.24503980e+00 -1.24510278e+00
  -1.24598896e+00 -1.24862527e+00 -1.25051150e+00 -1.24461665e+00
  -1.24669337e+00 -1.24776067e+00 -1.25201461e+00]
 [-8.85353941e-04 -2.14608873e-03  2.00139419e-03  1.01159727e-03
   8.20705727e-03 -1.92278379e-03 -1.24801974e-03 -7.49498485e-03
   6.81320612e-04  1.19773561e-02 -4.91669785e-03  1.12734086e-03
  -2.09373334e-03 -1.01205803e-02  5.41607334e-03]]
```

In [68]:
```
print('Preference', preference1)
print('Velocity' , velocity1)
```

```
Preference [1. 1. 1. 1. 1.]
Velocity [[-1.24770961e+00 -1.24804823e+00 -1.24775024e+00 -1.24807
908e+00
  -1.24804810e+00]
 [ 8.96167772e-05 -6.00367175e-05  1.59843644e-05  2.57898243e-05
  -7.76657377e-05]]
```

In [70]:
```
print('Preference', preference2)
print('Velocity', velocity2)
```

```
Preference [-1. -1. -1. -1. -1. -1.]
Velocity [[ 1.24827801e+00  1.24833059e+00  1.24830534e+00  1.24852
154e+00
   1.24835011e+00  1.24861310e+00]
 [-1.96089858e-05 -4.80345020e-05  1.32269230e-05 -8.55550602e-05
  -1.15651344e-05 -5.92693661e-06]]
```

In [ ]: