In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv('D3.csv')
df.head()
M = len(df)
M
```

Out[2]: 99

In [3]:
```
x1 = df.values[:, 0]
x2 = df.values[:, 1]
x3 = df.values[:, 2]
y = df.values[:, 3]
print('x1 = ', x1[: 99])
print('x2 = ', x2[: 99])
print('x3 = ', x3[: 99])
print('y = ', y[: 99])
```

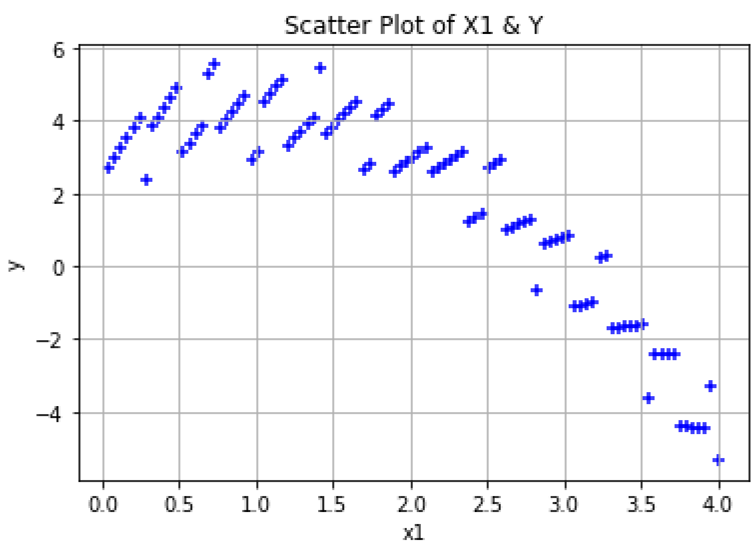
```
x1 =  [0.04040404 0.08080808 0.12121212 0.16161616 0.2020202  0.24242424
 0.28282828 0.32323232 0.36363636 0.4040404  0.44444444 0.48484848
 0.52525253 0.56565657 0.60606061 0.64646465 0.68686869 0.72727273
 0.76767677 0.80808081 0.84848485 0.88888889 0.92929293 0.96969697
 1.01010101 1.05050505 1.09090909 1.13131313 1.17171717 1.21212121
 1.25252525 1.29292929 1.33333333 1.37373737 1.41414141 1.45454545
 1.49494949 1.53535354 1.57575758 1.61616162 1.65656566 1.6969697
 1.73737374 1.77777778 1.81818182 1.85858586 1.8989899  1.93939394
 1.97979798 2.02020202 2.06060606 2.1010101  2.14141414 2.18181818
 2.22222222 2.26262626 2.3030303  2.34343434 2.38383838 2.42424242
 2.46464646 2.50505051 2.54545455 2.58585859 2.62626263 2.66666667
 2.70707071 2.74747475 2.78787879 2.82828283 2.86868687 2.90909091
 2.94949495 2.98989899 3.03030303 3.07070707 3.11111111 3.15151515
 3.19191919 3.23232323 3.27272727 3.31313131 3.35353535 3.39393939
 3.43434343 3.47474747 3.51515152 3.55555556 3.5959596  3.63636364
 3.67676768 3.71717172 3.75757576 3.7979798  3.83838384 3.87878788
 3.91919192 3.95959596 4.        ]
x2 =  [0.13494949 0.82989899 1.52484848 2.21979798 2.91474747 3.60969697
 0.30464646 0.99959596 1.69454545 2.38949495 3.08444444 3.77939394
 0.47434343 1.16929293 1.86424242 2.55919192 3.25414141 3.94909091
 0.6440404  1.3389899  2.03393939 2.72888889 3.42383838 0.11878788
 0.81373737 1.50868687 2.20363636 2.89858586 3.59353535 0.28848485
 0.98343434 1.67838384 2.37333333 3.06828283 3.76323232 0.45818182
 1.15313131 1.84808081 2.5430303  3.2379798  3.93292929 0.62787879
 1.32282828 2.01777778 2.71272727 3.40767677 0.10262626 0.79757576
 1.49252525 2.18747475 2.88242424 3.57737374 0.27232323 0.96727273
 1.66222222 2.35717172 3.05212121 3.74707071 0.4420202  1.1369697
 1.83191919 2.52686869 3.22181818 3.91676768 0.61171717 1.30666667
 2.00161616 2.69656566 3.39151515 0.08646465 0.78141414 1.47636364
 2.17131313 2.86626263 3.56121212 0.25616162 0.95111111 1.64606061
 2.3410101  3.0359596  3.73090909 0.42585859 1.12080808 1.81575758
 2.51070707 3.20565657 3.90060606 0.59555556 1.29050505 1.98545455
 2.68040404 3.37535354 0.07030303 0.76525253 1.46020202 2.15515152
 2.85010101 3.54505051 0.24      ]
x3 =  [0.88848485 1.3369697  1.78545455 2.23393939 2.68242424 3.13090909
 3.57939394 0.02787879 0.47636364 0.92484848 1.37333333 1.82181818
 2.27030303 2.71878788 3.16727273 3.61575758 0.06424242 0.51272727
 0.96121212 1.40969697 1.85818182 2.30666667 2.75515152 3.20363636
```

```
3.65212121 0.10060606 0.54909091 0.99757576 1.44606061 1.89454545
2.3430303  2.79151515 3.24       3.68848485 0.1369697  0.58545455
1.03393939 1.48242424 1.93090909 2.37939394 2.82787879 3.27636364
3.72484848 0.17333333 0.62181818 1.07030303 1.51878788 1.96727273
2.41575758 2.86424242 3.31272727 3.76121212 0.20969697 0.65818182
1.10666667 1.55515152 2.00363636 2.45212121 2.90060606 3.34909091
3.79757576 0.24606061 0.69454545 1.1430303  1.59151515 2.04
2.48848485 2.9369697  3.38545455 3.83393939 0.28242424 0.73090909
1.17939394 1.62787879 2.07636364 2.52484848 2.97333333 3.42181818
3.87030303 0.31878788 0.76727273 1.21575758 1.66424242 2.11272727
2.56121212 3.00969697 3.45818182 3.90666667 0.35515152 0.80363636
1.25212121 1.70060606 2.14909091 2.59757576 3.04606061 3.49454545
3.9430303  0.39151515 0.84      ]
y =  [ 2.6796499   2.96848981  3.25406475  3.53637472  3.81541972  4.09119974
  2.36371479  3.83296487  4.09894997  4.3616701   4.62112526  4.87731544
  3.13024065  3.37990089  3.62629616  3.86942645  5.30929177  5.54589212
  3.77922749  4.00929789  4.23610332  4.45964378  4.67991926  2.89692977
  3.1106753   4.52115587  4.72837146  4.93232207  5.13300772  3.33042839
  3.52458409  3.71547481  3.90310057  4.08746135  5.46855715  3.64638799
  3.82095385  3.99225473  4.16029065  4.32506159  4.48656756  2.64480856
  2.79978458  4.15149563  4.29994171  4.44512281  2.58703894  2.7256901
  2.86107628  2.9931975   3.12205373  3.247645    2.56997129  2.68903261
  2.80482896  2.91736034  3.02662674  3.13262817  1.23536462  1.3348361
  1.43104261  2.72398415  2.81366071  2.9000723   0.98321892  1.06310057
  1.13971724  1.21306894  1.28315566 -0.65002258  0.6135342   0.673826
  0.73085284  0.7846147   0.83511159 -1.1176565  -1.07368956 -1.03298759
 -0.99555059  0.23862143  0.26952848 -1.70282944 -1.67845234 -1.6573402
 -1.63949305 -1.62491086 -1.61359365 -3.60554141 -2.40075414 -2.39923185
 -2.40097453 -2.40598218 -4.41425481 -4.4257924  -4.44059497 -4.45866252
 -4.47999504 -3.30459253 -5.33245499]
```

In [4]:

```python
plt.scatter(x1, y, color='blue', marker = '+')
plt.grid()
plt.rcParams["figure.figsize"] = (10,6)
plt.xlabel('x1')
plt.ylabel('y')
plt.title('Scatter Plot of X1 & Y')
```

Out[4]: `Text(0.5, 1.0, 'Scatter Plot of X1 & Y')`

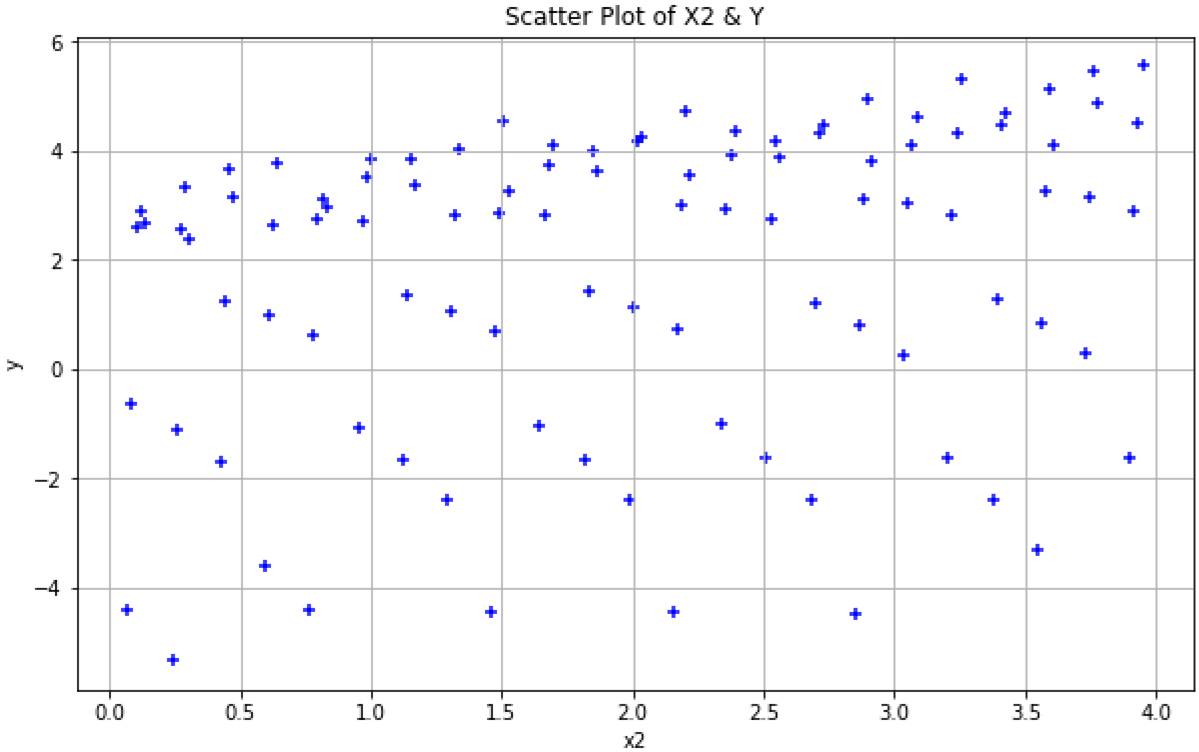


In [5]:

```python
plt.scatter(x2, y, color='blue', marker = '+')
plt.grid()
plt.rcParams["figure.figsize"] = (10,6)
```

```
plt.xlabel('x2')
plt.ylabel('y')
plt.title('Scatter Plot of X2 & Y')
```

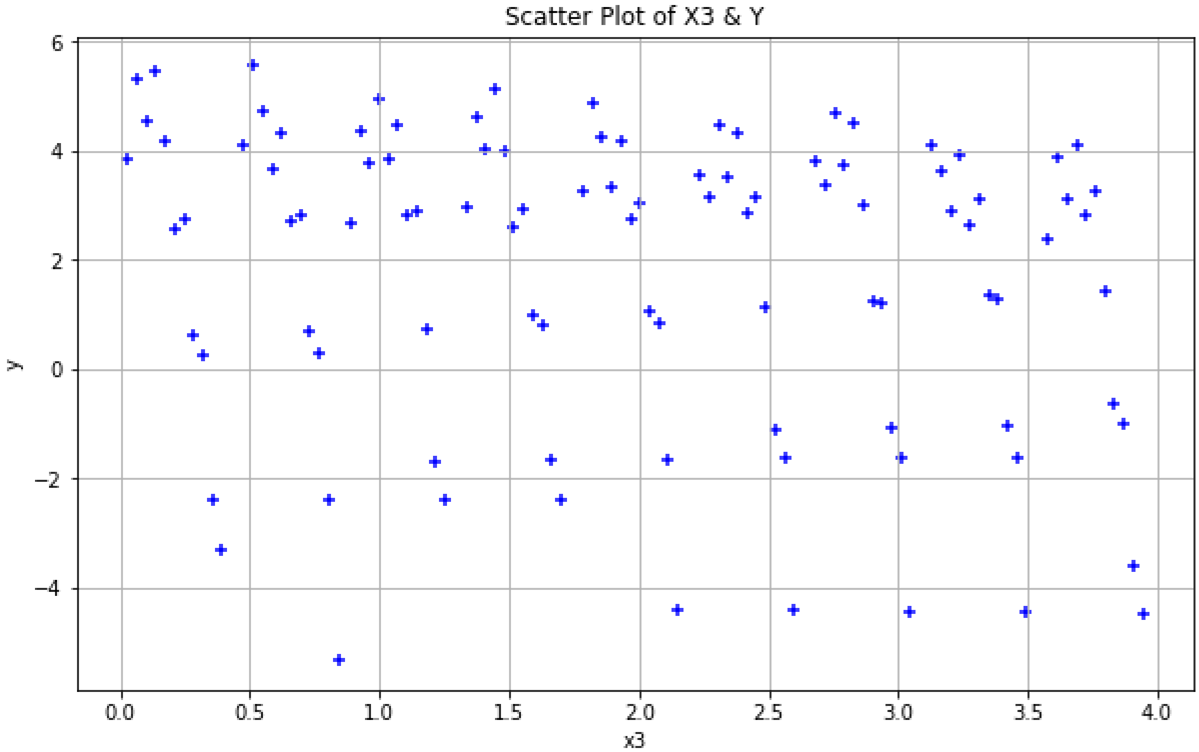
Out[5]: Text(0.5, 1.0, 'Scatter Plot of X2 & Y')



In [6]:

```
plt.scatter(x3, y, color='blue', marker = '+')
plt.grid()
plt.rcParams["figure.figsize"] = (10,6)
plt.xlabel('x3')
plt.ylabel('y')
plt.title('Scatter Plot of X3 & Y')
```

Out[6]: Text(0.5, 1.0, 'Scatter Plot of X3 & Y')



In [7]:
```python
x_0 = np.ones((M, 1))
x_0[:5]
```

Out[7]:
```
array([[1.],
       [1.],
       [1.],
       [1.],
       [1.]])
```

In [8]:
```python
x_1 = x1.reshape(M, 1)
x_1[:10]
```

Out[8]:
```
array([[0.04040404],
       [0.08080808],
       [0.12121212],
       [0.16161616],
       [0.2020202 ],
       [0.24242424],
       [0.28282828],
       [0.32323232],
       [0.36363636],
       [0.4040404 ]])
```

In [9]:
```python
x1 = np.hstack((x_0, x_1))
x1[:5]
```

Out[9]:
```
array([[1.        , 0.04040404],
       [1.        , 0.08080808],
       [1.        , 0.12121212],
       [1.        , 0.16161616],
       [1.        , 0.2020202 ]])
```

In [10]:
```python
theta = np.zeros(2)
```

```
theta
```

Out[10]: array([0., 0.])

In [11]:

```
def compute_cost(x1, y, theta):
    predictions = x1.dot(theta)
    errors = np.subtract(predictions, y)
    sqrErrors = np.square(errors)
    J = 1 / (2 * M) * np.sum(sqrErrors)
    return J
```

In [12]:

```
cost1 = compute_cost(x1, y, theta)
print('the cost for given values of theta_0 and theta_1= ', cost1)
```

```
the cost for given values of theta_0 and theta_1=  5.483015861682611
```

In [13]:

```
def gradient_descent(x1, y, theta, alpha, iterations):
    cost_history = np.zeros(iterations)
    for i in range(iterations):
         predictions = x1.dot(theta)
         errors = np.subtract(predictions, y)
         sum_delta = (alpha / M) * x1.transpose().dot(errors);
         theta = theta - sum_delta;
         cost_history[i] = compute_cost(x1, y, theta)
    return theta, cost_history
```

In [14]:

```
theta = [0., 0.]
iterations = 1500;
alpha = 0.01;
```

In [15]:

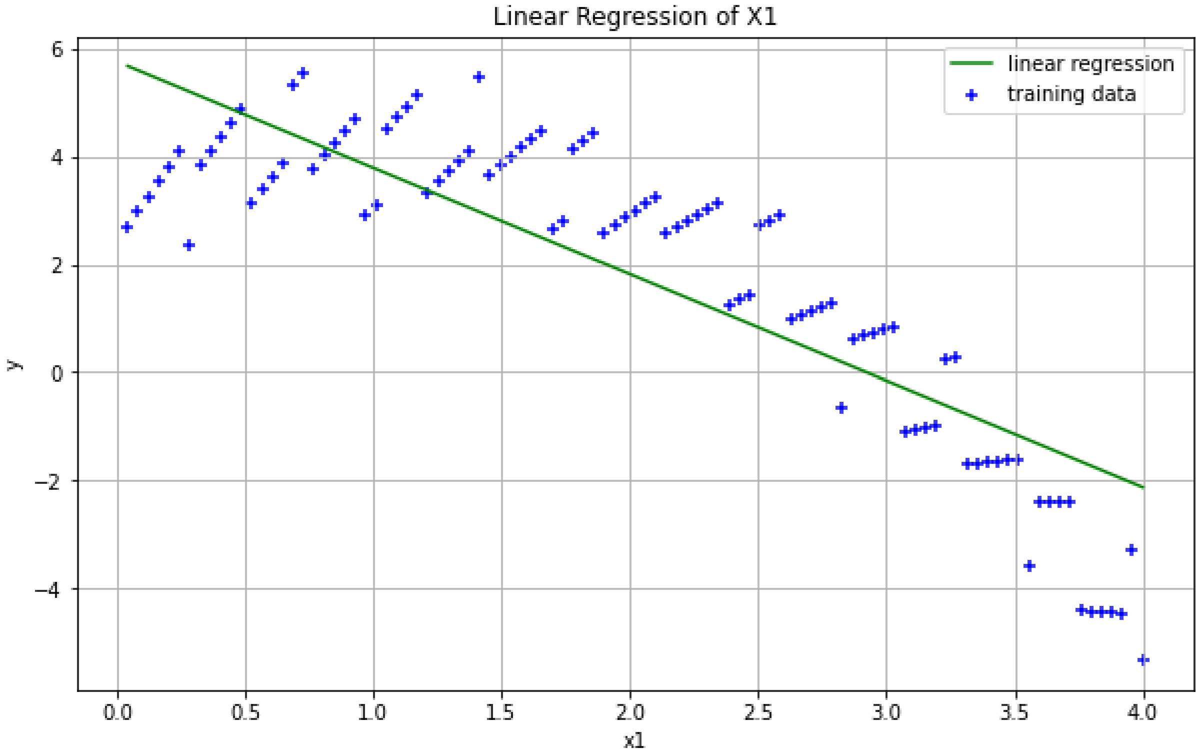
```
theta, cost_history = gradient_descent(x1, y, theta, alpha, iterations)
print('final value of theta =', theta)
print('cost_history = ', cost_history)
```

```
final value of theta = [ 5.75752967 -1.97114532]
cost_history =  [5.4416155  5.40304386 5.36697031 ... 0.98927932 0.98925005 0.98922091]
```

In [16]:

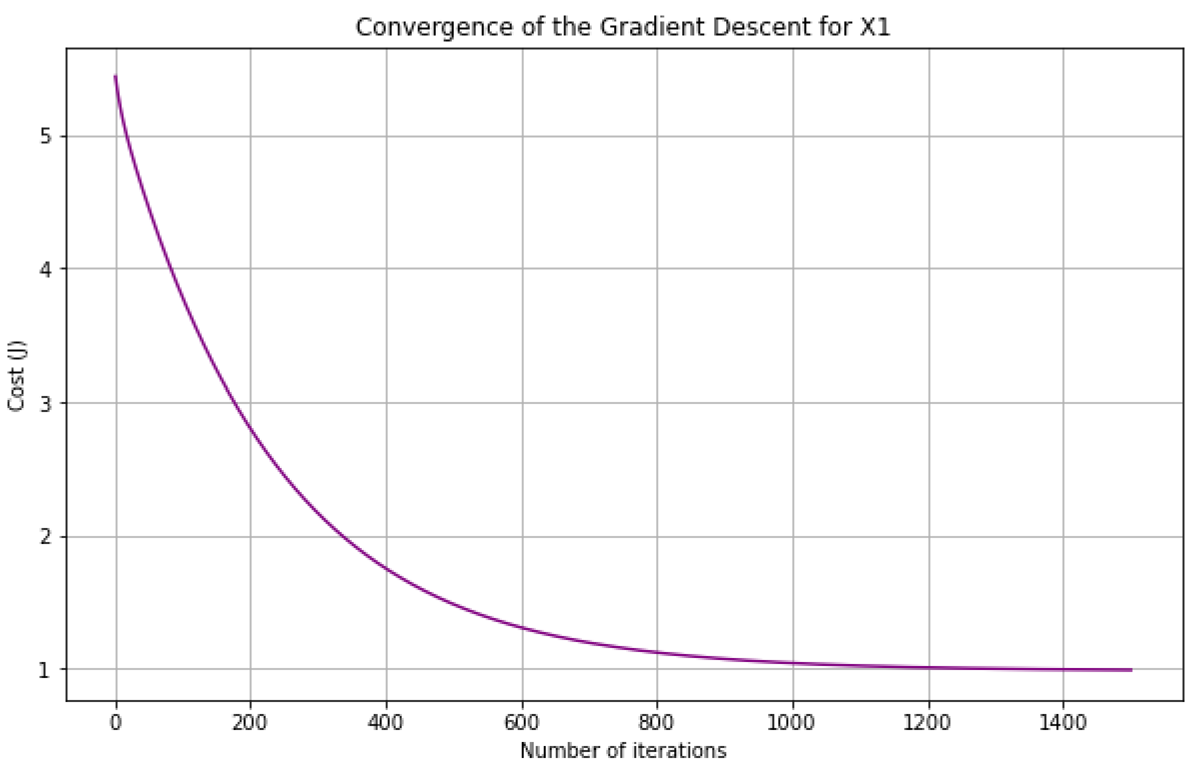
```
plt.scatter(x1[:,1], y, color='blue', marker='+', label='training data')
plt.plot(x1[:,1], x1.dot(theta), color='green', label='linear regression')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('x1')
plt.ylabel('y')
plt.title('Linear Regression of X1')
plt.legend()
```

Out[16]: <matplotlib.legend.Legend at 0x1c35b39fb50>



In [17]:
```
plt.plot(range(1,iterations+1),cost_history,color='purple')
plt.rcParams["figure.figsize"]=(10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of the Gradient Descent for X1')
```

Out[17]: Text(0.5, 1.0, 'Convergence of the Gradient Descent for X1')



In [18]:
```
x_0 = np.ones((M, 1))
x_0[:5]
```

Out[18]:
```
array([[1.],
       [1.],
       [1.],
       [1.],
       [1.]])
```

In [19]:
```
x_2 = x2.reshape(M, 1)
x_2[:5]
```

Out[19]:
```
array([[0.13494949],
       [0.82989899],
       [1.52484848],
       [2.21979798],
       [2.91474747]])
```

In [20]:
```
theta = np.zeros(2)
theta
```

Out[20]:
```
array([0., 0.])
```

In [21]:
```
x2 = np.hstack((x_0, x_2))
x2[:5]
```

Out[21]:
```
array([[1.        , 0.13494949],
       [1.        , 0.82989899],
       [1.        , 1.52484848],
       [1.        , 2.21979798],
       [1.        , 2.91474747]])
```

In [22]:
```
def compute_cost2(x2, y, theta):
    predictions = x2.dot(theta)
    errors = np.subtract(predictions, y)
    sqrErrors = np.square(errors)
    J = 1 / (2 * M) * np.sum(sqrErrors)
    return J
```

In [23]:
```
cost2 = compute_cost2(x2, y, theta)
print('the cost for given values of theta_0 and theta_1= ', cost2)
```

