2023-05-15_ChartTypes_011_VectorFields_Quiver

May 25, 2023

[1]:
```
import numpy as np
import scipy
import imageio

import matplotlib
import matplotlib.pyplot as plt
import matplotlib.cm as cm

matplotlib.rc('image', interpolation='nearest')
matplotlib.rc('figure',facecolor='white')
matplotlib.rc('image',cmap='viridis')
colors=plt.rcParams['axes.prop_cycle'].by_key()['color']
%matplotlib inline
```

1 Vector fields

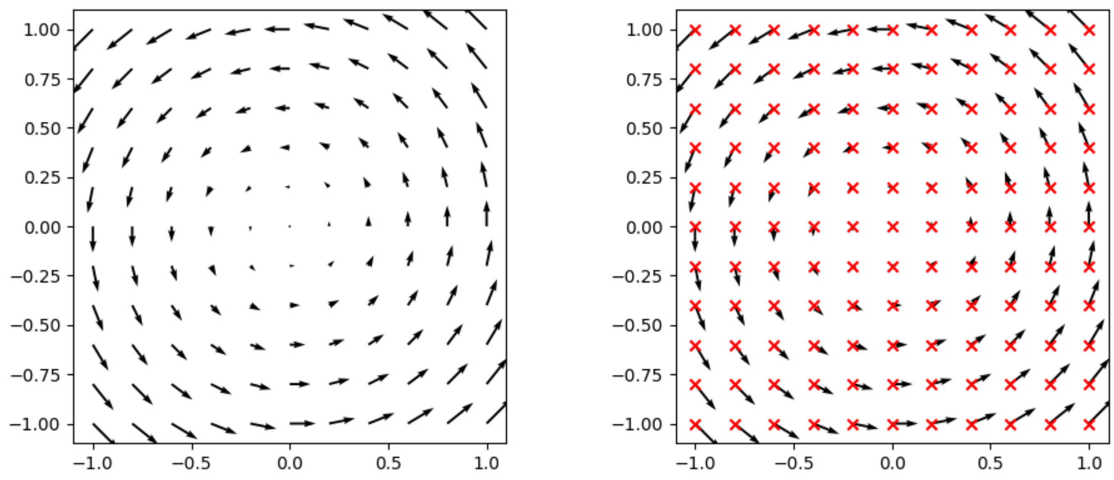
- vector field is function from $\mathbb{R}^m$ to $\mathbb{R}^n$, here focus on $m = n = 2$
- can write it as pair of functions $u(x, y), v(x, y)$ where x, y are horizontal and vertical coordinate in $\mathbb{R}^2$, u and v are horizontal and vertical coordinates of vector field
- think of a small arrow attached to each point in space
- applications:
- velocity fields of air, fluid
- force fields (e.g. from electric charges)
- gradients of functions during optimization

1.1 Quiver plots

[2]:
```
# generate a simple vector field
nPts1d=11
x = np.linspace(-1,1,num=nPts1d)
y = x
nPts=nPts1d**2
X, Y = np.meshgrid(x, y)
u = -Y
v = X
```

1.1.1 Basic example

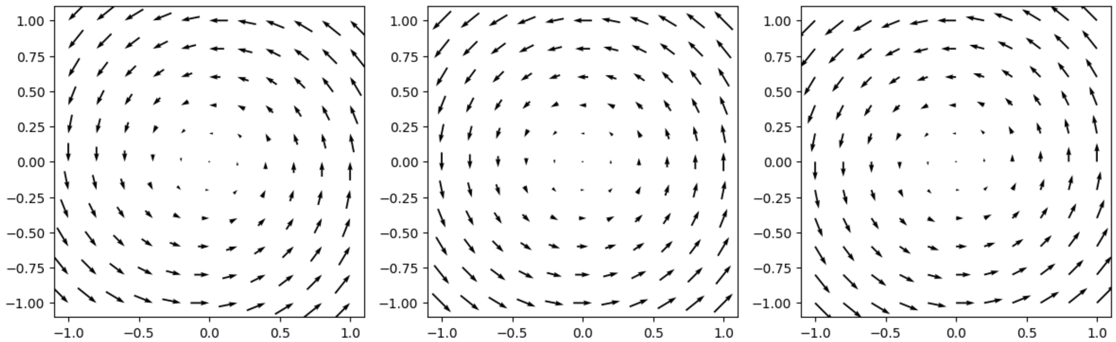
[3]:
```
fig=plt.figure(figsize=(10,4))
# for quiver plot: accurate aspect ratio is important! (we return to this in a␣
↪moment)
ax=fig.add_subplot(1,2,1,aspect=1.)
ax.quiver(X,Y,u,v)

# show data points for comparison
ax=fig.add_subplot(1,2,2,aspect=1.)
ax.quiver(X,Y,u,v)
ax.scatter(X.ravel(),Y.ravel(),marker="x",c="r")

plt.tight_layout()
plt.show()
```



1.1.2 Pivot of arrows

[4]:
```
# how do we attach arrows to data points? tail, mid, tip?

fig=plt.figure(figsize=(12,4))
ax=fig.add_subplot(1,3,1,aspect=1.)
ax.quiver(X,Y,u,v,pivot="tip")

ax=fig.add_subplot(1,3,2,aspect=1.)
ax.quiver(X,Y,u,v,pivot="mid")
#ax.scatter(X.ravel(),Y.ravel(),marker="x",c="r")

ax=fig.add_subplot(1,3,3,aspect=1.)
```

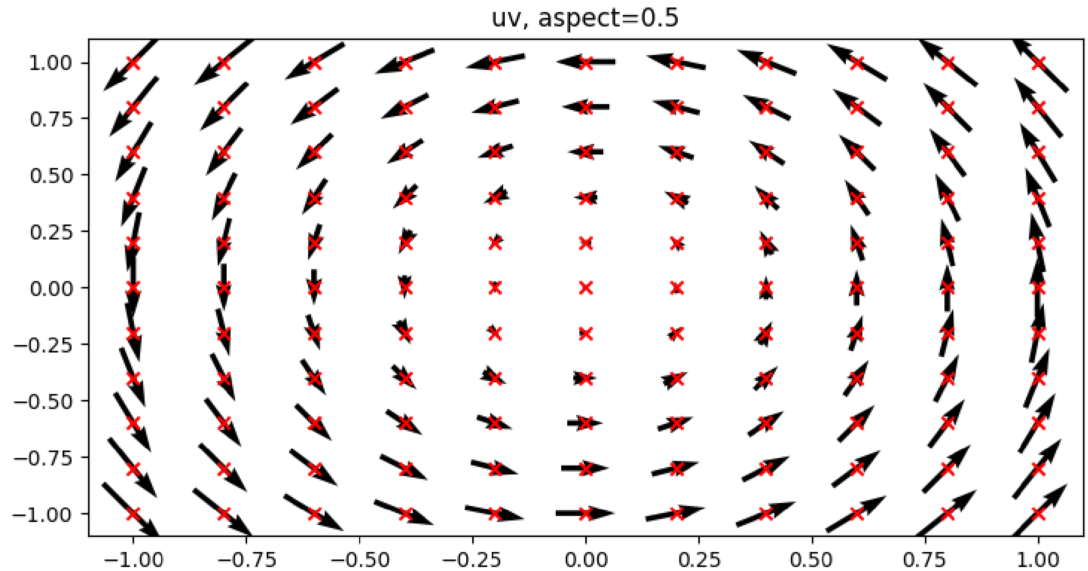
```
ax.quiver(X,Y,u,v,pivot="tail")


plt.tight_layout()
plt.show()
```

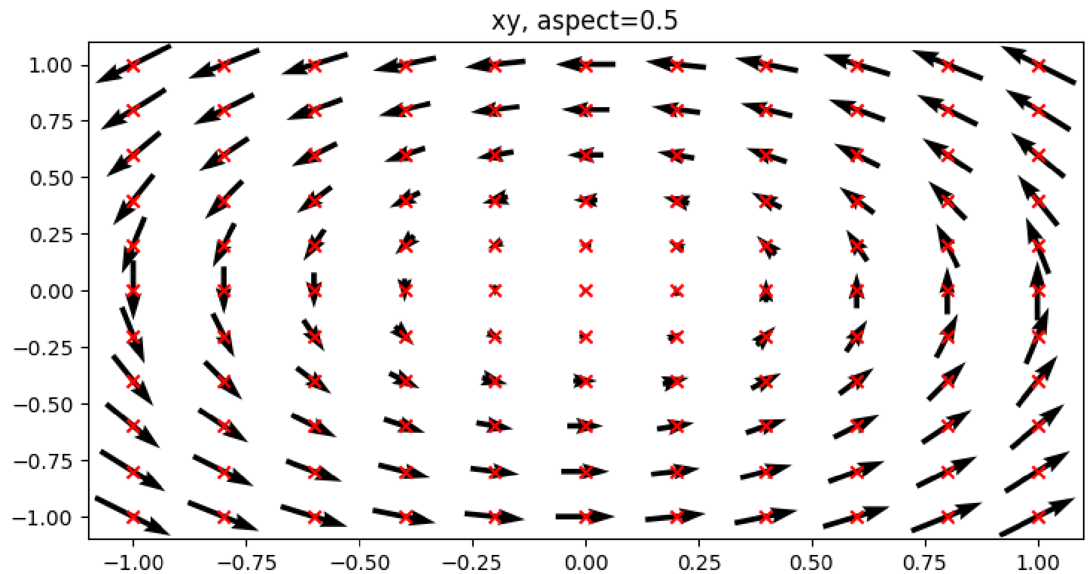


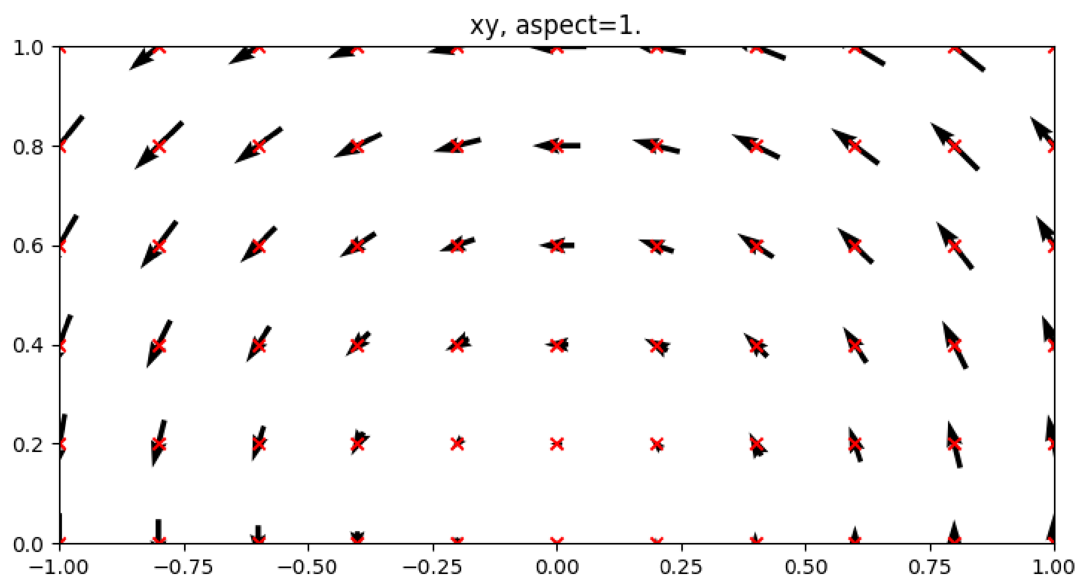
1.1.3 Aspect ratio and angles

[6]:
```
# should arrows be added to data points in "same axis scalings"
#   or in an "independent axis scaling"?
# for aspect ratios != 1, this can have different effects
# my personal opinion: avoid aspect ratio != 1 for quiver plots, since it will
# almost certainly collide with intuitive perception

# angles, orientation

fig=plt.figure(figsize=(8,12))

# uv: arrow orientation is based on aspect ratio 1
# independent of plot coordinate system, i.e. u=v=1 will result in 45° line
# this can be confusing when u,v "live in same coordinate system" as x,y
# e.g. as in a gradient field

ax=fig.add_subplot(3,1,1,aspect=.5)
plt.title("uv, aspect=0.5")
ax.quiver(X,Y,u,v,pivot="mid",angles="uv")
# show data points for comparison
ax.scatter(X.ravel(),Y.ravel(),marker="x",c="r")

# xy: arrow orientations are consistent with coordinate axis
# i.e. at each point arrow will point from (x,y) in direction (x+u,y+v)
# so arrow orientation is changed with axis aspect ratio
# can still be confusing if aspect ratio != 1
```

```
ax=fig.add_subplot(3,1,2,aspect=.5)
plt.title("xy, aspect=0.5")
ax.quiver(X,Y,u,v,pivot="mid",angles="xy")
# show data points for comparison
ax.scatter(X.ravel(),Y.ravel(),marker="x",c="r")

ax=fig.add_subplot(3,1,3,aspect=1.)
plt.title("xy, aspect=1.")
ax.quiver(X,Y,u,v,pivot="mid",angles="xy")
# show data points for comparison
ax.scatter(X.ravel(),Y.ravel(),marker="x",c="r")
ax.set_ylim([0.,1.])
ax.set_xlim([-1.,1.])


plt.tight_layout()
plt.show()
```

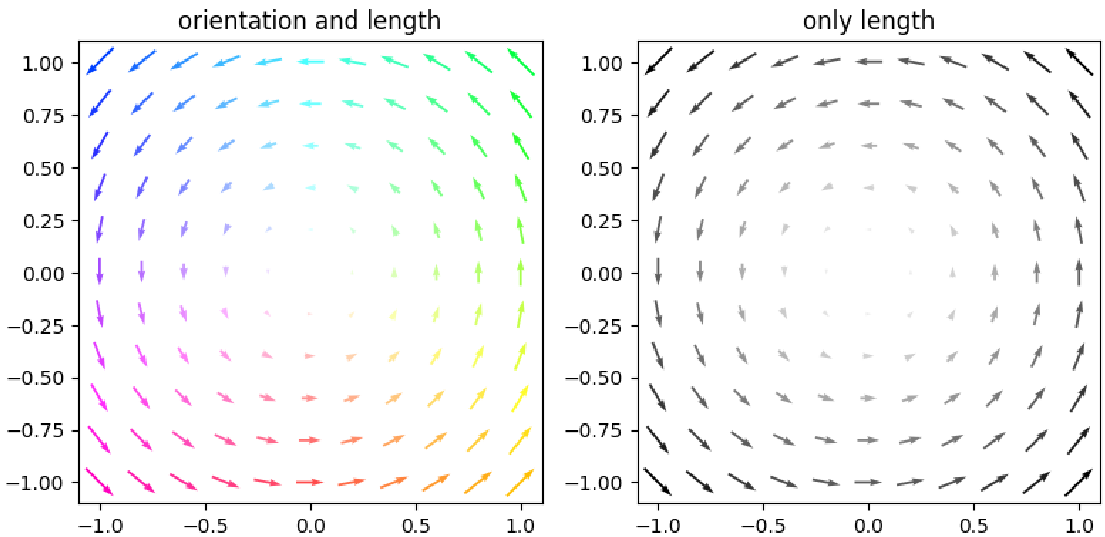

uv, aspect=0.5
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00


xy, aspect=0.5
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00


xy, aspect=1.
1.0
0.8
0.6
0.4
0.2
0.0
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

1.1.4 Color

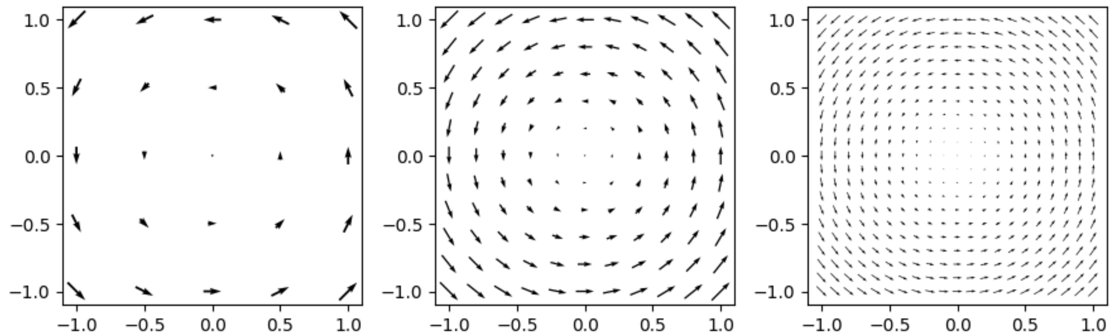
[7]:
```
# additional degree of freedom: color; use with care
# experiment: how about orientation as hue, length as saturation

# creat empty HSV array
colorsHSV=np.zeros((nPts,3))
# set Saturation to 1
colorsHSV[:,2]=1.
# set Value to (rescaled) magnitude
magnitude=((u**2+v**2)**0.5).ravel()
vmax=np.max(magnitude)
colorsHSV[:,1]=np.clip(magnitude/vmax,0.,1.)
# set Hue to orientation
phi=np.mod(np.arctan2(v,u).ravel()/2/np.pi,1.)
colorsHSV[:,0]=phi
# conver to RGB
colorsRGB=matplotlib.colors.hsv_to_rgb(colorsHSV)

# just visualize magnitude
colorsRGB2=cm.Greys(magnitude/vmax)
#colorsRGB2=cm.Reds(magnitude/vmax)
# use a colormap with 0=white

# elegant side effect of this coloring scheme: zero length arrows become␣
↪invisible,
# since color merges into background color


fig=plt.figure(figsize=(8,5))

ax=fig.add_subplot(1,2,1,aspect=1.)
ax.quiver(X,Y,u,v,color=colorsRGB,pivot="mid")
plt.title("orientation and length")

ax=fig.add_subplot(1,2,2,aspect=1.)
ax.quiver(X,Y,u,v,color=colorsRGB2,pivot="mid")
plt.title("only length")

plt.tight_layout()
plt.show()
```



