In [40]:
```
import tensorflow as tf
from tensorflow.keras import layers
import tensorflow_addons as tfa

import numpy as np
from scipy.integrate import odeint

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
from matplotlib import cm

from time import time

#%matplotlib notebook
```

In [41]:
```
def plot(pred_data, true_data, model_name):
    print(true_data.shape)
    print(pred_data.shape)
    stps=10
    # LYAPUNOV TIME
    dt = 0.01
    pl = pred_data.shape[0]
    xp = pred_data[:,0]; xt= true_data[stps:(pl+stps),0]
    yp = pred_data[:,1]; yt= true_data[stps:(pl+stps),1]
    zp = pred_data[:,2]; zt= true_data[stps:(pl+stps),2]
    rmse = np.sqrt((xp-xt)**2+(yp-yt)**2+(zp-zt)**2)

    try:
        lyind = np.min(np.argwhere(rmse > 2.71828))
        lyaptime = (lyind-1)*dt
    except:
        lyind = 0
        lyaptime = 0

    tp = np.linspace(0, dt*pl, pl)
    print('Model {} Lyapunov time = {}'.format(model_name, lyaptime))

    fig, ax = plt.subplots(2,2)
    fig.set_figwidth(10)
    fig.set_figheight(5)
    fig.suptitle('Rossler System Forecast (Type 2) - Model {} (true vs. pred)'.format(mo
    plt.subplots_adjust(hspace=0.5)
    ax[0,0].plot(tp, xp, c='orange', lw = 1)
    ax[0,0].plot(tp, xt, c='black', linestyle='dashed', lw = 0.75)
    ax[0,0].set_xlabel('time')
    ax[0,0].set_ylabel('x')
    ax[0,0].set_title('x-axis')
    ax[0,0].legend( ['predicted','true'])

    ax[0,1].plot(tp, yp, c='orange', lw = 1)
    ax[0,1].plot(tp, yt, c='black', linestyle='dashed', lw = 0.75)
    ax[0,1].set_xlabel('time')
    ax[0,1].set_ylabel('y')
    ax[0,1].set_title('y-axis')
    ax[0,1].legend( ['predicted','true'])

    ax[1,0].plot(tp, zp, c='orange', lw = 1)
    ax[1,0].plot(tp, zt, c='black', linestyle='dashed', lw = 0.75)
    ax[1,0].set_xlabel('time')
    ax[1,0].set_ylabel('z')
    ax[1,0].set_title('z-axis')
    ax[1,0].legend( ['predicted','true'])
```

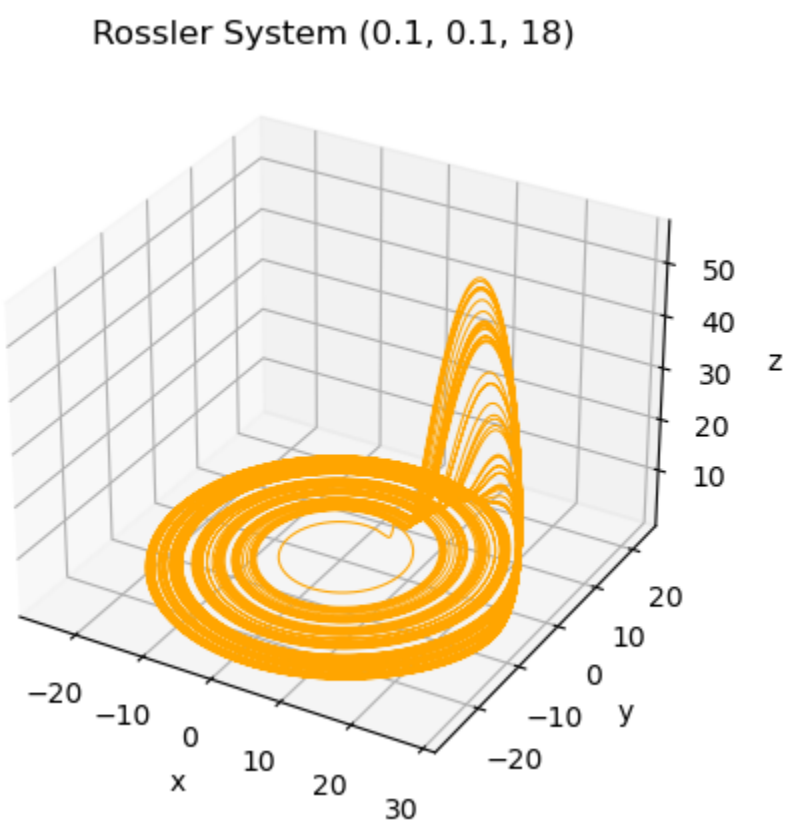
```
    ax[1,1].plot(tp, rmse, c='orange', lw = 1)
    ax[1,1].vlines(lyaptime, ymin = np.min(rmse), ymax=np.max(rmse), color = 'red')
    ax[1,1].set_xlabel('time')
    ax[1,1].set_ylabel('distance (rmse)')
    ax[1,1].set_title('Euclidean distance vs Time')
    plt.savefig('rossler-{}'.format(model_name), bbox_inches='tight',dpi = 200)

    # 3D Plot of Lorenz System
    ax = plt.figure().add_subplot(projection='3d')
    ax.plot(xt,yt,zt, lw=0.5, alpha=0.75, c='black')
    ax.plot(xp,yp,zp, lw=0.5, alpha=1, c='orange')
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('z')
    ax.set_title('{} Model - Rossler System (pred vs. true)'.format(model_name))  # Plot
    plt.savefig('3d-rossler-{}'.format(model_name), bbox_inches='tight', dpi = 200)
```

Generating Rossler System data

In [42]:
```
def Rossler(x, t, a = 0.1, b = 0.1, c = 18):
    x,y,z = x
    dfdt = [-y-z, x+a*y, b+ z*(x-c)]
    return dfdt

# initial conditions
dt = 0.01
train_pts = 50001
x0_train = [5,5,5]

t = np.linspace(0, dt*train_pts, train_pts)

ross_param = (0.1,0.1,18)
sol = odeint(Rossler, x0_train, t, args=ross_param)

x = sol[:,0]; y = sol[:,1]; z = sol[:,2]
input_data = np.array([x,y,z]).T
print('input_data shape:',input_data.shape)
```

```
input_data shape: (50001, 3)
```

In [43]:
```
# Plotting Lorenz System data
ax = plt.figure().add_subplot(projection='3d')
ax.plot(x,y,z, c='orange', lw = 0.75, alpha = 1)  # Plot contour curves

ax.set_xlabel('x')  # Plot contour curves
ax.set_ylabel('y')  # Plot contour curves
ax.set_zlabel('z')  # Plot contour curves
ax.set_title('Rossler System {}'.format(ross_param))  # Plot contour curves
plt.show()
```



