In [319…
```
import csv
import numpy as np
import math
import matplotlib.pyplot as plt
```

In [320…
```
# read file and store to numpy array
A = []
B = []
C = []
with open('data_minibatch.csv', newline='') as csvfile:
    reader = csv.reader(csvfile)
    rows = []
    for row in reader:
        rows.append([float(x) for x in row])
    # shuffle data
    shuffled_indices = np.random.permutation(len(rows))
    for i in range(len(rows)):
        A.append(rows[i][0])
        B.append(rows[i][1])
        C.append(rows[i][2])
A = np.array(A)
B = np.array(B)
C = np.array(C)
```

In [321…
```
def f(x):
    return np.sum(A + (B * x) + (1/2 * (C * math.pow(x,2))))
```

In [322…
```
def f_gradient(x):
    return np.sum(B + (C * x))
```

In [323…
```
def f_hessian():
    return np.sum(C)
```

In [324…
```
# 2.1
# f(x) is Lipchitz continiously differentiable if L > 0
L = np.sum(np.abs(C))
print('L: {:.3f}'.format(L))
if L > 0:
    print('f(x) is Lipchitz continiously differentiable')
print('Range of step size: 0 < t < {:.3f}\n'.format(2/L))

# find x*, p*
x_opt = -np.sum(B)/np.sum(C)
print('x* is {:.3f} (hessian of f(x*) = {:.3f})'.format(x_opt, f_hessian()))
p_opt = f(x_opt)
print('p* is {:.3f}'.format(p_opt))
```

```
L: 166.644
f(x) is Lipchitz continiously differentiable
Range of step size: 0 < t < 0.012

x* is -0.115 (hessian of f(x*) = 166.644)
p* is -0.441
```

In [325…
```
def search_direction(x, k, i):
    b = B[i:i+k]
    c = C[i:i+k]
    return - np.sum(b + (c * x))
```

In [326…
```
def mini_batch(N,k,t):
    epoch = 5
    # initial x_previous
    x_prev = 1
    f_hist = [f(x_prev)]
    for e in range(epoch):
        i = 0
        for n in range(1,N//k + 1):
            x = x_prev + (t * (search_direction(x_prev, k, i)))
            f_hist.append(f(x))
            x_prev = x
            i += k
    return f_hist
```

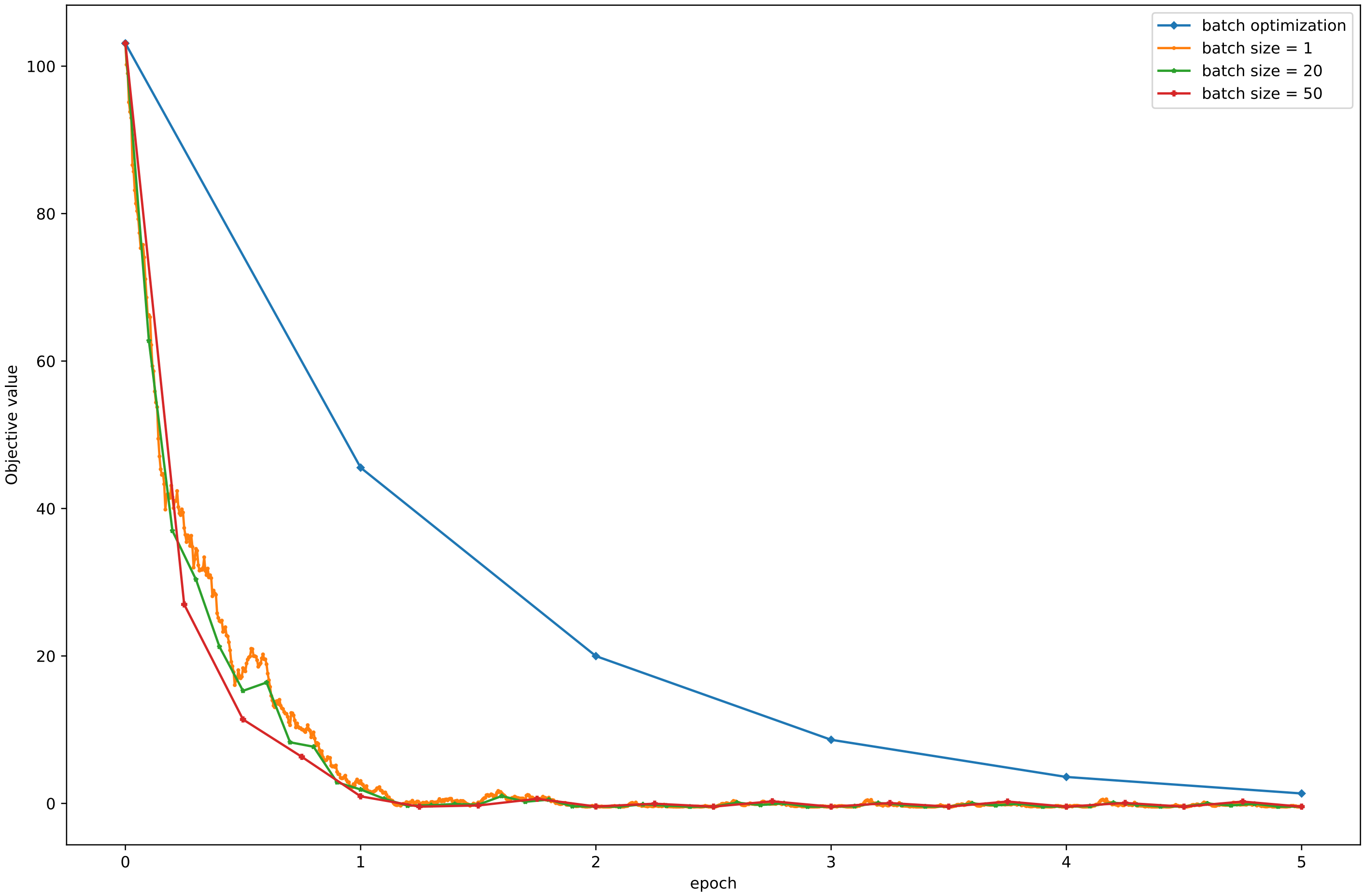
In [327…
```
# 2.2
def result(N,t):
    print('step size:', t)
    plt.figure(figsize=(15, 10))
```

```python
    k = N #batch_size
    f_hist = mini_batch(N,k,t)
    plt.plot([x/(N//k) for x in range(len(f_hist))], f_hist,marker='D', markersize=3, label='batch optimization')

    k = 1 #batch_size
    f_hist_1 = mini_batch(N,k,t)
    plt.plot([x/(N//k) for x in range(len(f_hist_1))], f_hist_1,marker='.', markersize=3, label='batch size = 1')

    k = 20 #batch_size
    f_hist_20 = mini_batch(N,k,t)
    plt.plot([x/(N//k) for x in range(len(f_hist_20))], f_hist_20,marker='*', markersize=3, label='batch size = 20')

    k = 50 #batch_size
    f_hist_50 = mini_batch(N,k,t)
    plt.plot([x/(N//k) for x in range(len(f_hist_50))], f_hist_50,marker='P', markersize=3, label='batch size = 50')
    plt.xlabel("epoch")
    plt.ylabel("Objective value")
    plt.legend(loc='upper right')
    plt.show()
```

In [328…

```python
N = A.shape[0]
t = 0.01
result(N, t)
```

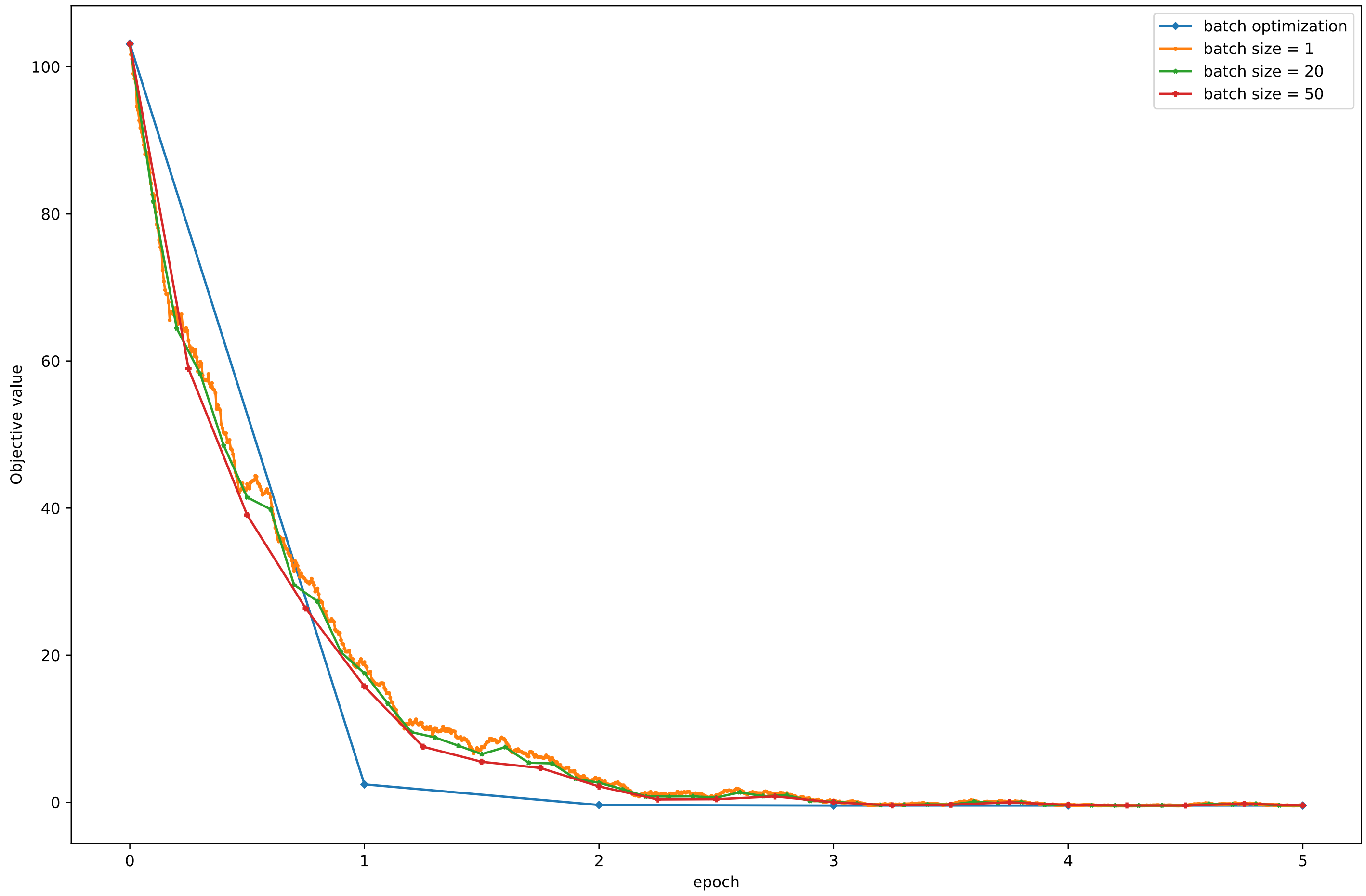
step size: 0.01



In [329…

```python
N = A.shape[0]
t = 0.005
result(N, t)
```

step size: 0.005



batch optimization
batch size = 1
batch size = 20
batch size = 50
Objective value
0
20
40
60
80
100
epoch
0
1
2
3
4
5