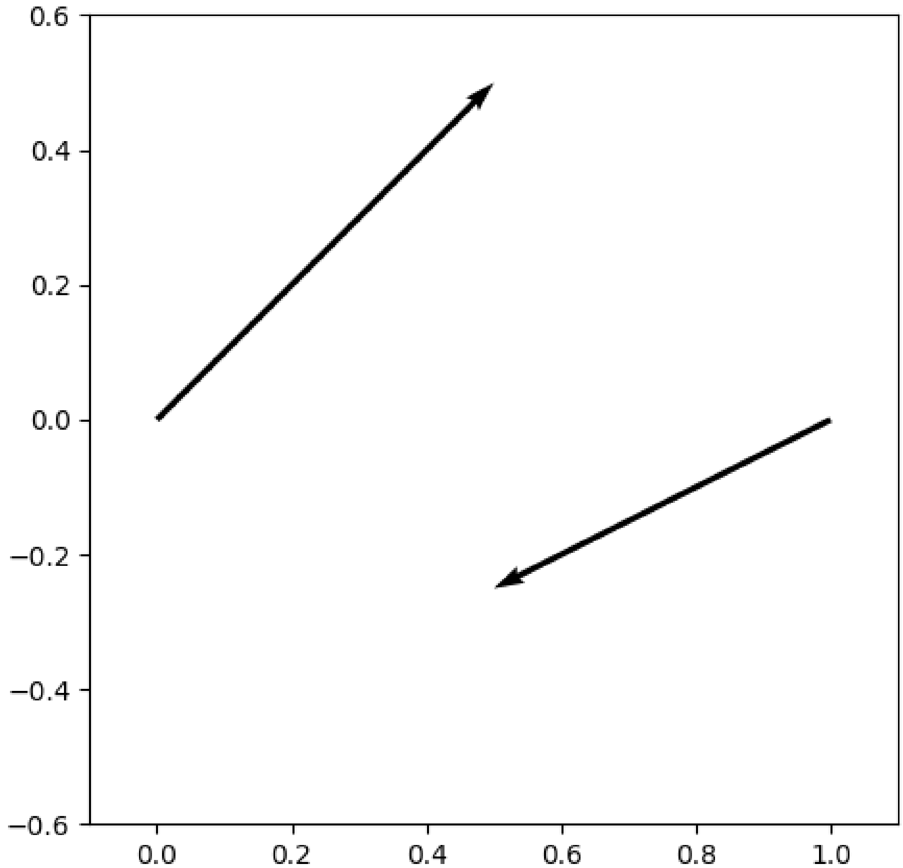
1.1.5 Number and length of arrows

[8]:
```
# generate a simple vector field at different resolutions
nPts1dList=[5,11,21]
nPlots=len(nPts1dList)

fig=plt.figure(figsize=(3*nPlots,3))


for i, nPts1d in enumerate(nPts1dList):
    x = np.linspace(-1,1,num=nPts1d)
    y = x
    nPts=nPts1d**2
    X, Y = np.meshgrid(x, y)
    u = -Y
    v = X

    ax=fig.add_subplot(1,nPlots,i+1,aspect=1.)
    ax.quiver(X,Y,u,v,pivot="mid")
    # show data points for comparison
    #ax.scatter(X.ravel(),Y.ravel(),marker="x",c="r")

plt.tight_layout()
plt.show()

# obvious trade-off:
# * more arrows: better spatial resolution
```

