Gabriel Van Dreel
800886655
September 12, 2021
ECGR 4105-C01

Homework 0

The source code developed for homework 0 was uploaded to the Github repository at the following link:

https://github.com/Gabrielvd616/ECGR4105/tree/main/Homework0

Problem 1

1. The linear models were individually found and recorded in section [1] for each explanatory variable as shown below.

   For x_1,

   $y = 5.99114009 - 2.06191424x_1$

   For x_2,

   $y = 0.75592882 + 0.53878239x_2$

   For x_3,

   $y = 2.81356727 - 0.50011833x_3$

2. The final regression model for x_1 was graphed in section [2] and its loss with respect to the iteration number was graphed in section [3]. The final regression model for x_2 was graphed in section [4] and its loss with respect to the iteration number was graphed in section [5]. The final regression model for x_3 was graphed in section [6] and its loss with respect to the iteration number was graphed in section [7].

3. The explanatory variable x_1 was observed to have the least loss given that its loss function approached a value of 1 and therefore best explained the output y.

4. As the learning rate α was decreased, it was observed to decrease the rate of convergence of the loss functions with respect to the number of gradient descent iterations. Increasing α was also observed to increase the rate of convergence of the loss functions with respect to the number of gradient descent iterations.

Problem 2

1. The best linear model for all three explanatory variables was found for $\alpha = 0.1$ and recorded in section [8] as shown below.

For x_1, x_2, and x_3,

$$y = 5.41374693 - 2.04203017x_1 + 0.56122181x_2 - 0.2921286x_3$$

2. The loss function for x_1, x_2, and x_3 was graphed with respect to the iteration number in section [10].

3. As the learning rate α was decreased, it was observed to decrease the rate of convergence of the loss function with respect to the number of gradient descent iterations. However, increasing α was only observed to increase the rate of convergence of the loss function with respect to the number of gradient descent iterations to a limited extent in that the loss function would not converge if α was too large (e.g. $\alpha = 0.15$).

4. The values of y for new values of (x_1, x_2, x_3) were predicted and recorded in section [9] using the above linear model for $(1, 1, 1)$, $(2, 0, 4)$, and $(3, 2, 1)$ as shown below.

$$y(1, 1, 1) = 3.640809979168586$$

$$y(2, 0, 4) = 0.16117220225655782$$

$$y(3, 2, 1) = 0.11797145619322791$$

In [1]:

```python
# Three basic data science libraries:
# numpy
# pandas
# matplotlib
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

'''
Computes gradient descent model parameters and cost for linear regression

Input parameters:
x : m x n np array of training samples where m is the number of training samples and n
    is the number of model parameters
y : m x 1 np array of training labels
theta : 1 x n np array of model parameters
alpha : scalar value for learning rate
iterations : scalar value for the number of iterations

Output parameters:
theta : 1 x n np array of final model parameters
cost_history : 1 x iterations array of cost values for each iteration
'''


def gradient_descent(x, y, theta, alpha, iterations):
    cost_history = np.zeros(iterations)

    for i in range(iterations):
        # Computes ML model prediction using column vector theta and x values using
        # matrix vector product
        predictions = x.dot(theta)
        errors = np.subtract(predictions, y)
```

```
        sum_delta = (alpha / m) * x.transpose().dot(errors)
        theta = theta - sum_delta

        # Computes value of cost function J for each iteration
        sqr_error = np.square(errors)
        cost_history[i] = 1 / (2 * m) * np.sum(sqr_error)

    return theta, cost_history


# Problem 1
# Initializes the number of iterations and the learning rate alpha
iterations = 1500
alpha = 0.1

# Reads labelled training data
df = pd.read_csv(r'D3.csv')
m = len(df)
x1 = df.values[:, 0]
x2 = df.values[:, 1]
x3 = df.values[:, 2]
y = df.values[:, 3]

# x1 training
x_10 = np.ones((m, 1))
x_11 = x1.reshape(m, 1)
x1 = np.hstack((x_10, x_11))
theta1 = np.zeros(2)
theta1, cost_history1 = gradient_descent(x1, y, theta1, alpha, iterations)

# x2 training
x_20 = np.ones((m, 1))
x_21 = x2.reshape(m, 1)
x2 = np.hstack((x_20, x_21))
theta2 = np.zeros(2)
theta2, cost_history2 = gradient_descent(x2, y, theta2, alpha, iterations)

# x3 training
x_30 = np.ones((m, 1))
x_31 = x3.reshape(m, 1)
x3 = np.hstack((x_30, x_31))
theta3 = np.zeros(2)
theta3, cost_history3 = gradient_descent(x3, y, theta3, alpha, iterations)

# Reports the mutually exclusive linear models found for x1, x2, x3
print('Final value of theta1 =', theta1)
print('Final value of theta2 =', theta2)
print('Final value of theta3 =', theta3)
```