Training Input Data

In [44]:
```
out_step = 1
in_steps = 10
train_pts = 50000
samples = train_pts-in_steps-out_step

train_input = np.zeros((samples, in_steps, 3))
print('RNN data shape (batches, timesteps, dim):',train_input.shape)

for i in range(samples):
    train_input[i,:,:] = input_data[i:i+in_steps,:]

print('first batch {x_1,x_2,...,x_10}):\n', train_input[0])
```

```
RNN data shape (batches, timesteps, dim): (49989, 10, 3)
first batch {x_1,x_2,...,x_10}):
 [[5.         5.         5.        ]
 [4.90284096 5.05453801 4.38925053]
 [4.81088657 5.1081851  3.84957934]
 [4.72347347 5.16098909 3.3733529 ]
 [4.64001068 5.21299154 2.95363278]
 [4.55997289 5.26422851 2.58413974]
 [4.48289413 5.3147311  2.25921224]
 [4.40836182 5.3645261  1.97376265]
 [4.33601136 5.41363641 1.72323233]
 [4.26552105 5.46208153 1.50354705]]
```

Training Output Data

In [45]:
```
train_output = np.zeros((samples, 3))
for i in range(samples):
    train_output[i] = input_data[i+in_steps]
```

```
print('output shape:',train_output.shape)
print('first target batch x_{11}:', train_output[0])
```

```
output shape: (49989, 3)
first target batch x_{11}: [4.19660751 5.50987795 1.3110738 ]
```

1) Build GRU Network

In [46]:
```
input_dim = 3; output_size=3
gru_units = 100
lr = tf.keras.optimizers.schedules.PiecewiseConstantDecay([50,100],[1e-2,1e-3,1e-4])

LayerGRU = layers.RNN(layers.GRUCell(gru_units), input_shape=(None, input_dim))

GRUmodel = tf.keras.Sequential([LayerGRU,
                                layers.Dense(output_size)])

GRUmodel.compile(loss=tf.keras.losses.Huber(),
                 optimizer=tf.keras.optimizers.Adam(learning_rate=lr),
                 metrics=['mse'])
```

2) Build LSTM Network

In [47]:
```
input_dim = 3; output_size=3
lstm_units = 100
lr = tf.keras.optimizers.schedules.PiecewiseConstantDecay([50,100],[1e-2,1e-3,1e-4])

LayerLSTM = layers.RNN(layers.LSTMCell(lstm_units), input_shape=(None, input_dim))

LSTMmodel = tf.keras.Sequential([LayerLSTM,
                                 layers.Dense(output_size)])

LSTMmodel.compile(loss=tf.keras.losses.Huber(),
                  optimizer=tf.keras.optimizers.Adam(learning_rate=lr),
                  metrics=['mse'])
```

3) Build ESN Network

In [48]:
```
input_dim = 3; output_size=3
esn_units = 100
lr = tf.keras.optimizers.schedules.PiecewiseConstantDecay([50,100],[1e-2,1e-3,1e-4])

ESNmodel = tf.keras.Sequential([tfa.layers.ESN(esn_units, connectivity = 0.05, leaky = 0
                                layers.Dense(output_size)])

ESNmodel.compile(loss=tf.keras.losses.Huber(),
                 optimizer=tf.keras.optimizers.Adam(learning_rate=lr),
                 metrics=['mse'])
```

In [49]:
```
historyGRU = GRUmodel.fit(train_input, train_output, verbose=1, epochs=150)
```

```
Epoch 1/150
1563/1563 [==============================] - 21s 13ms/step - loss: 0.2832 - mse: 3.3718
Epoch 2/150
1563/1563 [==============================] - 20s 13ms/step - loss: 0.0844 - mse: 1.5674
Epoch 3/150
1563/1563 [==============================] - 20s 13ms/step - loss: 0.0534 - mse: 0.8904
Epoch 4/150
1563/1563 [==============================] - 20s 13ms/step - loss: 0.0330 - mse: 0.4680
Epoch 5/150
```

```
Epoch 125/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1701 - mse: 0.6619
Epoch 126/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1694 - mse: 0.6583
Epoch 127/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1686 - mse: 0.6546
Epoch 128/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1679 - mse: 0.6503
Epoch 129/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1672 - mse: 0.6467
Epoch 130/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1664 - mse: 0.6426
Epoch 131/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1658 - mse: 0.6390
Epoch 132/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1650 - mse: 0.6348
Epoch 133/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1643 - mse: 0.6314
Epoch 134/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1636 - mse: 0.6275
Epoch 135/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1630 - mse: 0.6239
Epoch 136/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1623 - mse: 0.6204
Epoch 137/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1616 - mse: 0.6168
Epoch 138/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1609 - mse: 0.6131
Epoch 139/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1603 - mse: 0.6097
Epoch 140/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1596 - mse: 0.6063
Epoch 141/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1590 - mse: 0.6028
Epoch 142/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1583 - mse: 0.5988
Epoch 143/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1577 - mse: 0.5955
Epoch 144/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1570 - mse: 0.5921
Epoch 145/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1564 - mse: 0.5887
Epoch 146/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1558 - mse: 0.5856
Epoch 147/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1552 - mse: 0.5823
Epoch 148/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1546 - mse: 0.5788
Epoch 149/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1540 - mse: 0.5754
Epoch 150/150
1563/1563 [==============================] - 2s 1ms/step - loss: 0.1533 - mse: 0.5723
```

Test Input data

In [52]:

```
# MODEL FORECAST
x0_test = train_output[-1]
print('test initial cond:', x0_test)
test_pts = 5001
t_test = np.linspace(t[-1], t[-1]+dt*test_pts, test_pts)
solt = odeint(Rossler, x0_test, t_test, args=ross_param)
xt = solt[:,0]; yt = solt[:,1]; zt = solt[:,2]
test_data = np.array([xt,yt,zt]).T
```

```
samplest = test_pts-in_steps-out_step
test_input = np.zeros((samplest-1, in_steps, 3))
test_target = np.zeros((samplest-1, 3))

for i in range(samplest-1):
    test_input[i,:,:] = test_data[i:i+in_steps,:]
    test_target[i,:] = test_data[i+in_steps+1]

print('RNN test data shape (batches, timesteps, dim):',test_input.shape)
print('RNN target data shape:',test_target.shape)
```

```
test initial cond: [1.38611198e+00 1.94653704e+01 1.91561353e-02]
RNN test data shape (batches, timesteps, dim): (4989, 10, 3)
RNN target data shape: (4989, 3)
```

Forecast one step into the future for test data sequences

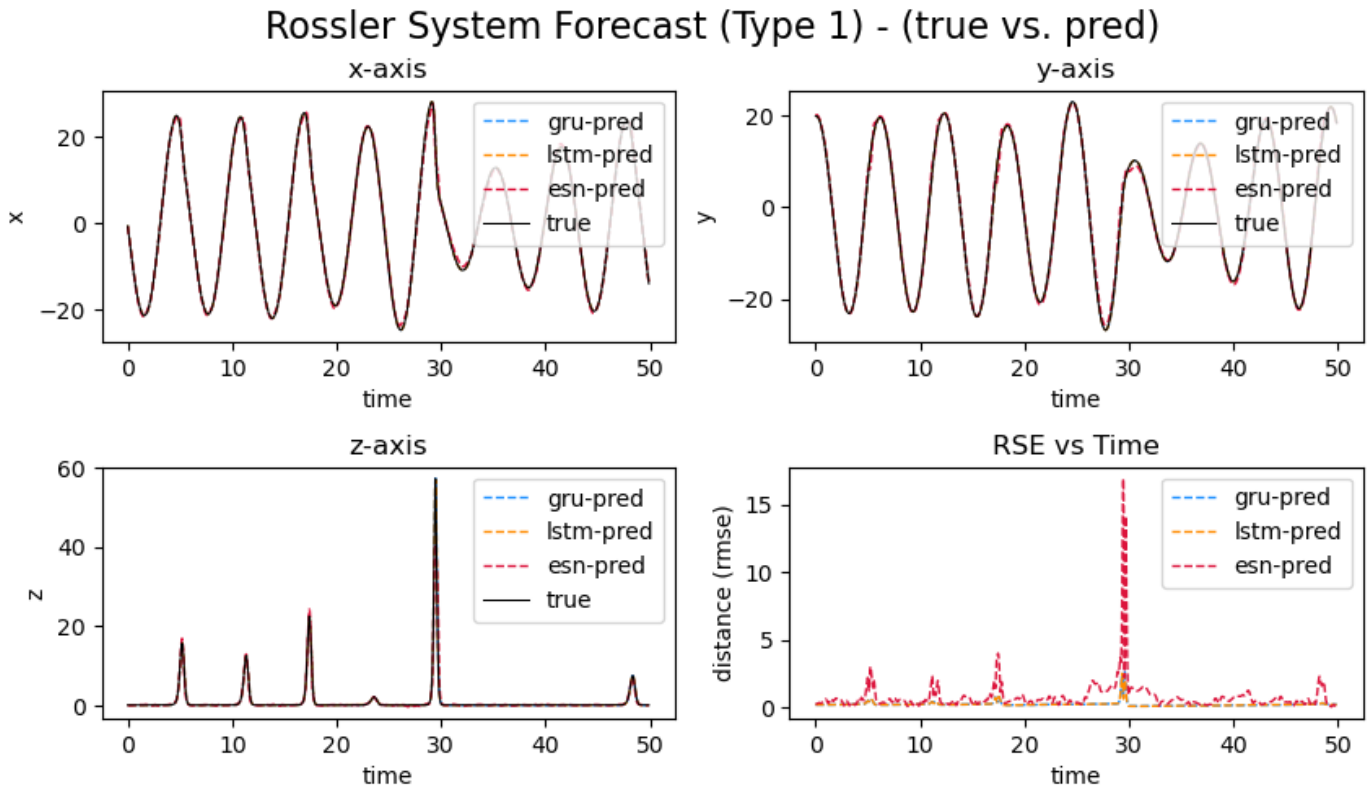
In [53]:

```
def forecast_t1_plot(models, test_input, test_target, units):
    pres_list = []
    for model in models:
        pres_list.append(model.predict(test_input))

    pl = test_target.shape[0]
    tp = np.linspace(0, dt*pl, pl)

    fig, ax = plt.subplots(2,2)
    fig.set_figwidth(10)
    fig.set_figheight(5)
    fig.suptitle('Rossler System Forecast (Type 1) - (true vs. pred)', fontsize=16)
    plt.subplots_adjust(hspace=0.5)

    xg = pres_list[0][:,0]; yg = pres_list[0][:,1]; zg = pres_list[0][:,2]
    xl = pres_list[1][:,0]; yl = pres_list[1][:,1]; zl = pres_list[1][:,2]
    xe = pres_list[2][:,0]; ye = pres_list[2][:,1]; ze = pres_list[2][:,2]
    xt = test_target[:,0]; yt = test_target[:,1]; zt = test_target[:,2]

    grmse = np.sqrt((xg-xt)**2+(yg-yt)**2+(zg-zt)**2)
    lrmse = np.sqrt((xl-xt)**2+(yl-yt)**2+(zl-zt)**2)
    ermse = np.sqrt((xe-xt)**2+(ye-yt)**2+(ze-zt)**2)

    ax[0,0].plot(tp, xg, c='dodgerblue', linestyle='dashed', lw = 1)
    ax[0,0].plot(tp, xl, c='darkorange', linestyle='dashed', lw = 1)
    ax[0,0].plot(tp, xe, c='crimson', linestyle='dashed', lw = 1)
    ax[0,0].plot(tp, xt, c='black', lw = 0.75)
    ax[0,0].set_xlabel('time')
    ax[0,0].set_ylabel('x')
    ax[0,0].set_title('x-axis')
    ax[0,0].legend(['gru-pred','lstm-pred','esn-pred','true'], loc = 'upper right')

    ax[0,1].plot(tp, yg, c='dodgerblue', linestyle='dashed', lw = 1)
    ax[0,1].plot(tp, yl, c='darkorange', linestyle='dashed', lw = 1)
    ax[0,1].plot(tp, ye, c='crimson', linestyle='dashed', lw = 1)
    ax[0,1].plot(tp, yt, c='black', lw = 0.75)
    ax[0,1].set_xlabel('time')
    ax[0,1].set_ylabel('y')
    ax[0,1].set_title('y-axis')
    ax[0,1].legend(['gru-pred','lstm-pred','esn-pred','true'], loc = 'upper right')

    ax[1,0].plot(tp, zg, c='dodgerblue', linestyle='dashed', lw = 1)
    ax[1,0].plot(tp, zl, c='darkorange', linestyle='dashed', lw = 1)
    ax[1,0].plot(tp, ze, c='crimson', linestyle='dashed', lw = 1)
    ax[1,0].plot(tp, zt, c='black', lw = 0.75)
    ax[1,0].set_xlabel('time')
    ax[1,0].set_ylabel('z')
```

```
    ax[1,0].set_title('z-axis')
    ax[1,0].legend(['gru-pred','lstm-pred','esn-pred','true'], loc = 'upper right')

    ax[1,1].plot(tp, grmse, c='dodgerblue', linestyle='dashed', lw = 1)
    ax[1,1].plot(tp, lrmse, c='darkorange', linestyle='dashed', lw = 1)
    ax[1,1].plot(tp, ermse, c='crimson', linestyle='dashed', lw = 1)
    ax[1,1].set_xlabel('time')
    ax[1,1].set_ylabel('distance (rmse)')
    ax[1,1].set_title('RSE vs Time')
    ax[1,1].legend(['gru-pred','lstm-pred','esn-pred','true'], loc = 'upper right')
    plt.savefig('rossler-t1-{}'.format(units), bbox_inches='tight',dpi = 200)
```

