In [36]:
```
%matplotlib inline
import torch
import numpy as np
import pandas as pd
from torch import nn
import matplotlib.pyplot as plt
```

In [37]:
```
train_data_x=pd.read_csv("T:\project\programming\DeepLearning\experiment\dataset\data
train_data_y=pd.read_csv("T:\project\programming\DeepLearning\experiment\dataset\data

train_features=torch.tensor(train_data_x.values,dtype=torch.float32)
train_features=torch.cat([train_features,torch.ones(49,1)],dim=1)
train_res=torch.tensor(train_data_y.values,dtype=torch.float32)

print(train_features)
```

```
tensor([[2.3684, 1.0000],
        [2.5400, 1.0000],
        [2.5421, 1.0000],
        [2.5491, 1.0000],
        [2.7867, 1.0000],
        [2.9117, 1.0000],
        [3.0356, 1.0000],
        [3.1147, 1.0000],
        [3.1582, 1.0000],
        [3.3276, 1.0000],
        [3.3793, 1.0000],
        [3.4122, 1.0000],
        [3.4216, 1.0000],
        [3.5316, 1.0000],
        [3.6393, 1.0000],
        [3.6733, 1.0000],
        [3.9256, 1.0000],
        [4.0499, 1.0000],
        [4.2483, 1.0000],
        [4.3440, 1.0000],
        [4.3827, 1.0000],
        [4.4231, 1.0000],
        [4.6102, 1.0000],
        [4.6881, 1.0000],
        [4.9777, 1.0000],
        [5.0360, 1.0000],
        [5.0685, 1.0000],
        [5.4161, 1.0000],
        [5.4396, 1.0000],
        [5.4563, 1.0000],
        [5.5698, 1.0000],
        [5.6016, 1.0000],
        [5.6878, 1.0000],
        [5.7216, 1.0000],
        [5.8539, 1.0000],
        [6.1978, 1.0000],
        [6.3511, 1.0000],
        [6.4797, 1.0000],
        [6.7384, 1.0000],
        [6.8638, 1.0000],
        [7.0223, 1.0000],
        [7.0782, 1.0000],
        [7.1514, 1.0000],
        [7.4664, 1.0000],
        [7.5974, 1.0000],
```

```
       [7.7441, 1.0000],
       [7.7730, 1.0000],
       [7.8265, 1.0000],
       [7.9306, 1.0000]])
```

In [38]:

```
def squared_loss(y_hat, y):
    """均方损失。"""
    return ((y_hat - y.reshape(y_hat.shape))**2 / 2).mean()
```

In [39]:

```
def set_learning_rate(optimizer, lr):
    for param_group in optimizer.param_groups:
        param_group['lr'] = lr
```

In [40]:

```
def train(net,optimizer,loss,train_features,train_res,get_step=None):
    theta=-1
    print(net[0].bias)
    loss_list=[]
    theta_bias=[]
    while  torch.norm(list(net.parameters())[0].squeeze()-theta,p=2,dim=0).item()**2
        theta=torch.tensor(list(net.parameters())[0].squeeze())
        l=loss(net(train_features),train_res)
        optimizer.zero_grad()
        l.backward()
        if get_step!=None:
            set_learning_rate(optimizer,get_step(loss,net,train_features,train_res))

        optimizer.step()
        #准备可视化的数据
        #print("l:",l)
        loss_list.append(l.item())
        #print(torch.norm(list(net.parameters())[0].squeeze()-theta,p=2,dim=0).item()*
        theta_bias.append(torch.norm(list(net.parameters())[0].squeeze()-theta,p=2,di
    return loss_list,theta_bias,theta
```

In [41]:

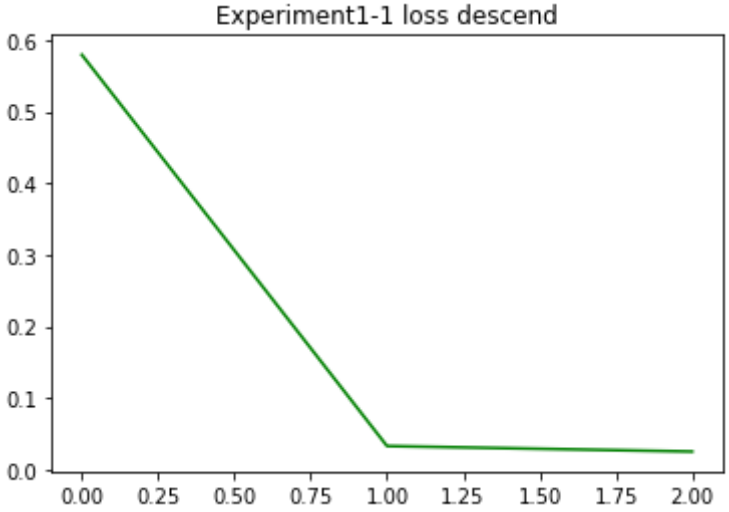
```
#初始化超参数
net=nn.Sequential( nn.Linear(train_features.shape[1],1) )
net[0].weight.data.fill_(0)
net[0].bias.data.fill_(0)
optimizer=torch.optim.SGD(net.parameters(), 0.03)
loss = squared_loss
```