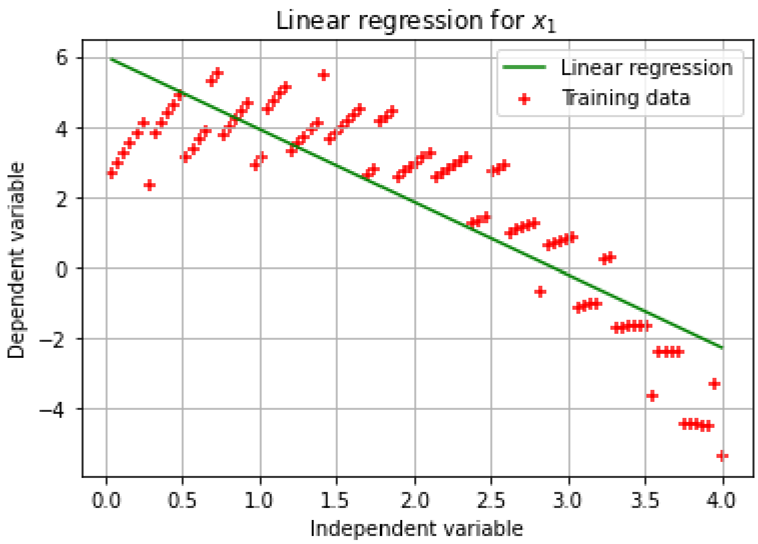
```
Final value of theta1 = [ 5.99114009 -2.06191424]
Final value of theta2 = [0.75592882 0.53878239]
Final value of theta3 = [ 2.81356727 -0.50011833]
```

In [2]:

```
# Plots linear regression for x1
plt.figure(1)
plt.scatter(x1[:, 1], y, color='red', marker='+', label='Training data')
plt.plot(x1[:, 1], x1.dot(theta1), color='green', label='Linear regression')
plt.rcParams['figure.figsize'] = (10, 6)
plt.grid()
plt.xlabel('Independent variable')
```

```
plt.ylabel('Dependent variable')
plt.title('Linear regression for $x_{1}$')
plt.legend()
```