In [54]:
```
models = [GRUmodel, LSTMmodel, ESNmodel]
units=100
forcast_t1_plot(models, test_input, test_target, units)
```

```
156/156 [==============================] - 1s 3ms/step
156/156 [==============================] - 1s 4ms/step
156/156 [==============================] - 0s 1ms/step
```



Forecast N steps into the future until divergence

In [55]:
```
def forecast_t2(model, test_input, in_steps, samples):
    # warmup we have get in_steps' predictions as initial conditions
    # to allow model forecast from predictions
    hat=np.zeros((samples, in_steps, 3))

    # use first 10 inputs (10,10,3) to get an initial condition
    for j in range(in_steps):
        tin = np.zeros((1, in_steps, 3))
        tin[0] = test_input[j]
        hat[0,j] = model.predict(tin)
    print('init cond terminated')

    # generate predictions from initial last 10 predictions
    predseq = np.zeros((samples-in_steps, 3))
    for i in range(1,samples-in_steps):
        tin = np.zeros((1, in_steps, 3))
```

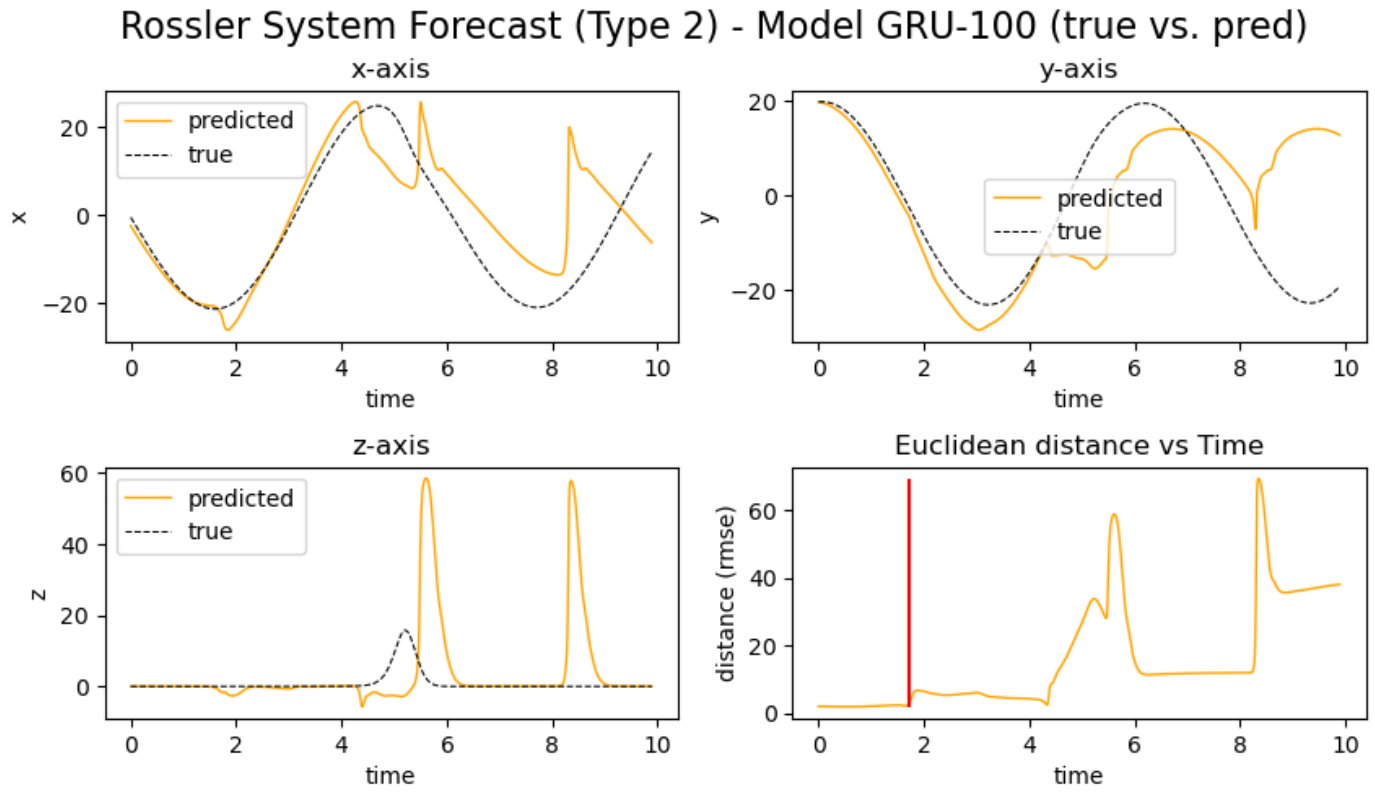
```
        tin[0] = hat[i-1]
        pred = model(tin)
        predseq[i-1] = pred
        # put last 9 steps from 'i-1' as first 9 steps in 'i'
        hat[i,0:(in_steps-1)]=hat[i-1,1:in_steps]
        hat[i,-1] = pred[0]
    print('forecast terminated')
    return(predseq)
```

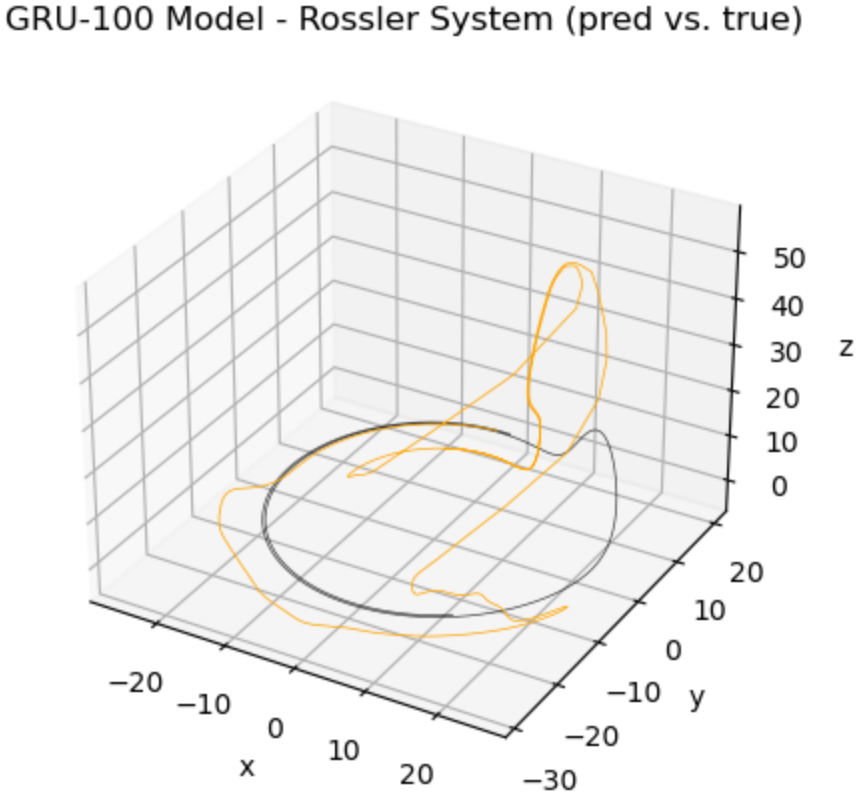
In [56]:
```
gru_pred=forcast_t2(model=GRUmodel,test_input=test_input,in_steps=10,samples=1000)
```

```
1/1 [==============================] - 0s 12ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 12ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 12ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 13ms/step
init cond terminated
forecast terminated
```

In [57]:
```
plot(pred_data=gru_pred[:-1], true_data=test_data[:-1], model_name='GRU-{}'.format(gru_u
#Model GRU-10 Lyapunov time = 0.17
#Model GRU-25 Lyapunov time = 0.950
#Model GRU-50 Lyapunov time = 1.35
#Model GRU-100 Lyapunov time = 1.71
```

```
(5000, 3)
(989, 3)
Model GRU-100 Lyapunov time = 1.71
```