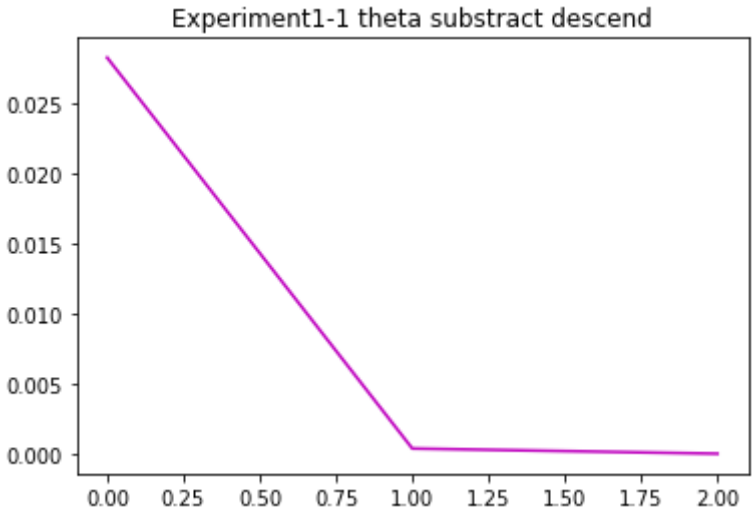
In [42]:

```
loss_list,theta_bias,theta=train(net,optimizer,loss,train_features,train_res)
epoches=np.arange(len(loss_list))
fig, ax = plt.subplots()
fmt='g-'
ax.set_title('Experiment1-1 loss descend')
ax.plot(epoches, loss_list,fmt)

fig_2, ax_2 = plt.subplots()
fmt='m-'
ax_2.set_title('Experiment1-1 theta substract descend')
ax_2.plot(epoches, theta_bias,fmt)


print("Experiment1-1 theta substraction ",torch.norm(list(net.parameters())[0].squeez
print("Experiment1-1 loss ",squared_loss(net(train_features),train_res).data)
```

```
Parameter containing:
```

```
tensor([0.], requires_grad=True)
C:\Users\Young\AppData\Local\Temp\ipykernel_25020\3571246903.py:7: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(True), rather than torch.tensor(sourceTensor).
  theta=torch.tensor(list(net.parameters())[0].squeeze())
Experiment1-1 theta substraction  8.055307634362513e-06
Experiment1-1 loss  tensor(0.0247)
```





In [43]:

```
train_data2_x=pd.read_csv("T:\project\programming\DeepLearning\experiment\dataset\data
train_data2_y=pd.read_csv("T:\project\programming\DeepLearning\experiment\dataset\data
print(train_data2_x.shape)
#标准化
train_data2_x[:] = train_data2_x[:].apply(
    lambda x: (x - x.mean()) / (x.std()))

train2_features=torch.tensor(train_data2_x.values,dtype=torch.float32)
train2_features=torch.cat([torch.ones(46,1),train2_features],dim=1)
train2_res=torch.tensor(train_data2_y.values,dtype=torch.float32)

train2_features
```

Out[43]:

```
(46, 2)
tensor([[ 1.0000e+00, -4.9598e-01, -2.2617e-01],
        [ 1.0000e+00,  4.9987e-01, -2.2617e-01],
        [ 1.0000e+00, -7.2502e-01, -1.5266e+00],
        [ 1.0000e+00,  1.2468e+00,  1.0743e+00],
        [ 1.0000e+00, -1.6724e-02,  1.0743e+00],
        [ 1.0000e+00, -5.7813e-01, -2.2617e-01],
        [ 1.0000e+00, -7.1133e-01, -2.2617e-01],
```

```
        [ 1.0000e+00, -7.6984e-01, -2.2617e-01],
        [ 1.0000e+00, -6.2793e-01, -2.2617e-01],
        [ 1.0000e+00, -7.2740e-02,  1.0743e+00],
        [ 1.0000e+00,  1.9484e-03, -2.2617e-01],
        [ 1.0000e+00, -1.3498e-01, -2.2617e-01],
        [ 1.0000e+00,  3.0866e+00,  2.3747e+00],
        [ 1.0000e+00, -9.0926e-01, -2.2617e-01],
        [ 1.0000e+00,  3.7539e-01,  1.0743e+00],
        [ 1.0000e+00, -8.4452e-01, -1.5266e+00],
        [ 1.0000e+00, -9.4909e-01, -2.2617e-01],
        [ 1.0000e+00,  7.6004e-01,  1.0743e+00],
        [ 1.0000e+00,  1.2854e+00,  1.0743e+00],
        [ 1.0000e+00, -2.8809e-01, -2.2617e-01],
        [ 1.0000e+00, -1.3747e-01, -1.5266e+00],
        [ 1.0000e+00, -4.9100e-01, -2.2617e-01],
        [ 1.0000e+00, -4.5355e-02,  1.0743e+00],
        [ 1.0000e+00,  2.3546e+00, -2.2617e-01],
        [ 1.0000e+00, -1.1184e+00, -2.2617e-01],
        [ 1.0000e+00, -6.7274e-01, -2.2617e-01],
        [ 1.0000e+00,  6.5672e-01, -2.2617e-01],
        [ 1.0000e+00,  2.5091e-01, -2.2617e-01],
        [ 1.0000e+00,  7.9489e-01, -2.2617e-01],
        [ 1.0000e+00, -1.9847e-01, -1.5266e+00],
        [ 1.0000e+00, -1.2429e+00, -2.8271e+00],
        [ 1.0000e+00,  5.1741e-02,  1.0743e+00],
        [ 1.0000e+00,  1.4173e+00, -2.2617e-01],
        [ 1.0000e+00, -2.3332e-01,  1.0743e+00],
        [ 1.0000e+00, -6.9888e-01, -2.2617e-01],
        [ 1.0000e+00, -9.4535e-01, -2.2617e-01],
        [ 1.0000e+00,  1.6626e-01,  1.0743e+00],
        [ 1.0000e+00,  2.7592e+00,  1.0743e+00],
        [ 1.0000e+00,  2.0361e-01,  1.0743e+00],
        [ 1.0000e+00, -4.1631e-01, -1.5266e+00],
        [ 1.0000e+00,  2.9821e-01, -2.2617e-01],
        [ 1.0000e+00,  7.0776e-01,  1.0743e+00],
        [ 1.0000e+00, -9.9390e-01, -2.2617e-01],
        [ 1.0000e+00, -1.4271e+00, -1.5266e+00],
        [ 1.0000e+00, -1.8228e-01,  1.0743e+00],
        [ 1.0000e+00, -9.9017e-01, -2.2617e-01]])
```

In [44]:

```
#初始化超参数
net=nn.Sequential( nn.Linear(train2_features.shape[1],1) )
net[0].weight.data.fill_(0)
net[0].bias.data.fill_(0)
optimizer=torch.optim.SGD(net.parameters(), 0.03)
loss = squared_loss
```

In [45]:

```
def get_step(loss,net,train_features,train_res):
    a=torch.tensor(0.0,requires_grad=True)
    #weight=net[0].weight.clone().detach().requires_grad_(True)
    test=torch.mm(train_features,torch.transpose(net[0].weight+a*net[0].weight.grad.
    l=loss(test,train_res)
    gd_1 = torch.autograd.grad(l, a, create_graph=True)
    gd_2 = torch.autograd.grad(gd_1, a)
    #print(float(gd_1[0].data)/float(gd_2[0].data))
    #net[0].weight.data=weight
    return float(gd_1[0].data)/float(gd_2[0].data)
```

In [46]:

```
loss_list_train2,theta_bias_train2,theta_train2=train(net,optimizer,loss,train2_featur

epoches_train2=np.arange(len(loss_list_train2))
```

```
fig_train2, ax_train2 = plt.subplots()
fmt='g-'
ax_train2.set_title('Experiment1-2 loss descend')
ax_train2.plot(epoches_train2, loss_list_train2,fmt)

fig_2_train2, ax_2_train2 = plt.subplots()
fmt='m-'
ax_2_train2.set_title('Experiment1-2 theta substract descend')
ax_2_train2.plot(epoches_train2, theta_bias_train2,fmt)

print("Experiment1-2 theta substraction ",torch.norm(list(net.parameters())[0].squeez
print("Experiment1-2 loss ",squared_loss(net(train2_features),train2_res).data)
```

```
Parameter containing:
tensor([0.], requires_grad=True)
Experiment1-2 theta substraction  2.384185791015625e-07
Experiment1-2 loss  tensor(2.0665e+09)
C:\Users\Young\AppData\Local\Temp\ipykernel_25020\3571246903.py:7: UserWarning: To cop
y construct from a tensor, it is recommended to use sourceTensor.clone().detach() or s
ourceTensor.clone().detach().requires_grad_(True), rather than torch.tensor(sourceTens
or).
  theta=torch.tensor(list(net.parameters())[0].squeeze())
```