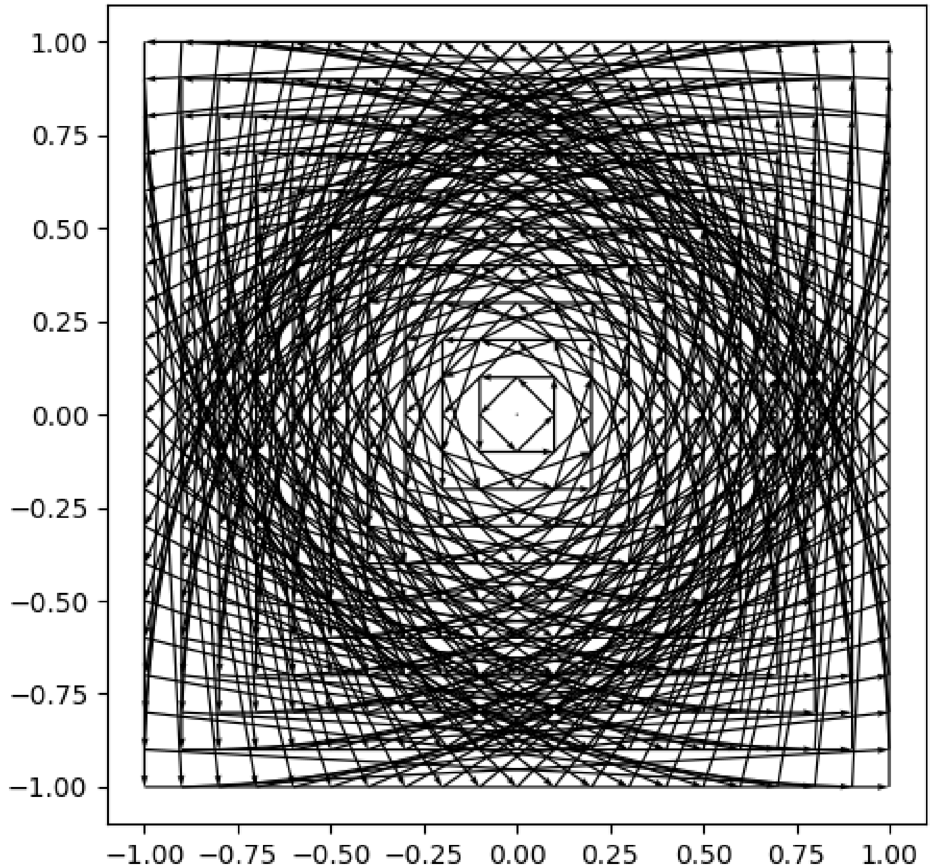
```
# * with too many arrows the plot becomes hard to read
```



[9]:
```
# length of the arrows:
# without specific instructions uses a simple auto-scale algorithm to produce
# a nice quiver plot with arrows not overlapping each other
# so the arrow lengths that we give to quiver are not exactly
# the lengths of the drawn arrows

# in principle vector field might encode function \R^2 \to \R^2
# then f(x,y)=(x+u,y+v)
# plotting arrows that literally go from (x,y) to f(x,y) will almost
# always look chaotic for large deformations
# still: sometimes we may want to plot arrows of exact length
# this can be done as follows


fig=plt.figure()
ax=fig.add_subplot(aspect=1.)
ax.quiver([0,1],[0,0],[0.5,-0.5],[0.5,-0.25],\
        pivot="tail",angles="xy",scale_units="xy",scale=1.)
plt.xlim([-0.1,1.1])
plt.ylim([-0.6,0.6])
plt.tight_layout()
plt.show()
```



[10]:
```
# using this to visualize maps from R^2 to R^2 as explicit arrows is almost
# always not working (if one has large displacements)
fig=plt.figure()
ax=fig.add_subplot(aspect=1.)
ax.quiver(X,Y,u-X,v-Y,pivot="tail",angles="xy",scale_units="xy",scale=1.)
plt.tight_layout()
plt.show()
# we will return to this challenge with different ideas a bit later
```



1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


```
[ ]:
```

```
[ ]:
```

```
[ ]:
```