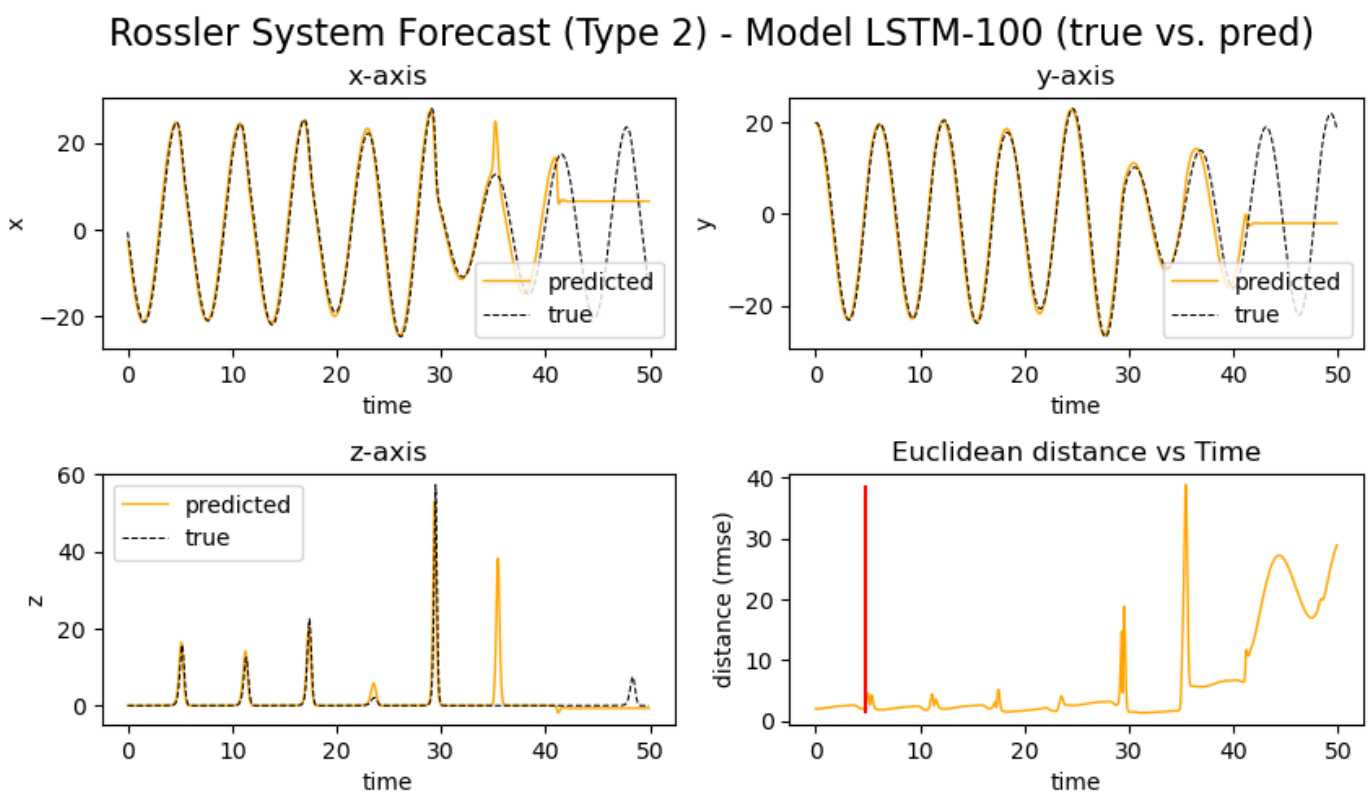


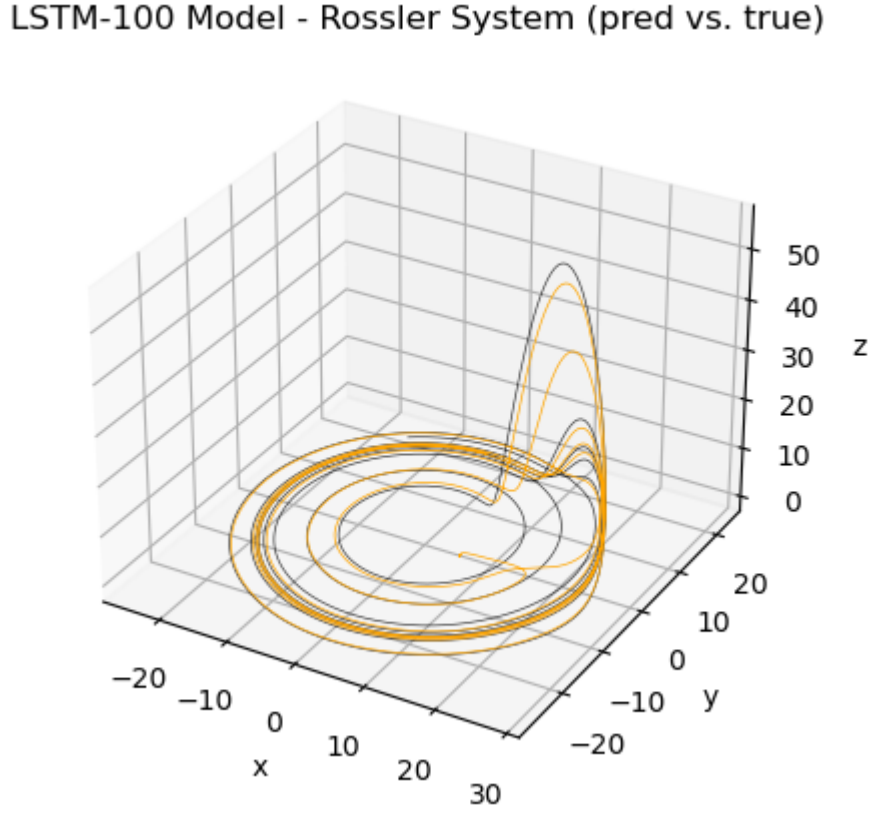
In [58]:
```
lstm_pred=forecast_t2(model=LSTMmodel,test_input=test_input,in_steps=10,samples=5000)
```

```
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 12ms/step
1/1 [==============================] - 0s 11ms/step
1/1 [==============================] - 0s 11ms/step
1/1 [==============================] - 0s 12ms/step
1/1 [==============================] - 0s 13ms/step
init cond terminated
forecast terminated
```

In [59]:
```
plot(pred_data=lstm_pred[:-1], true_data=test_data[:-1], model_name='LSTM-{}'.format(lst
#Model LSTM-10 Lyapunov time = 0.11
#Model LSTM-25 Lyapunov time = 2.68
#Model LSTM-50 Lyapunov time = 1.7
#Model LSTM-100 Lyapunov time = 4.79
```

```
(5000, 3)
(4989, 3)
Model LSTM-100 Lyapunov time = 4.79
```





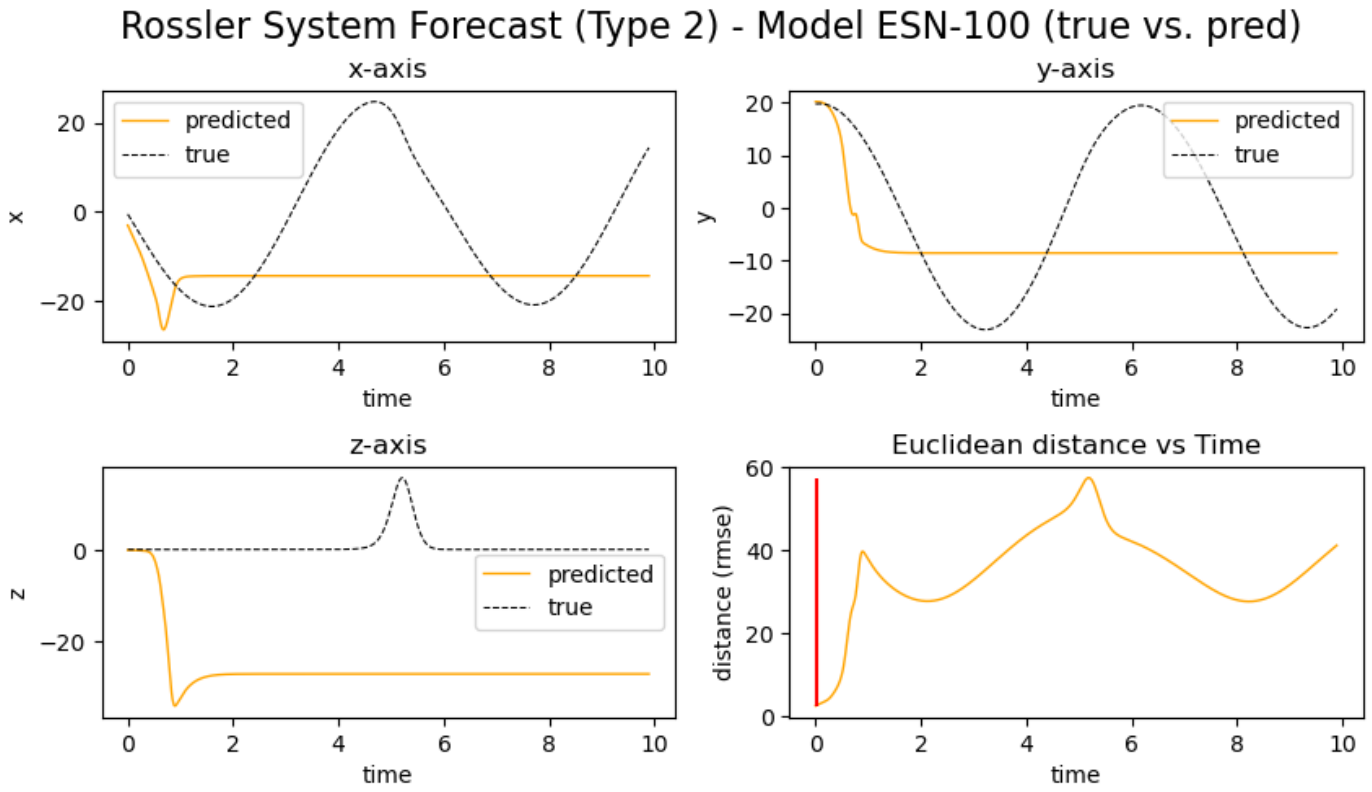
In [60]:
```
esn_pred=forecast_t2(model=ESNmodel,test_input=test_input,in_steps=10,samples=1000)
```

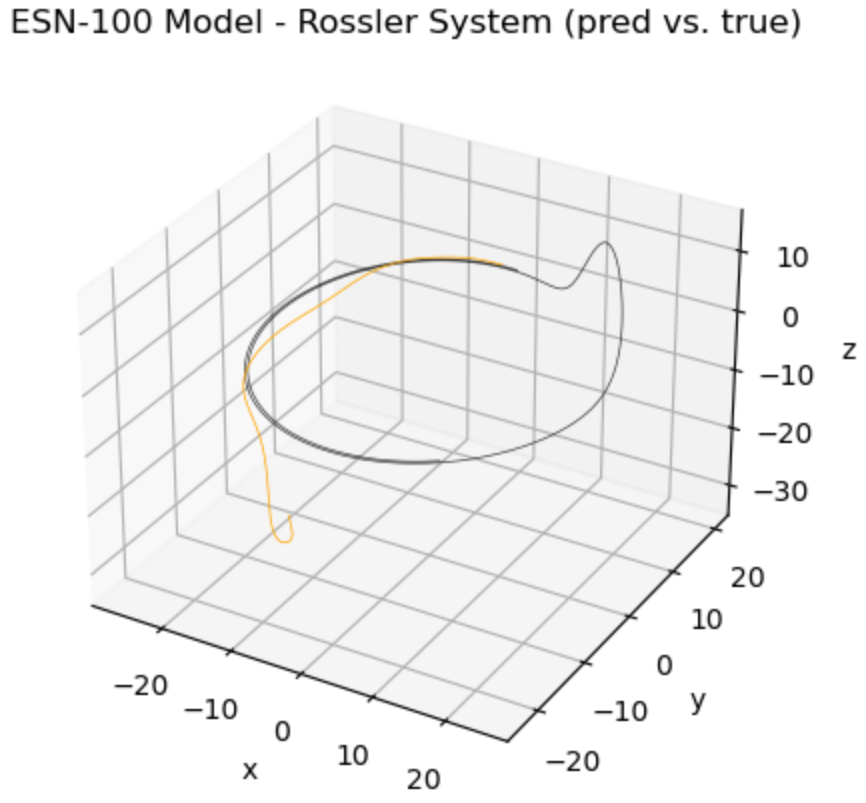
```
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
```

```
init cond terminated
forecast terminated
```

In [61]:
```
plot(pred_data=esn_pred[:-1], true_data=test_data[:-1], model_name='ESN-{}'.format(esn_u
#Model ESN-10 Lyapunov time = 0.04
#Model ESN-25 Lyapunov time = 0.3
#Model ESN-50 Lyapunov time = 2.02
#Model ESN-100 Lyapunov time = 1.65
```

```
(5000, 3)
(989, 3)
Model ESN-100 Lyapunov time = 0.02
```





