aj563_sf3_code

May 20, 2021

1 Task 1

1.1 Task 1.1

[48]:
```
import numpy as np
from CartPole import *
from sklearn import linear_model
```

[49]:
```
    def start_the_cart(initial_values1, initial_values2=None,␣
↪initial_values3=None, steps=10, remap_angle=False, visual=False,␣
↪display_plots=True, variable = None):

        cp = CartPole(visual=visual)
        cp.cart_location, cp.cart_velocity, cp.pole_angle, cp.pole_velocity =␣
↪initial_values1



        for step in range(steps):
            if visual:
                cp.drawPlot()
            cp.performAction()
            if remap_angle:
                cp.remap_angle()
            inter= [cp.cart_location, cp.cart_velocity, cp.pole_angle, cp.
↪pole_velocity]
            try:
                x_history = np.vstack((x_history, np.array(inter)))
            except:
                x_history = np.vstack((np.array(initial_values1),np.
↪array(inter)))



        x_axis=range(len(x_history))

        if initial_values2:
```

```
            cp.cart_location, cp.cart_velocity, cp.pole_angle, cp.pole_velocity␣
↪= initial_values2



            for step in range(steps):
                if visual: cp.drawPlot()
                cp.performAction()
                if remap_angle: cp.remap_angle()
                inter= [cp.cart_location, cp.cart_velocity, cp.pole_angle, cp.
↪pole_velocity]
                try:
                    y_history = np.vstack((y_history, np.array(inter)))
                except:
                    y_history = np.vstack((np.array(initial_values2),np.
↪array(inter)))

        if initial_values3:
            cp.cart_location, cp.cart_velocity, cp.pole_angle, cp.pole_velocity␣
↪= initial_values3



            for step in range(steps):
                if visual: cp.drawPlot()
                cp.performAction()
                if remap_angle: cp.remap_angle()
                inter= [cp.cart_location, cp.cart_velocity, cp.pole_angle, cp.
↪pole_velocity]
                try:
                    z_history = np.vstack((z_history, np.array(inter)))
                except:
                    z_history = np.vstack((np.array(initial_values3),np.
↪array(inter)))

        if display_plots:
            fig, axs = plt.subplots(2, 2, figsize=(10, 7))

            axs[0,0].plot(x_axis, [x[0] for x in x_history],label='First')
            if initial_values2: axs[0,0].plot(x_axis, [x[0] for x in␣
↪y_history],label='Second')
            if initial_values3: axs[0,0].plot(x_axis, [x[0] for x in␣
↪z_history],label='Third')

            axs[0,1].plot(x_axis, [x[1] for x in x_history])
            if initial_values2: axs[0,1].plot(x_axis, [x[1] for x in y_history])
```

```
        if initial_values3: axs[0,1].plot(x_axis, [x[1] for x in z_history])

        axs[1,0].plot(x_axis, [x[2] for x in x_history])
        if initial_values2: axs[1,0].plot(x_axis, [x[2] for x in y_history])
        if initial_values3: axs[1,0].plot(x_axis, [x[2] for x in z_history])

        axs[1,1].plot(x_axis, [x[3] for x in x_history])
        if initial_values2: axs[1,1].plot(x_axis, [x[3] for x in y_history])
        if initial_values3: axs[1,1].plot(x_axis, [x[3] for x in z_history])

        #Set titles
        axs[0,0].set_title('Cart location')
        axs[0,0].set_xlabel('Steps')
        axs[0,0].set_ylabel('x')

        axs[0,1].set_title('Cart velocity')
        axs[0,1].set_xlabel('Steps')
        axs[0,1].set_ylabel('x_dot')


        axs[1,0].set_title('Pole angle')
        axs[1,0].set_xlabel('Steps')
        axs[1,0].set_ylabel('theta')

        axs[1,1].set_title('Pole velocity')
        axs[1,1].set_xlabel('Steps')
        axs[1,1].set_ylabel('theta_dot')

        if variable: fig.suptitle(('Effect of different initial {} on cart␣
↪dynamics').format(variable),fontsize=16)

        fig.legend()
        fig.tight_layout()


    return x_history[-1]
```

