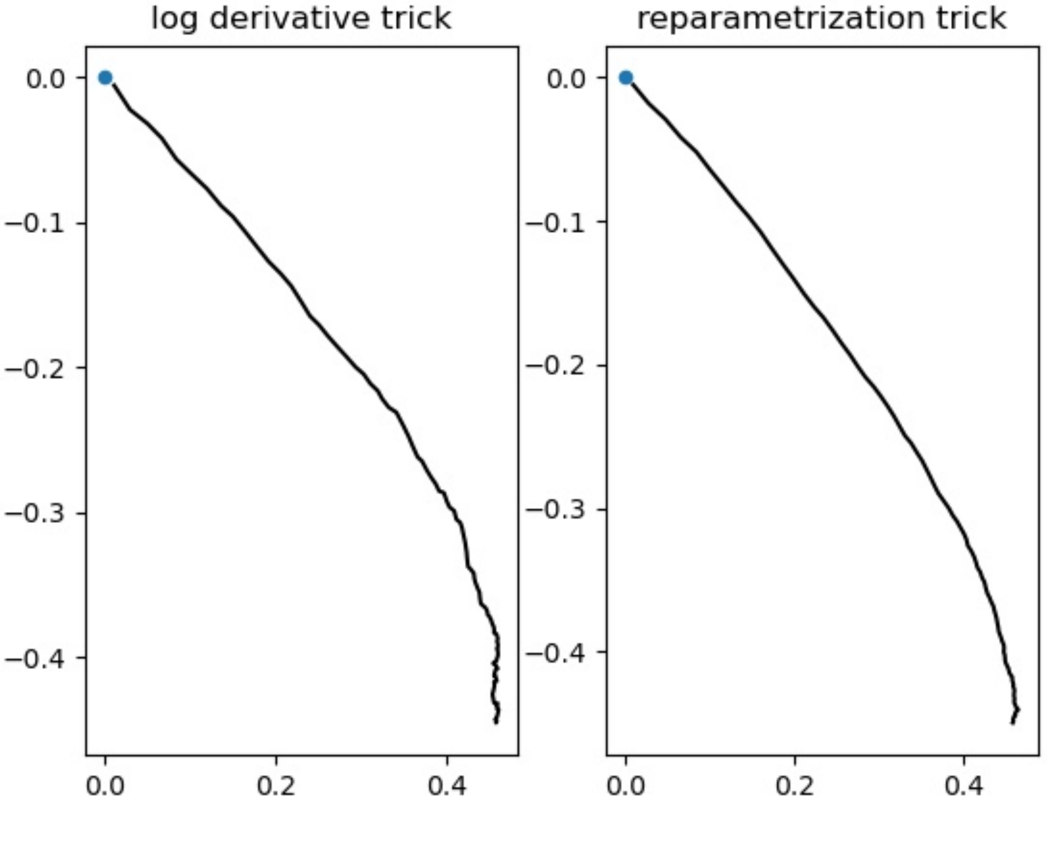
```
import torch
from torch.distributions.normal import Normal
import math

import matplotlib.pyplot as plt

### (a) log-derivative trick

def df_dmu(x, mu, sigma):

    return x*torch.sin(x)*(x-mu)/sigma**2 + mu-1


def df_dt(x, mu, sigma):

    return x*torch.sin(x)*(-1+(x-mu)**2/sigma**2) + sigma - 1


epochs = 200
lr = 1e-2
batch = 16

x_batch = torch.zeros(256,1)

# initialize
mu = torch.tensor([0.0])
t = torch.tensor([0.0])

mu_lst_a =[]
t_lst_a = []

for epoch in range(epochs+1):
    x_batch = Normal(loc=mu, scale=torch.exp(t)).sample(x_batch.size())

    mu -= df_dmu(x=x_batch, mu=mu, sigma=torch.exp(t)).mean() * lr
    t -= df_dt(x=x_batch, mu=mu, sigma=torch.exp(t)).mean() * lr

    if epoch%2==0:
        mu_lst_a.append(mu.item())
        t_lst_a.append(t.item())


    if epoch%100==0:
        print('epoch:', epoch, 'mu:',mu.item(), 'sigma:',math.exp(t.item()), 'tau:', t.item())

        X = Normal(loc=mu.item(), scale=math.exp(t.item())).sample((1,5000))
                          print('f:',((X*torch.sin(X)).mean()+   0.5*(mu.item()-1)**2   +   math.exp(t.item())   -
math.log(math.exp(t.item()))).item())



### (b) reparameterization trick
```

```
def dphi_dx(x):
    return torch.sin(x)+x*torch.cos(x)

def df_dmu(y, mu, sigma):
    return dphi_dx(mu+sigma*y) + mu - 1

def df_dt(y, mu, sigma):
    return dphi_dx(mu+sigma*y)*sigma*y + sigma - 1


y_batch = torch.zeros(256,1)

# initialize
mu = torch.tensor([0.0])
t = torch.tensor([0.0])

mu_lst_b = []
t_lst_b = []

for epoch in range(epochs+1):
    y_batch = Normal(loc=0, scale=1).sample(x_batch.size())

    mu -= df_dmu(y=y_batch, mu=mu, sigma=torch.exp(t)).mean() * lr
    t -= df_dt(y=y_batch, mu=mu, sigma=torch.exp(t)).mean() * lr


    if epoch%2==0:
        mu_lst_b.append(mu.item())
        t_lst_b.append(t.item())

    if epoch%100==0:
        print('epoch:', epoch, 'mu:',mu.item(), 'sigma:',math.exp(t.item()), 'tau:', t.item())

        X = Normal(loc=mu.item(), scale=math.exp(t.item())).sample((1,5000))
        print('f:',((X*torch.sin(X)).mean()+ 0.5*(mu.item()-1)**2 + math.exp(t.item()) - math.log(math.exp(t.item()))).item())


plt.subplot(1, 2, 1)
plt.scatter(0,0, s=20)
plt.plot(mu_lst_a, t_lst_a, color='k')
plt.title('log derivative trick')

plt.subplot(1, 2, 2)
plt.scatter(0,0, s=20)
plt.plot(mu_lst_b, t_lst_b, color='k')
plt.title('reparametrization trick')

plt.savefig('log derivative vs reparametrization.png')
plt.show()
```


log derivative trick
reparametrization trick
0.0
−0.1
−0.2
−0.3
−0.4
0.0
0.2
0.4
0.0
−0.1
−0.2
−0.3
−0.4
0.0
0.2
0.4