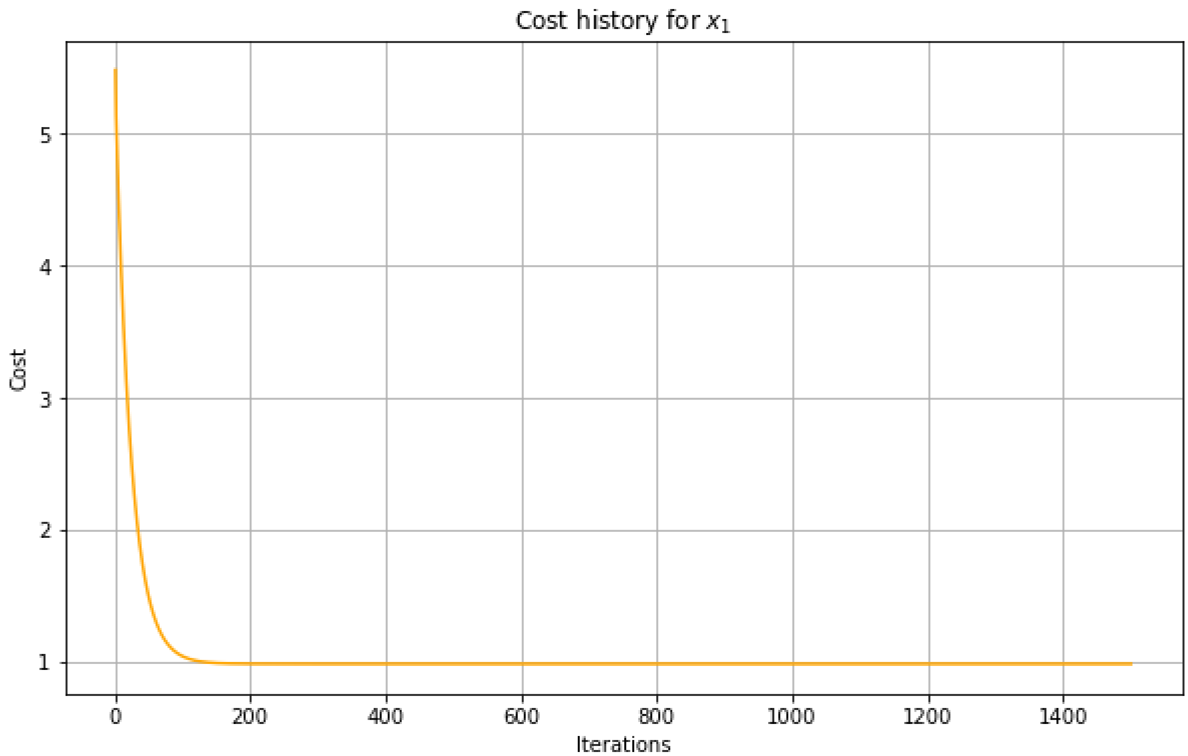
Out[2]: <matplotlib.legend.Legend at 0x8f44970>



In [3]:

```
# Plots cost history for x1
plt.figure(2)
plt.plot(np.linspace(1, iterations, iterations), cost_history1, color='orange',
         label='Cost history')
plt.rcParams['figure.figsize'] = (10, 6)
plt.grid()
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Cost history for $x_{1}$')
```

Out[3]: Text(0.5, 1.0, 'Cost history for x_{1}')

