Linear regression with multiple variables

Implementation and collecting the actual values of parameters respectively.

In [0]:
```
import pandas as pd
import matplotlib.pyplot as plt
import torch
import torch.optim as optim
import numpy as np
### Load the data

# First we load the entire CSV file into an m x 3
D = torch.tensor(pd.read_csv("/content/drive/My Drive/Colab Notebooks/assignment 4/data_trai
n.csv", header=None).values, dtype=torch.float)

# We extract all rows and the first 2 columns, and then transpose it
x_dataset = D[:, 0:3].t()

# We extract all rows and the last column, and transpose it
y_dataset = D[:, 3].t()

# And make a convenient variable to remember the number of input columns
n = 3
#
iteration = 1000

### Model definition ###

# First we define the trainable parameters A and b
params = np.empty((1*3,iteration), dtype=float)
B = np.empty((1*1, iteration), dtype=float)
losses = []
A = torch.randn((1, n), requires_grad=True)
b = torch.randn(1, requires_grad=True)

# Then we define the prediction model
def model(x_input):
    return A.mm(x_input) + b


### Loss function definition ###

def loss(y_predicted, y_target):
    return ((y_predicted - y_target)**2).sum()

### Training the model ###

# Setup the optimizer object, so it optimizes a and b.
optimizer = optim.Adam([A, b], lr=0.01)

# Main optimization loop
for t in range(iteration):
    # Set the gradients to 0.
    optimizer.zero_grad()
    # Compute the current predicted y's from x_dataset
    y_predicted = model(x_dataset)
    # See how far off the prediction is
    current_loss = loss(y_predicted, y_dataset)
    # Compute the gradient of the loss with respect to A and b.
    current_loss.backward()
    # Update A and b accordingly.
    optimizer.step()
    params[:, t] = A.detach().numpy()
    B[:, t] = b.item()
    losses.append(current_loss)
    #print(f"t = {t}, loss = {current_loss}, A = {A.detach().numpy()}, b = {b.item()}")

#print(params[1,999])
#print(B[:, 999])
```

Plotting the actual values.

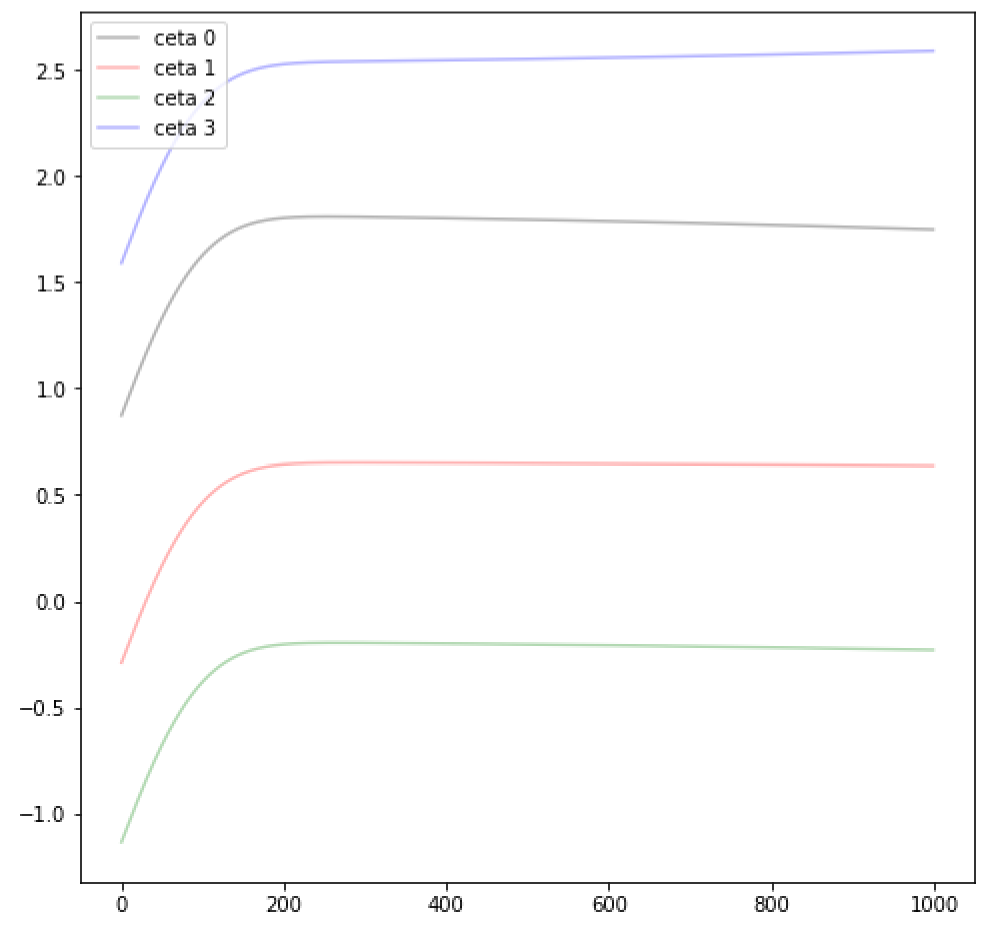
In [50]:
```
import matplotlib.pyplot as plt

print(params[:,999])
print(params[1][999])

plt.clf()
plt.figure(figsize=(8, 8))
plt.plot(range(0,iteration), B[0], c='black', label='ceta 0', alpha=0.3)
plt.plot(range(0,iteration), params[0], c='r', label='ceta 1', alpha=0.3)
plt.plot(range(0,iteration), params[1], c='g', label='ceta 2', alpha=0.3)
plt.plot(range(0,iteration), params[2], c='b', label='ceta 3', alpha=0.3)
plt.legend(loc='upper left')
plt.show()
```

```
[ 0.63614947 -0.23046547  2.59024858]
-0.23046547174453735

<Figure size 432x288 with 0 Axes>
```



plot the J ceta.

In [56]:
```
import matplotlib.pyplot as plt

plt.clf()
plt.figure(figsize=(8,8))
plt.plot(range(0,iteration),losses,c='b', label='J ceta', alpha=0.3)
plt.legend(loc='best')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```