LSTM trying different activations, recurrent_activations

activation = selu, tanh, relu

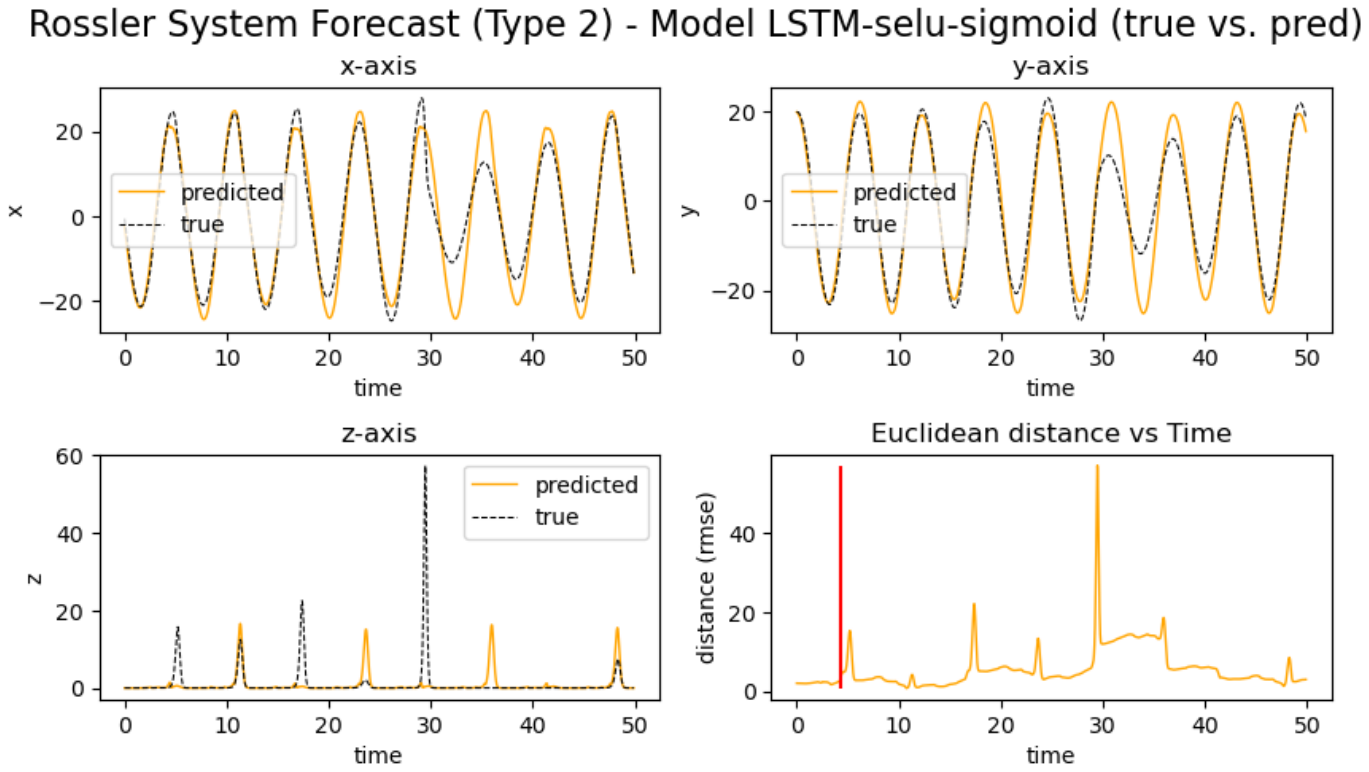
recurrent_activation = selu, tanh, relu

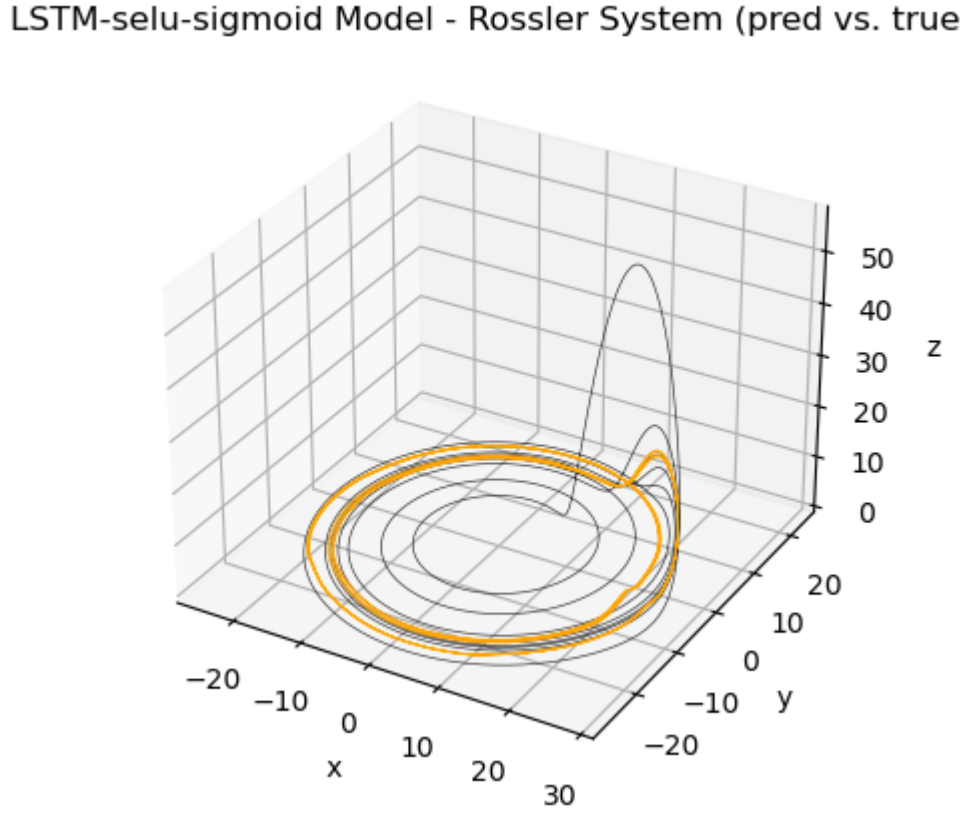
In [66]:
```
input_dim = 3; output_size=3
lstm_units = 100
lr = tf.keras.optimizers.schedules.PiecewiseConstantDecay([50,100],[1e-2,1e-3,1e-4])

LayerLSTMb = layers.RNN(layers.LSTMCell(lstm_units, activation = 'selu',
                        recurrent_activation = 'sigmoid'), input_shape=(None, input_dim)
#sigmoid
LSTMmodelb = tf.keras.Sequential([LayerLSTM,
                                  layers.Dense(output_size)])

LSTMmodelb.compile(loss=tf.keras.losses.Huber(),
                   optimizer=tf.keras.optimizers.Adam(learning_rate=lr),
                   metrics=['mse'])

historyLSTMmodelb = LSTMmodelb.fit(train_input, train_output, verbose=1, epochs=150)

# relu didnt work well
# Model LSTM-tanh-selu Lyapunov time = 4.32
# Model LSTM-selu-sigmoid Lyapunov time = 4.23
```

```
Epoch 1/150
1563/1563 [==============================] - 24s 15ms/step - loss: 0.3082 - mse: 4.7177
Epoch 2/150
1563/1563 [==============================] - 24s 15ms/step - loss: 0.1339 - mse: 2.7148
Epoch 3/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0967 - mse: 1.8557
Epoch 4/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0697 - mse: 1.2057
Epoch 5/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0498 - mse: 0.7540
Epoch 6/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0352 - mse: 0.4580
Epoch 7/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0243 - mse: 0.2632
Epoch 8/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0164 - mse: 0.1476
Epoch 9/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0107 - mse: 0.0809
Epoch 10/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0072 - mse: 0.0453
Epoch 11/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0051 - mse: 0.0266
Epoch 12/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0039 - mse: 0.0175
Epoch 13/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0030 - mse: 0.0115
Epoch 14/150
1563/1563 [==============================] - 23s 15ms/step - loss: 0.0024 - mse: 0.0076
Epoch 15/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0019 - mse: 0.0052
Epoch 16/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0015 - mse: 0.0036
Epoch 17/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0013 - mse: 0.0029
Epoch 18/150
1563/1563 [==============================] - 22s 14ms/step - loss: 0.0011 - mse: 0.0023
Epoch 19/150
1563/1563 [==============================] - 22s 14ms/step - loss: 9.3738e-04 - mse: 0.0
```

In [71]:
```
plot(pred_data=LSTMmodelb_pred[:-1], true_data=test_data[:-1],model_name='LSTM-selu-sigm
```

```
1/1 [==============================] - 0s 12ms/step
1/1 [==============================] - 0s 12ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 12ms/step
1/1 [==============================] - 0s 12ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 12ms/step
init cond terminated
forecast terminated
(5000, 3)
(4989, 3)
Model LSTM-selu-sigmoid Lyapunov time = 4.23
```





ESN hyper-parameter search

In [39]:
```
leaky_list = [0.01,0.1,0.5]
spectral_radius_list = [0.5,0.8,0.95]
sparsity_list = [0.05, 0.1]
neurons = 200
lr = tf.keras.optimizers.schedules.PiecewiseConstantDecay([50,100],[1e-2,1e-3,1e-4])

ESNmodel_list = []; ESNmodel_history = []; ESNmodel_pred = []
for leak in leaky_list:
    for spec_rad in spectral_radius_list:
        for sparse in sparsity_list:
            print('Model unit{} leaky {} spec rad {} saprsity {}'.format(neurons,int(100

            model = tf.keras.Sequential([tfa.layers.ESN(neurons, connectivity = sparse,
                                         layers.Dense(3)])

            model.compile(loss=tf.keras.losses.Huber(),
                          optimizer=tf.keras.optimizers.Adam(learning_rate=lr),
                          metrics=['mse'])

            ESNmodel_list.append(model)

            historyESN = model.fit(train_input, train_output, verbose=1, epochs=150)
            ESNmodel_history.append(historyESN)

            esn_pred=forcast_t2(model=model,test_input=test_input,in_steps=10,samples=10
            ESNmodel_pred.append(esn_pred)
            plot(pred_data=esn_pred[:-1], true_data=test_data[:-1],
                 model_name='ESN-unit{}-leak{}-sr{}-sprs{}'.format(neurons,int(100*leak)
```

```
Model unit200 leaky 1 spec rad 50 saprsity 5
Epoch 1/150
1563/1563 [==============================] - 13s 7ms/step - loss: 4.7879 - mse: 55.1602
Epoch 2/150
1563/1563 [==============================] - 12s 7ms/step - loss: 3.9048 - mse: 39.7554
Epoch 3/150
```