```
the cost for given values of theta_0 and theta_1=  5.483015861682611
```

In [24]:
```
def gradient_descent2(x2, y, theta, alpha, iterations):
    cost_history2 = np.zeros(iterations)
    for i in range(iterations):
        predictions = x2.dot(theta)
        errors = np.subtract(predictions, y)
        sum_delta = (alpha / M) * x2.transpose().dot(errors);
        theta = theta - sum_delta;
        cost_history2[i] = compute_cost(x2, y, theta)
    return theta, cost_history2
```

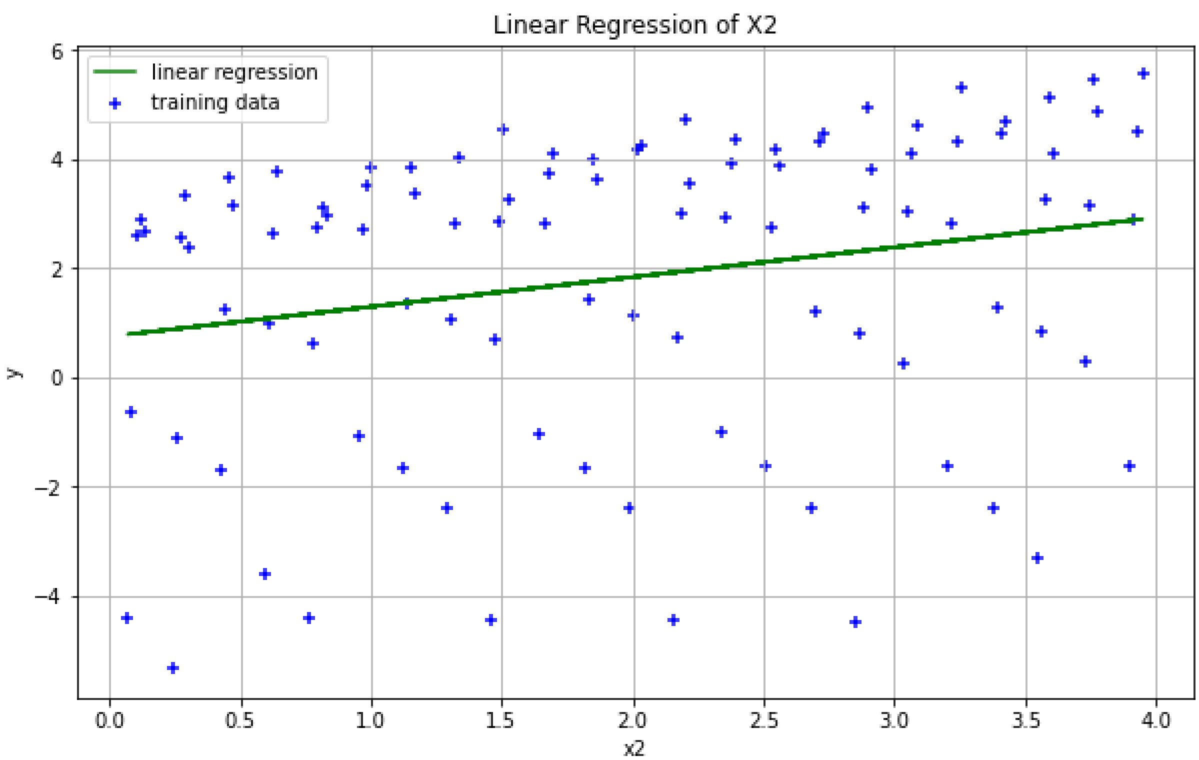
In [25]:
```
theta = [0., 0.]
iterations = 1500;
alpha = 0.01;
```

In [26]:
```
theta, cost_history2= gradient_descent2(x2, y, theta, alpha, iterations)
print('final value of theta =', theta)
print('cost_history = ', cost_history2)
```