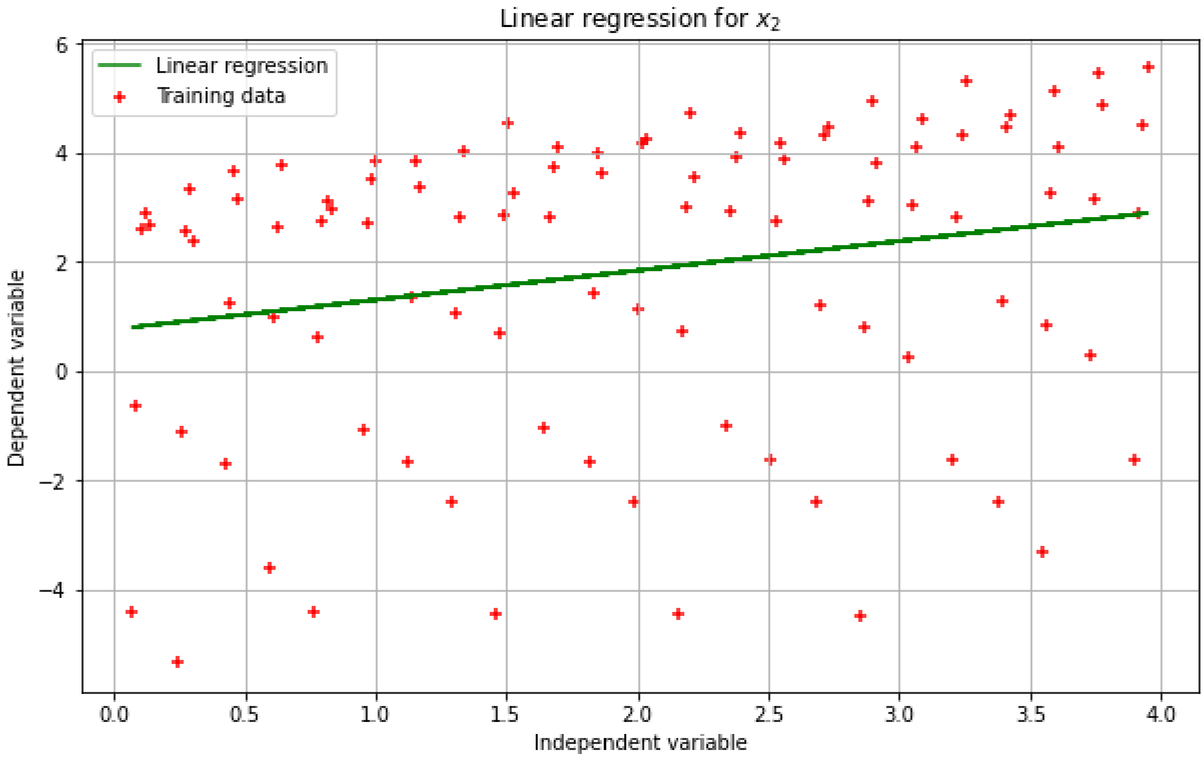


In [4]:

```
# Plots linear regression for x2
plt.figure(3)
plt.scatter(x2[:, 1], y, color='red', marker='+', label='Training data')
plt.plot(x2[:, 1], x2.dot(theta2), color='green', label='Linear regression')
plt.rcParams['figure.figsize'] = (10, 6)
plt.grid()
plt.xlabel('Independent variable')
plt.ylabel('Dependent variable')
plt.title('Linear regression for $x_{2}$')
plt.legend()
```

Out[4]:

```
<matplotlib.legend.Legend at 0x925e0d0>
```

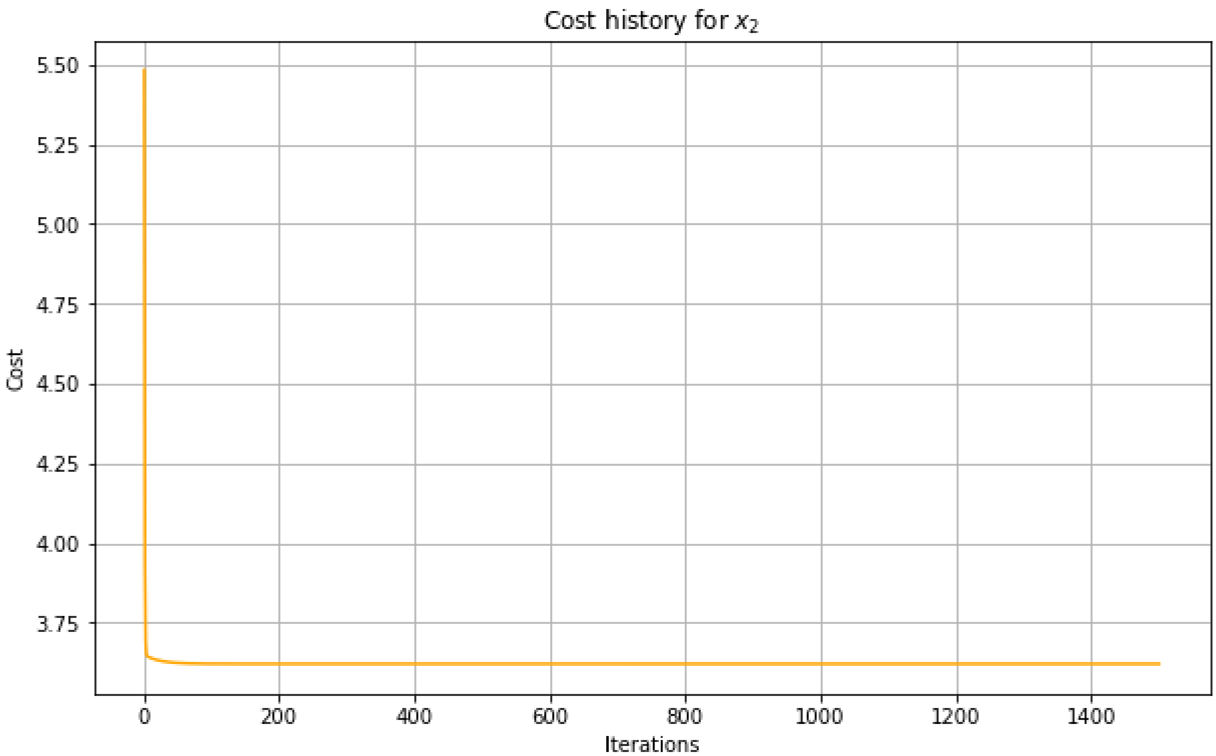


In [5]:

```
# Plots cost history for x2
plt.figure(4)
plt.plot(np.linspace(1, iterations, iterations), cost_history2, color='orange',
         label='Cost history')
plt.rcParams['figure.figsize'] = (10, 6)
plt.grid()
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Cost history for $x_{2}$')
```

Out[5]:

```
Text(0.5, 1.0, 'Cost history for $x_{2}$')
```

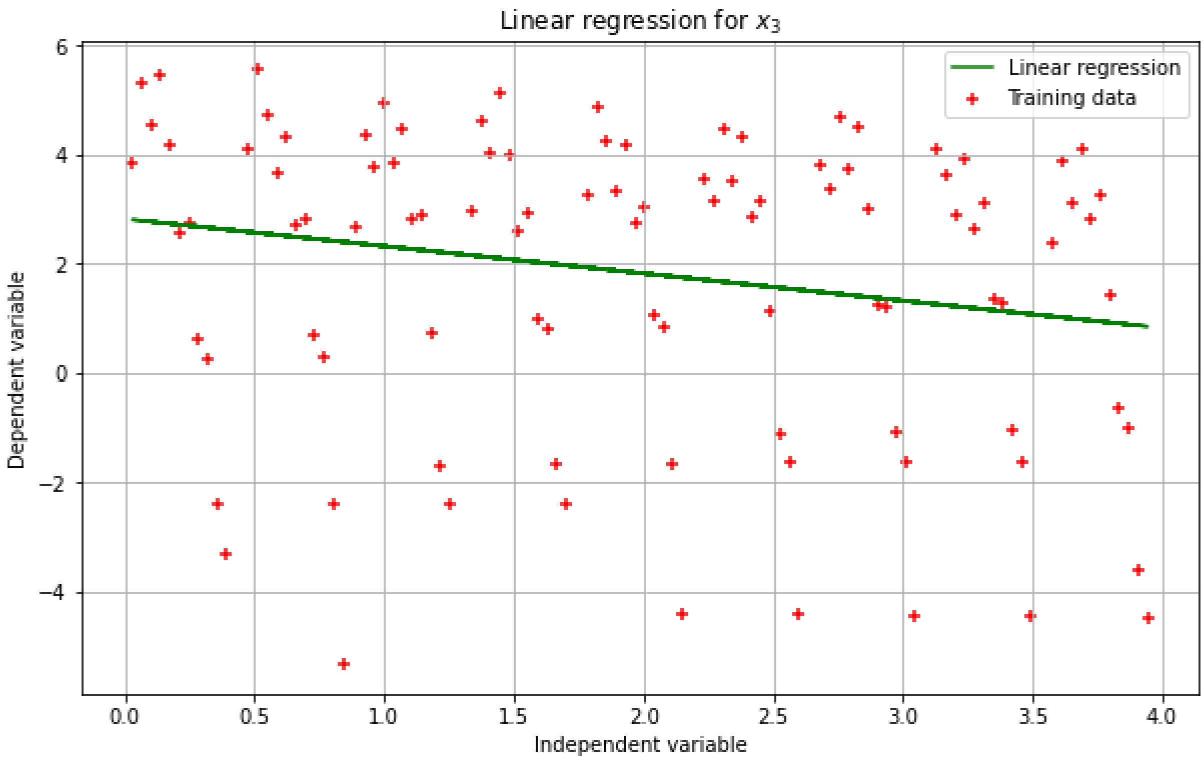


In [6]:

```
# Plots linear regression for x3
plt.figure(5)
plt.scatter(x3[:, 1], y, color='red', marker='+', label='Training data')
plt.plot(x3[:, 1], x3.dot(theta3), color='green', label='Linear regression')
plt.rcParams['figure.figsize'] = (10, 6)
plt.grid()
plt.xlabel('Independent variable')
plt.ylabel('Dependent variable')
plt.title('Linear regression for $x_{3}$')
plt.legend()
```

Out[6]:

```
<matplotlib.legend.Legend at 0x580ef70>
```

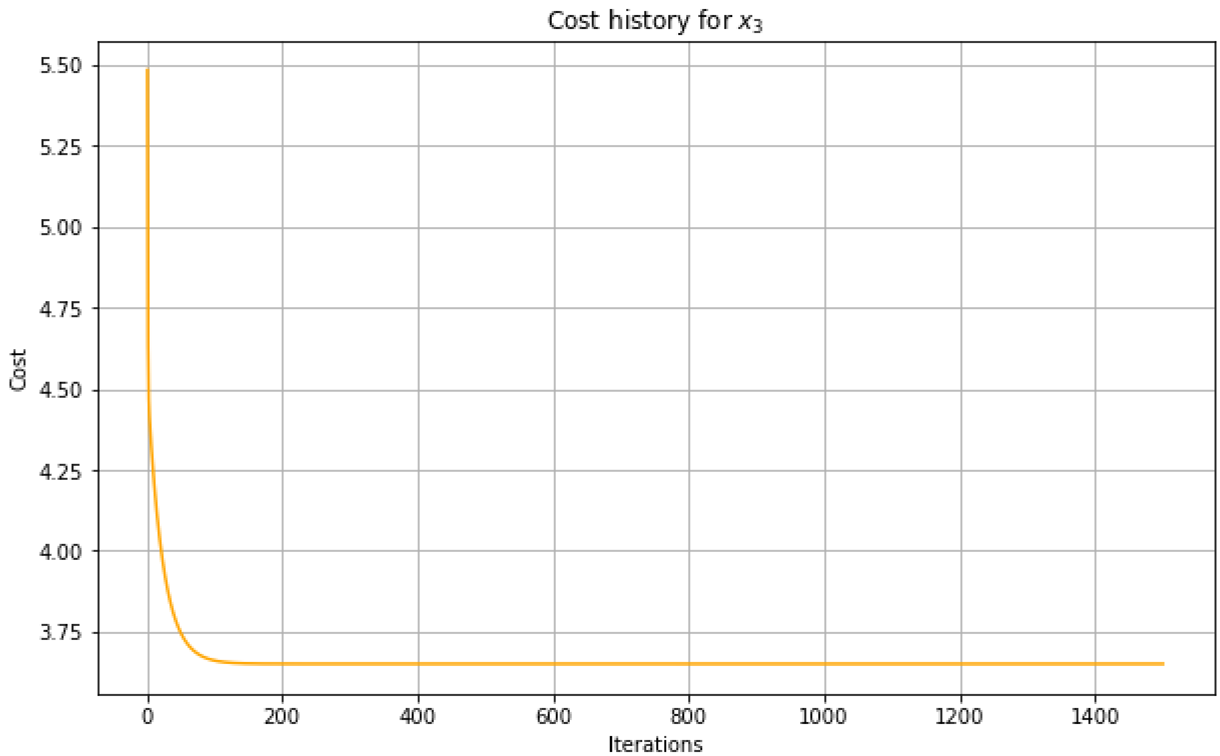


In [7]:

```
# Plots cost history for x3
plt.figure(6)
plt.plot(np.linspace(1, iterations, iterations), cost_history3, color='orange',
         label='Cost history')
plt.rcParams['figure.figsize'] = (10, 6)
plt.grid()
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Cost history for $x_{3}$')
```

Out[7]:

```
Text(0.5, 1.0, 'Cost history for $x_{3}$')
```



In [8]:

```
# Problem 2
# Reformats labelled training data
x = df.values[:, :3]
x = np.hstack((np.ones((m, 1)), x.reshape(m, 3)))

# Retrains ML model with x1, x2, x3
theta = np.zeros(4)
theta, cost_history = gradient_descent(x, y, theta, alpha, iterations)
print('Final value of theta =', theta)
```

```
Final value of theta = [ 5.41374693 -2.04203017  0.56122181 -0.2921286 ]
```

In [9]:

```
# Predicts the value of y for (x1, x2, x3) = (1, 1, 1), (2, 0, 4), (3, 2, 1)
y1 = theta.dot(np.array([1, 1, 1, 1]))
y2 = theta.dot(np.array([1, 2, 0, 4]))
y3 = theta.dot(np.array([1, 3, 2, 1]))
print(y1, y2, y3)
```

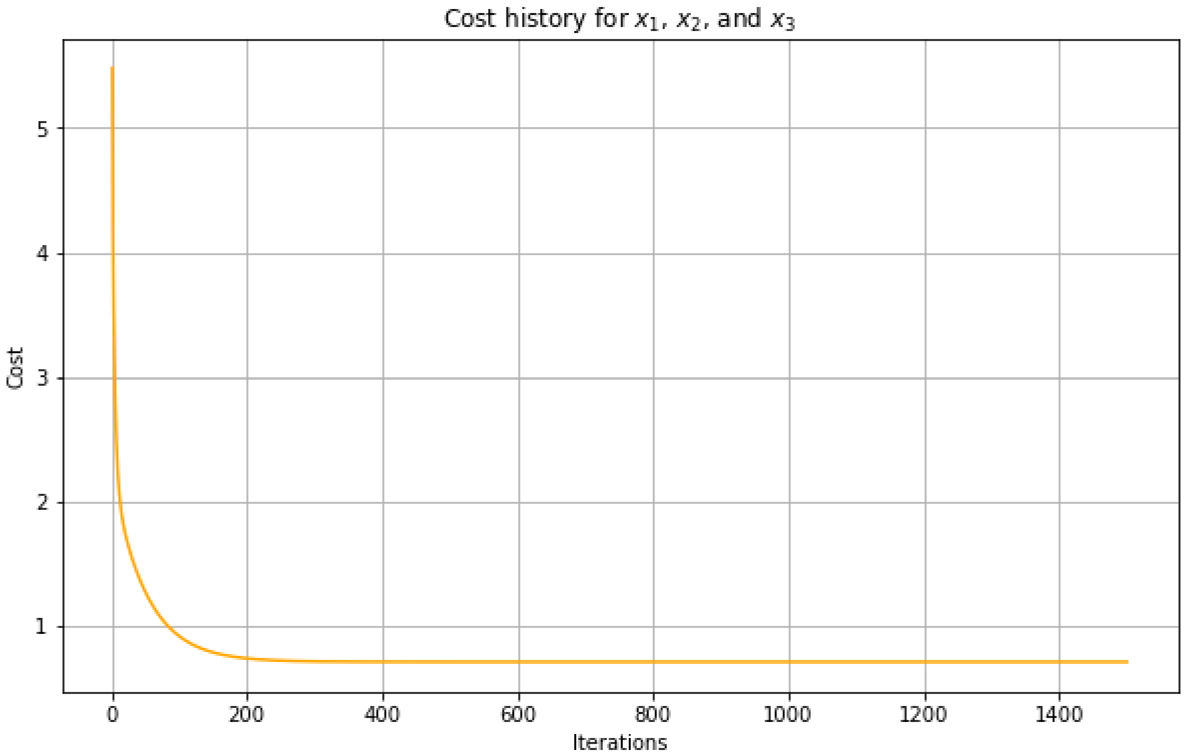
```
3.640809979168586 0.16117220225655782 0.11797145619322791
```

In [10]:

```
# Plots cost history for x1, x2, x3
plt.figure(7)
plt.plot(np.linspace(1, iterations, iterations), cost_history, color='orange',
         label='Cost history')
plt.rcParams['figure.figsize'] = (10, 6)
plt.grid()
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Cost history for $x_{1}$, $x_{2}$, and $x_{3}$')
```

Out[10]:

```
Text(0.5, 1.0, 'Cost history for $x_{1}$, $x_{2}$, and $x_{3}$')
```


Cost history for x_1, x_2, and x_3
Cost
5
4
3
2
1
0
200
400
600
800
1000
1200
1400
Iterations