In [2]:

```python
%pylab inline
%config InlineBackend.figure_format = 'retina'
from ipywidgets import interact
import tensorflow as tf
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout, GRU, CuDNNLSTM, Flatten
import math
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Dropout, LSTM
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.metrics import mean_absolute_error
import numpy as np
from numpy import pi
import tensorflow as tf
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [3]:

```python
class Particle_Tracking_Training_Data(tf.Module):
    def __init__(self, Nt, rings=True):
        self.Nt = int(Nt)
        self.Ny = self.Nx = 256
        self.d = 3
        ximg = [[[i, j] for i in np.arange(self.Ny)]
            for j in np.arange(self.Nx)]
        self.ximg = np.float32(ximg)

        x = np.arange(self.Nx) - self.Nx//2
        y = np.arange(self.Ny) - self.Ny//2
        X0, Y0 = np.meshgrid(x, y)
        self.X = np.float32(X0)
        self.Y = np.float32(Y0)

        if rings:
            self.ring_indicator = 1.
        else:
            self.ring_indicator = 0.

        self._gen_video = tf.function(
            input_signature=(
                tf.TensorSpec(
                    shape=[self.Ny, self.Nx, self.Nt, None], dtype=tf.float32),
                tf.TensorSpec(shape=[self.Nt, None], dtype=tf.float32),
                tf.TensorSpec(shape=[], dtype=tf.float32),
                tf.TensorSpec(shape=[], dtype=tf.float32),
                tf.TensorSpec(shape=[], dtype=tf.float32),)
        )(self._gen_video)

        self._gen_labels = tf.function(
            input_signature=(
                tf.TensorSpec(
                    shape=[self.Ny, self.Nx, self.Nt, None], dtype=tf.float32),)
        )(self._gen_labels)

    def __call__(self, kappa, a, IbackLevel, Nparticles, sigma_motion):
        ## random brownian motion paths
        ## Nt, Nparticles, 3
        xi = self._sample_motion(Nparticles, sigma_motion)

        #### translate track positions to img coords
        ## Ny, Nx, Nt, Np, 2
        XALL = (self.ximg[:, :, None, None, :]
                - xi[None, None, :, :, :2])
        ## Ny, Nx, Nt, Np
        r = tf.math.sqrt(XALL[..., 0]**2 + XALL[..., 1]**2)
        z = xi[..., 2]

        ### generate video
        I = self._gen_video(r, z, kappa, a, IbackLevel)

        ### generate labels
        labels = self._gen_labels(r)

        return I, labels, xi

    @staticmethod
    def rand(n):
        return tf.random.uniform([n], dtype=tf.float32)

    @tf.function(
        input_signature=(
            tf.TensorSpec(shape=[], dtype=tf.int32),
            tf.TensorSpec(shape=[], dtype=tf.float32),))
    def _sample_motion(self, Nparticles, sigma_motion):
        #### boundaries
        b_lower = tf.constant(
            [-10, -10, -30.], tf.float32)
        b_upper = tf.constant(
            [self.Nx+10, self.Ny+10, 30.], tf.float32)
        #### uniform random initial possitions
        U = tf.random.uniform(
            [1, Nparticles, self.d],
            dtype=tf.float32)
        X0 = b_lower + (b_upper - b_lower)*U
        #### normal increments
        dX = tf.random.normal(
            [self.Nt, Nparticles, self.d],
            stddev=sigma_motion,
            dtype=tf.float32)
        #### unbounded Brownian motion
        X = X0 + tf.math.cumsum(dX, axis=0)
        #### reflected brownian motion
        ## note that this is imperfect,
        ## if increments are very large it wont work
        X = tf.math.abs(X - b_lower) + b_lower
        X = -tf.math.abs(b_upper - X) + b_upper
        return X

    def _gen_video(self, r, z, kappa, a, IbackLevel):
        uw = (0.5 + self.rand(1))/2.
        un = tf.floor(3*self.rand(1))
        uampRing = 0.2 + 0.8*self.rand(1)
        ufade = 15 + 10*self.rand(1)
        rmax = ufade*(un/uw)**(2./3.)
        ufadeMax = 0.85
        fade = (1. - ufadeMax*tf.abs(tf.tanh(z/ufade)))
        core = tf.exp(-(r**2/(8.*a))**2)
        ring = fade*(tf.exp(-(r - z)**4/(a)**4)
                + 0.5*uampRing*tf.cast(r<z, tf.float32))
        I = tf.transpose(
            tf.reduce_sum(
                fade*(core + self.ring_indicator*ring),
                axis=3),
            [2, 0, 1]) # Nt, Ny, Nx
        I += IbackLevel*tf.sin(
            self.rand(1)*6*pi/512*tf.sqrt(
                self.rand(1)*(self.X - self.rand(1)*512)**2
                    + self.rand(1)*(self.Y - self.rand(1)*512)**2))
        I += tf.random.normal(
            [self.Nt, self.Ny, self.Nx],
            stddev=kappa,
            dtype=tf.float32)
        Imin = tf.reduce_min(I)
        Imax = tf.reduce_max(I)
        I = (I - Imin)/(Imax - Imin)
        I = tf.round(I*tf.maximum(256., tf.round(2**16*self.rand(1))))
        return I

    def _gen_labels(self, r):
        R_detect = 3.
        ## (Ny, Nx, Nt)
        detectors = tf.reduce_sum(
            tf.cast(r[::2, ::2, :, :] < R_detect, tf.int32),
            axis=3)
        ## (Nt, Ny, Nx)
        P = tf.transpose(
            tf.cast(detectors > 0, tf.int32), [2, 0, 1])
        ## (Nt, Ny, Nx, 2)
        labels = tf.stack([1-P, P], 3)
        return labels
```

In [4]:

```python
Nt = 300 ## number of frames for each video
kappa = 0.1 ## standard deviation of background noise added to image
a = 3. ## scale factor for the size of particle spots (not true size of particles)
IbackLevel = 0.1 ## relative intensity of randomly generated background pattern; in (0, 1)
Nparticles = 3 ## the number of particles (more => slower)
sigma_motion = 2 ## the standard deviation for particle brownian motion; should be in (0, 10)


pt = Particle_Tracking_Training_Data(Nt) ## create object instance

vid, labels, tracks = pt(kappa, a, IbackLevel, Nparticles, sigma_motion)
```

Generating the Data

In [7]:

```python
@interact(t=(0, Nt-1, 1))
def plotfn(t=0):
    fig = figure(1, [14, 7])
    imshow(vid[t], origin='lower')
    #xlim(-10, 265)
    #ylim(-10, 265)
```

Detecting the Particles

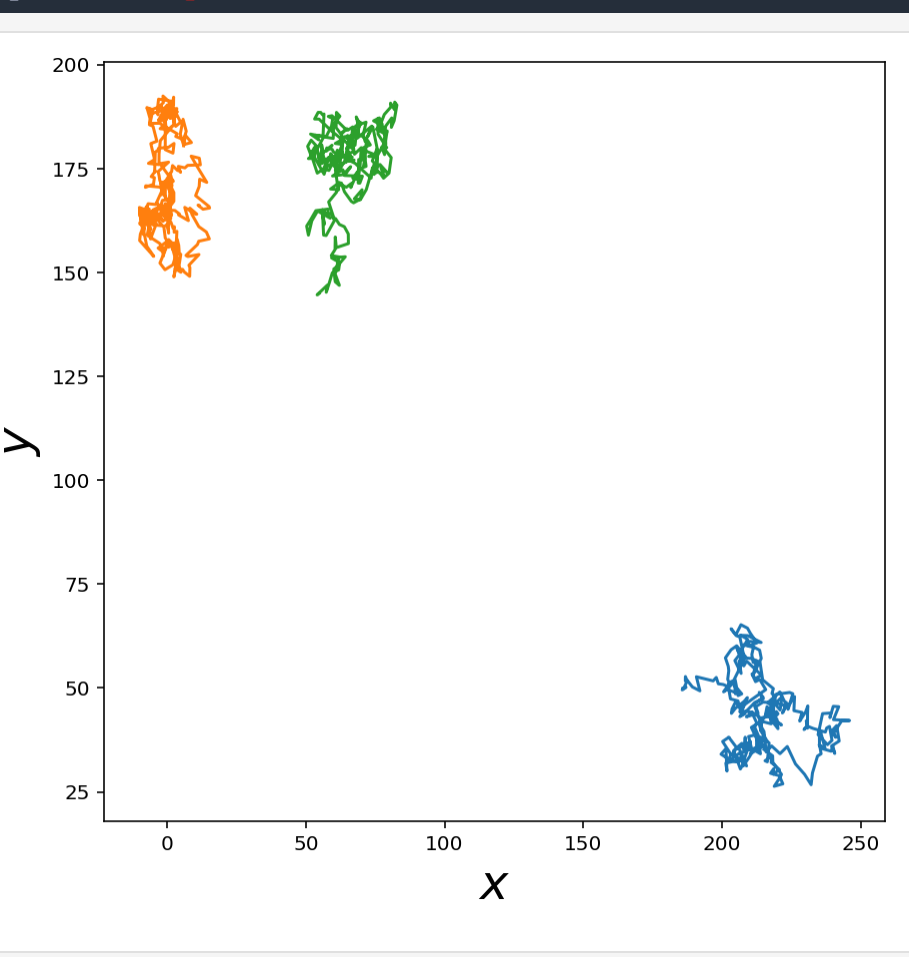
In [5]:

```python
@interact(t=(0, Nt-1, 1))
def plotfn(t=0, show_tracks=True):
    fig = figure(1, [14, 7])
    imshow(vid[t], origin='lower')
    if show_tracks:
        plot(tracks[t, :, 0], tracks[t, :, 1], 'rx')
    xlim(-10, 265)
    ylim(-10, 265)
```

Tracking the Particles

In [9]:

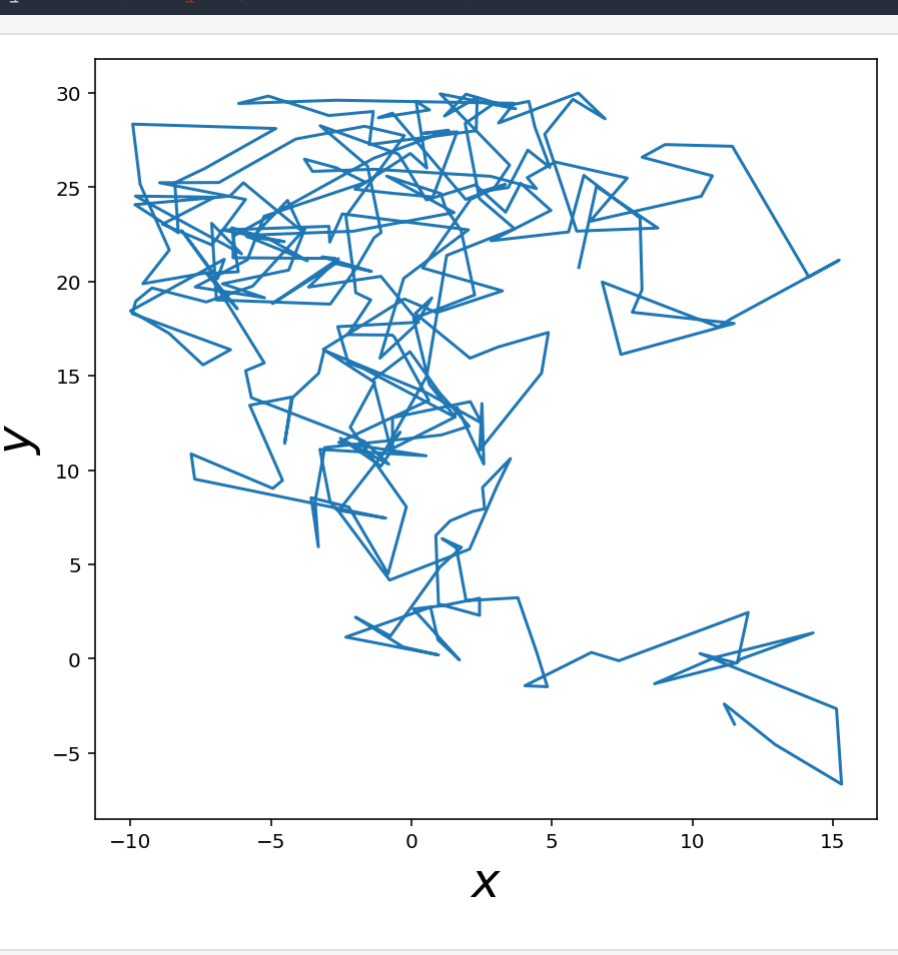
```python
figure(1, [7, 7])
plot(tracks[..., 0], tracks[..., 1])
#xlim(-12, 300)
#ylim(-12, 300)
xlabel(r'$x$', fontsize=24)
ylabel(r'$y$', fontsize=24);
```



In [ ]:

In [11]:

```python
figure(1, [7, 7])
plot(tracks[:,1,0], tracks[:,1,2])
#xlim(-12, 300)
#ylim(-12, 300)
xlabel(r'$x$', fontsize=24)
ylabel(r'$y$', fontsize=24);
```



In [115…

```python
X=np.array(tracks[:,0,0])
X_scaled=(X-np.mean(X))/np.std(X)
Y=np.array(tracks[:,0,1])
Y_scaled=(Y-np.mean(Y))/np.std(Y)
```

In [169…

```python
def get_data(data):
    a_train=[]
    b_train=[]
    for i in range(10,len(data)):
        a_train.append(data[i-10:i])
        b_train.append(data[i])
    a_train, b_train = np.array(a_train), np.array(b_train)
    return(a_train, b_train)
```

In [170…

```python
X_train, y_train=get_data(X_scaled[:250])
X_test, y_test=get_data(X_scaled[250:])
Xy_train, yy_train=get_data(Y_scaled[:250])
Xy_test, yy_test=get_data(Y_scaled[250:])
```

In [ ]:

In [ ]:

In [247…

```python
X_train = []
y_train = []
X_train_scaled=X_scaled[:250]
#train_data=train_data.numpy()
for i in range(30,len(X_scaled)):
    X_train.append(X_scaled[i-30:i])
    y_train.append(X_scaled[i])
X_train, y_train = np.array(X_train), np.array(y_train)
```

In [248…

```python
Xy_train = []
yy_train = []
Xy_train_scaled=Y_scaled[:250]
#train_data=train_data.numpy()
```

In [ ]:

```python
X_test = []
y_test = []

for i in range(30,len(X_scaled)):
    X_train.append(X_scaled[i-30:i])
    y_train.append(X_scaled[i])
X_train, y_train = np.array(X_train), np.array(y_train)

X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
```

In [ ]:

In [ ]:

In [ ]:

In [210…

In [ ]:

In [ ]:

In [76]:

```python
opt=tf.keras.optimizers.RMSprop(
    learning_rate=0.001
)
model = Sequential()
model.add(CuDNNLSTM(units=32,return_sequences=True, input_shape=(X_train.shape[1],1)))
#model.add(CuDNNLSTM(units=256,return_sequences=True))
#model.add(CuDNNLSTM(units=128,return_sequences=True))
#model.add(Dropout(0.2))
model.add(CuDNNLSTM(units=16,return_sequences=True))
#model.add(CuDNNLSTM(units=32,return_sequences=True))
model.add(Dropout(0.2))
model.add(CuDNNLSTM(units=8,return_sequences=True))
#model.add(CuDNNLSTM(units=8,return_sequences=True))
model.add(CuDNNLSTM(units=4,return_sequences=True))
model.add(Flatten())
model.add(Dense(1))
model.compile(optimizer=opt,loss='mean_squared_error',metrics=["mae"])
```

In [171…

```python
model=Sequential()
model.add(CuDNNLSTM(units=128,return_sequences=True, input_shape=(X_train.shape[1],1)) )
model.add(Dropout(0.2))
model.add(CuDNNLSTM(units=64,return_sequences=True) )
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(units=1))
# Compiling the RNN
model.compile(optimizer='rmsprop',loss='mean_squared_error')
# Fitting to the training set
#model.fit(X_train,y_train,epochs=50,batch_size=16)
```

In [172…

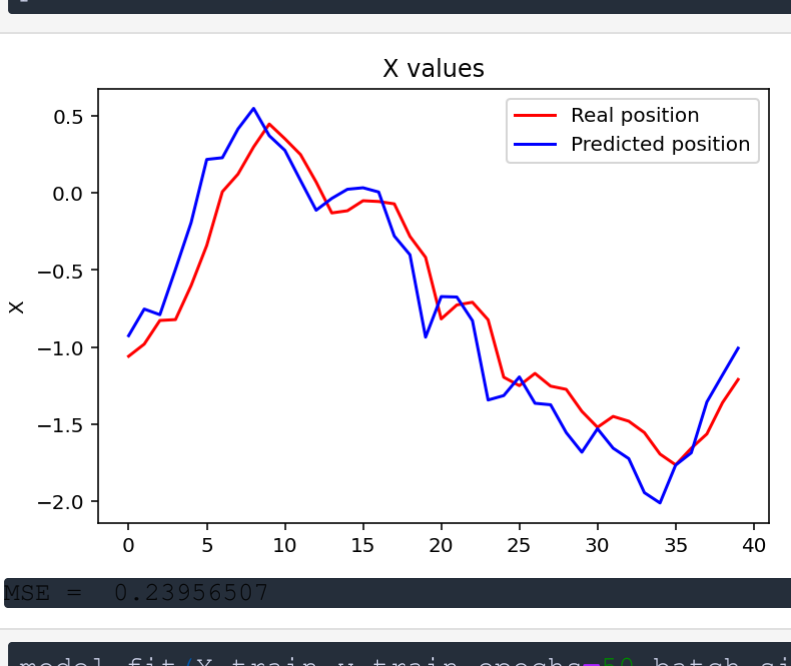
```python
model.fit(Xy_train,yy_train,epochs=50,batch_size=16,validation_data=(Xy_test[:2], yy_test[:2]))
y_pred = model.predict(Xy_test)
```

```
Epoch 1/50
15/15 [==============================] - 3s 51ms/step - loss: 0.3942 - val_loss: 0.0273
Epoch 2/50
15/15 [==============================] - 0s 17ms/step - loss: 0.2671 - val_loss: 0.0733
Epoch 3/50
15/15 [==============================] - 0s 14ms/step - loss: 0.2214 - val_loss: 0.0130
Epoch 4/50
15/15 [==============================] - 0s 15ms/step - loss: 0.2123 - val_loss: 0.0633
Epoch 5/50
15/15 [==============================] - 0s 14ms/step - loss: 0.2002 - val_loss: 0.0469
Epoch 6/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1866 - val_loss: 0.0533
Epoch 7/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1691 - val_loss: 0.1987
Epoch 8/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1565 - val_loss: 0.1699
Epoch 9/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1450 - val_loss: 0.0362
Epoch 10/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1434 - val_loss: 0.0080
Epoch 11/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1394 - val_loss: 0.0292
Epoch 12/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1349 - val_loss: 0.0361
Epoch 13/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1274 - val_loss: 0.0794
Epoch 14/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1211 - val_loss: 0.0272
Epoch 15/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1308 - val_loss: 0.1241
Epoch 16/50
15/15 [==============================] - 0s 15ms/step - loss: 0.1114 - val_loss: 0.1256
Epoch 17/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1246 - val_loss: 0.0750
Epoch 18/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1083 - val_loss: 0.0807
Epoch 19/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1047 - val_loss: 0.3538
Epoch 20/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1097 - val_loss: 0.1290
Epoch 21/50
15/15 [==============================] - 0s 14ms/step - loss: 0.1005 - val_loss: 0.0386
Epoch 22/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0917 - val_loss: 0.0923
Epoch 23/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0891 - val_loss: 0.1447
Epoch 24/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0866 - val_loss: 0.0242
Epoch 25/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0836 - val_loss: 0.0344
Epoch 26/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0791 - val_loss: 0.0398
Epoch 27/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0812 - val_loss: 0.0130
Epoch 28/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0772 - val_loss: 0.0553
Epoch 29/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0874 - val_loss: 0.0650
Epoch 30/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0785 - val_loss: 0.0113
Epoch 31/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0745 - val_loss: 0.0046
Epoch 32/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0760 - val_loss: 0.0064
Epoch 33/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0692 - val_loss: 0.0457
Epoch 34/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0792 - val_loss: 0.0433
Epoch 35/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0727 - val_loss: 0.0077
Epoch 36/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0660 - val_loss: 0.0548
Epoch 37/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0626 - val_loss: 0.1822
Epoch 38/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0756 - val_loss: 0.0087
Epoch 39/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0612 - val_loss: 0.0471
Epoch 40/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0723 - val_loss: 0.0122
Epoch 41/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0667 - val_loss: 0.0166
Epoch 42/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0701 - val_loss: 0.0294
Epoch 43/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0719 - val_loss: 0.0903
Epoch 44/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0723 - val_loss: 0.0355
Epoch 45/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0615 - val_loss: 0.0617
Epoch 46/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0592 - val_loss: 0.0164
Epoch 47/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0651 - val_loss: 0.0809
Epoch 48/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0608 - val_loss: 0.0430
Epoch 49/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0639 - val_loss: 0.0327
Epoch 50/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0631 - val_loss: 0.0348
```

In [173…

```python
plt.plot(y_pred, color='red',label='Real position')
plt.plot(yy_test, color='blue',label='Predicted position');
plt.title('X values')
#plt.xlabel('Date')
plt.ylabel('X')
plt.legend()
plt.show()

MSE1=mean_squared_error(y_pred, yy_test, squared=False)
print("MSE = ", MSE1)
```




```
MSE =  0.23956507
```

In [174…

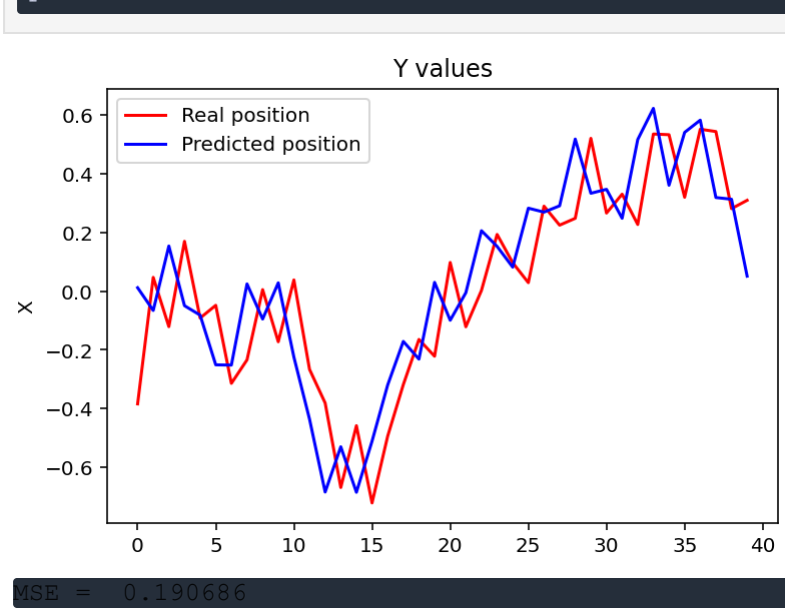
```python
model.fit(X_train,y_train,epochs=50,batch_size=16, validation_data=(X_test[:2], y_test[:2]))
x_pred = model.predict(X_test)
```

```
Epoch 1/50
15/15 [==============================] - 0s 20ms/step - loss: 0.0490 - val_loss: 0.0503
Epoch 2/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0478 - val_loss: 0.0663
Epoch 3/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0443 - val_loss: 0.0869
Epoch 4/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0480 - val_loss: 0.0574
Epoch 5/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0452 - val_loss: 0.0537
Epoch 6/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0460 - val_loss: 0.0717
Epoch 7/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0409 - val_loss: 0.0620
Epoch 8/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0392 - val_loss: 0.0819
Epoch 9/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0424 - val_loss: 0.0612
Epoch 10/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0402 - val_loss: 0.0589
Epoch 11/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0414 - val_loss: 0.0709
Epoch 12/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0466 - val_loss: 0.0882
Epoch 13/50
15/15 [==============================] - 0s 16ms/step - loss: 0.0358 - val_loss: 0.0707
Epoch 14/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0393 - val_loss: 0.1082
Epoch 15/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0424 - val_loss: 0.0858
Epoch 16/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0446 - val_loss: 0.0722
Epoch 17/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0465 - val_loss: 0.1270
Epoch 18/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0450 - val_loss: 0.0623
Epoch 19/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0453 - val_loss: 0.0855
Epoch 20/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0437 - val_loss: 0.0955
Epoch 21/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0375 - val_loss: 0.0755
Epoch 22/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0426 - val_loss: 0.0796
Epoch 23/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0400 - val_loss: 0.0636
Epoch 24/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0423 - val_loss: 0.0743
Epoch 25/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0378 - val_loss: 0.0720
Epoch 26/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0398 - val_loss: 0.0781
Epoch 27/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0428 - val_loss: 0.0781
Epoch 28/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0373 - val_loss: 0.0836
Epoch 29/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0444 - val_loss: 0.0846
Epoch 30/50
15/15 [==============================] - 0s 16ms/step - loss: 0.0395 - val_loss: 0.1001
Epoch 31/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0420 - val_loss: 0.0917
Epoch 32/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0385 - val_loss: 0.0747
Epoch 33/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0390 - val_loss: 0.0739
Epoch 34/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0404 - val_loss: 0.0949
Epoch 35/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0396 - val_loss: 0.0648
Epoch 36/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0358 - val_loss: 0.0788
Epoch 37/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0376 - val_loss: 0.0836
Epoch 38/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0494 - val_loss: 0.0925
Epoch 39/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0369 - val_loss: 0.0889
Epoch 40/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0383 - val_loss: 0.0903
Epoch 41/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0369 - val_loss: 0.0861
Epoch 42/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0359 - val_loss: 0.0741
Epoch 43/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0395 - val_loss: 0.0815
Epoch 44/50
15/15 [==============================] - 0s 16ms/step - loss: 0.0371 - val_loss: 0.0868
Epoch 45/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0413 - val_loss: 0.0813
Epoch 46/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0415 - val_loss: 0.0933
Epoch 47/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0372 - val_loss: 0.0847
Epoch 48/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0385 - val_loss: 0.0904
Epoch 49/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0394 - val_loss: 0.1009
Epoch 50/50
15/15 [==============================] - 0s 15ms/step - loss: 0.0371 - val_loss: 0.0846
```

In [175…

```python
plt.plot(x_pred, color='red',label='Real position')
plt.plot(y_test, color='blue',label='Predicted position');
plt.title('Y values')
#plt.xlabel('Date')
plt.ylabel('X')
plt.legend()
plt.show()

MSE1=mean_squared_error(x_pred, y_test, squared=False)
print("MSE = ", MSE1)
```




```
MSE =  0.190686
```

In [139…

```python
x_pred
```

Out[139…

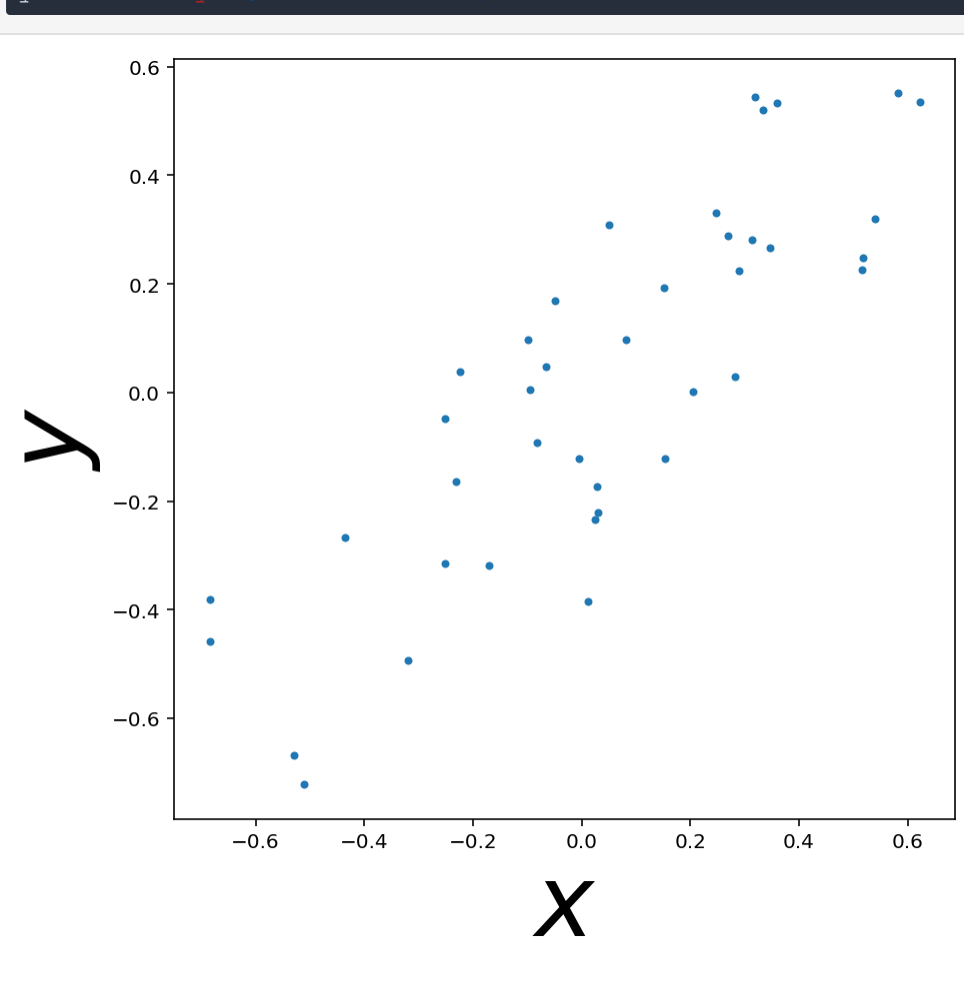
```
array([[ 0.02891346],
       [-0.25876385],
       [-0.09110858],
       [ 0.08409639],
       [-0.03282701],
       [-0.1471768 ],
       [ 0.1513299 ],
       [ 0.12525119],
       [ 0.06137033],
       [ 0.47542968],
       [ 0.11650269],
       [ 0.24012011],
       [ 0.12309794],
       [ 0.4640012 ],
       [ 0.4176118 ],
       [ 0.29948637],
       [ 0.4742349 ],
       [ 0.4235113 ],
       [ 0.16309182],
       [ 0.21584761]], dtype=float32)
```

In [176…

```python
figure(1, [7, 7])
plot(y_test,x_pred,'.')
#xlim(50, 200)
#ylim(50, 200)



xlabel(r'$x$', fontsize=50)
ylabel(r'$y$', fontsize=50);
```

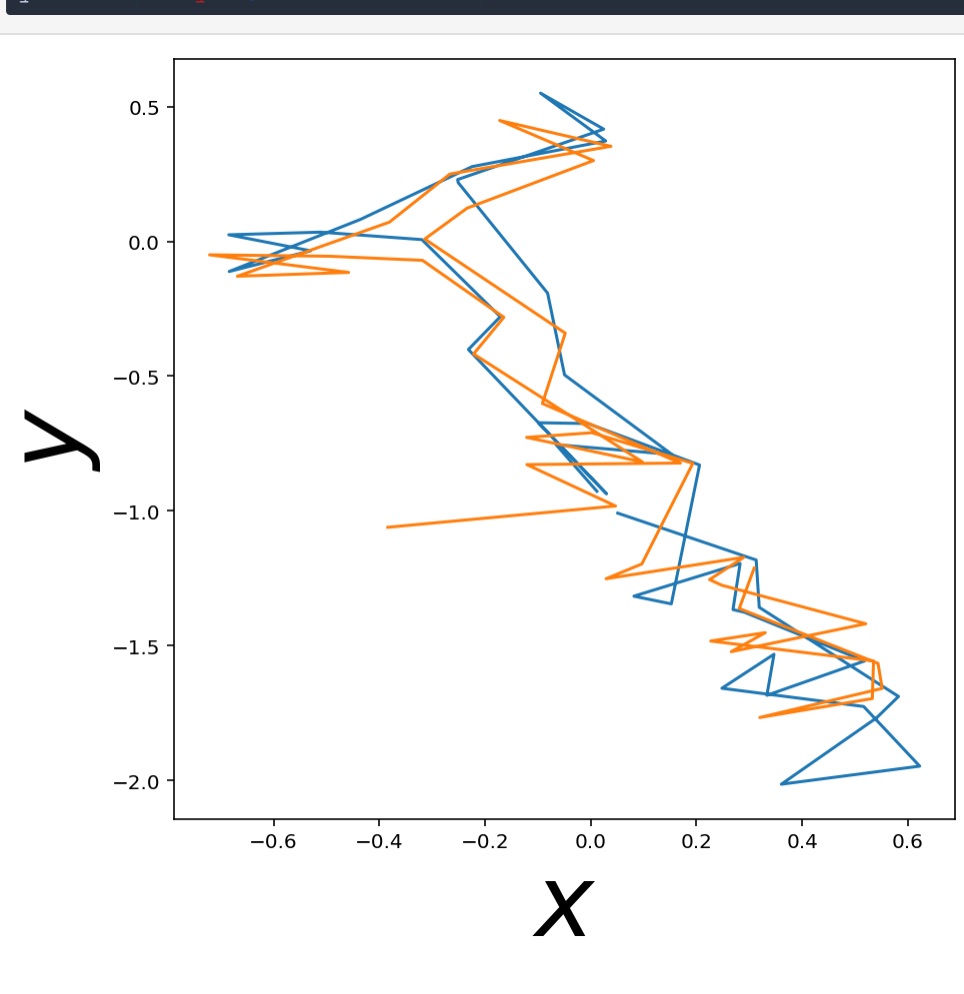


In [ ]:

In [178…

```python
figure(1, [7, 7])
plot(y_test,yy_test)
#xlim(50, 200)
#ylim(50, 200)
plot(x_pred, y_pred)

xlabel(r'$x$', fontsize=50)
ylabel(r'$y$', fontsize=50);
```



In [227…

```python
tracks[250:,1,0]
```