1.1.1 Stable equilibrium

[50]:
```
#history = start_the_cart([0,0,np.pi,1],[0,0,np.pi,5],[0,0,np.
↪pi,10],visual=False,remap_angle=False,variable = 'Pole Velocity')
```

1.1.2 Complete rotation of pendulum

[51]:
```
#history = start_the_cart([0,0,np.pi,15],visual=False,remap_angle=False)
```

1.2 Task 1.2

[52]:
```
initialize = np.array([np.random.uniform(-5,5), np.random.uniform(-10, 10), np.
↪random.uniform(-np.pi,np.pi), np.random.uniform(-15,15)])
#print(initialize)
```

[53]:
```
variable_number={0:'Cart location',1:'Cart velocity',2:'Pole angle',3:'Pole␣
↪velocity'}
```

[54]:
```
def one_step(variable, x_axis_range, x_axis_intervals):



    x = initialize.copy()
    x_axis = np.linspace(x_axis_range[0],x_axis_range[1], x_axis_intervals)
    steps=1
    for i in x_axis:
        x[variable] = i
        y = start_the_cart(x, steps=steps, display_plots=False)

        try:
            final_y = np.vstack((final_y, np.array(y)))
        except:
            final_y = np.array(y)

    fig, axs = plt.subplots(2, 2, figsize=(10, 7))
    axs[0,0].plot(x_axis, [y[0] for y in final_y])

    axs[0,1].plot(x_axis, [y[1] for y in final_y])

    axs[1,0].plot(x_axis, [y[2] for y in final_y])

    axs[1,1].plot(x_axis, [y[3] for y in final_y])


    #Set titles

    axs[0,0].set_xlabel('Initial value of {}'.format(variable_number[variable]))
    axs[0,0].set_ylabel('Cart location')


    axs[0,1].set_xlabel('Initial value of {}'.format(variable_number[variable]))
    axs[0,1].set_ylabel('Cart velocity')


    axs[1,0].set_xlabel('Initial value of {}'.format(variable_number[variable]))
    axs[1,0].set_ylabel('Pole angle')
```

```
    axs[1,1].set_xlabel('Initial value of {}'.format(variable_number[variable]))
    axs[1,1].set_ylabel('Pole velocity')

    fig.suptitle(('Effect of initial {} on cart dynamics after {} step'.
 ↪format(variable_number[variable],steps)),fontsize=16)

    fig.tight_layout()
```

1.2.1 Vary cart location

```
[55]: #one_step(0,[-5,5],15)
```

1.2.2 Vary cart velocity

```
[56]: #one_step(1,[-10,10],15)
```

1.2.3 Vary pole angle

```
[57]: #one_step(2,[-np.pi,np.pi],15)
```

1.2.4 Vary pole velocity

```
[58]: #one_step(3,[-15,15],15)
```

1.2.5 Creating a variable "y", the difference between x

```
[59]: def one_step_difference(variable, x_axis_range, x_axis_intervals):

    x = initialize.copy()
    x_axis = np.linspace(x_axis_range[0],x_axis_range[1], x_axis_intervals)
    steps=1
    for i in x_axis:
        x[variable] = i
        x_t = start_the_cart(x, steps=steps, display_plots=False)
        y = x_t-x

        try:
            final_y = np.vstack((final_y, np.array(y)))
        except:
            final_y = np.array(y)


    fig, axs = plt.subplots(2, 2, figsize=(10, 7))
```

```
    axs[0,0].plot(x_axis, [y[0] for y in final_y])

    axs[0,1].plot(x_axis, [y[1] for y in final_y])

    axs[1,0].plot(x_axis, [y[2] for y in final_y])

    axs[1,1].plot(x_axis, [y[3] for y in final_y])


    #Set titles

    axs[0,0].set_xlabel('Initial value of {}'.format(variable_number[variable]))
    axs[0,0].set_ylabel('Cart location change')


    axs[0,1].set_xlabel('Initial value of {}'.format(variable_number[variable]))
    axs[0,1].set_ylabel('Cart velocity change')


    axs[1,0].set_xlabel('Initial value of {}'.format(variable_number[variable]))
    axs[1,0].set_ylabel('Pole angle change')


    axs[1,1].set_xlabel('Initial value of {}'.format(variable_number[variable]))
    axs[1,1].set_ylabel('Pole velocity change')

    fig.suptitle(('Effect of initial {} on cart dynamics after {} step'.
 ↪format(variable_number[variable],steps)),fontsize=16)

    fig.tight_layout()

    return final_y
```

[60]:
```
def one_step_difference_2(variable, x_axis_range, x_axis_intervals):

    x = initialize.copy()
    x_axis = np.linspace(x_axis_range[0],x_axis_range[1], x_axis_intervals)
    steps=1
    for i in x_axis:
        x[variable] = i
        x_t = start_the_cart(x, steps=steps,␣
 ↪display_plots=False,remap_angle=True)
        y = x_t-x

        try:
            final_y = np.vstack((final_y, np.array(y)))
        except:
```

```
            final_y = np.array(y)


    return x_axis, final_y
```

[61]:
```
def together_plot_1_2():
    x_axis1, c_l_real = one_step_difference_2(0,[-5,5],15)
    x_axis2,c_v_real = one_step_difference_2(1,[-10,10],15)
    x_axis3,p_a_real = one_step_difference_2(2,[-np.pi,np.pi],15)
    x_axis4,p_v_real = one_step_difference_2(3,[-15,15],15)

    fig, axs = plt.subplots(2, 2, figsize=(15, 12))
    for i in range(4):
        axs[0,0].plot(x_axis1, [y[i] for y in c_l_real],label='{}'.
 ↪format(variable_number[i]))





        axs[0,1].plot(x_axis2, [y[i] for y in c_v_real],label='Real c_v')


        axs[1,0].plot(x_axis3, [y[i] for y in p_a_real],label='Real p_a')


        axs[1,1].plot(x_axis4, [y[i] for y in p_v_real],label='Real p_v')



        axs[0,0].legend()


    axs[0,0].set_xlabel('Initial value of cart location')
    axs[0,0].set_ylabel('Cart dynamics changes')

    axs[0,1].set_xlabel('Initial value of cart velocity')
    axs[0,1].set_ylabel('Cart dynamics changes')

    axs[1,0].set_xlabel('Initial value of pole angle')
    axs[1,0].set_ylabel('Cart dynamics changes')

    axs[1,1].set_xlabel('Initial value of pole velocity')
    axs[1,1].set_ylabel('Cart dynamics changes')

    fig.tight_layout()

    plt.show()
```

1.2.6 (i) Scans of single relationships

[62]:
```
#together_plot_1_2()
```

1.2.7 (ii) Contour plots

[63]:
```
different_pairs = [[0,1],[0,2],[0,3],[1,2],[1,3],[2,3]]
axes_ranges = {0 : np.linspace(-5,5,10), 1 : np.linspace(-10,10,10), 2 : np.
↪linspace(-np.pi,np.pi,10), 3 : np.linspace(-15,15,10)}

def axes_for_pairs(index_pair):
    range_of_variables = []
    for index in index_pair:
        range_of_variables.append(axes_ranges[index])
    return range_of_variables

def contours_of_pairs(index_pair, range_of_variables):


    index_1, index_2 = index_pair
    range_1, range_2 = range_of_variables

    initial_grid = np.zeros((len(range_1),len(range_2),4))
    final_grid = np.zeros((len(range_1),len(range_2),4))

    for i,value_1 in enumerate(range_1):
        for j, value_2 in enumerate(range_2):
            x = initialize.copy()
            x[index_1] = value_1
            x[index_2] = value_2
            initial_grid[i,j] = x
            final_grid[i,j] = np.array(start_the_cart(x, steps=1,␣
↪display_plots=False))

    y_grid = final_grid - initial_grid
    y_grid = np.moveaxis(y_grid, -1, 0)

    fig, axs = plt.subplots(2, 2, figsize=(12, 9))

    axs[0,0].contourf(range_1, range_2, y_grid[0].T, vmin=y_grid.min(),␣
↪vmax=y_grid.max())
    axs[0,0].set_title('cart_location')
    axs[0,0].set_xlabel('{} initial value'.format(variable_number[index_1]))
    axs[0,0].set_ylabel('{} initial value'.format(variable_number[index_2]))

    axs[0,1].contourf(range_1, range_2, y_grid[1].T, vmin=y_grid.min(),␣
↪vmax=y_grid.max())
```

```python
    axs[0,1].set_title('cart_velocity')
    axs[0,1].set_xlabel('{} initial value'.format(variable_number[index_1]))
    axs[0,1].set_ylabel('{} initial value'.format(variable_number[index_2]))

    axs[1,0].contourf(range_1, range_2, y_grid[2].T, vmin=y_grid.min(),␣
 ↪vmax=y_grid.max())
    axs[1,0].set_title('pole_angle')
    axs[1,0].set_xlabel('{} initial value'.format(variable_number[index_1]))
    axs[1,0].set_ylabel('{} initial value'.format(variable_number[index_2]))

    axs[1,1].contourf(range_1, range_2, y_grid[3].T, vmin=y_grid.min(),␣
 ↪vmax=y_grid.max())
    axs[1,1].set_title('pole_velocity')
    axs[1,1].set_xlabel('{} initial value'.format(variable_number[index_1]))
    axs[1,1].set_ylabel('{} initial value'.format(variable_number[index_2]))

    fig.tight_layout()
```

[94]:
```python
"""
for indices in different_pairs:
     print('Plots of {} and {}'.
 ↪format(variable_number[indices[0]],variable_number[indices[1]]))
     contours_of_pairs(indices, axes_for_pairs(indices))
     plt.show()
"""
```

[94]:
```
"\nfor indices in different_pairs:\n     print('Plots of {} and
{}'.format(variable_number[indices[0]],variable_number[indices[1]]))\n
contours_of_pairs(indices, axes_for_pairs(indices))\n     plt.show()\n"
```

1.3 Task 1.3

[65]:
```python
def get_xy_pairs(n):

    for iteration in range(n):
        random_point= np.array([np.random.uniform(-5,5), np.random.uniform(-10,␣
 ↪10), np.random.uniform(-np.pi,np.pi), np.random.uniform(-15,15)])
        y = start_the_cart(random_point,steps=1,␣
 ↪remap_angle=True,display_plots=False)
        try:
            final_y = np.vstack((final_y, np.array(y)))
            final_x = np.vstack((final_x, np.array(random_point)))
        except:
            final_y = np.array(y)
            final_x = np.array(random_point)
```

```python
    return final_x,final_y-final_x
```

[66]:
```python
x,y= get_xy_pairs(500)
```

[67]:
```python
#Create train and test sets
proportion = 0.95
number_of_samples = 500
cutoff = int(proportion*number_of_samples)
train_x= x[:cutoff]
test_x=x[cutoff:]

train_y=y[:cutoff]
test_y=y[cutoff:]
```

[68]:
```python
model = linear_model.LinearRegression()
model.fit(train_x,train_y)
n=model.predict(test_x)
```

[69]:
```python
def predict(train_x, test_x, train_y):
    W = np.matmul(np.linalg.pinv(train_x),train_y)
    prediction = np.matmul(test_x,W)
    return prediction,W
```

[70]:
```python
m,W = predict(train_x, test_x, train_y)
```

1.3.1 First plot. Real and predicted plotted against initial

[71]:
```python
def vertical_plot(input,next_step,pred_next_step):

    fig, axs = plt.subplots(2, 2, figsize=(10, 7))

    axs[0,0].scatter([x[0] for x in input],[y[0] for y in␣
 ↪next_step],label='Real value')
    axs[0,0].scatter([x[0] for x in input],[y[0] for y in␣
 ↪pred_next_step],label='Predicted Value')
    axs[0,0].set_xlabel('Cart location initial value')
    axs[0,0].set_ylabel('Cart location final value')
    axs[0,0].legend()

    axs[0,1].scatter([x[1] for x in input],[y[1] for y in next_step])
    axs[0,1].scatter([x[1] for x in input],[y[1] for y in pred_next_step])
    axs[0,1].set_xlabel('Cart velocity initial value')
    axs[0,1].set_ylabel('Cart velocity final value')
```

```
    axs[1,0].scatter([x[2] for x in input],[y[2] for y in next_step])
    axs[1,0].scatter([x[2] for x in input],[y[2] for y in pred_next_step])
    axs[1,0].set_xlabel('Pole angle initial value')
    axs[1,0].set_ylabel('Pole angle final value')

    axs[1,1].scatter([x[3] for x in input],[y[3] for y in next_step])
    axs[1,1].scatter([x[3] for x in input],[y[3] for y in pred_next_step])
    axs[1,1].set_xlabel('Pole velocity initial value')
    axs[1,1].set_ylabel('Pole velocity final value')

    fig.suptitle('Predictions vs Real values after 1 step plotted against␣
↪various initial values')
    fig.tight_layout()
```

[72]:
```
#vertical_plot(test_x,test_y+test_x,m+test_x)
```

[73]:
```
def rmse_calc(A,B):
    squared = (A-B)**2
    cl_mse =0
    cv_mse=0
    pa_mse=0
    pv_mse=0
    for row in squared:

        cl_mse+=row[0]
        cv_mse+=row[1]
        pa_mse+=row[2]
        pv_mse+=row[3]
    cl_mse=(cl_mse/len(squared))**0.5
    cv_mse=(cv_mse/len(squared))**0.5
    pa_mse=(pa_mse/len(squared))**0.5
    pv_mse=(pv_mse/len(squared))**0.5


    return cl_mse,cv_mse,pa_mse,pv_mse
```

[74]:
```
rmse_calc(test_y,n)
```

[74]:
```
(0.11861650064178725, 1.3658805420008469, 2.119815738456754, 3.232319734005139)
```

[75]:
```
rmse_calc(test_y,m)
```

[75]:
```
(0.12023238299058406,
 1.386829387917551,
 2.0989986937181055,
 3.2341823869798754)
```

1.3.2 second plot. This one is real vs predicted only

[76]:
```
def real_vs_predicted(real,predicted):
    fig, axs = plt.subplots(2, 2, figsize=(10, 7))

    axs[0,0].scatter([x[0] for x in real],[y[0] for y in predicted])

    axs[0,0].set_xlabel('Cart location real value')
    axs[0,0].set_ylabel('Cart location predicted value')


    axs[0,1].scatter([x[1] for x in real],[y[1] for y in predicted])
    axs[0,1].set_xlabel('Cart velocity real value')
    axs[0,1].set_ylabel('Cart velocity predicted value')

    axs[1,0].scatter([x[2] for x in real],[y[2] for y in predicted])
    axs[1,0].set_xlabel('Pole angle real value')
    axs[1,0].set_ylabel('Pole angle predicted value')

    axs[1,1].scatter([x[3] for x in real],[y[3] for y in predicted])
    axs[1,1].set_xlabel('Pole velocity real value')
    axs[1,1].set_ylabel('Pole velocity predicted value')

    fig.suptitle('Predictions vs Real values after 1 step')
    fig.tight_layout()
```

[77]:
```
#real_vs_predicted(test_y+test_x,m+test_x)
```

1.3.3 Scans with varying parameters

[78]:
```
def one_step_difference_with_predictions_2(variable, x_axis_range,␣
 ↪x_axis_intervals):
    x = initialize.copy()
    x_axis = np.linspace(x_axis_range[0],x_axis_range[1], x_axis_intervals)
    steps=1

    for i in x_axis:
        x[variable] = i
        x_t = start_the_cart(x, steps=steps,␣
 ↪display_plots=False,remap_angle=True)
        y = x_t-x
        pred = model.predict([x])
        try:
            final_y = np.vstack((final_y, np.array(y)))
            final_pred = np.vstack((final_pred,np.array(pred)))
        except:
            final_y = np.array(y)
```

```
            final_pred = np.array(pred)



    return x_axis,final_y,final_pred
```

[79]:
```
def together_plot():
    x_axis1, c_l_real,c_l_pred =␣
↪one_step_difference_with_predictions_2(0,[-5,5],15)
    x_axis2,c_v_real,c_v_pred =␣
↪one_step_difference_with_predictions_2(1,[-10,10],15)
    x_axis3,p_a_real,p_a_pred = one_step_difference_with_predictions_2(2,[-np.
↪pi,np.pi],15)
    x_axis4,p_v_real,p_v_pred =␣
↪one_step_difference_with_predictions_2(3,[-15,15],15)

    fig, axs = plt.subplots(2, 2, figsize=(17, 11))
    for i in range(4):
        axs[0,0].plot(x_axis1, [y[i] for y in c_l_real],label='Real {}'.
↪format(variable_number[i]))
        axs[0,0].scatter(x_axis1, [y[i] for y in c_l_pred],label='Predicted {}'.
↪format(variable_number[i]))




        axs[0,1].plot(x_axis2, [y[i] for y in c_v_real],label='Real c_v')
        axs[0,1].scatter(x_axis2, [y[i] for y in c_v_pred],label='Predicted␣
↪c_v')

        axs[1,0].plot(x_axis3, [y[i] for y in p_a_real],label='Real p_a')
        axs[1,0].scatter(x_axis3, [y[i] for y in p_a_pred],label='Predicted␣
↪p_a')

        axs[1,1].plot(x_axis4, [y[i] for y in p_v_real],label='Real p_v')
        axs[1,1].scatter(x_axis4, [y[i] for y in p_v_pred],label='Predicted␣
↪p_v')


        axs[0,0].legend()


    axs[0,0].set_xlabel('Initial value of cart location')
    axs[0,0].set_ylabel('Cart dynamics changes')

    axs[0,1].set_xlabel('Initial value of cart velocity')
    axs[0,1].set_ylabel('Cart dynamics changes')
```

```
    axs[1,0].set_xlabel('Initial value of pole angle')
    axs[1,0].set_ylabel('Cart dynamics changes')

    axs[1,1].set_xlabel('Initial value of pole velocity')
    axs[1,1].set_ylabel('Cart dynamics changes')

    fig.tight_layout()


    plt.show()
```

[80]:
```
#together_plot()
```

1.4 Task 1.4

[81]:
```
def future_predictions_from_predictions(initial_conditions,␣
↪time_steps,remap_angle=True):
    final_y=np.array(initial_conditions)
    final_pred=np.array(initial_conditions)

    for i in range(time_steps):
        real= start_the_cart(final_y[i], steps=1,␣
↪display_plots=False,remap_angle=remap_angle)
        pred = np.matmul(final_pred[i],W)
        pred+=final_pred[i]


        final_y = np.vstack((final_y, np.array(real)))
        final_pred = np.vstack((final_pred,np.array(pred)))

    x_axis=np.linspace(0,time_steps,time_steps+1)

    fig, axs = plt.subplots(2, 2, figsize=(12, 10))
    axs[0,0].plot(x_axis, [y[0] for y in final_y],label='Real value')
    axs[0,0].plot(x_axis, [y[0] for y in final_pred],label='Predicted value')
    axs[0,0].legend()

    axs[0,1].plot(x_axis, [y[1] for y in final_y])
    axs[0,1].plot(x_axis, [y[1] for y in final_pred])

    axs[1,0].plot(x_axis, [y[2] for y in final_y])
    axs[1,0].plot(x_axis, [y[2] for y in final_pred])

    axs[1,1].plot(x_axis, [y[3] for y in final_y])
    axs[1,1].plot(x_axis, [y[3] for y in final_pred])
```

```
    #Set titles

    axs[0,0].set_xlabel('Time steps')
    axs[0,0].set_ylabel('Cart location')


    axs[0,1].set_xlabel('Time steps')
    axs[0,1].set_ylabel('Cart velocity')


    axs[1,0].set_xlabel('Time steps')
    axs[1,0].set_ylabel('Pole angle')


    axs[1,1].set_xlabel('Time steps')
    axs[1,1].set_ylabel('Pole velocity')

    fig.suptitle('Predicted (from predictions) vs Real change in cart dynamics␣
↪over time with initial conditions {}'.format(initial_conditions),fontsize=16)

    fig.tight_layout()
```

[82]:
```
#future_predictions_from_predictions([[10,5,2,6]],10,remap_angle=True)
```

[83]:
```
#future_predictions_from_predictions([[0,0,np.pi,15]],10,remap_angle=True)
```

[84]:
```
def future_predictions_from_real(initial_conditions,␣
↪time_steps,remap_angle=True):
    final_y=np.array(initial_conditions)
    final_pred=np.array(initial_conditions)

    for i in range(time_steps):
        real= start_the_cart(final_y[i], steps=1,␣
↪display_plots=False,remap_angle=remap_angle)
        pred = np.matmul(final_y[i],W)
        pred+=final_y[i]


        final_y = np.vstack((final_y, np.array(real)))
        final_pred = np.vstack((final_pred,np.array(pred)))

    x_axis=np.linspace(0,time_steps,time_steps+1)

    fig, axs = plt.subplots(2, 2, figsize=(12, 10))
```

```
    axs[0,0].plot(x_axis, [y[0] for y in final_y],label='Real value')
    axs[0,0].plot(x_axis, [y[0] for y in final_pred],label='Predicted value')
    axs[0,0].legend()

    axs[0,1].plot(x_axis, [y[1] for y in final_y])
    axs[0,1].plot(x_axis, [y[1] for y in final_pred])

    axs[1,0].plot(x_axis, [y[2] for y in final_y])
    axs[1,0].plot(x_axis, [y[2] for y in final_pred])

    axs[1,1].plot(x_axis, [y[3] for y in final_y])
    axs[1,1].plot(x_axis, [y[3] for y in final_pred])


    #Set titles

    axs[0,0].set_xlabel('Time steps')
    axs[0,0].set_ylabel('Cart location')


    axs[0,1].set_xlabel('Time steps')
    axs[0,1].set_ylabel('Cart velocity')


    axs[1,0].set_xlabel('Time steps')
    axs[1,0].set_ylabel('Pole angle')


    axs[1,1].set_xlabel('Time steps')
    axs[1,1].set_ylabel('Pole velocity')

    fig.suptitle('Predicted (from recent dynamics) vs Real change in cart␣
↪dynamics over time with initial conditions {}'.
↪format(initial_conditions),fontsize=16)

    fig.tight_layout()
```

[85]:
```
#future_predictions_from_real([[10,5,2,6]],10,remap_angle=True)
```

[86]:
```
#future_predictions_from_real([[0,0,np.pi,15]],10,remap_angle=True)
```

1.4.1 Here I will train the model on non-remapped angle to see the effect

[87]:
```
def get_xy_pairs_2(n):

    for iteration in range(n):
```

```
        random_point= np.array([np.random.uniform(-5,5), np.random.uniform(-10,␣
↪10), np.random.uniform(-np.pi,np.pi), np.random.uniform(-15,15)])
        y = start_the_cart(random_point,steps=1,␣
↪remap_angle=False,display_plots=False)
        try:
            final_y = np.vstack((final_y, np.array(y)))
            final_x = np.vstack((final_x, np.array(random_point)))
        except:
            final_y = np.array(y)
            final_x = np.array(random_point)


    return final_x,final_y-final_x
```

[88]:
```
x2,y2= get_xy_pairs_2(500)
```

[89]:
```
#Create train and test sets
proportion = 0.95
number_of_samples = 500
cutoff = int(proportion*number_of_samples)
train_x2= x2[:cutoff]
test_x2=x2[cutoff:]

train_y2=y2[:cutoff]
test_y2=y2[cutoff:]
```

[90]:
```
r,B = predict(train_x2, test_x2, train_y2)
```

[91]:
```
print(B)
```

```
[[ 1.57065926e-03  2.09828323e-02 -7.71697493e-04 -1.32795899e-03]
 [ 2.00738062e-01  2.75503518e-03  4.36027474e-03  2.88063797e-02]
 [ 4.49557298e-02  2.36752739e-01  1.63781782e-01  1.19756697e+00]
 [ 2.37050721e-03  2.53349442e-02  1.97746279e-01 -4.52693684e-02]]
```

[92]:
```
np.linalg.eigvals(W)
```

[92]:
```
array([ 0.04308479, -0.04819714, -0.15212995, -0.86964171])
```

[93]:
```
np.linalg.eigvals(B)
```

[93]:
```
array([ 0.56116959,  0.06452381, -0.06584885, -0.43700644])
```

[ ]:
```

```