```
final value of theta = [0.7392744 0.5453018]
cost_history =  [5.2669085  5.07623409 4.90799786 ... 3.6201926  3.62019244 3.62019228]
```

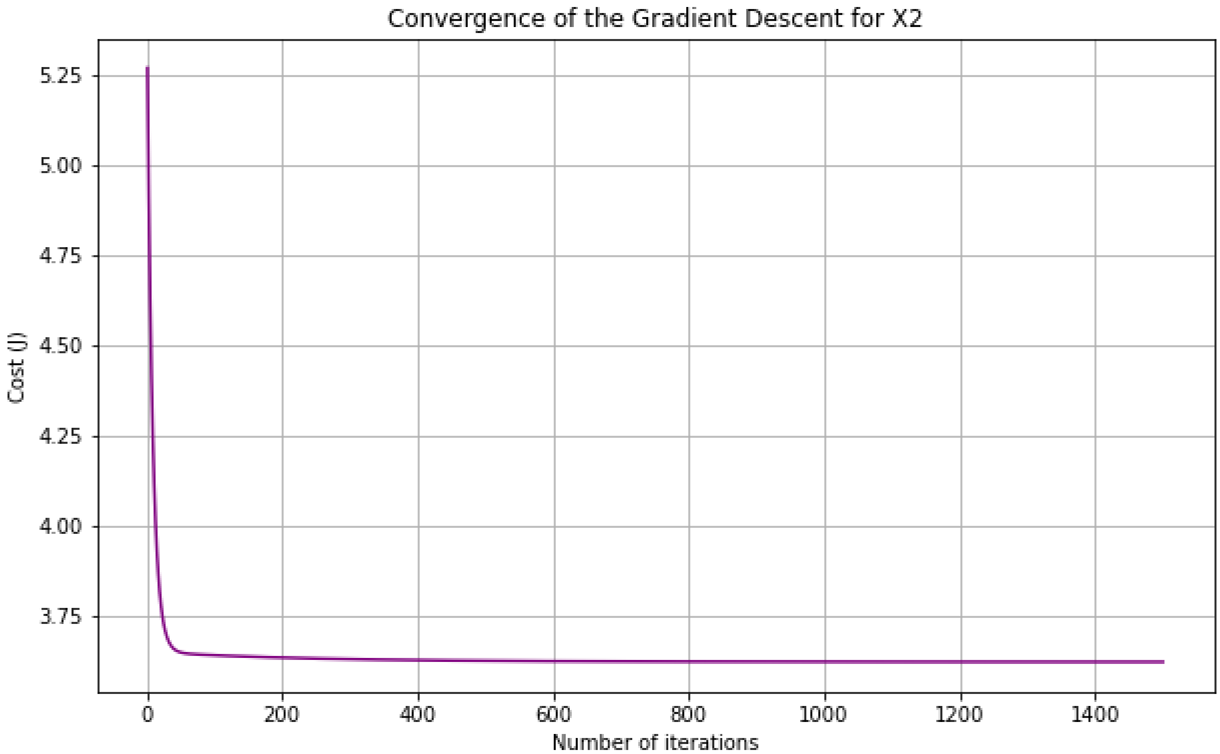
In [27]:
```
plt.scatter(x2[:,1], y, color='blue', marker='+', label='training data')
plt.plot(x2[:,1], x2.dot(theta), color='green', label='linear regression')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('x2')
plt.ylabel('y')
plt.title('Linear Regression of X2')
plt.legend()
```

Out[27]: `<matplotlib.legend.Legend at 0x1c35b34a040>`



In [28]:
```
plt.plot(range(1,iterations+1),cost_history2,color='purple')
plt.rcParams["figure.figsize"]=(10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of the Gradient Descent for X2')
```

Out[28]: `Text(0.5, 1.0, 'Convergence of the Gradient Descent for X2')`



In [29]:
```
x_0 = np.ones((M, 1))
x_0[:5]
```

Out[29]:
```
array([[1.],
       [1.],
       [1.],
       [1.],
       [1.]])
```

In [30]:
```
x_3 = x3.reshape(M, 1)
x_3[:10]
```

Out[30]:
```
array([[0.88848485],
       [1.3369697 ],
       [1.78545455],
       [2.23393939],
       [2.68242424],
       [3.13090909],
       [3.57939394],
       [0.02787879],
       [0.47636364],
       [0.92484848]])
```

In [31]:
```
x3 = np.hstack((x_0, x_3))
x3[:5]
```

Out[31]:
```
array([[1.        , 0.88848485],
       [1.        , 1.3369697 ],
       [1.        , 1.78545455],
       [1.        , 2.23393939],
       [1.        , 2.68242424]])
```

In [32]:
```
theta = np.zeros(2)
```

```
theta
```

Out[32]: array([0., 0.])

In [33]:

```
def compute_cost3(x3, y, theta):
    predictions = x3.dot(theta)
    errors = np.subtract(predictions, y)
    sqrErrors = np.square(errors)
    J = 1 / (2 * M) * np.sum(sqrErrors)
    return J
```

In [34]:

```
cost = compute_cost3(x3, y, theta)
print('the cost for given values of theta_0 and theta_1= ', cost)
```

```
the cost for given values of theta_0 and theta_1=  5.483015861682611
```

In [35]:

```
def gradient_descent3(x3, y, theta, alpha, iterations):
    cost_history3 = np.zeros(iterations)
    for i in range(iterations):
        predictions = x3.dot(theta)
        errors = np.subtract(predictions, y)
        sum_delta = (alpha / M) * x3.transpose().dot(errors);
        theta = theta - sum_delta;
        cost_history3[i] = compute_cost3(x3, y, theta)
    return theta, cost_history3
```

In [36]:

```
theta = [0., 0.]
iterations = 1500;
alpha = 0.01;
```

In [37]:

```
theta, cost_history3 = gradient_descent(x3, y, theta, alpha, iterations)
print('final value of theta =', theta)
print('cost_history = ', cost_history3)
```

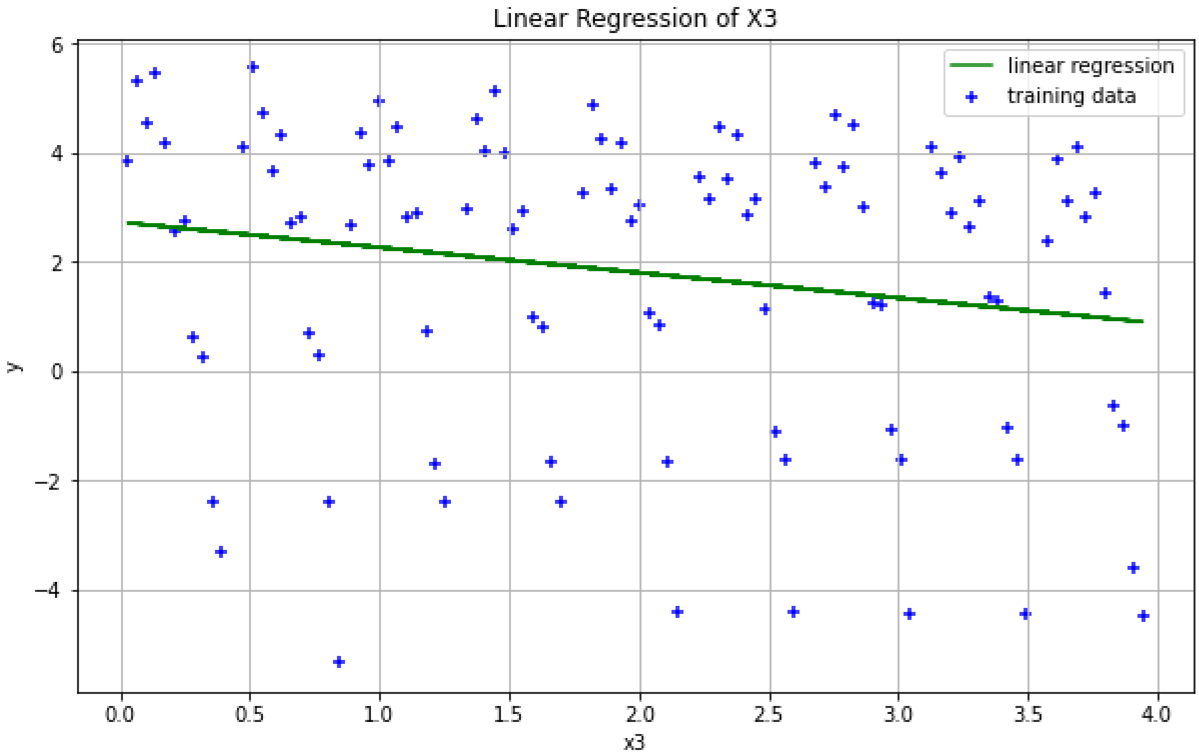
```
final value of theta = [ 2.71943299 -0.46300206]
cost_history =  [5.366643   5.26340773 5.17178032 ... 3.65144217 3.65143712 3.6514321 ]
```

In [38]:

```
plt.scatter(x3[:,1], y, color='blue', marker='+', label='training data')
plt.plot(x3[:,1], x3.dot(theta), color='green', label='linear regression')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('x3')
plt.ylabel('y')
plt.title('Linear Regression of X3')
plt.legend()
```

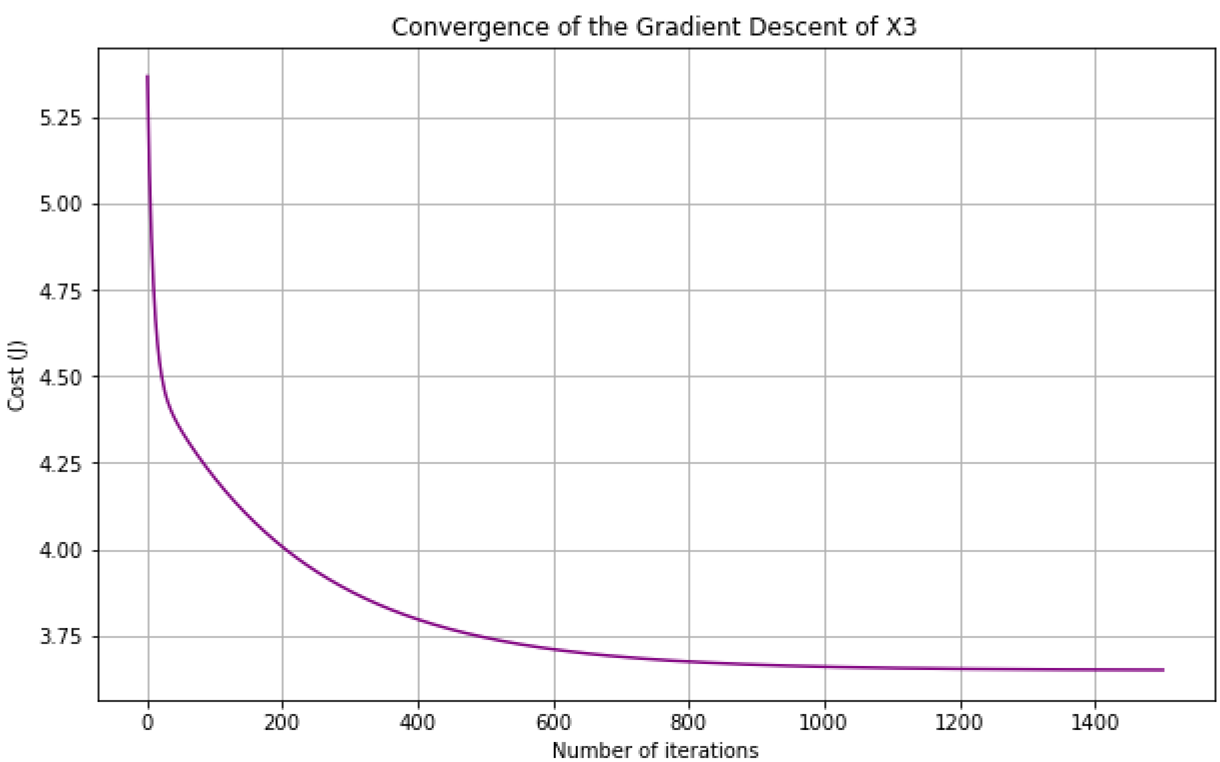
Out[38]:

```
<matplotlib.legend.Legend at 0x1c35b490610>
```



In [39]:
```
plt.plot(range(1,iterations+1),cost_history3,color='purple')
plt.rcParams["figure.figsize"]=(10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of the Gradient Descent of X3')
```

Out[39]: Text(0.5, 1.0, 'Convergence of the Gradient Descent of X3')



In [40]:
```
# Problem 2
```

In [41]:
```
def hypothesis(theta, X, n):
    h = np.ones((X.shape[0],1))
    theta = theta.reshape(1,n+1)
    for i in range(0,X.shape[0]):
        h[i] = float(np.matmul(theta, X[i]))
    h = h.reshape(X.shape[0])
    return h
```

In [42]:
```
def BGD(theta, alpha, num_iters, h, X, y, n):
    cost = np.ones(num_iters)
    for i in range(0,num_iters):
        theta[0] = theta[0] - (alpha/X.shape[0]) * sum(h - y)
        for j in range(1,n+1):
            theta[j] = theta[j] - (alpha/X.shape[0]) * sum((h-y) * X.transpose()[j])
        h = hypothesis(theta, X, n)
        cost[i] = (1/X.shape[0]) * 0.5 * sum(np.square(h - y))
    theta = theta.reshape(1,n+1)
    return theta, cost
```

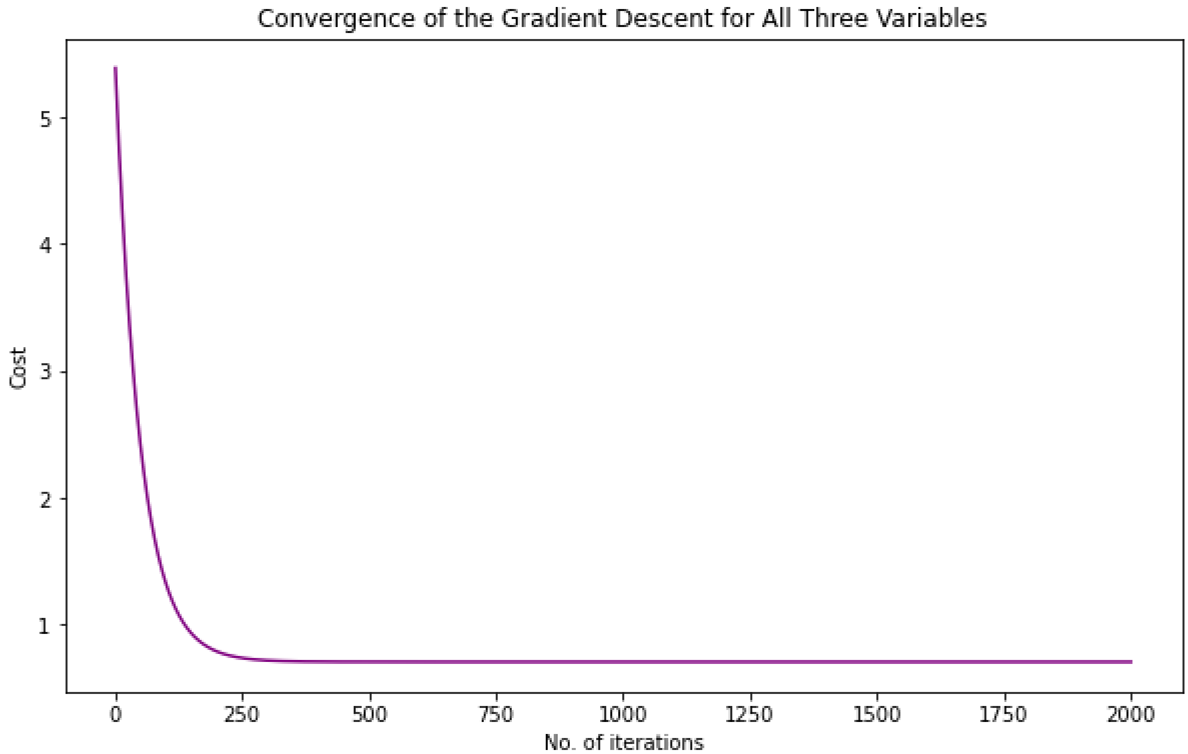
In [43]:
```
def linear_regression(X, y, alpha, num_iters):
    n = X.shape[1]
    one_column = np.ones((X.shape[0],1))
    X = np.concatenate((one_column, X), axis = 1)
    theta = np.zeros(n+1)
    h = hypothesis(theta, X, n)
    theta, cost = BGD(theta,alpha,num_iters,h,X,y,n)
    return theta, cost
```

In [44]:
```
X_train = df.values[:,[0,1,2]]
y_train = df.values[:,3]
```

In [45]:
```
mean = np.ones(X_train.shape[1])
std = np.ones(X_train.shape[1])
for i in range(0, X_train.shape[1]):
    mean[i] = np.mean(X_train.transpose()[i])
    std[i] = np.std(X_train.transpose()[i])
    for j in range(0, X_train.shape[0]):
        X_train[j][i] = (X_train[j][i] - mean[i])/std[i]
```

In [46]:
```
theta, cost = linear_regression(X_train, y_train, 0.01, 2000)
cost = list(cost)
n_iterations = [x for x in range(1,2001)]
plt.plot(n_iterations, cost, color = 'purple')
plt.xlabel('No. of iterations')
plt.ylabel('Cost')
plt.title('Convergence of the Gradient Descent for All Three Variables')
print('final value of theta =', theta)
```

```
final value of theta = [[ 1.82565676 -2.3578133   0.65275329 -0.33677367]]
```


Convergence of the Gradient Descent for All Three Variables
Cost
5
4
3
2
1
0
250
500
750
1000
1250
1500
1750
2000
No. of iterations

In [ ]:

In [ ]: