train_DerivativeNet

April 27, 2022

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import torch
from torch.optim import Adam
import torch.nn as nn
from torch.utils.data import DataLoader

from binarypredictor.dataset import FunctionDataset
from binarypredictor.net import DerivativeNet
```

[2]:
```
@torch.enable_grad()
def epoch(net, train_loader, loss_func, optimizer):
    epoch_losses = np.zeros([len(train_loader), ])

    for i, d in enumerate(train_loader):
        # Forward
        inp, targets = d[:, :, 0], d[:, :, 1]
        targets[abs(targets) > 1000] = 0
        out = net(inp.float())

        # Calculate the loss
        loss = loss_func(out, targets)
        epoch_losses[i] = loss

        # Backward
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    print('Epoch loss: ', epoch_losses.mean())
    return epoch_losses.mean()
```

[3]:
```
@torch.enable_grad()
def train(net, train_loader, test_loader):
    # Hyperparameters
    nr_epochs = 100
    lr = 1e-4

    # Workers
    loss_func = nn.MSELoss()
    optimizer = Adam(net.parameters(), lr=lr)

    # Losses
    losses = np.zeros([nr_epochs, ])

    # Baseline for best network
    best_loss = epoch(net, train_loader, loss_func, optimizer)
    best_net = net

    # Training
    for i in range(nr_epochs):
        losses[i] = epoch(net, train_loader, loss_func, optimizer)

        if losses[i] < best_loss:
            best_net = net
            best_loss = losses[i]

    return best_net, losses
```

```
[5]: in_features = 500
```

```
[11]: fd_0 = FunctionDataset(100000, 'first_500.csv', o=0, step=1/in_features)
fd_0.create_functions()
fd_1 = FunctionDataset(100000, 'second_500.csv', o=1, step=1/in_features)
fd_1.create_functions()
```

```
[12]: loader_0 = DataLoader(fd_0, batch_size=256, shuffle=True)
loader_1 = DataLoader(fd_1, batch_size=256, shuffle=True)
net_0 = DerivativeNet(train=True, hidden_layers=4, in_features=in_features,␣
 ↪out_features=in_features, hidden_size_linear=in_features)
net_1 = DerivativeNet(train=True, hidden_layers=4, in_features=in_features,␣
 ↪out_features=in_features, hidden_size_linear=in_features)
```

```
[13]: best_net_0, _ = train(net_0, loader_0, None)
print('First derivative net trained!')
best_net_1, _ = train(net_1, loader_1, None)
print('Second derivative net trained!')
```

```
Epoch loss:  2.141631111845641
Epoch loss:  0.72308164027036
Epoch loss:  0.712935806235389
Epoch loss:  0.7005866077702368
Epoch loss:  0.6947975653364226
Epoch loss:  0.6950505816418192
Epoch loss:  0.6927652602915264
Epoch loss:  0.6917025411068021
Epoch loss:  0.6916406401587875
Epoch loss:  0.6880453264012056
Epoch loss:  0.6865461246131936
Epoch loss:  0.6851625777114078
Epoch loss:  0.6821990715115881
Epoch loss:  0.6829578919941203
Epoch loss:  0.6825998208254499
Epoch loss:  0.6800632400585868
Epoch loss:  0.6857611119289837
Epoch loss:  0.6814216616970804
Epoch loss:  0.6803094455043374
Epoch loss:  0.6785694651896387
Epoch loss:  0.6806339725204136
Epoch loss:  0.6793249387417912
Epoch loss:  0.6789765987554779
Epoch loss:  0.6778939729151519
Epoch loss:  0.6796131370317601
Epoch loss:  0.6790404903995412
Epoch loss:  0.6754831487260511
Epoch loss:  0.6746833552332485
Epoch loss:  0.6753490936878087
Epoch loss:  0.67426039945439
Epoch loss:  0.6755255098690462
Epoch loss:  0.67386927339427
Epoch loss:  0.6747270292028442
Epoch loss:  0.6748946052987862
Epoch loss:  0.6753156252224427
Epoch loss:  0.6755895448462738
Epoch loss:  0.6738218768783237
Epoch loss:  0.6720664562929012
Epoch loss:  0.6738617799013776
Epoch loss:  0.672211969371342
Epoch loss:  0.6717692852172705
Epoch loss:  0.6739272461522876
Epoch loss:  0.6747108614048385
Epoch loss:  0.6705804643271219
Epoch loss:  0.6718914902880978
Epoch loss:  0.6728559107426793
Epoch loss:  0.6715047949415338
Epoch loss:  0.6721693823099746
Epoch loss:  0.6702049735485746
Epoch loss:  0.670686953131805
Epoch loss:  0.6713555619082487
Epoch loss:  0.6705283019167688
Epoch loss:  0.6738973557186858
Epoch loss:  0.6694414280259701
Epoch loss:  0.6692536102460168
Epoch loss:  0.6677662334631166
Epoch loss:  0.6695751294760448
Epoch loss:  0.6705554669622875
Epoch loss:  0.6683975937573806
Epoch loss:  0.6686250272461826
Epoch loss:  0.6687925958724887
Epoch loss:  0.6697291197907894
Epoch loss:  0.6679691691380327
Epoch loss:  0.6676008599950835
Epoch loss:  0.668853519975072
Epoch loss:  0.6672992713920906
Epoch loss:  0.668050455544001
Epoch loss:  0.665219666402968
Epoch loss:  0.6667094787063501
Epoch loss:  0.6673106305739459
Epoch loss:  0.6663882332994505
Epoch loss:  0.6662029832068002
Epoch loss:  0.6648745299757594
Epoch loss:  0.6657134792231538
Epoch loss:  0.6646642161299811
Epoch loss:  0.6647407165573685
Epoch loss:  0.6636316177942564
Epoch loss:  0.6625020067633876
Epoch loss:  0.6636761896445623
Epoch loss:  0.6632022824891083
Epoch loss:  0.6654327096384199
Epoch loss:  0.6616897703436635
Epoch loss:  0.6618282436714757
Epoch loss:  0.660773760522418
Epoch loss:  0.6610787630538502
Epoch loss:  0.6635293227708553
Epoch loss:  0.6626529310213025
Epoch loss:  0.6605543303672615
Epoch loss:  0.6609143192505897
Epoch loss:  0.6598857263165057
Epoch loss:  0.6600865147760152
Epoch loss:  0.660109784673242
Epoch loss:  0.6589682601830539
Epoch loss:  0.6590790524507117
Epoch loss:  0.6579833976600481
Epoch loss:  0.6569386225222321
Epoch loss:  0.6584952219063059
Epoch loss:  0.6576064815911491
Epoch loss:  0.6572837873797892
Epoch loss:  0.6587972024365154
Epoch loss:  0.6558526030281926
First derivative net trained!
Epoch loss:  0.5836066185155689
Epoch loss:  0.00035208335489152556
Epoch loss:  0.00017804432268221112
Epoch loss:  0.00013299656957384945
Epoch loss:  0.00015890914629145925
Epoch loss:  0.00015387008618831377
Epoch loss:  0.00010811288642719813
Epoch loss:  0.0003608074193997734
Epoch loss:  5.5438926433930005e-05
Epoch loss:  9.511945337899561e-05
Epoch loss:  0.00010433977351097725
Epoch loss:  0.00020589420515985843
Epoch loss:  0.0003577519619610081
Epoch loss:  2.886662888156427e-05
Epoch loss:  6.879234758025283e-05
Epoch loss:  0.00011601181180911068
Epoch loss:  0.00016226012465970032
Epoch loss:  0.0001456589529845257
Epoch loss:  8.464085747344334e-05
Epoch loss:  0.00013345244640317793
Epoch loss:  8.075671628416564e-05
Epoch loss:  0.00022544959323256343
Epoch loss:  3.87065012146288e-05
Epoch loss:  0.00012923558895873492
Epoch loss:  5.856370177666914e-05
Epoch loss:  0.00024471719344637446
Epoch loss:  0.00010005202535546192
Epoch loss:  0.0002504941392374621
Epoch loss:  1.926758559327662e-05
Epoch loss:  6.817820135054722e-05
Epoch loss:  4.494457793360039e-05
Epoch loss:  0.00010083932201922535
Epoch loss:  0.0002775141112155804
Epoch loss:  1.1002277492049738e-05
Epoch loss:  3.445091851507178e-05
Epoch loss:  0.0002337213261206673
Epoch loss:  4.2506919896096196e-05
Epoch loss:  9.988773769489393e-05
Epoch loss:  8.007912137101272e-05
Epoch loss:  9.046537928478084e-05
Epoch loss:  0.00011226804458962232
Epoch loss:  5.624839846962033e-05
Epoch loss:  8.596491702554306e-05
Epoch loss:  0.00010786078342682552
Epoch loss:  3.6751339692859346e-05
Epoch loss:  0.00014065982843435671
Epoch loss:  0.00016188935781887118
Epoch loss:  2.7271516370217893e-05
Epoch loss:  8.011339760251174e-05
Epoch loss:  4.2958601882318085e-05
Epoch loss:  8.342205997705651e-05
Epoch loss:  6.615854049926074e-05
Epoch loss:  0.0001010428678011075
Epoch loss:  7.091713555173393e-05
Epoch loss:  0.00015376963121799955
Epoch loss:  2.833537609709645e-05
Epoch loss:  6.526298101541198e-05
Epoch loss:  7.863640437632137e-05
Epoch loss:  4.669676155174475e-05
Epoch loss:  0.00023951476504921828
Epoch loss:  1.3752089289412901e-05
Epoch loss:  2.2851422941397794e-05
Epoch loss:  0.00024041224956292348
Epoch loss:  3.9564326447748095e-05
Epoch loss:  1.3487920023994903e-05
Epoch loss:  3.6132472326748926e-05
Epoch loss:  0.00012499142451099284
Epoch loss:  0.00011382774755824022
Epoch loss:  2.8579803040109563e-05
Epoch loss:  3.199275105685966e-05
Epoch loss:  0.0001135098463954821
Epoch loss:  2.30092763247093e-05
Epoch loss:  5.727769470307548e-05
Epoch loss:  7.486649124215048e-05
Epoch loss:  9.41899017906068e-05
Epoch loss:  3.321605143302055e-05
Epoch loss:  6.401709392973672e-05
Epoch loss:  5.0223161722273814e-05
Epoch loss:  7.233164823821211e-05
Epoch loss:  7.952672300453204e-05
Epoch loss:  3.430916246864346e-05
Epoch loss:  6.259043016318006e-05
Epoch loss:  0.00011312870734735303
Epoch loss:  1.2443006271426633e-05
Epoch loss:  5.146878313489349e-05
Epoch loss:  7.180968174239737e-05
Epoch loss:  0.00011468862184617889
Epoch loss:  8.764800261819906e-05
Epoch loss:  2.5520068787992314e-05
Epoch loss:  3.44294668985591e-05
Epoch loss:  0.0001028607304686519
Epoch loss:  9.919848728148752e-06
Epoch loss:  0.00012096686586772803
Epoch loss:  7.362909780944684e-05
Epoch loss:  1.0923338072701825e-05
Epoch loss:  4.650385296751556e-05
Epoch loss:  5.485047448791288e-05
Epoch loss:  4.040306477965524e-05
Epoch loss:  0.00012340383192979592
Epoch loss:  9.288041873626072e-06
Epoch loss:  7.733674855324613e-05
Second derivative net trained!
```

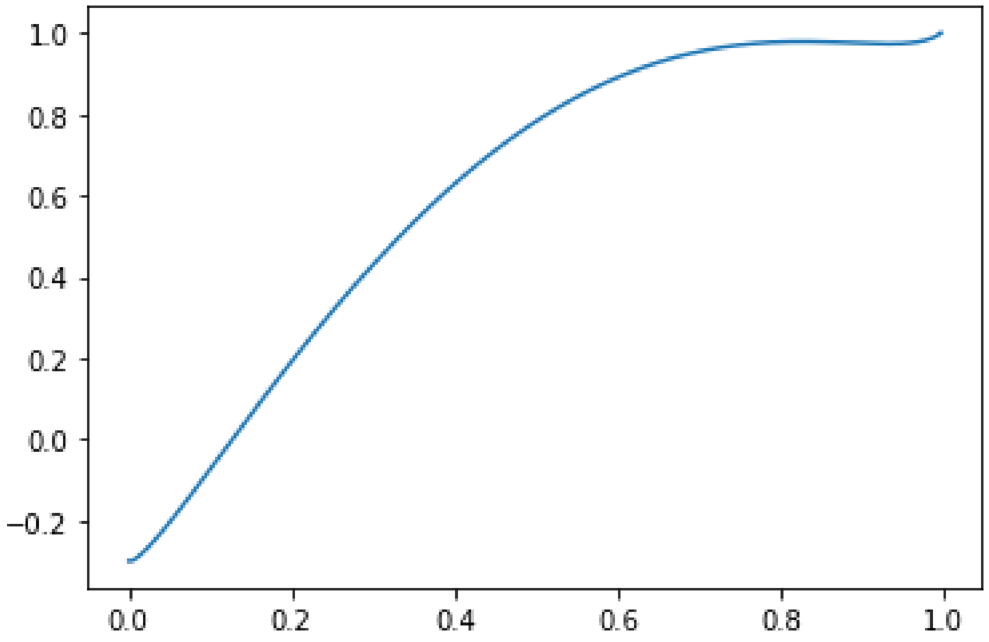
[14]:
```
torch.save(net_0, 'FirstDerivativeNet_500.pth')
torch.save(net_1, 'SecondDerivativeNet_500.pth')
```

[9]:
```
#net_0 = torch.load('FirstDerivativeNet_250.pth')
#net_1 = torch.load('SecondDerivativeNet_250.pth')
```

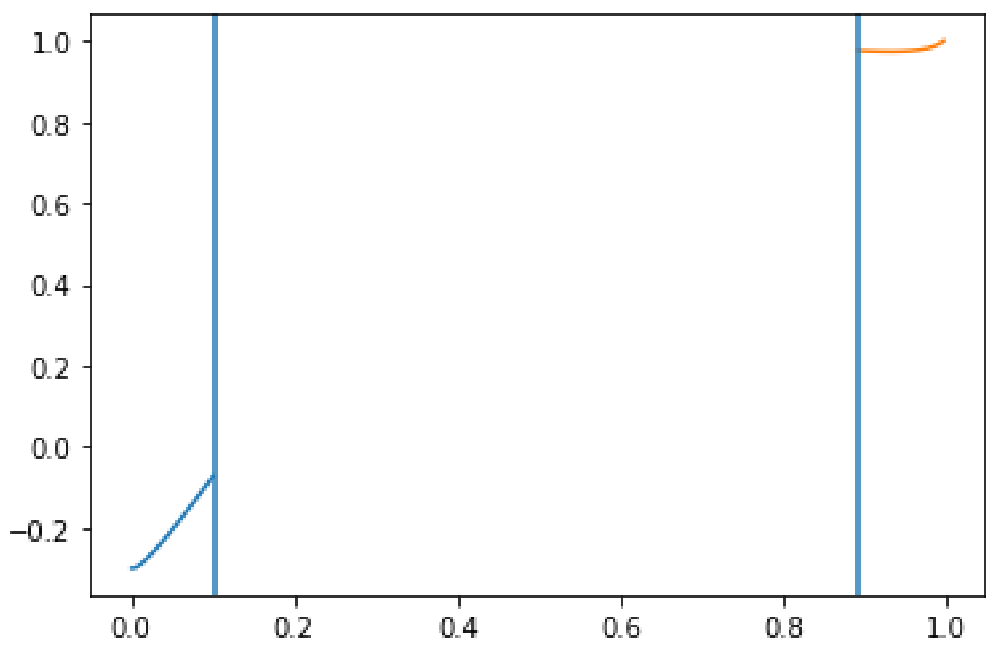
[10]:
```
x = torch.arange(0., 1., step=1/in_features)
```

[11]:
```
fd_test = FunctionDataset(50, 'None.csv', o=0, step=1/in_features,␣
 ↪overwrite=True)

for d in fd_test:
    print('##########################')
    plt.plot(x, d[:, 0])
    plt.show()
    #plt.plot(x, d[:, 1])

    out = net_0(d[:, 0])
    #plt.plot(x, out.detach())
    #plt.show()

    second_der = net_1(out)
    #plt.plot(x, second_der.detach())
    #plt.show()

    idx = torch.where(torch.diff(torch.sign(second_der)) != 0)[0]
    idx = idx[idx != 0]
    if len(idx) > 0:
        min_diff = 5
        valid_idx = [idx[i] for i in range(1, len(idx) - 1) if idx[i + 1] -␣
 ↪idx[i] > min_diff and abs(idx[i] - idx[i - 1]) > min_diff]
        if idx[-1] - idx[0] > min_diff:
            valid_idx += [idx[0]] + [idx[-1]]
        idx = torch.tensor(sorted(valid_idx), dtype=torch.int64)

    print(idx)
    x_idx = x[idx]

    x_split, f_split, s_split = torch.tensor_split(x, idx), torch.
 ↪tensor_split(d[:, 0], idx), torch.tensor_split(second_der, idx)

    x_split_, f_split_ = [], []
    # Remove curve parts with negative curvature
    for i, s in enumerate(s_split):
        if torch.sum(torch.sign(s)) >= 0:
            x_split_.append(x_split[i])
            f_split_.append(f_split[i])

    for x_, f_ in zip(x_split_, f_split_):
        plt.plot(x_, f_)
    for i in x_idx:
        plt.axvline(i)
    plt.show()
```

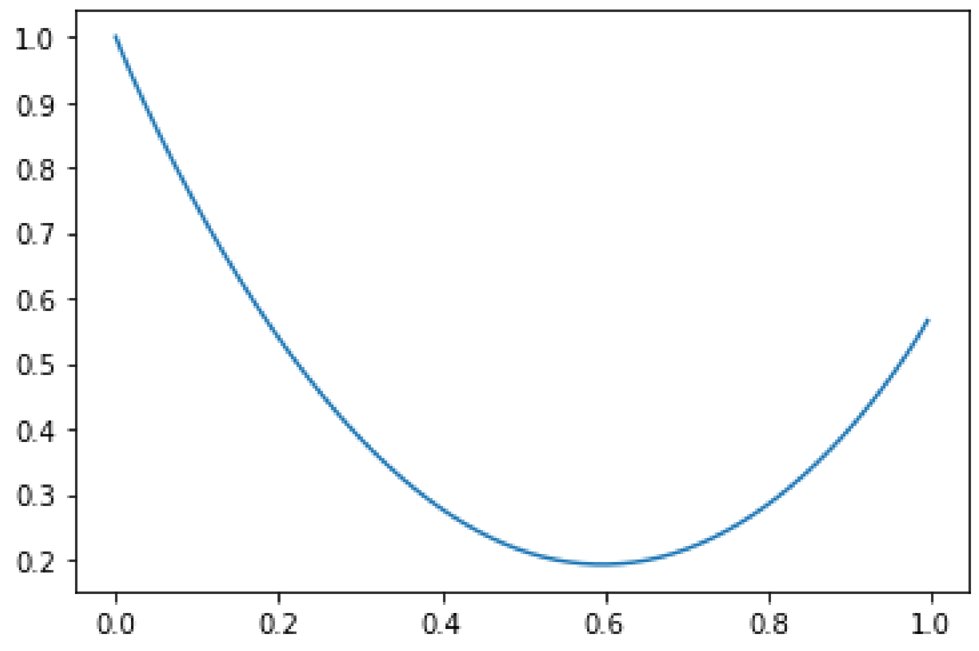
```
##########################
```



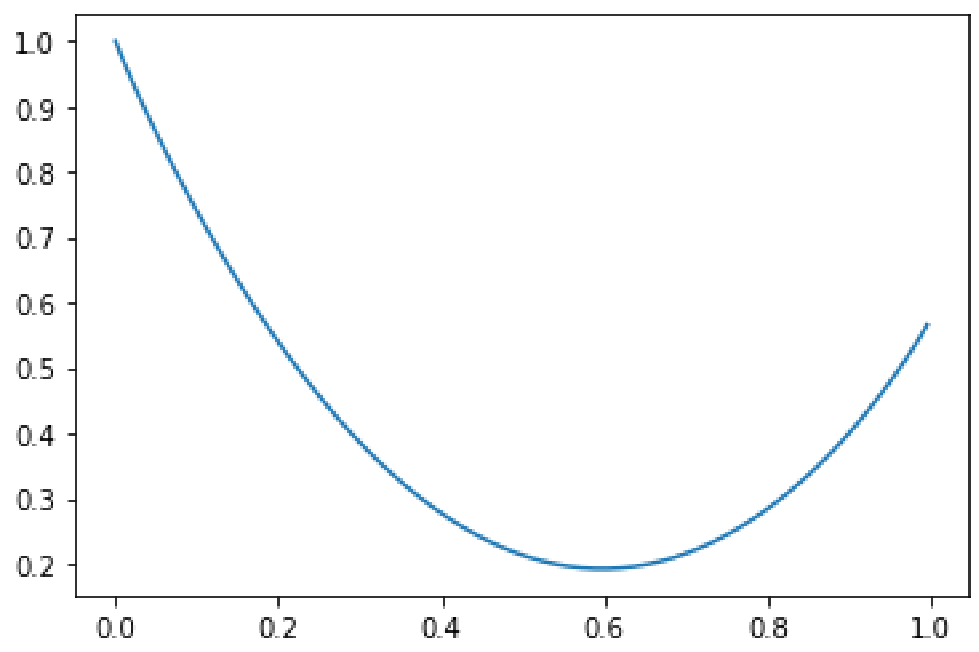
```
tensor([ 26, 223])
```



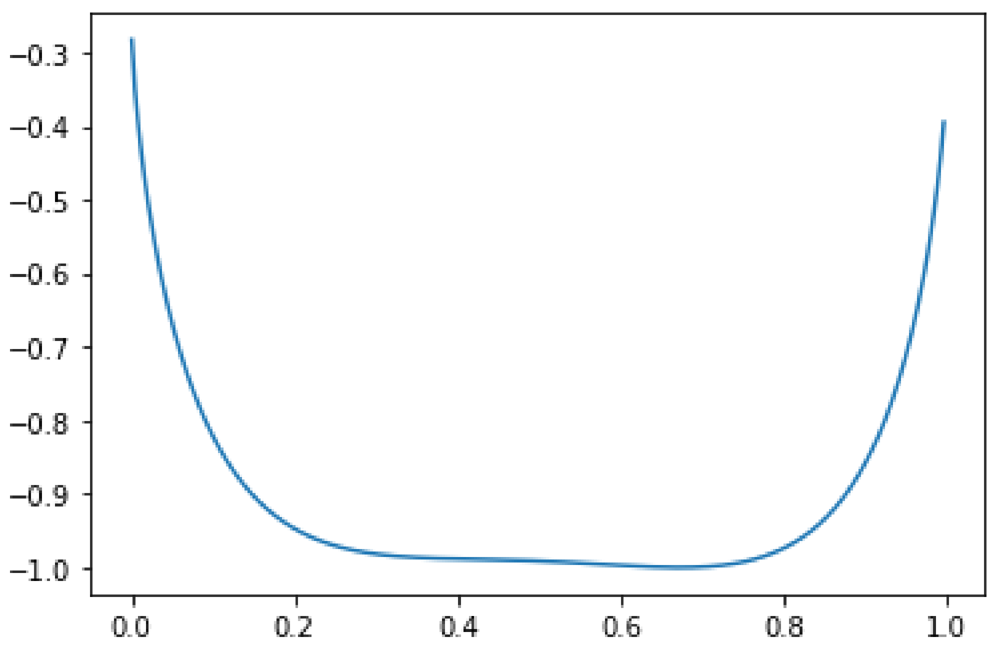
```
##########################
```



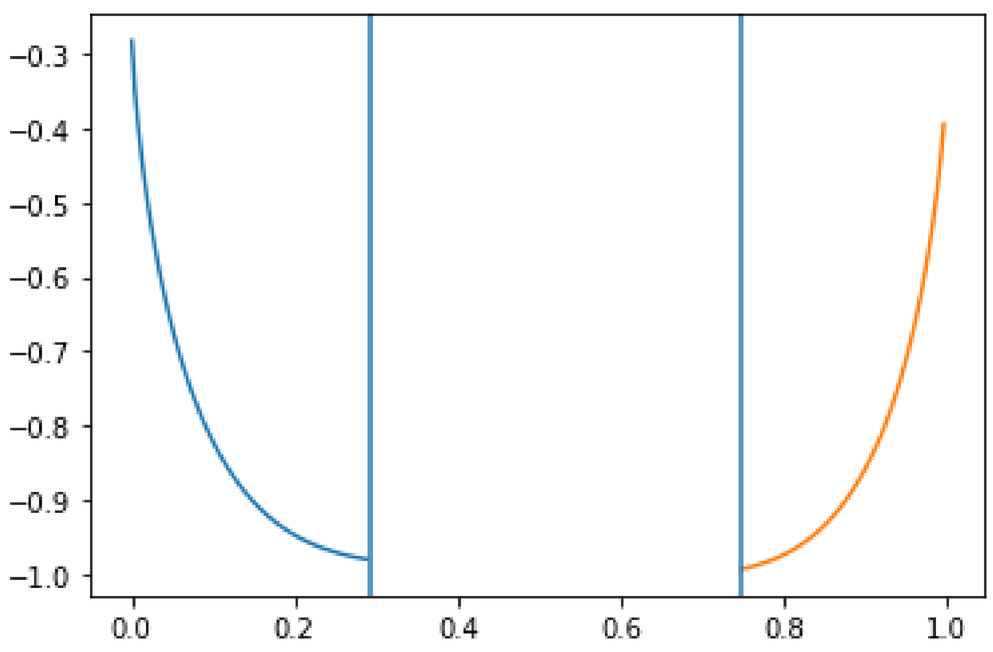
```
tensor([], dtype=torch.int64)
```



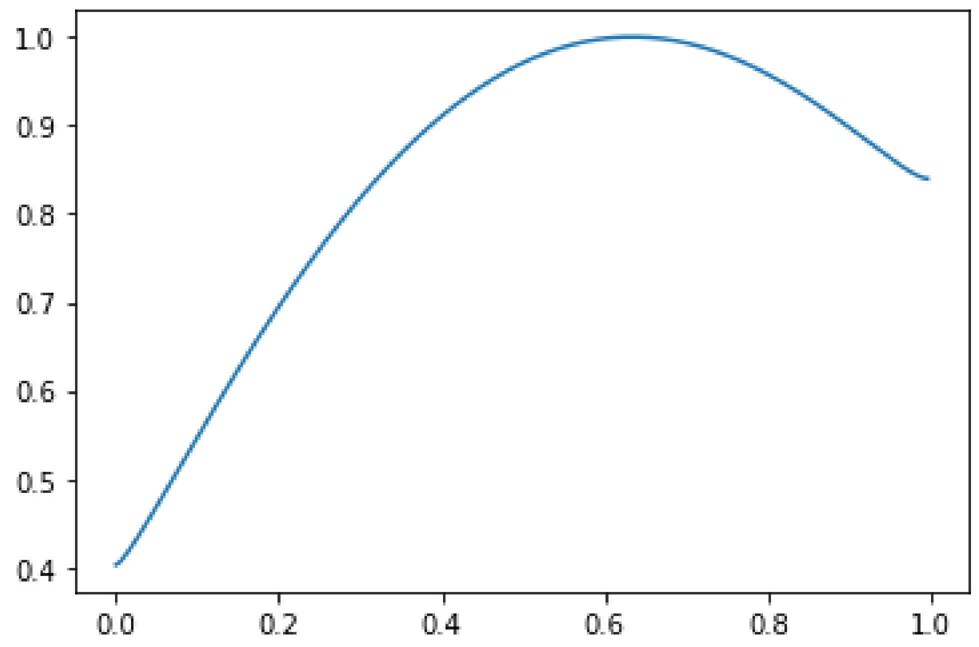
```
##########################
```



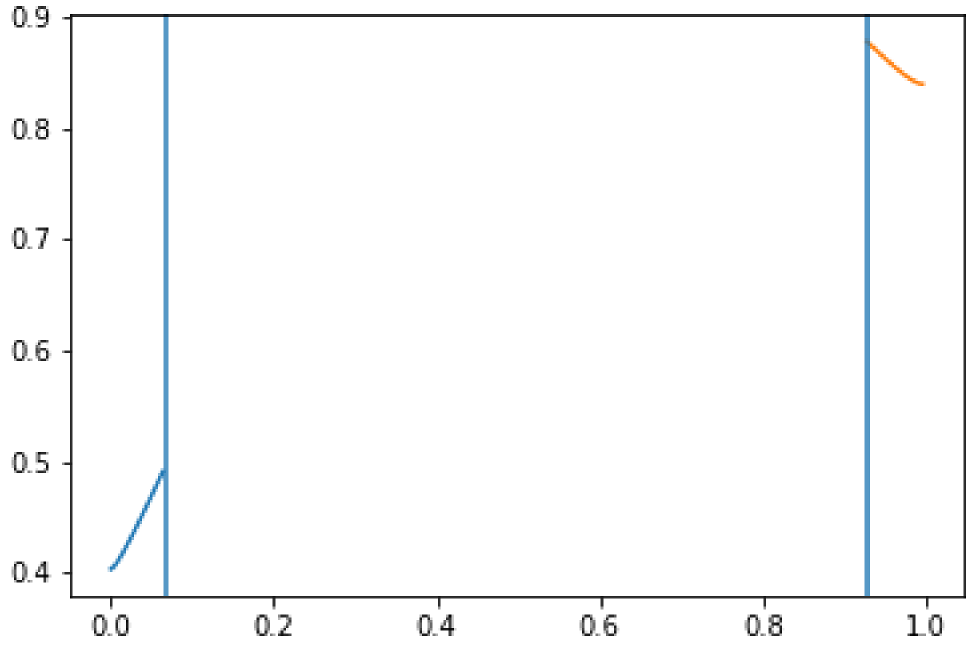
```
tensor([ 73, 187])
```



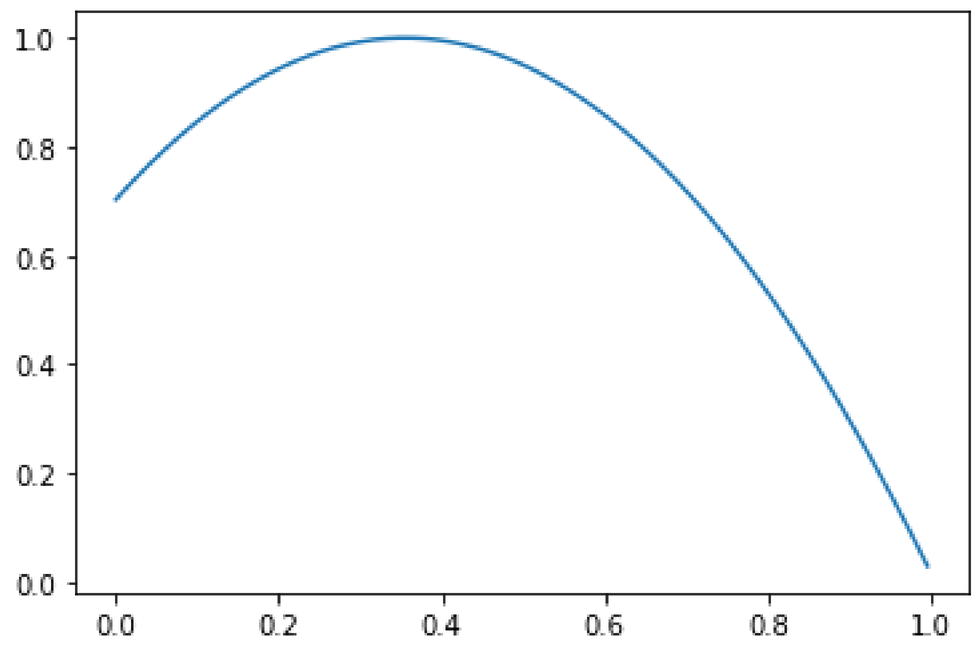
```
##########################
```



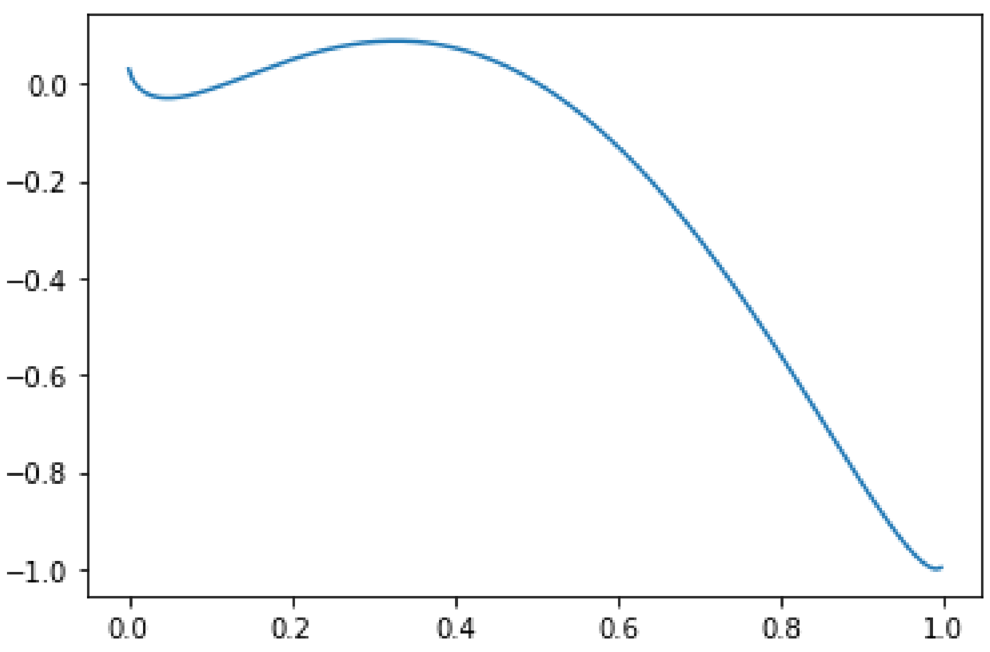
```
tensor([ 17, 232])
```



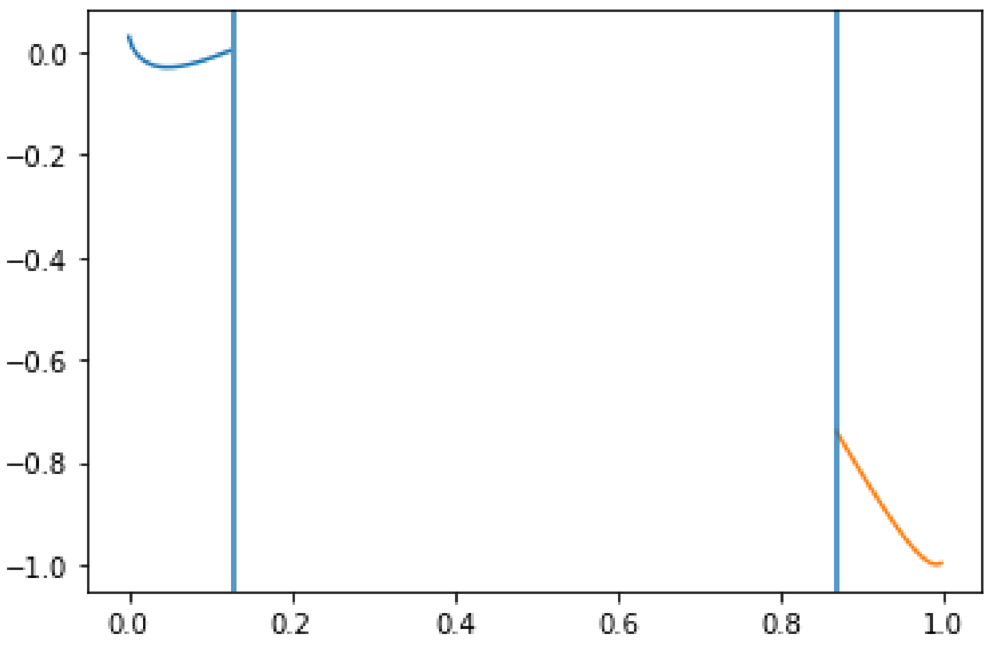
```
##########################
```



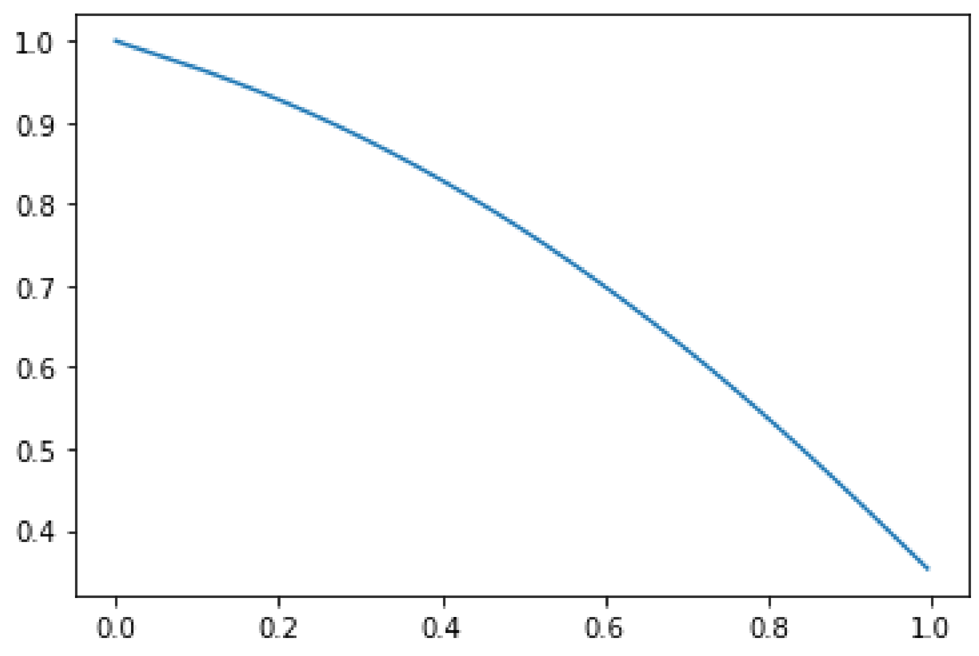
```
tensor([], dtype=torch.int64)
##########################
```



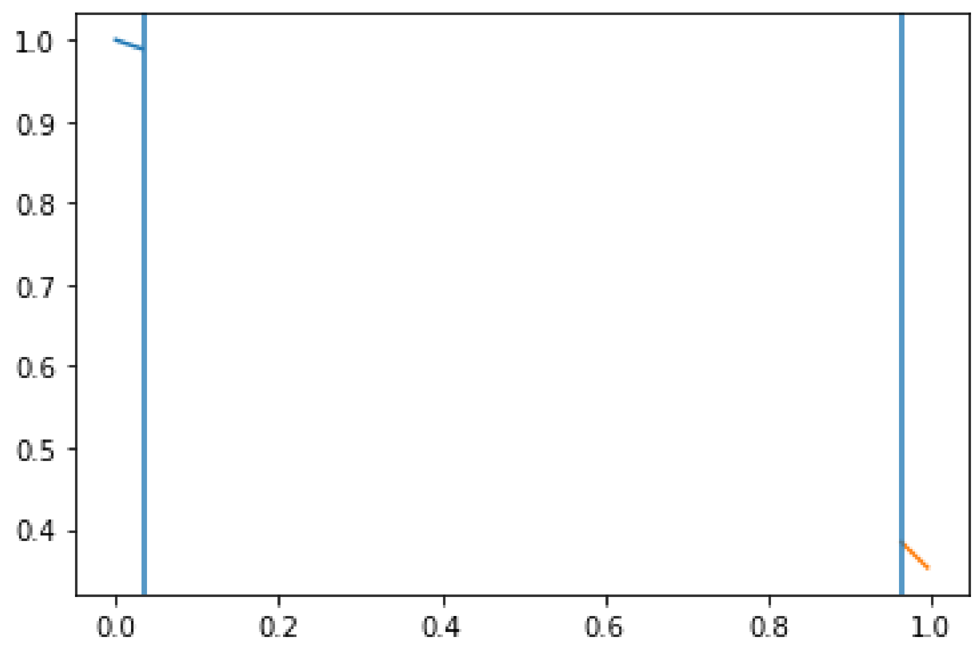
```
tensor([ 32, 217])
```



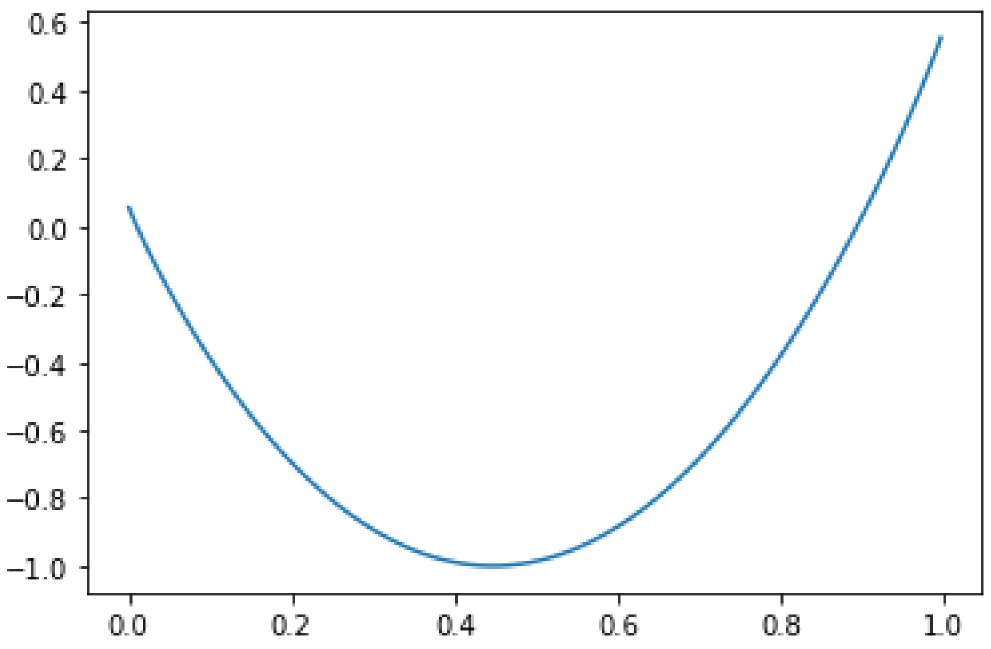
```
##########################
```


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0

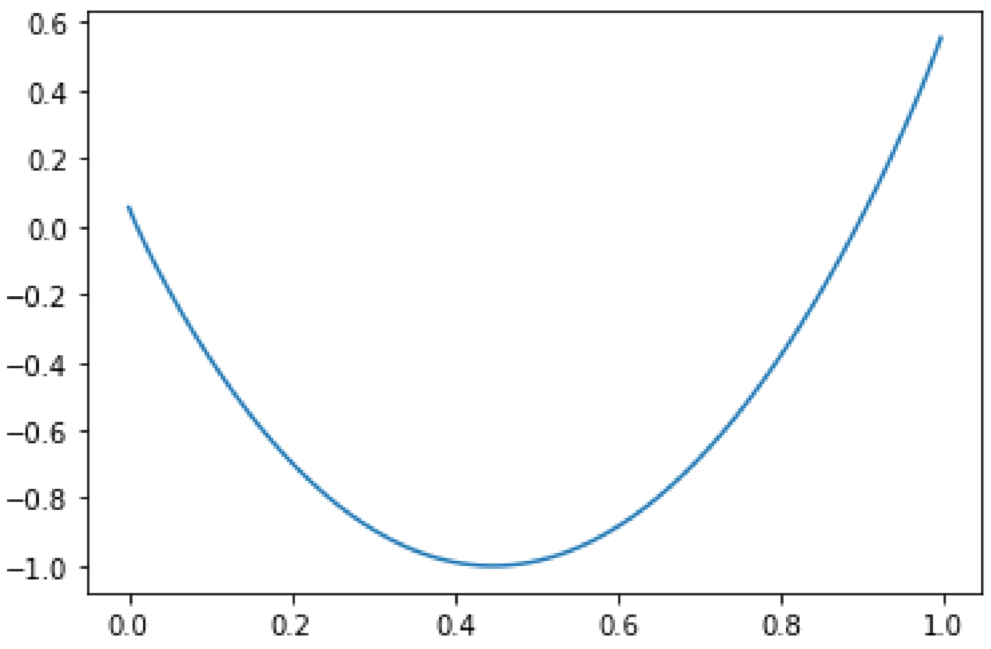
```
tensor([  9, 241])
```


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0

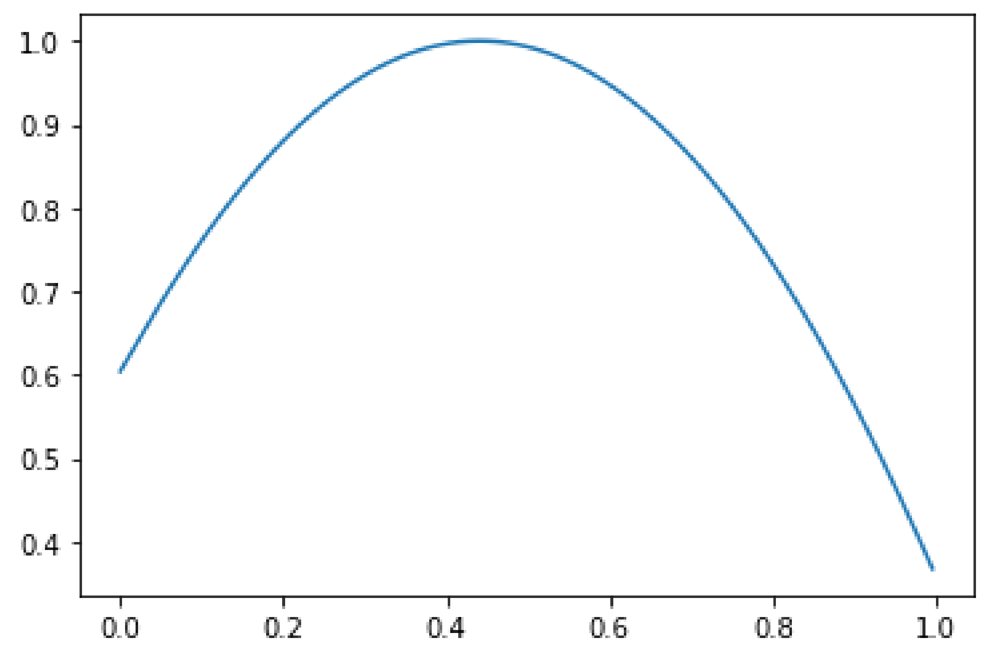
```
#########################
```



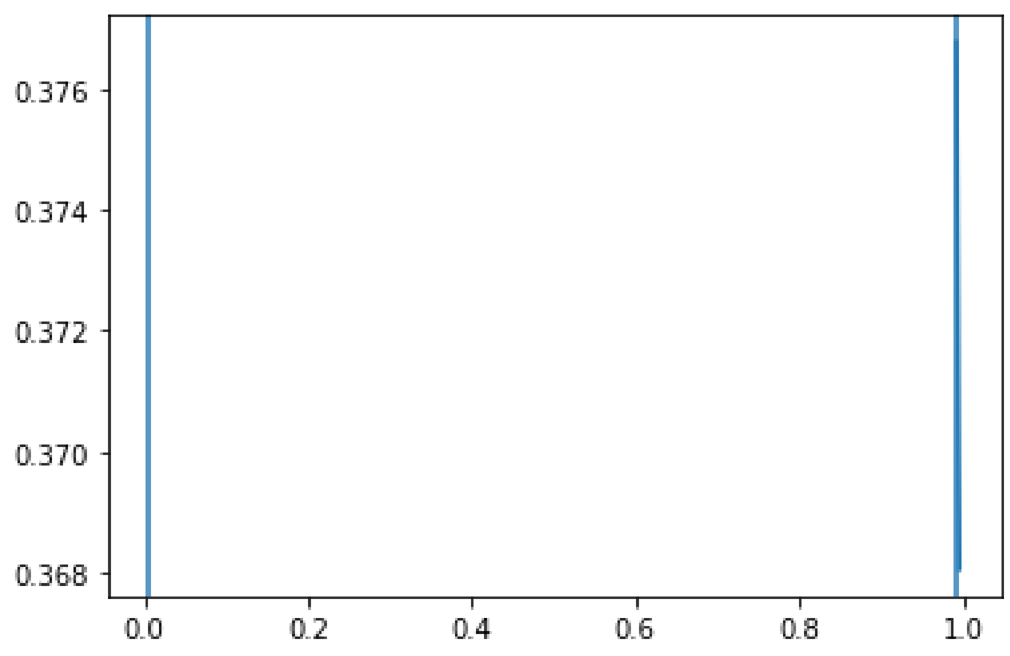
```
tensor([], dtype=torch.int64)
```



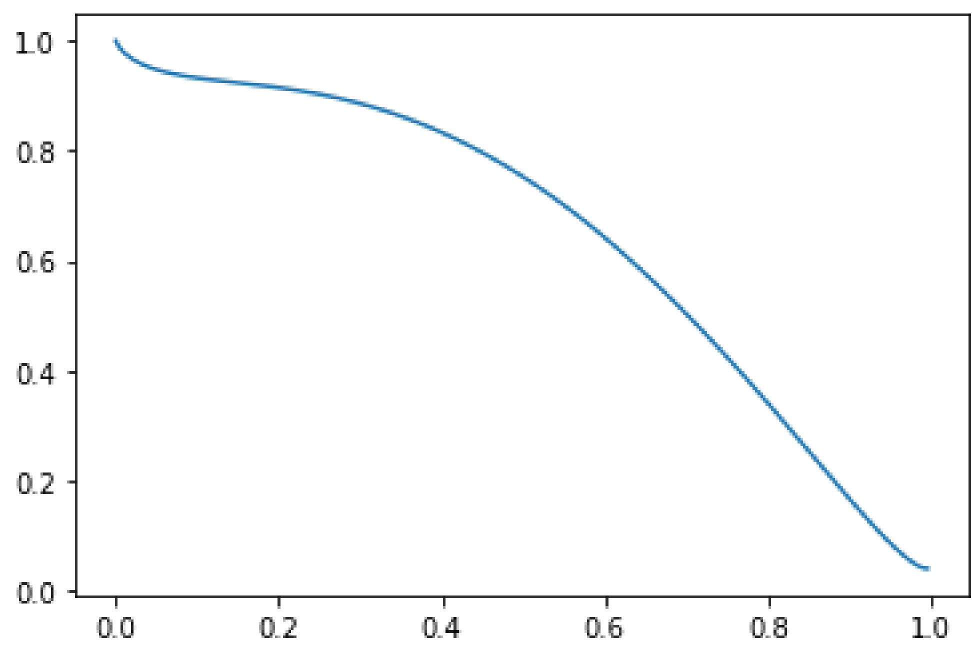
```
##########################
```


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0

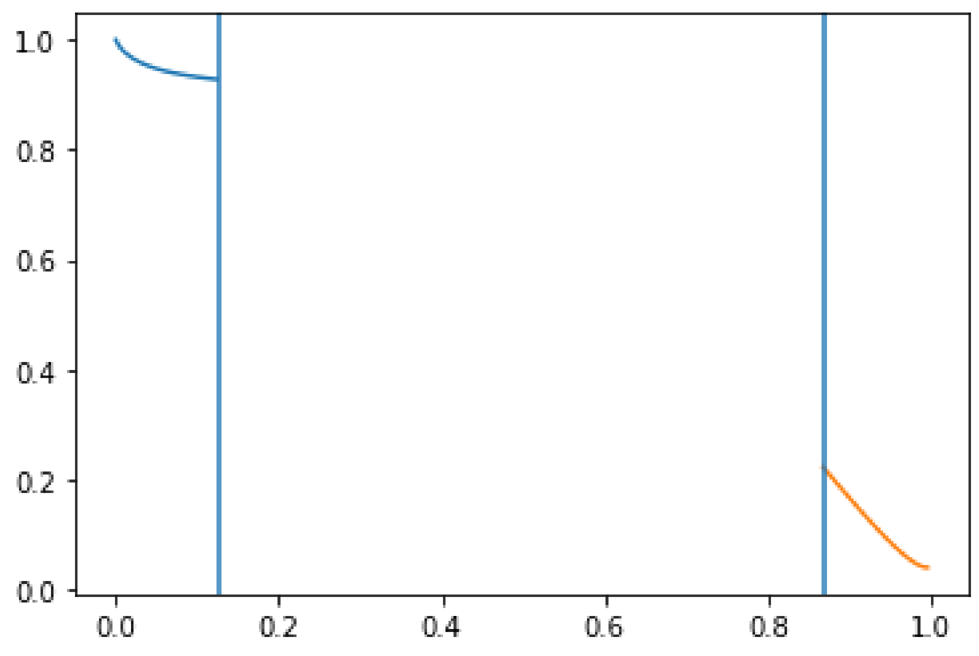
```
tensor([  1, 248])
```


0.376
0.374
0.372
0.370
0.368
0.0
0.2
0.4
0.6
0.8
1.0

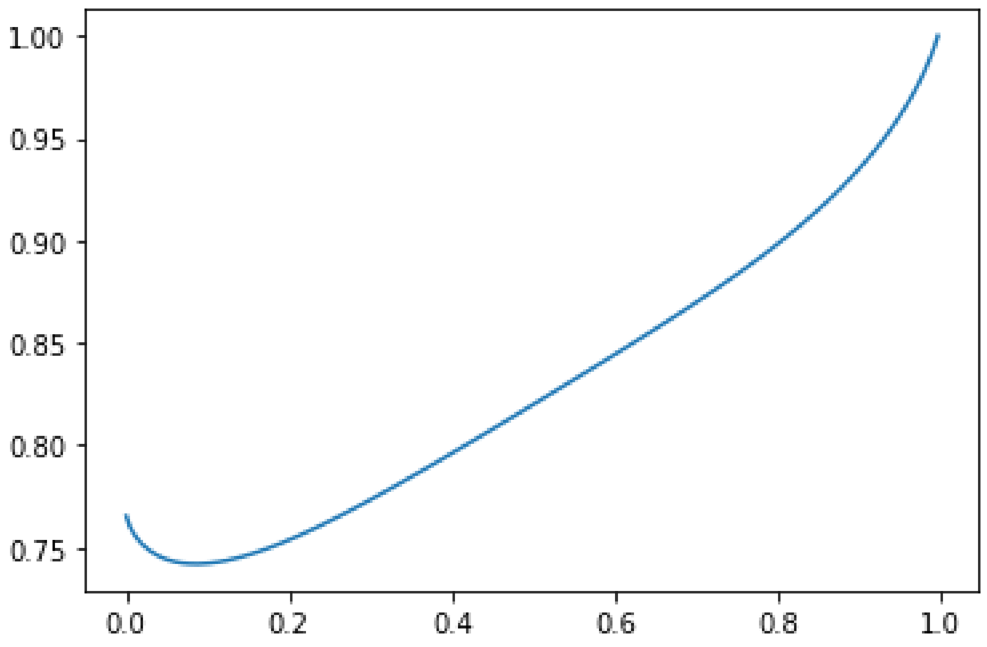
```
##########################
```



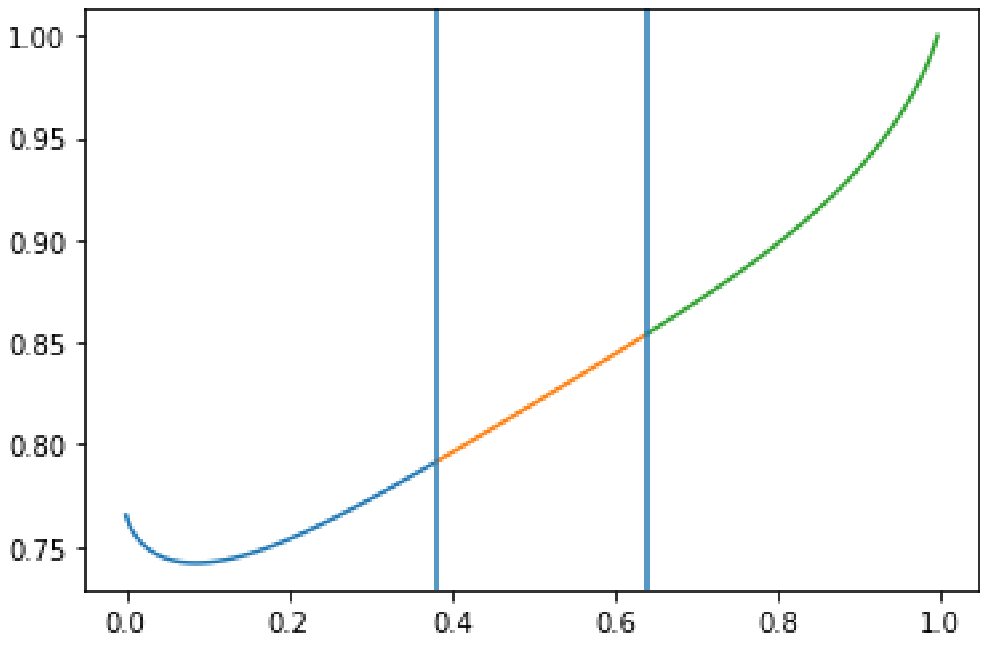
```
tensor([ 32, 217])
```



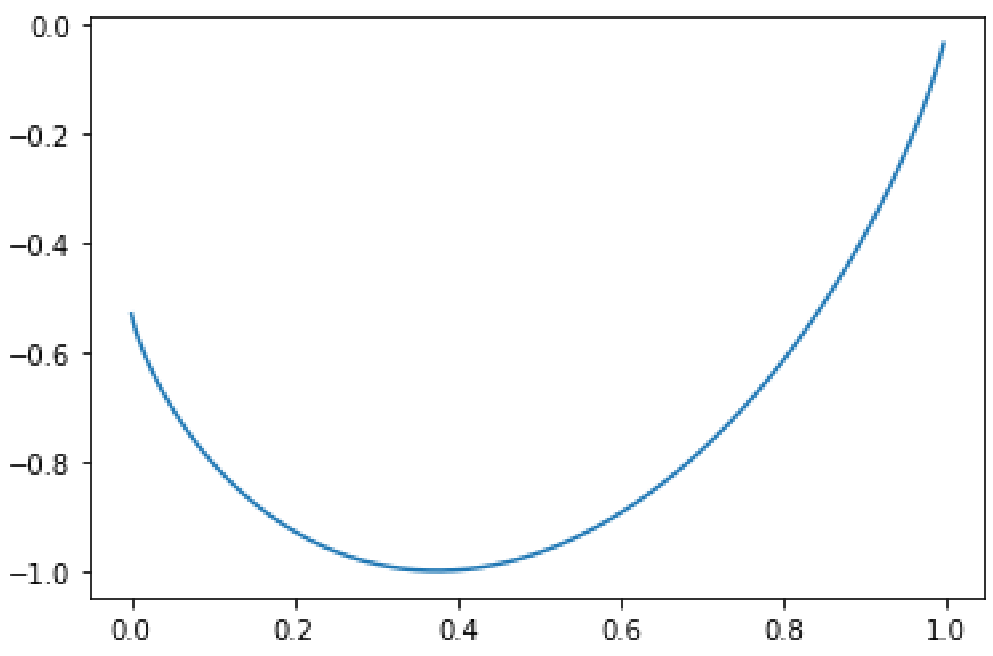
```
##########################
```



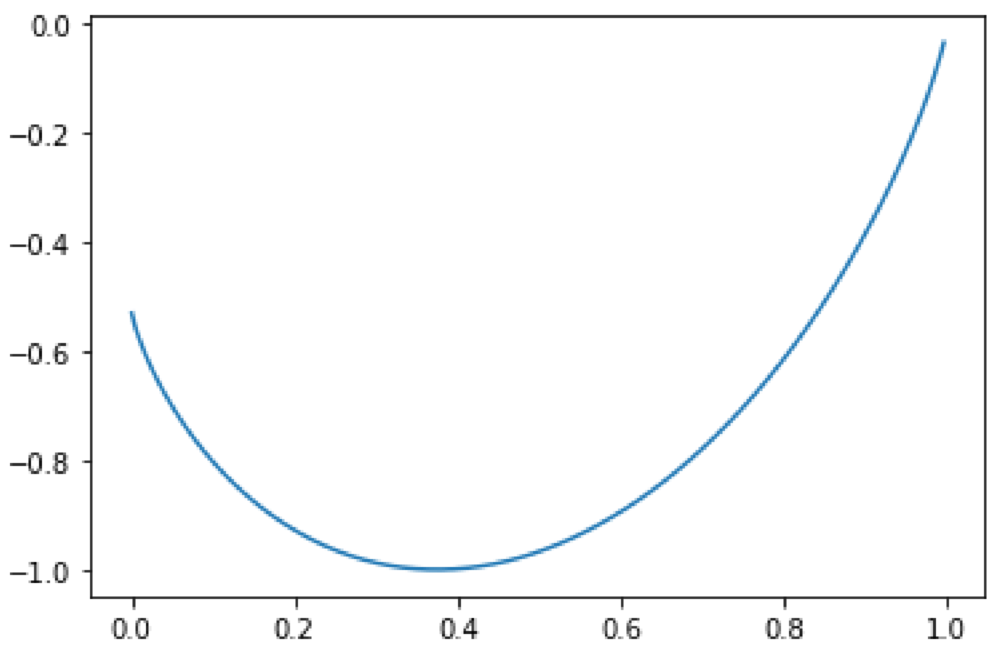
```
tensor([ 95, 160])
```



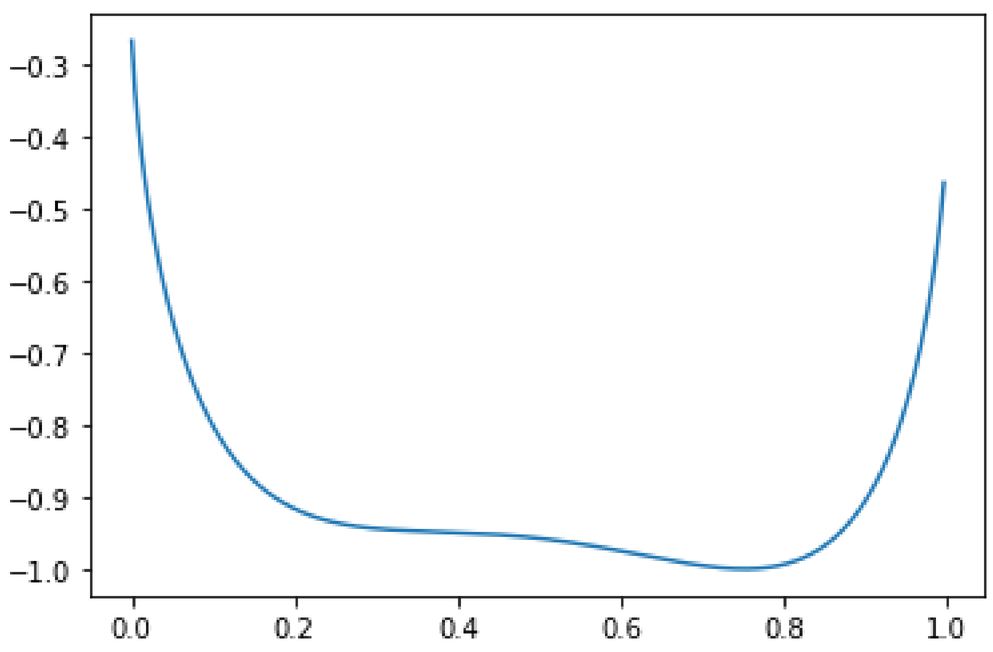
```
##########################
```



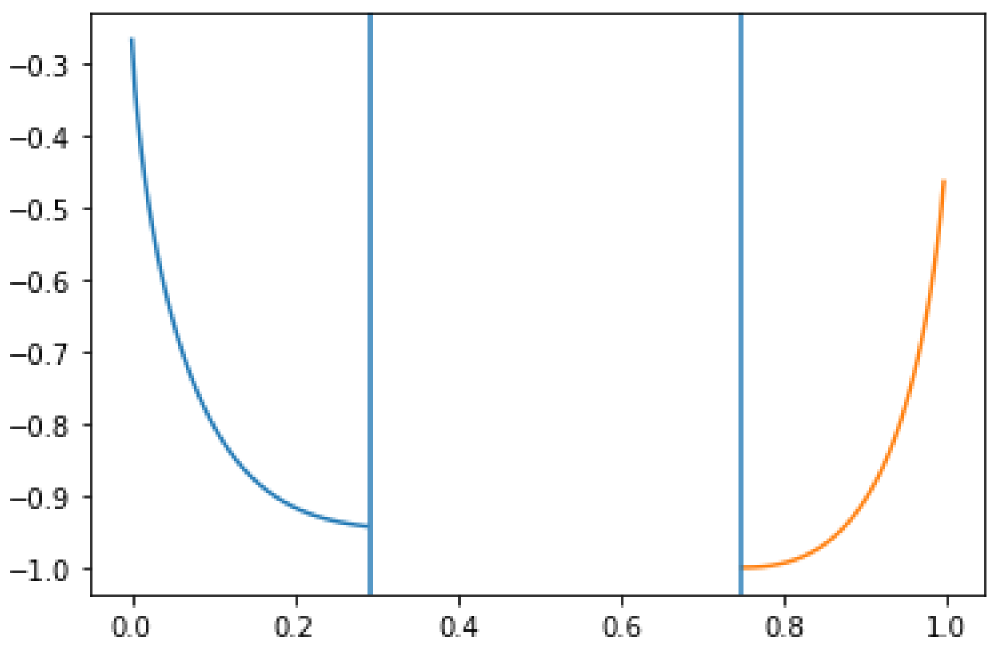
```
tensor([], dtype=torch.int64)
```



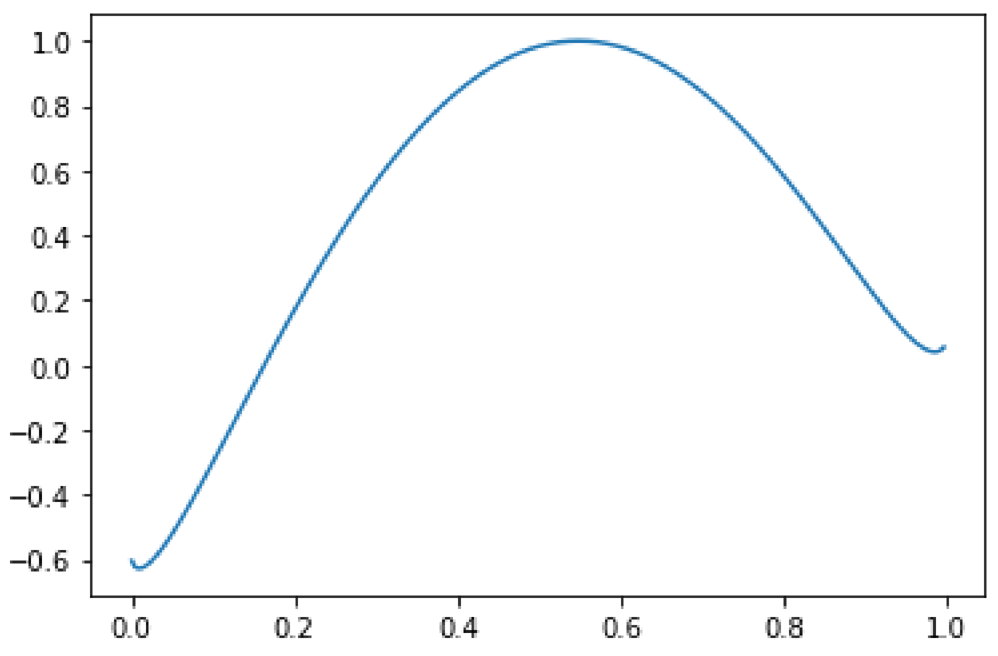
```
#########################
```



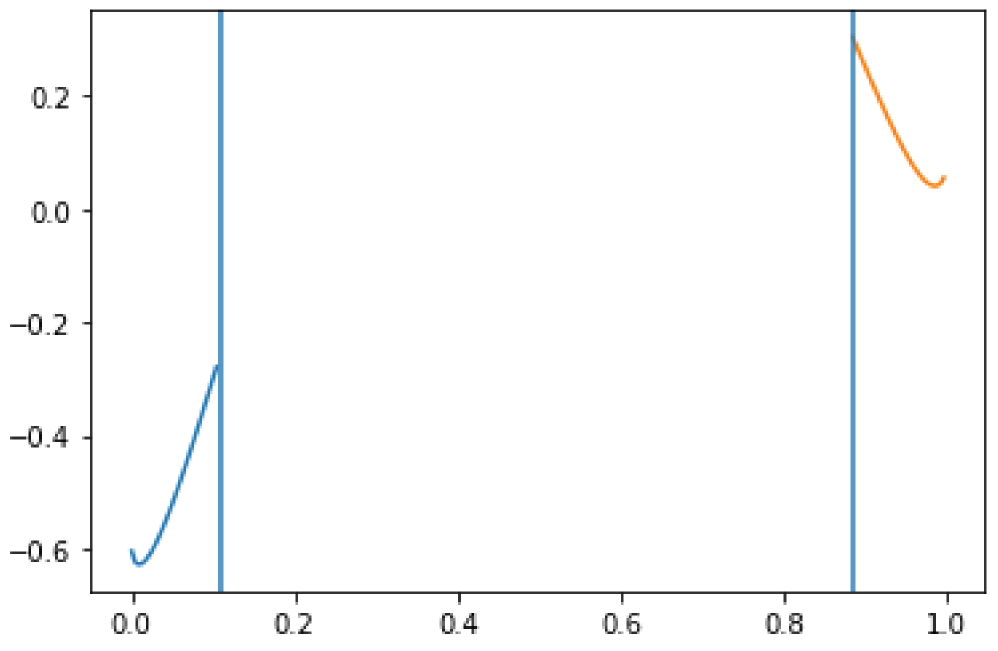
```
tensor([ 73, 187])
```



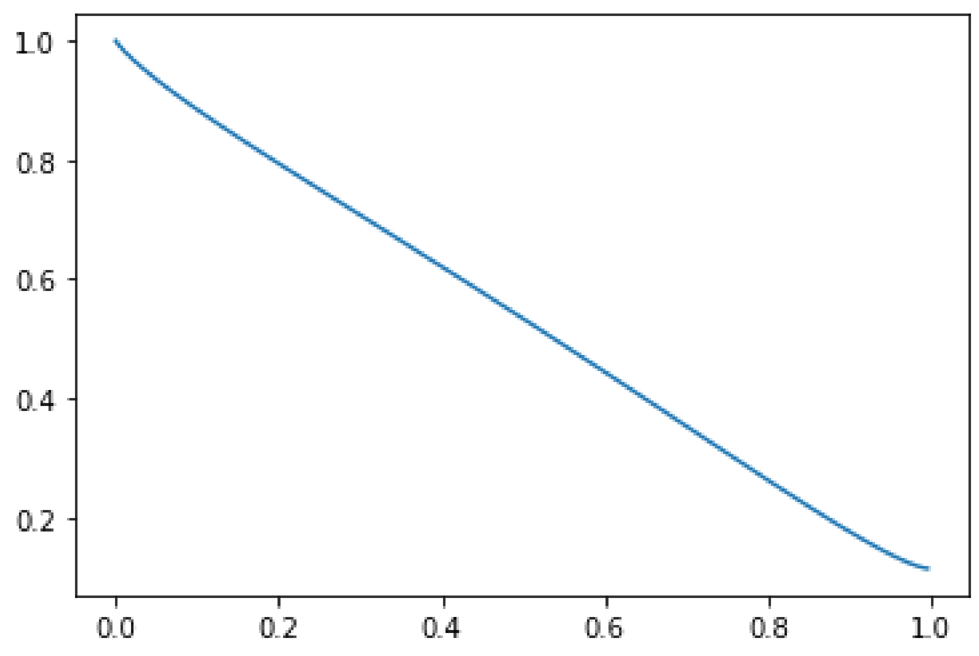
```
##########################
```



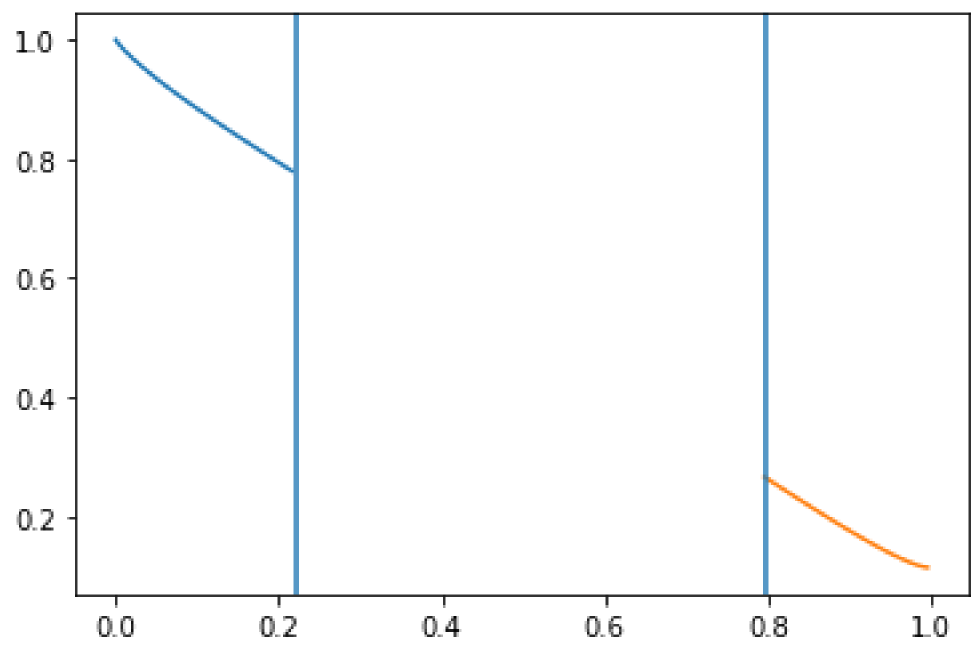
```
tensor([ 27, 221])
```



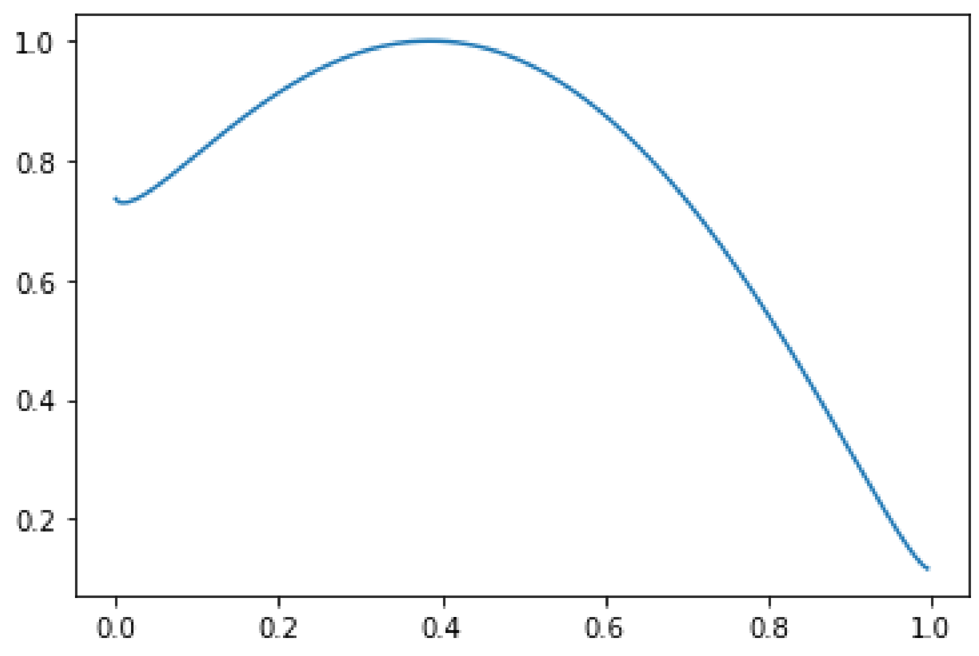
```
##########################
```



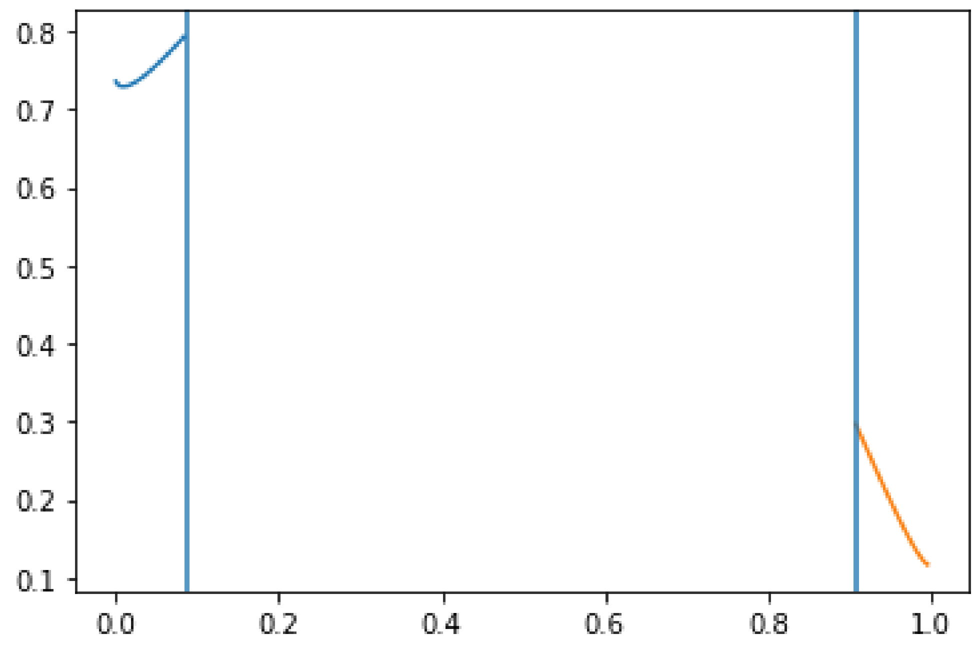
```
tensor([ 55, 199])
```



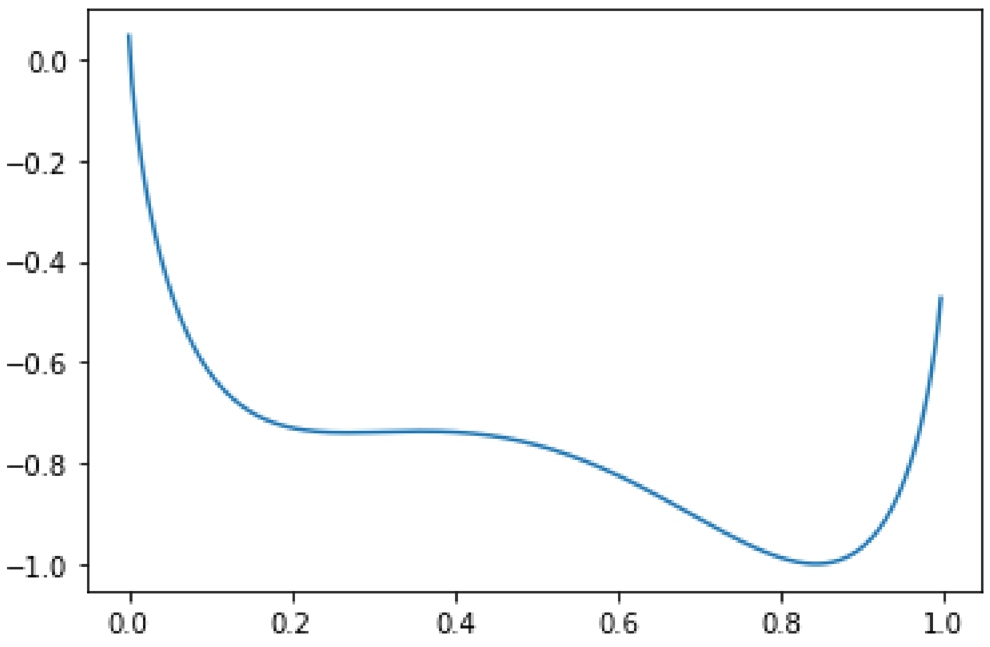
```
#########################
```



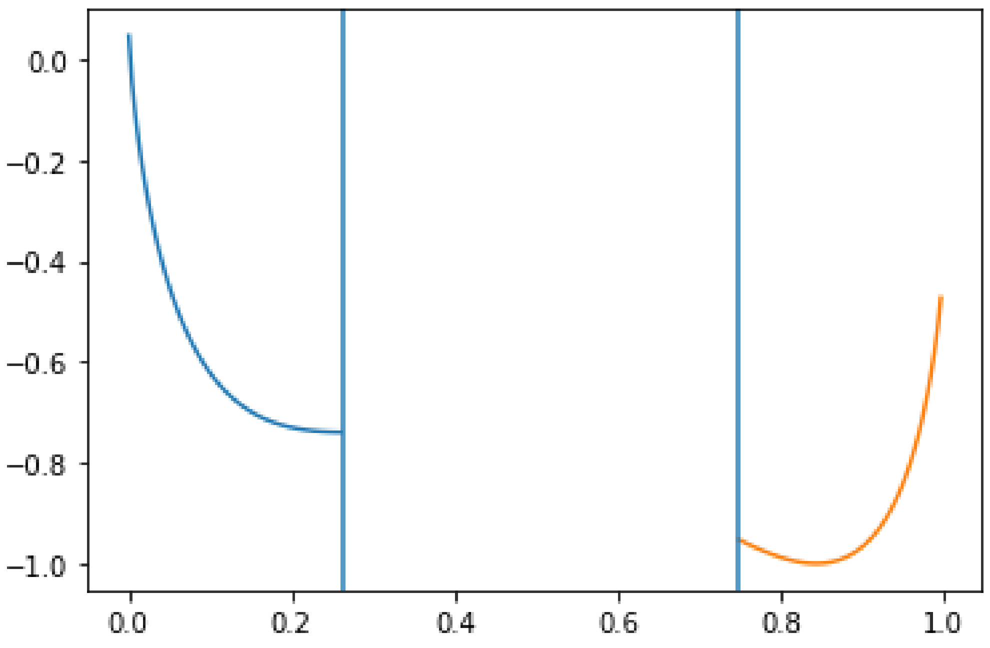
```
tensor([ 22, 227])
```



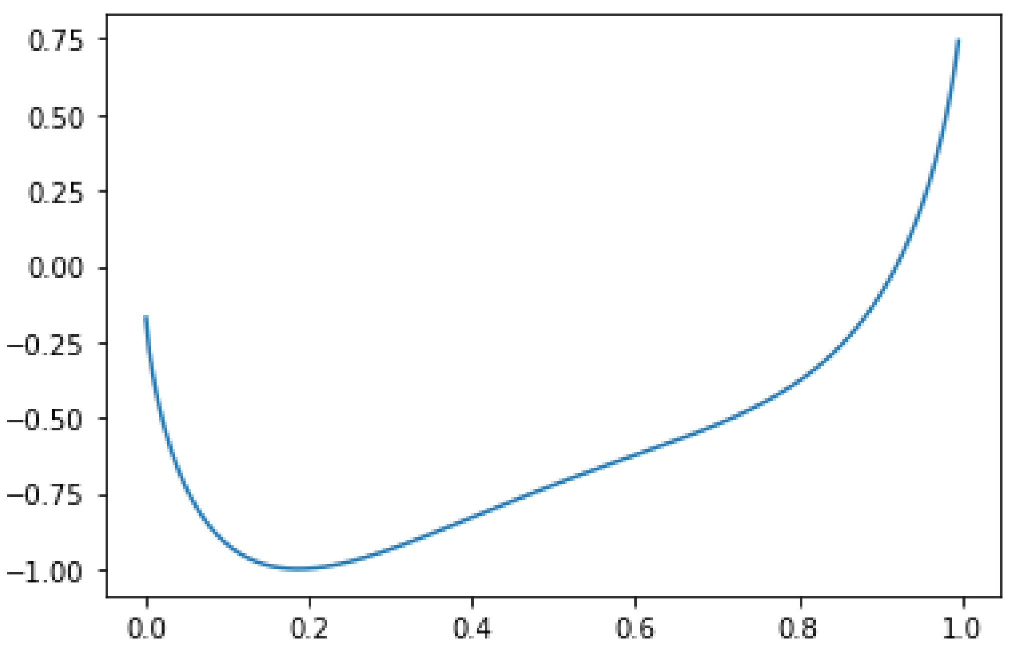
```
##########################
```



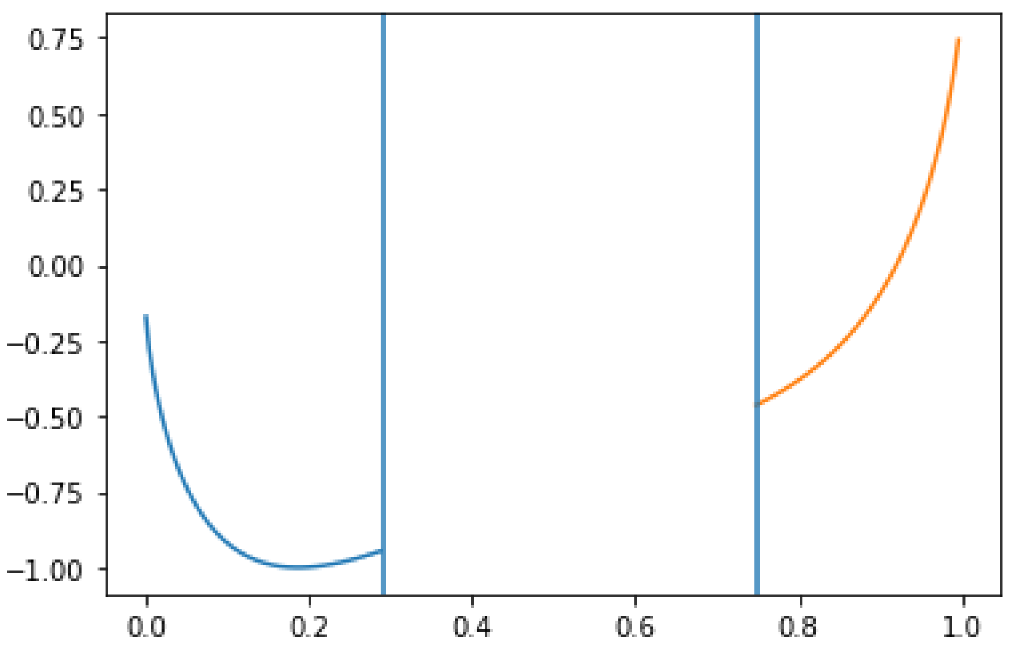
```
tensor([ 66, 187])
```



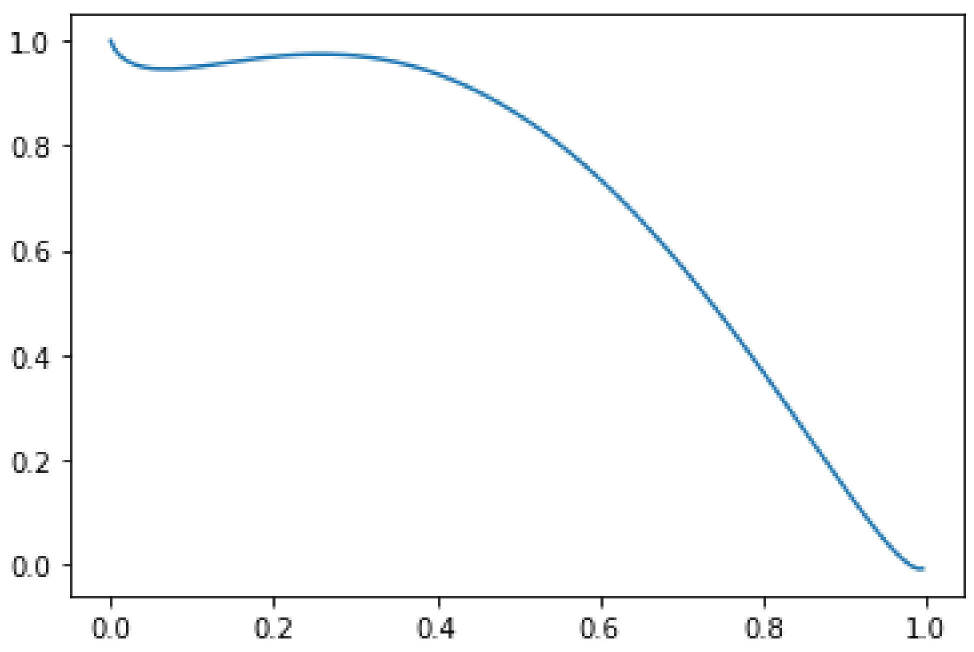
```
##########################
```



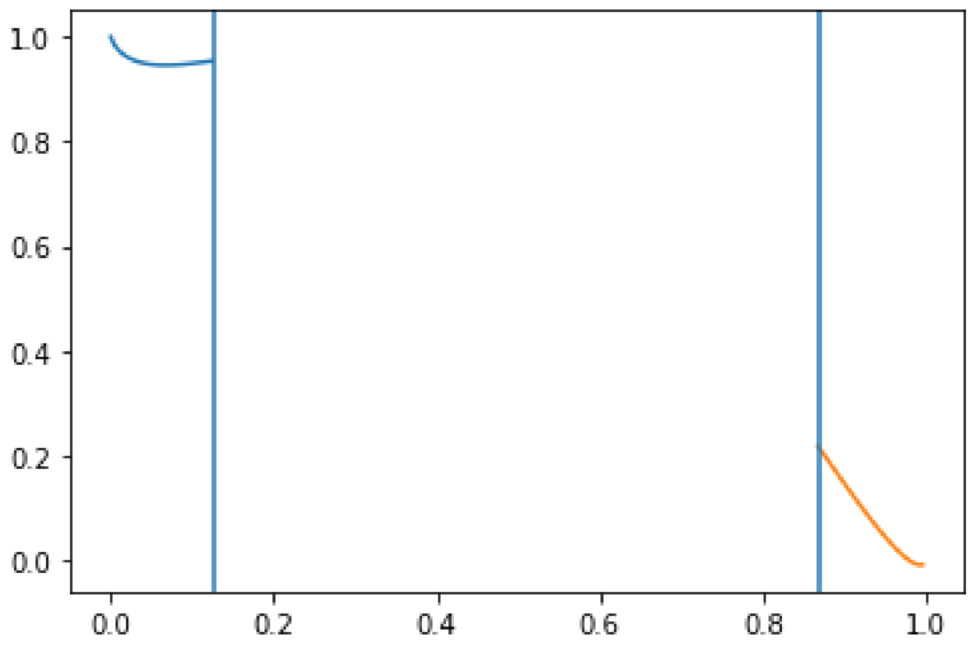
```
tensor([ 73, 187])
```



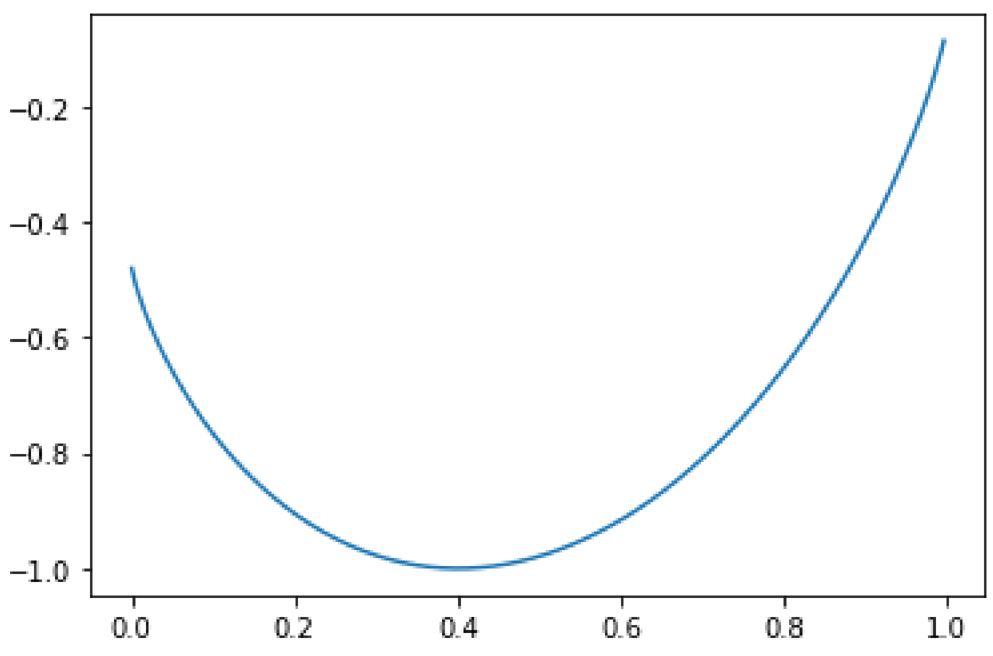
```
##########################
```



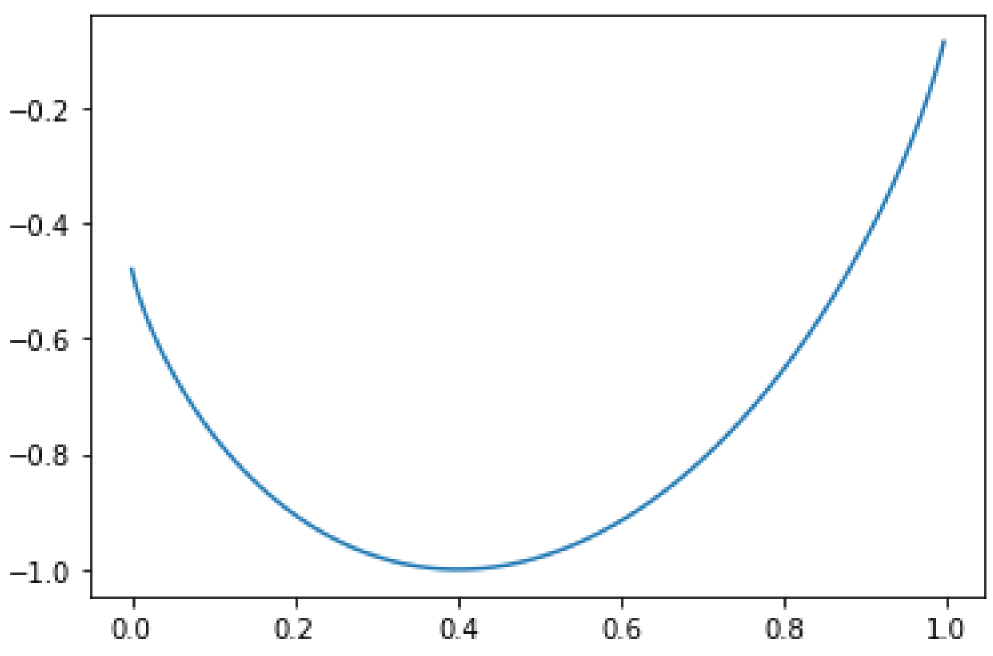
```
tensor([ 32, 217])
```



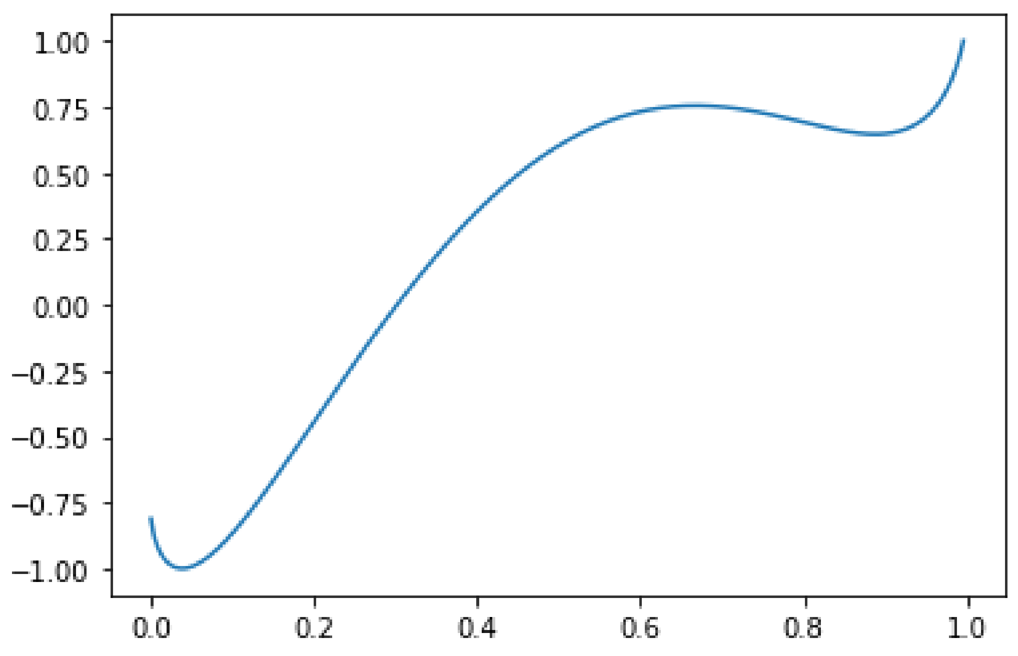
```
#########################
```



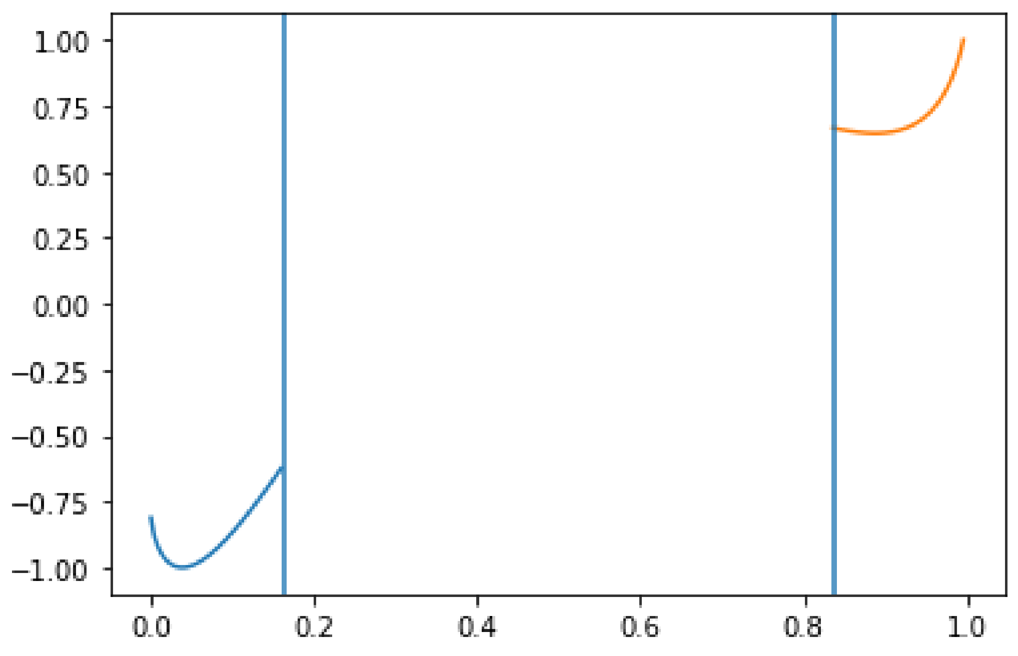
```
tensor([], dtype=torch.int64)
```



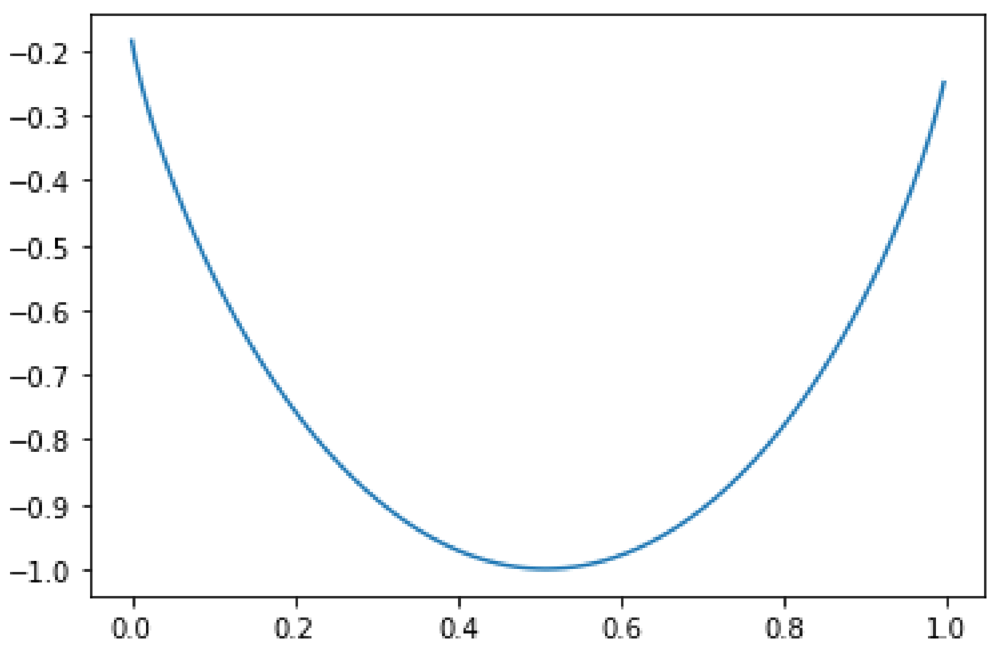
```
##########################
```



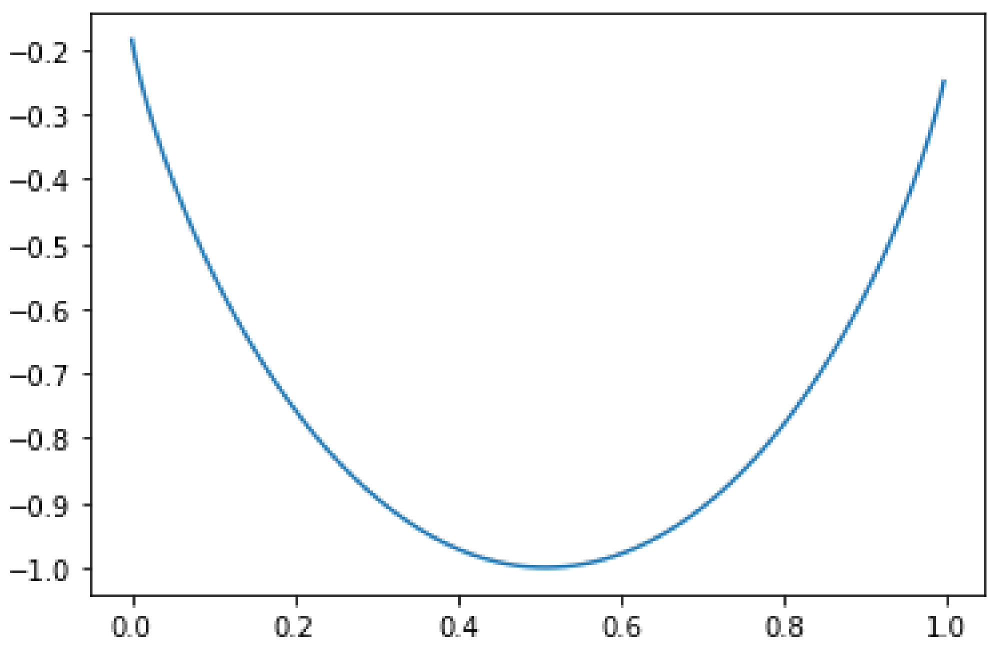
```
tensor([ 41, 209])
```



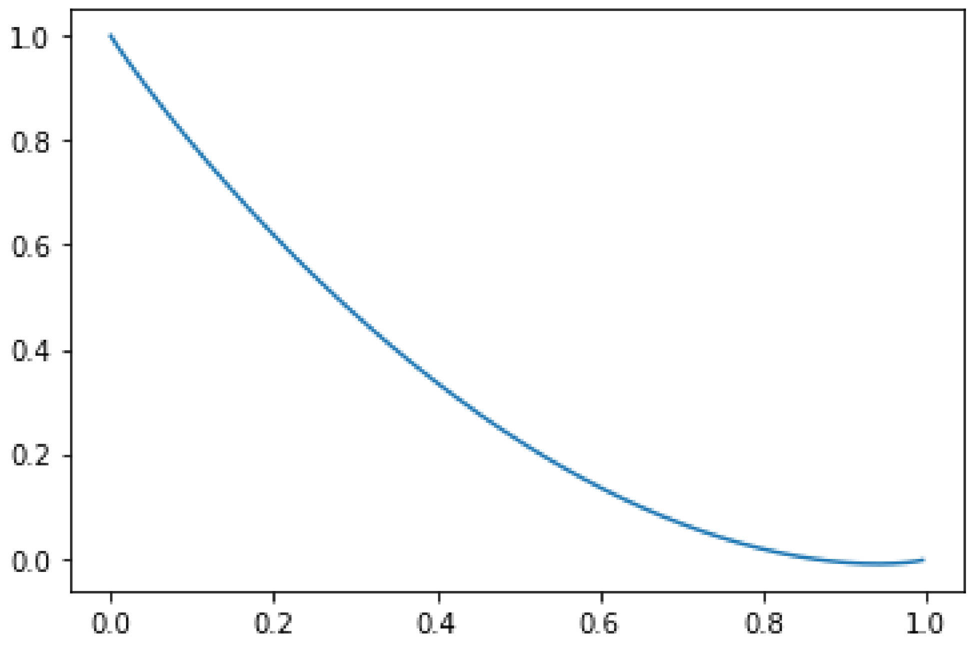
```
#########################
```



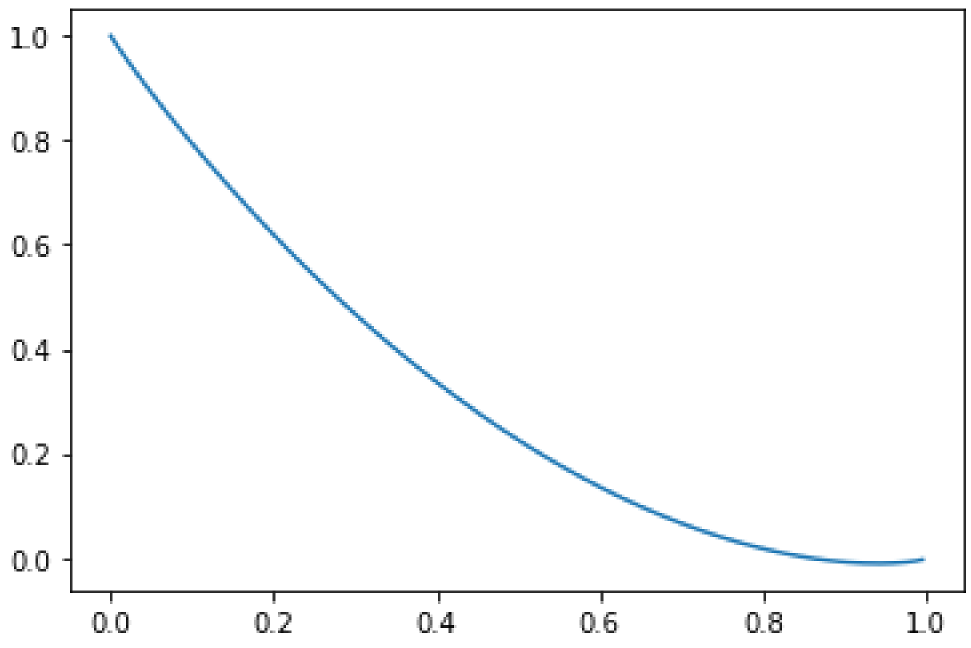
```
tensor([], dtype=torch.int64)
```



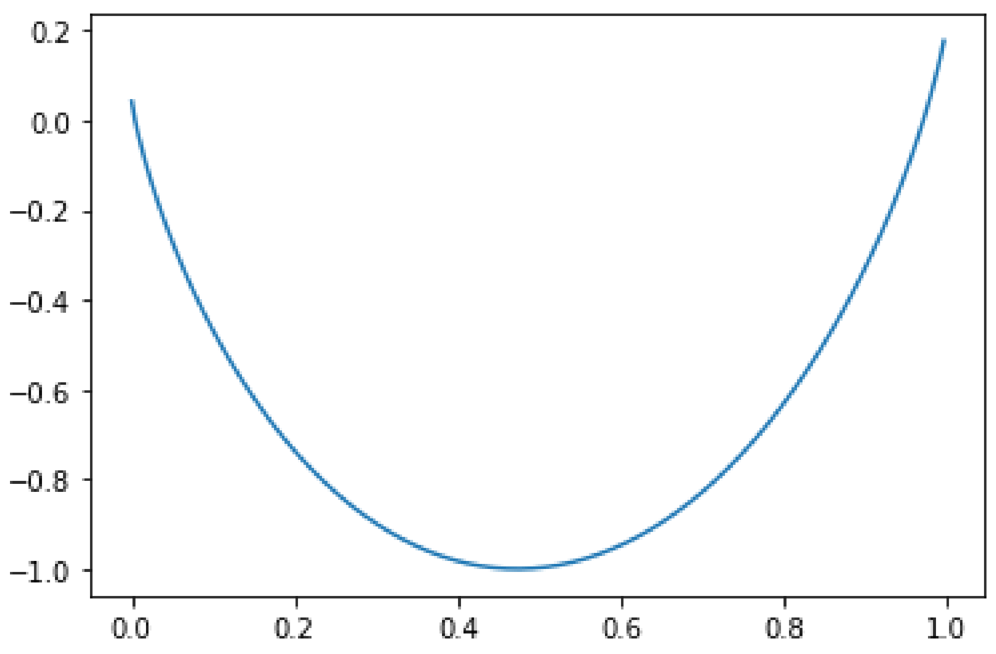
```
##########################
```



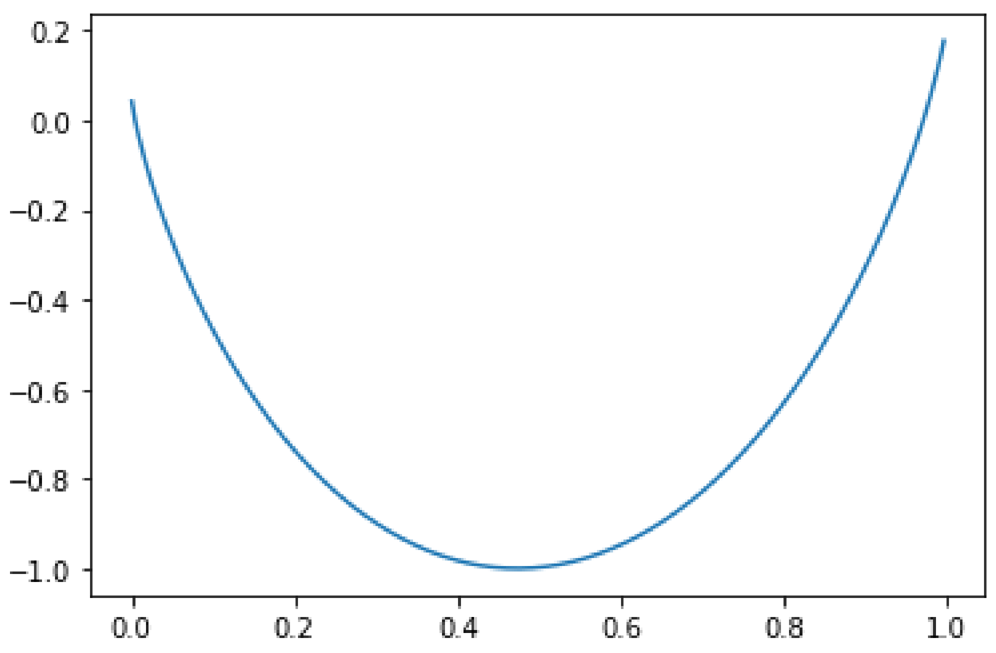
```
tensor([], dtype=torch.int64)
```



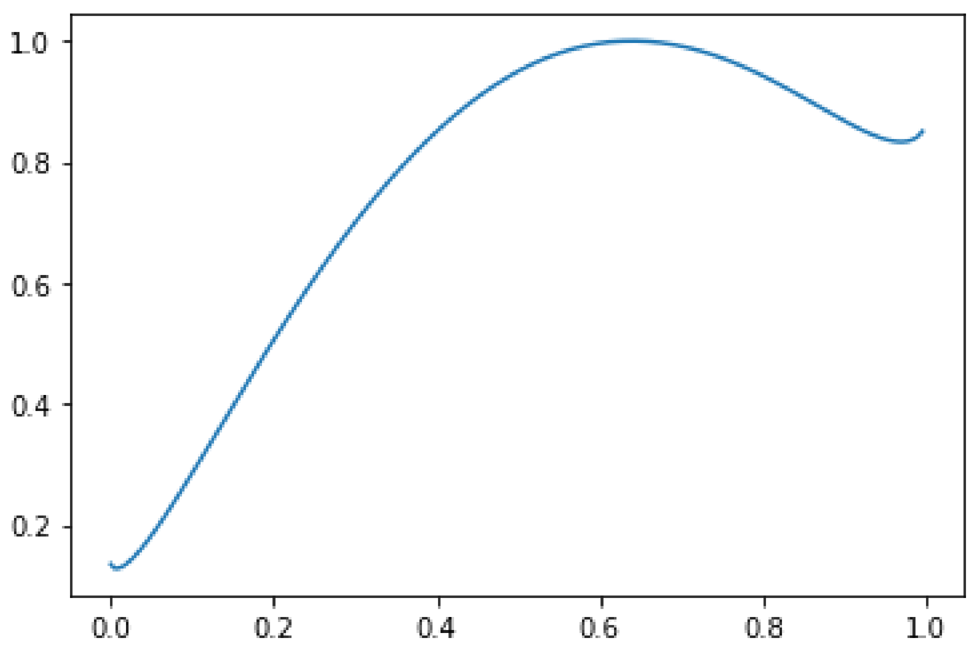
```
########################
```



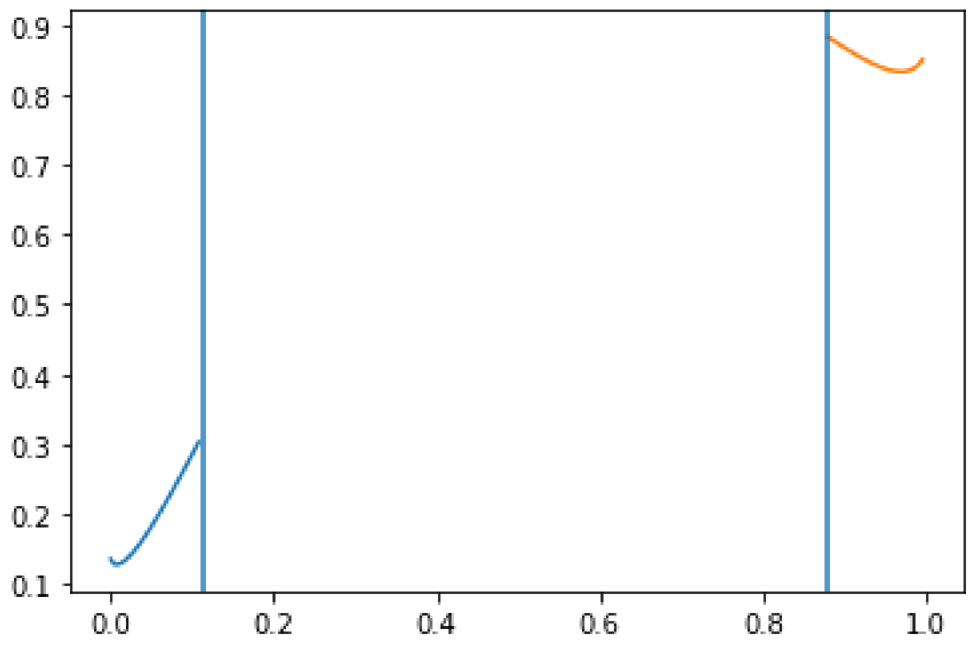
```
tensor([], dtype=torch.int64)
```



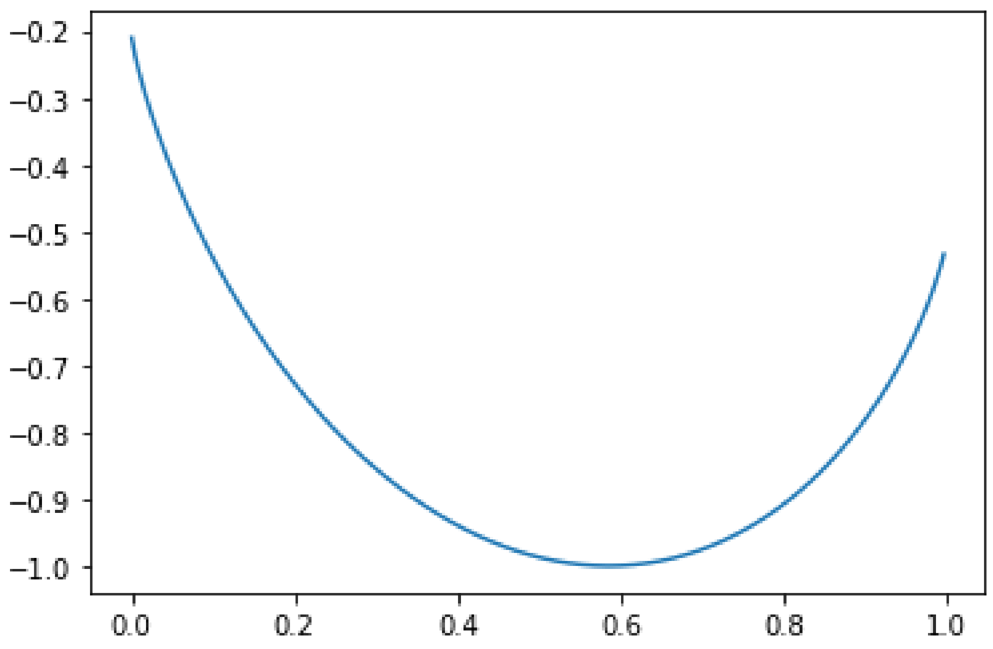
```
#########################
```



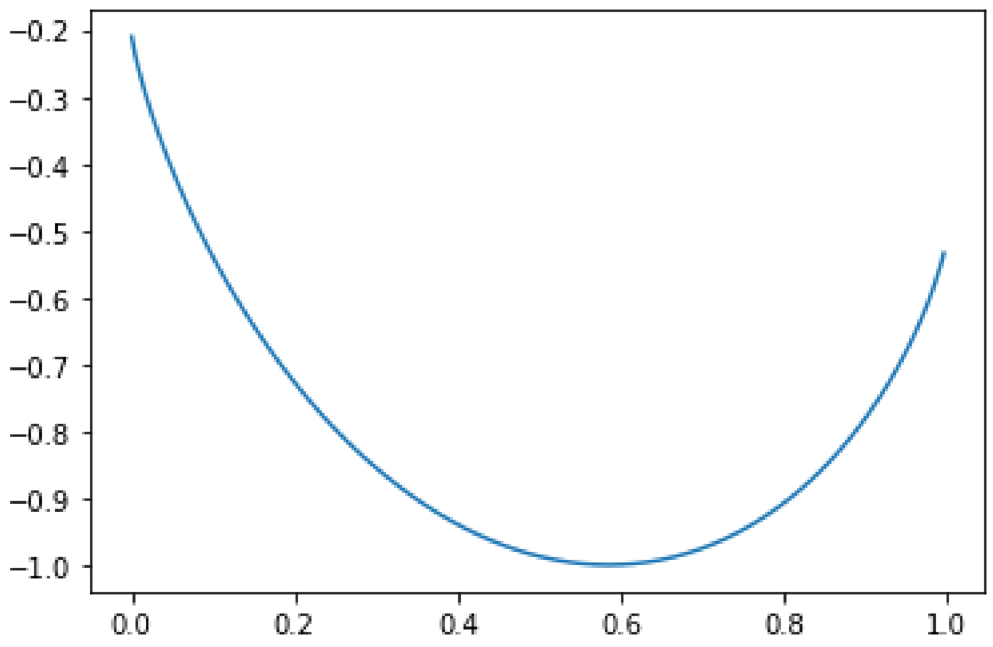
```
tensor([ 28, 220])
```



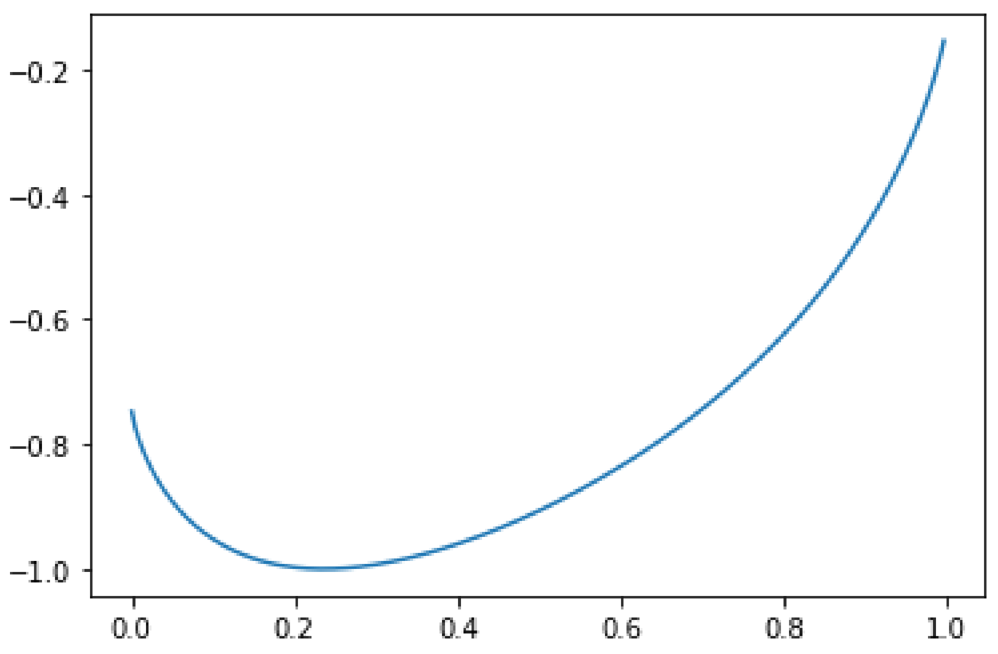
```
#########################
```



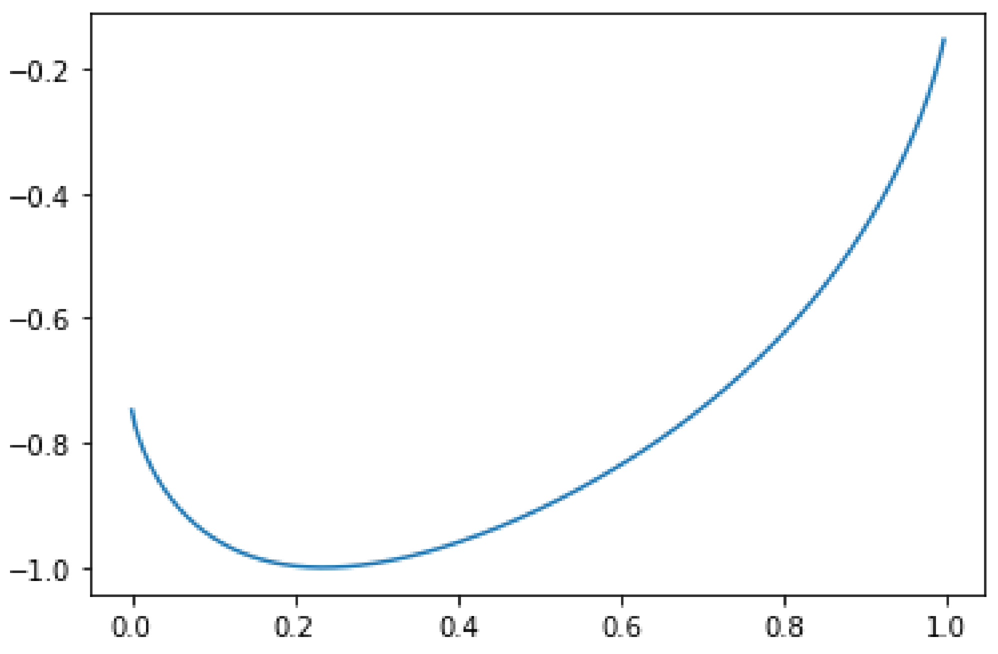
```
tensor([], dtype=torch.int64)
```



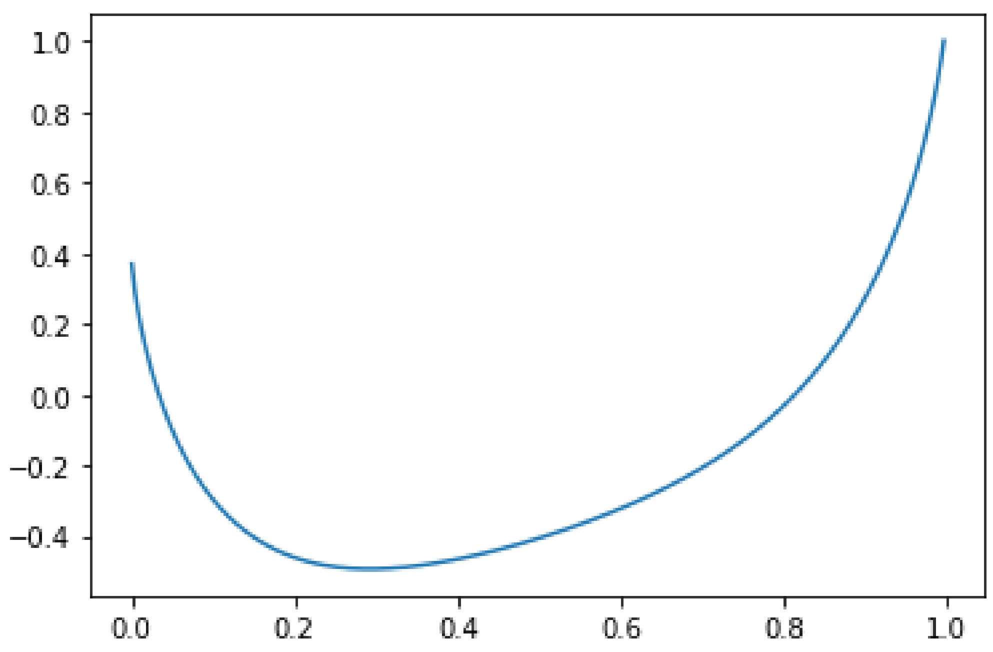
```
########################
```



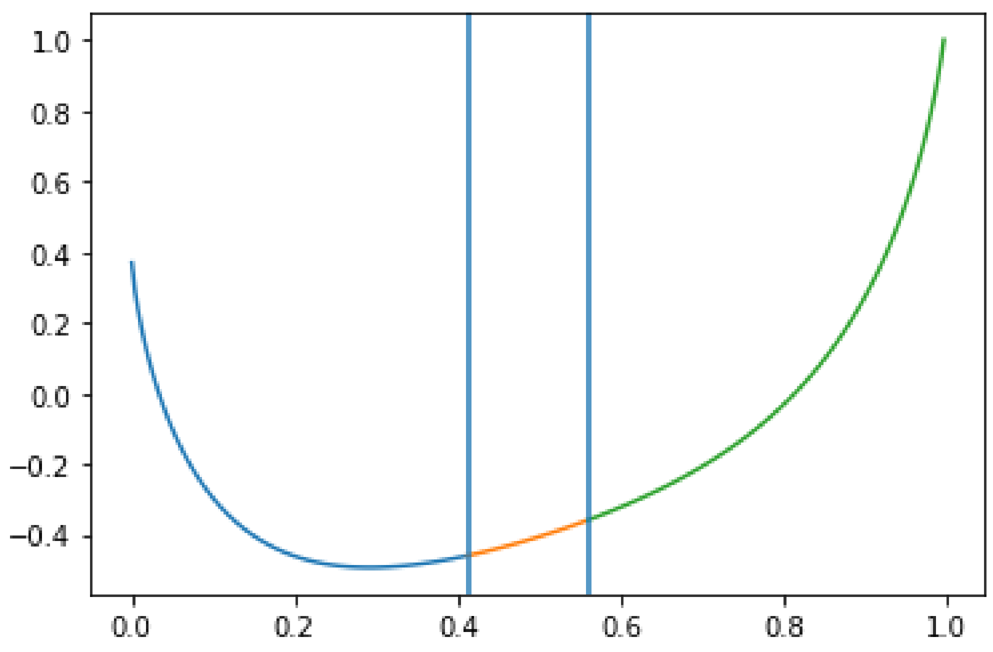
```
tensor([], dtype=torch.int64)
```



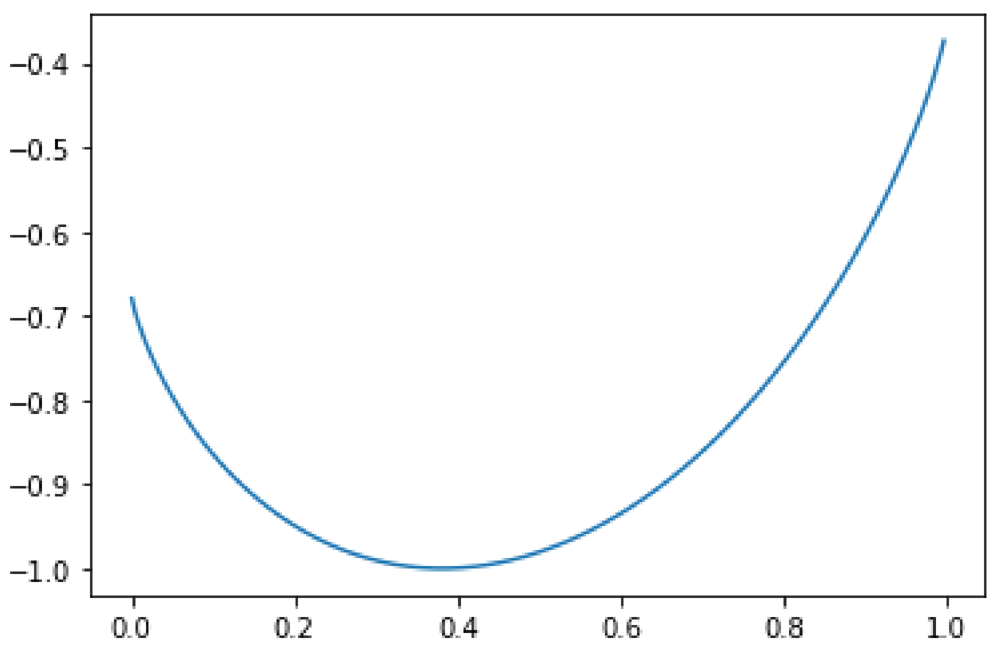
```
##########################
```



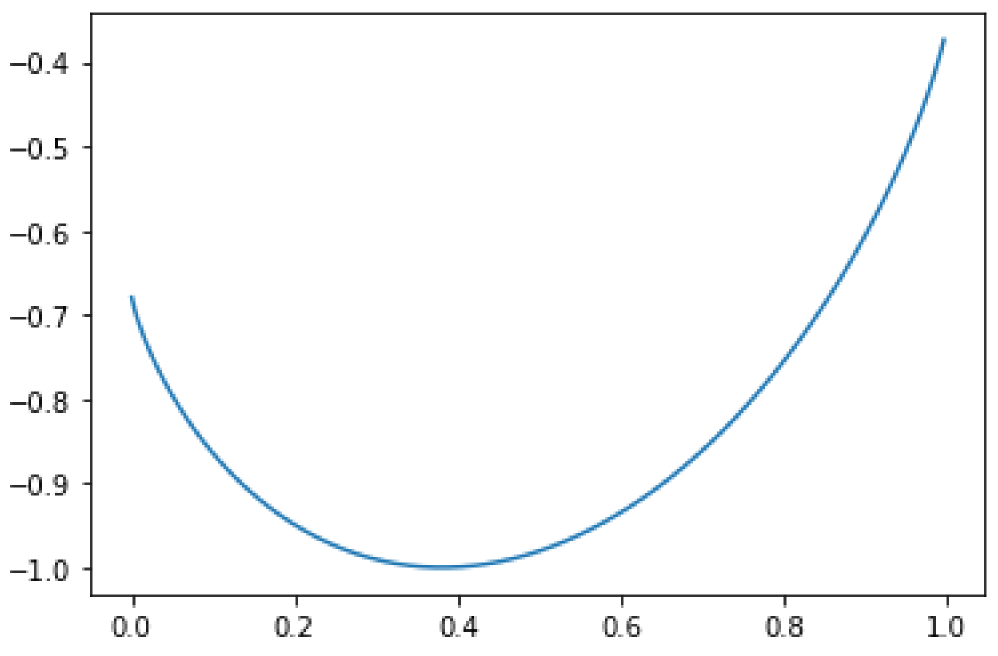
```
tensor([103, 140])
```



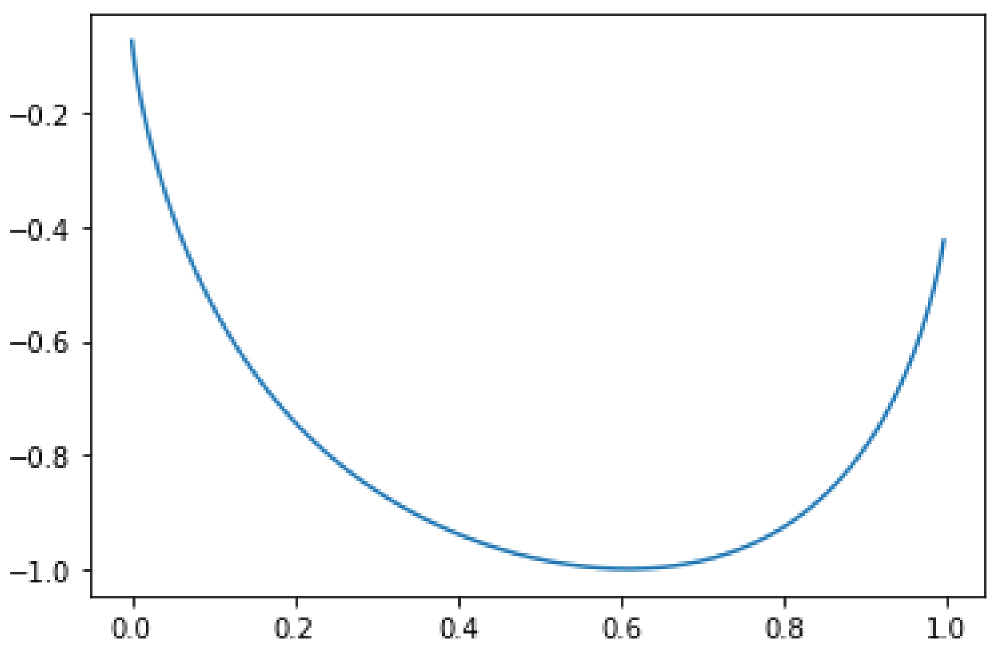
```
##########################
```



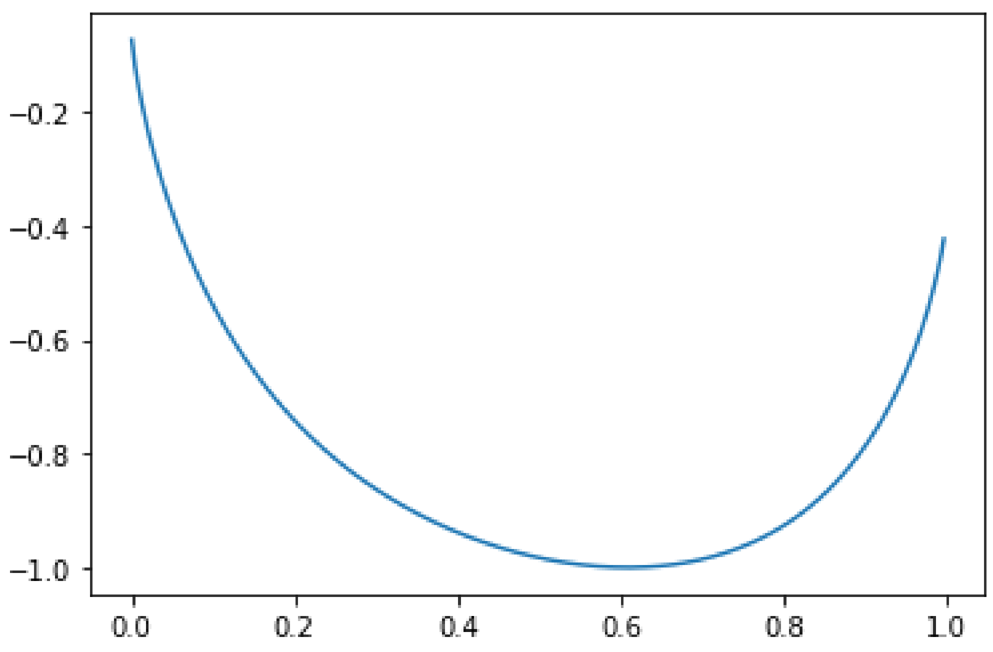
```
tensor([], dtype=torch.int64)
```



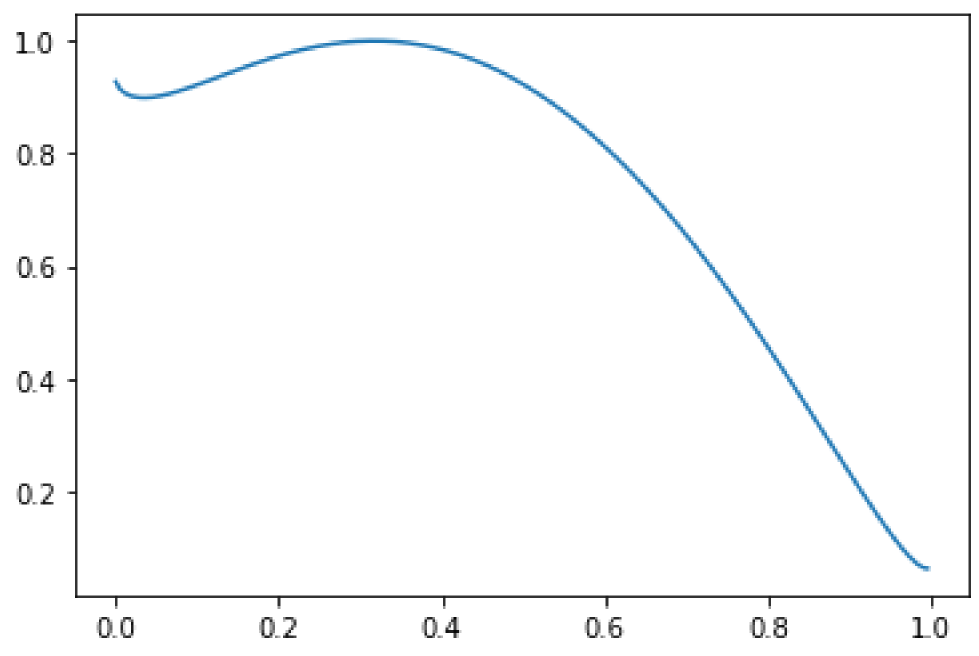
```
#########################
```



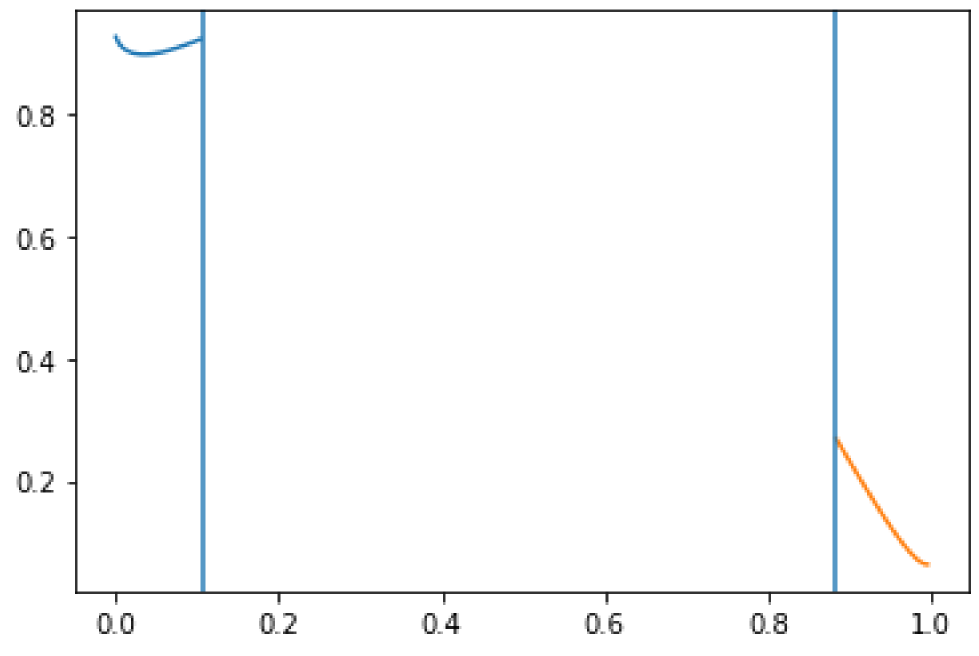
```
tensor([], dtype=torch.int64)
```



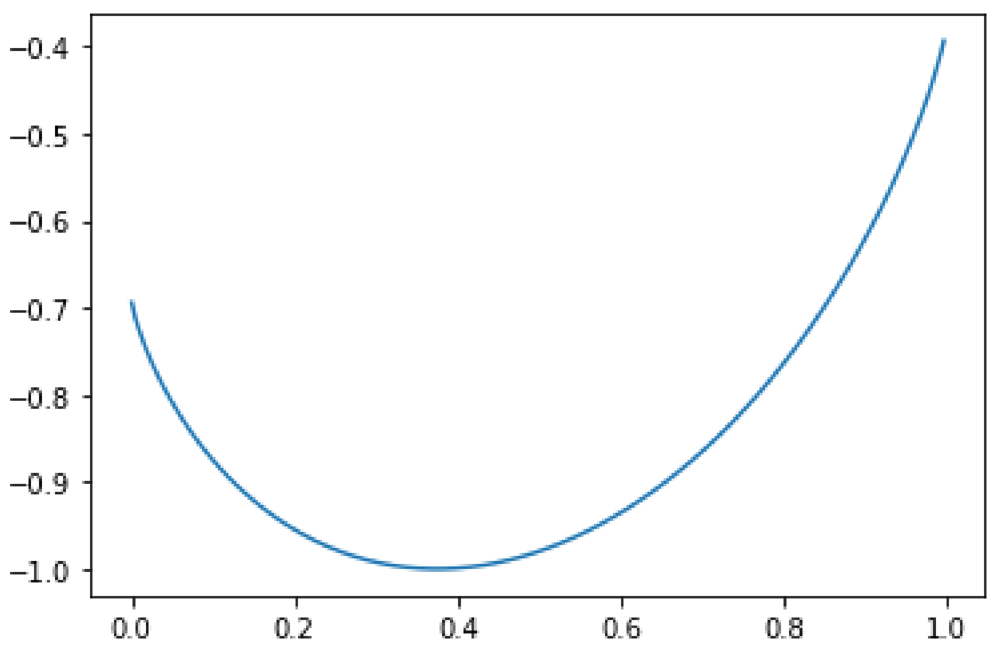
```
#########################
```



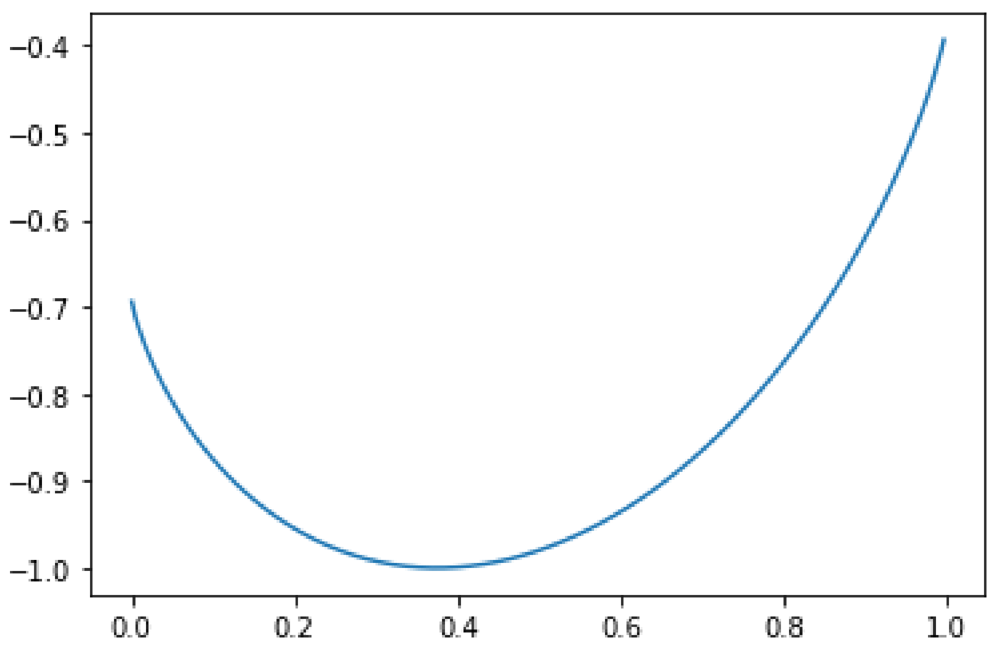
```
tensor([ 27, 221])
```



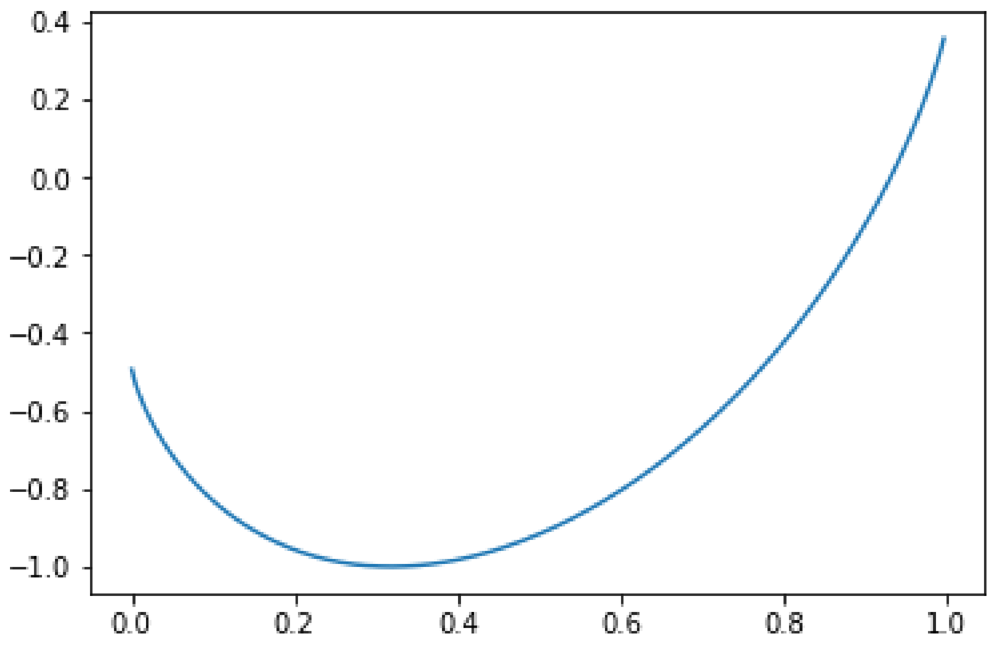
```
##########################
```



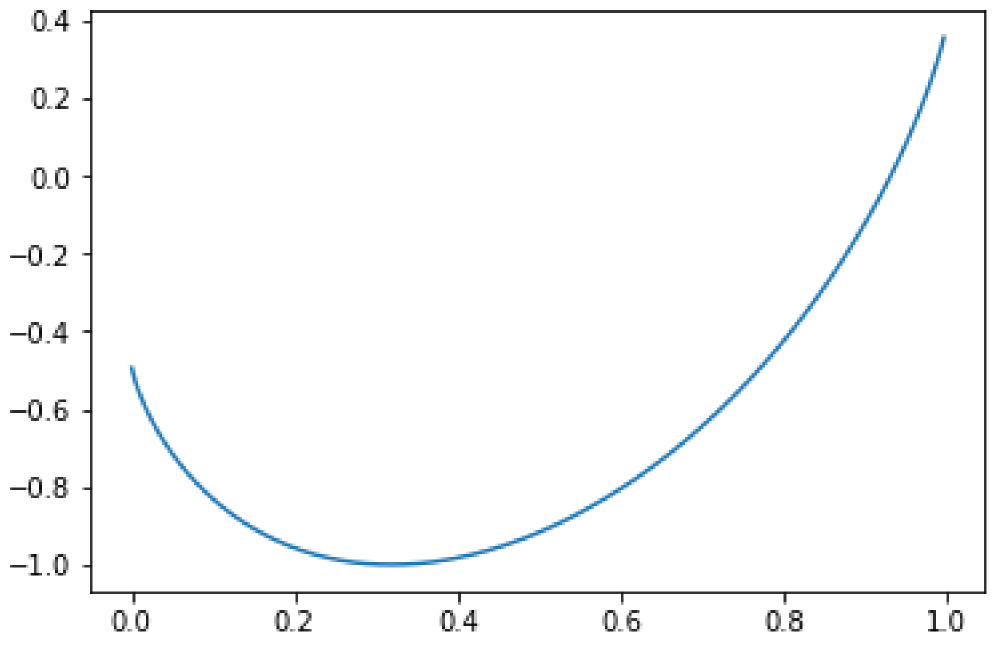
```
tensor([], dtype=torch.int64)
```



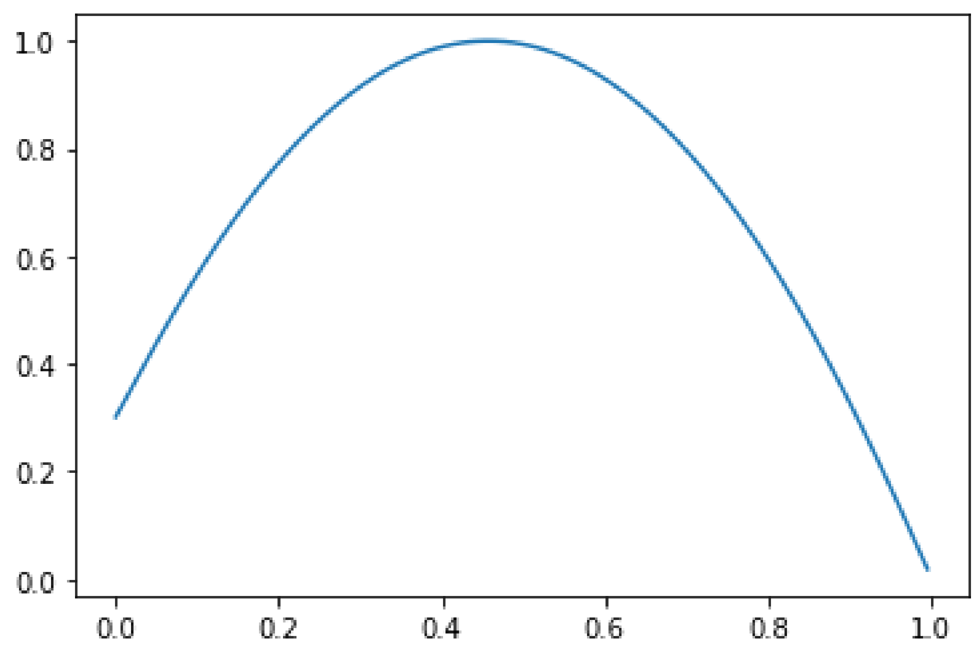
```
##########################
```



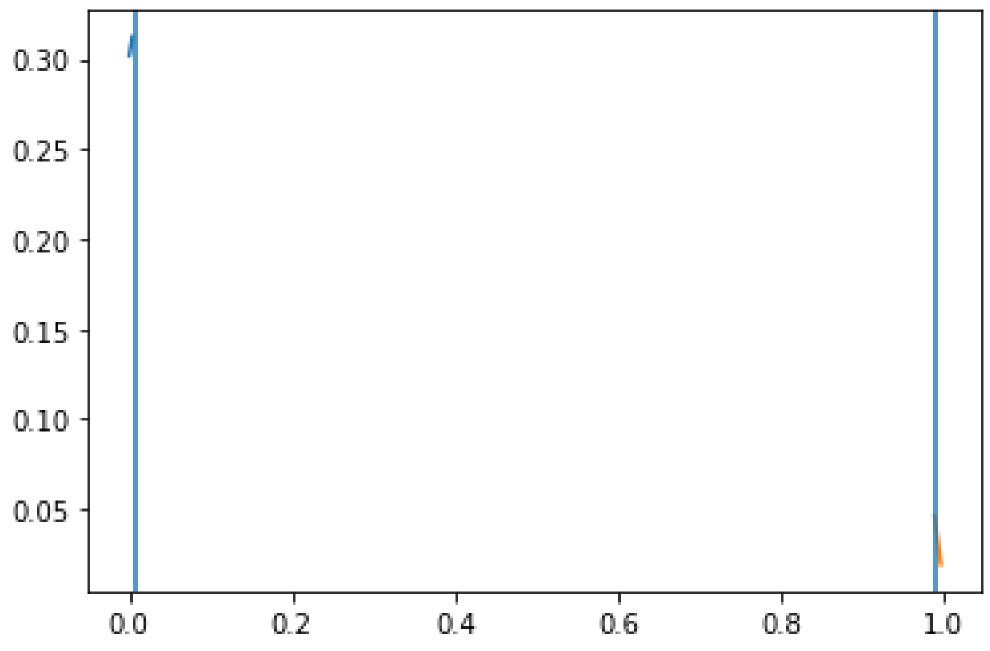
```
tensor([], dtype=torch.int64)
```



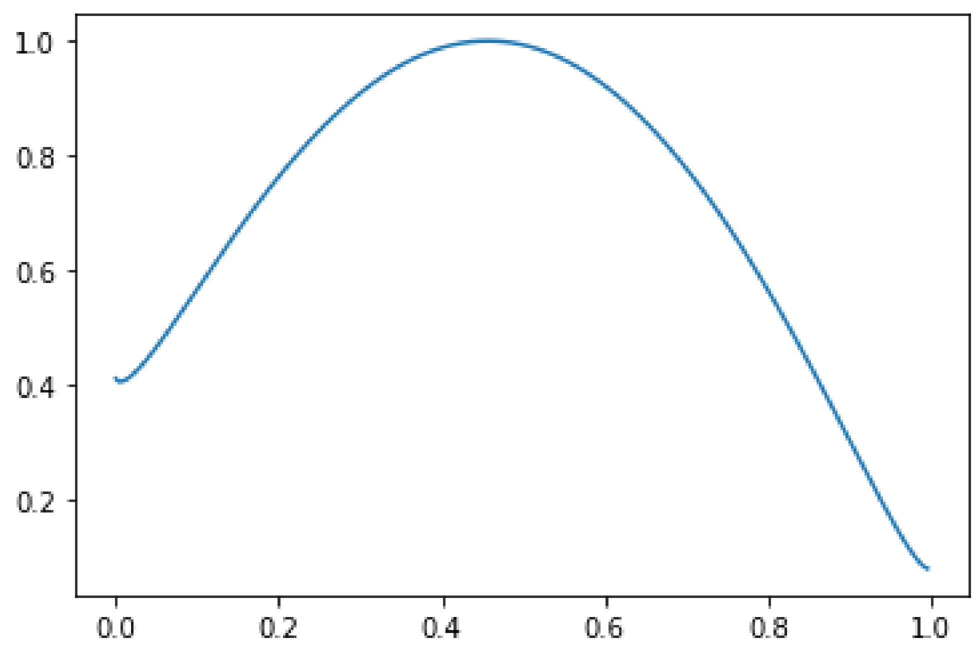
```
########################
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

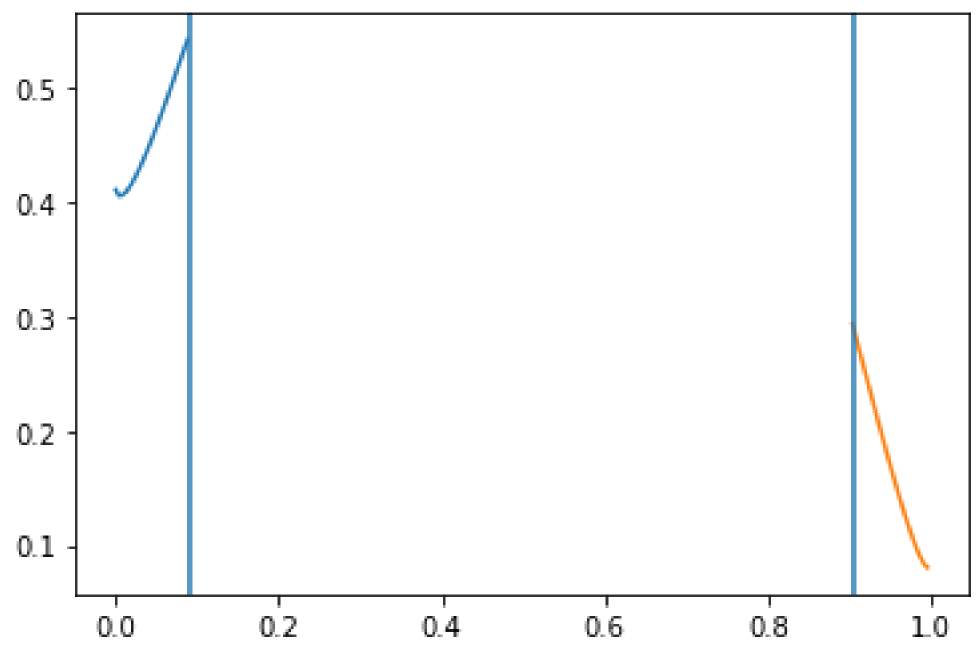
```
tensor([  2, 247])
```


0.30
0.25
0.20
0.15
0.10
0.05
0.0
0.2
0.4
0.6
0.8
1.0

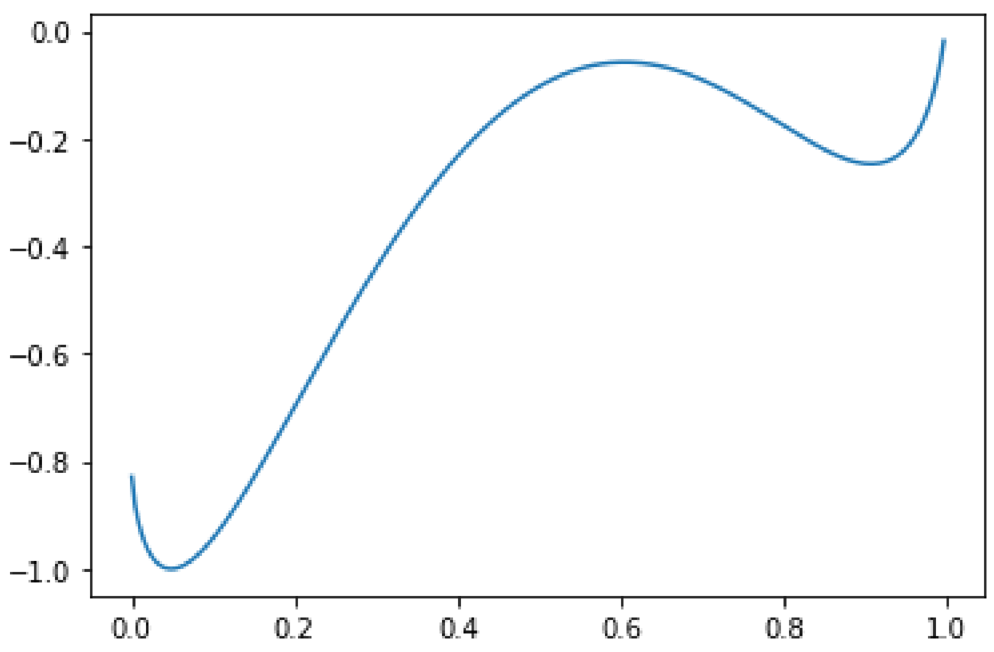
```
#########################
```



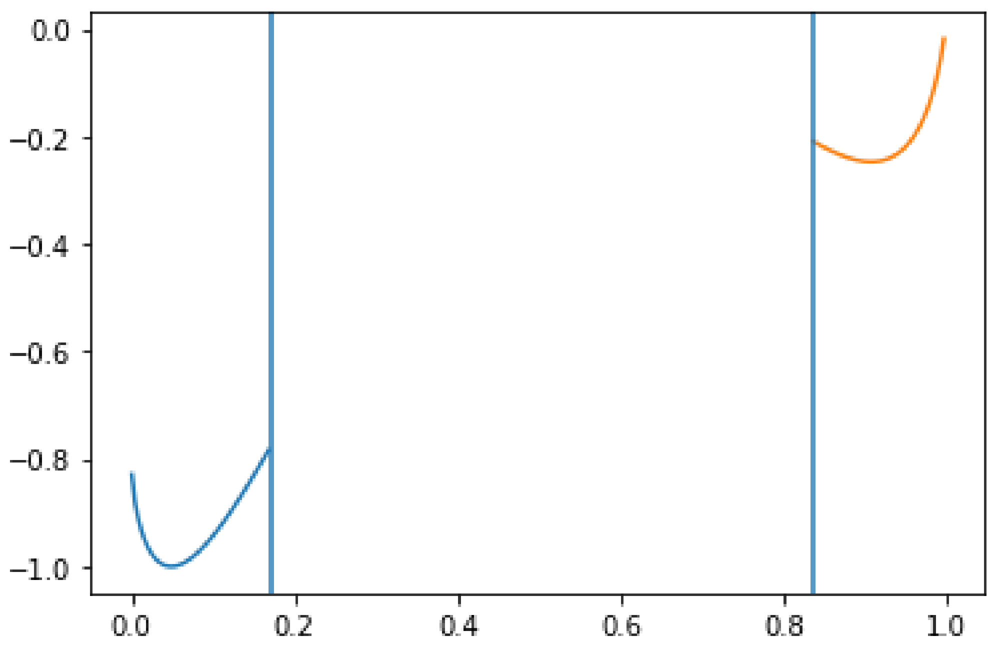
```
tensor([ 23, 226])
```



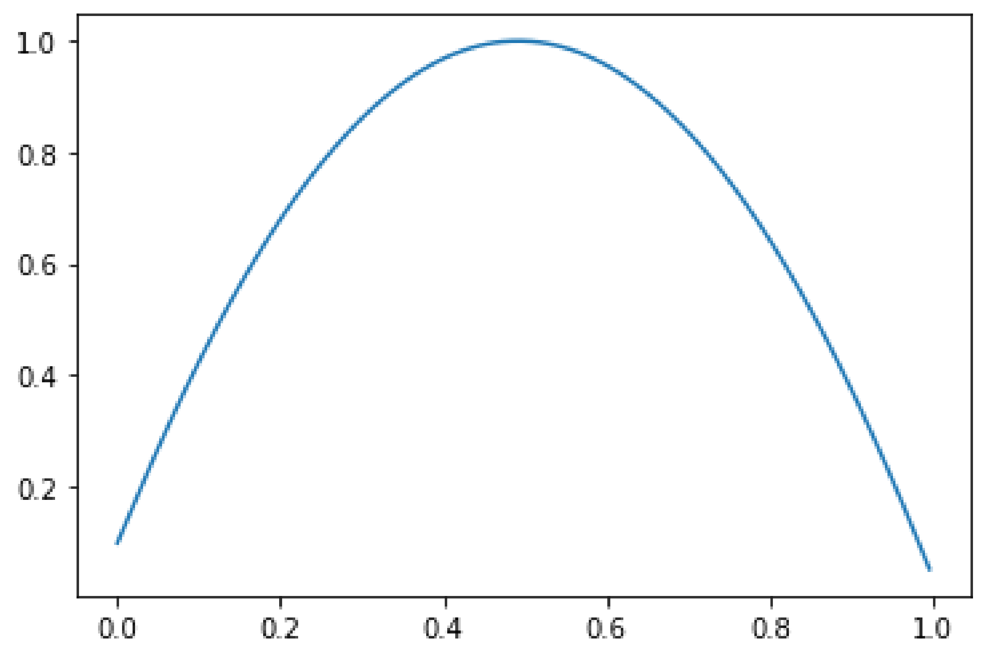
```
##########################
```



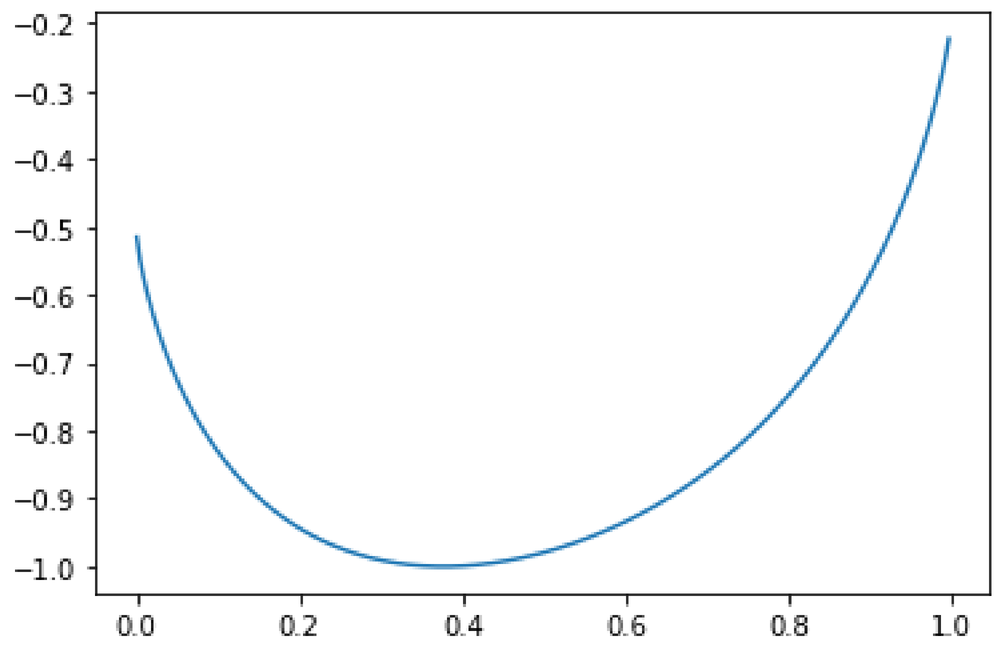
```
tensor([ 43, 209])
```



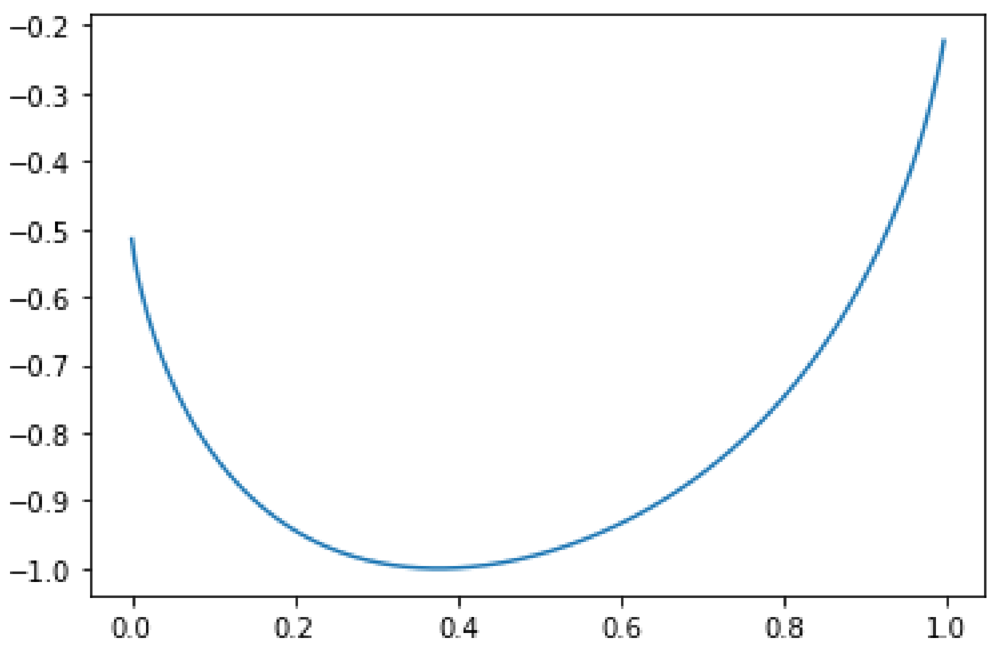
```
##########################
```



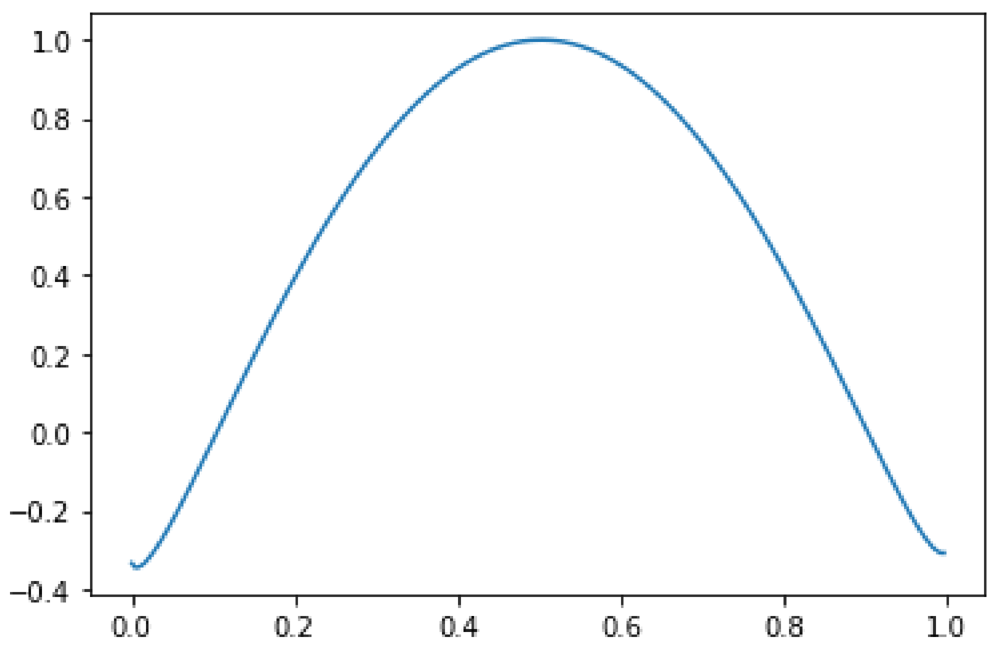
```
tensor([], dtype=torch.int64)
##########################
```



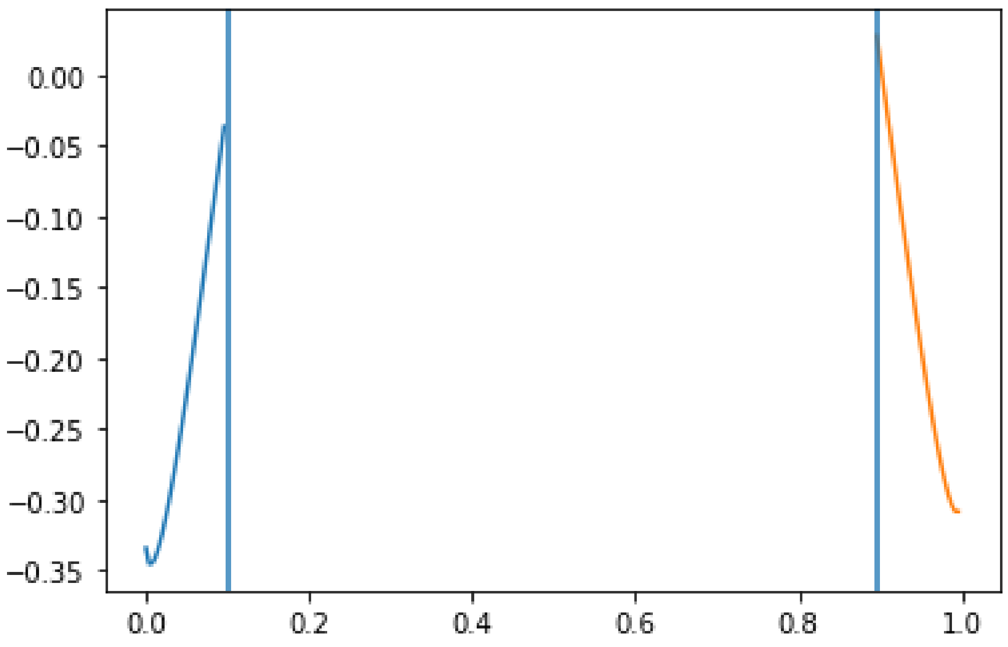
```
tensor([], dtype=torch.int64)
```



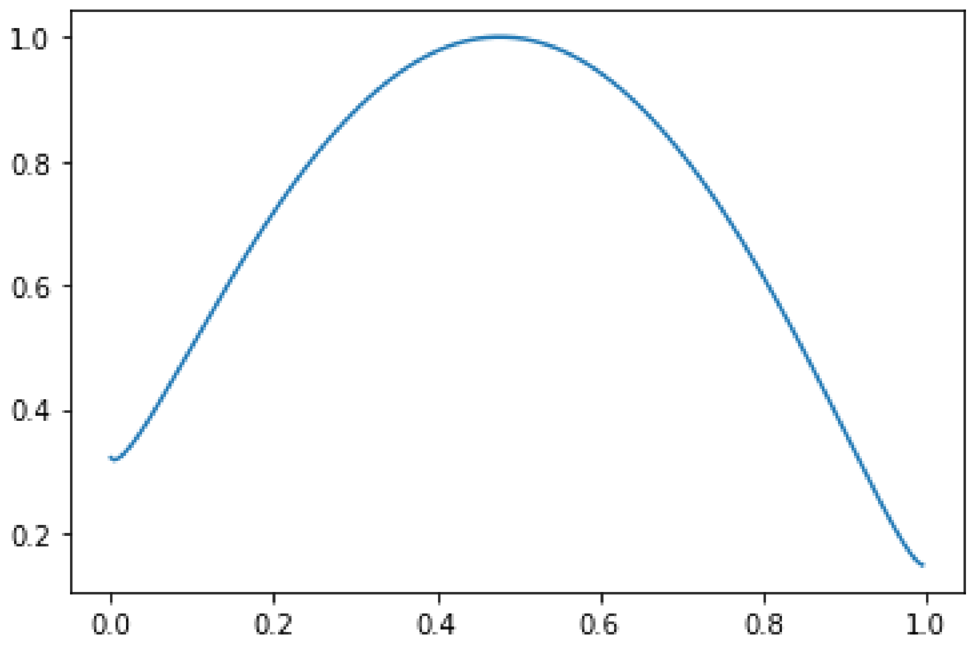
```
##########################
```



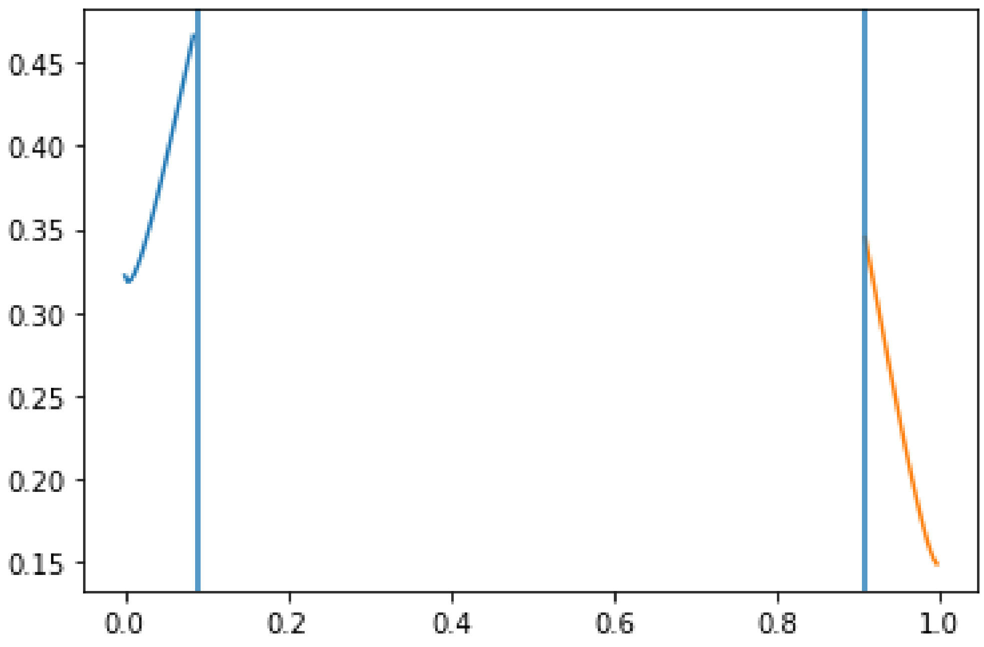
```
tensor([ 25, 224])
```



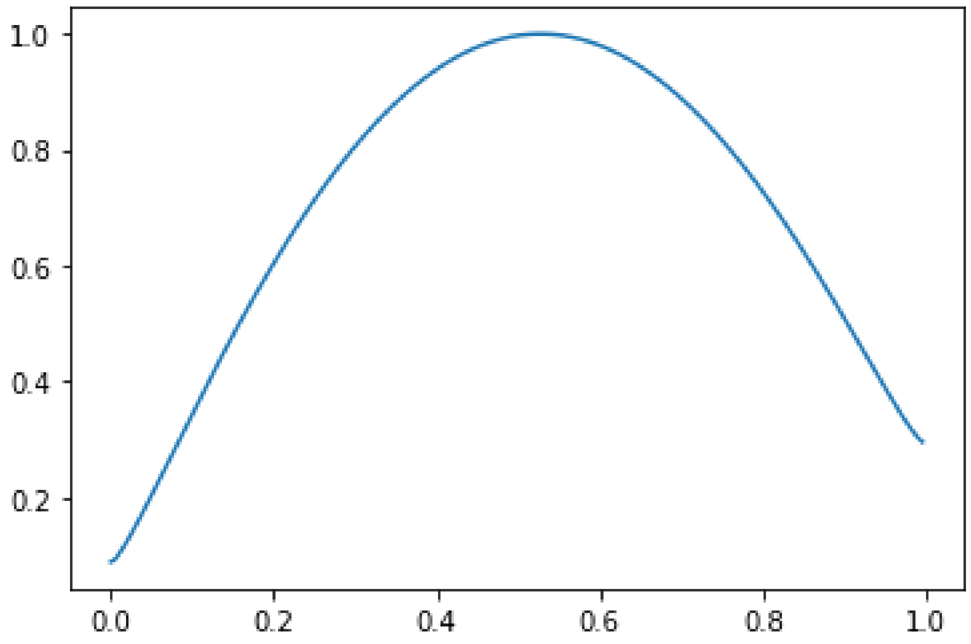
```
#########################
```



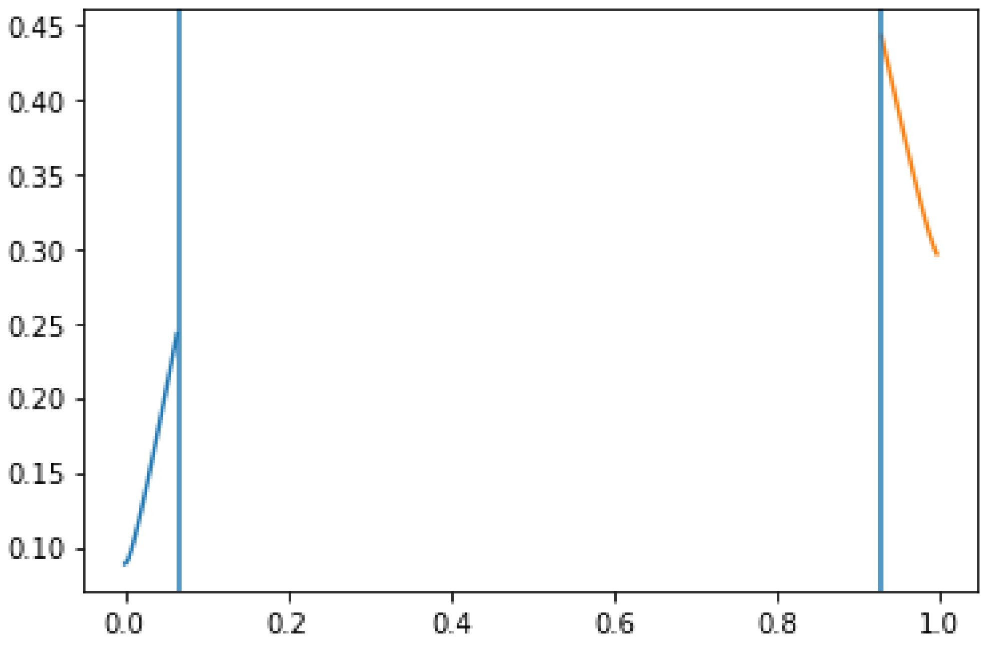
```
tensor([ 22, 227])
```



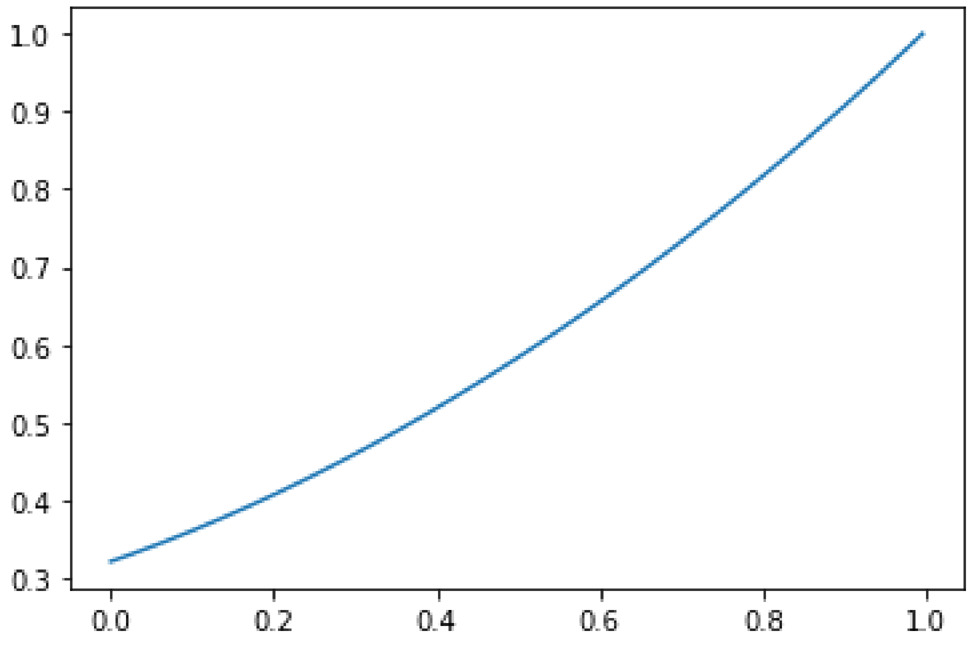
```
##########################
```



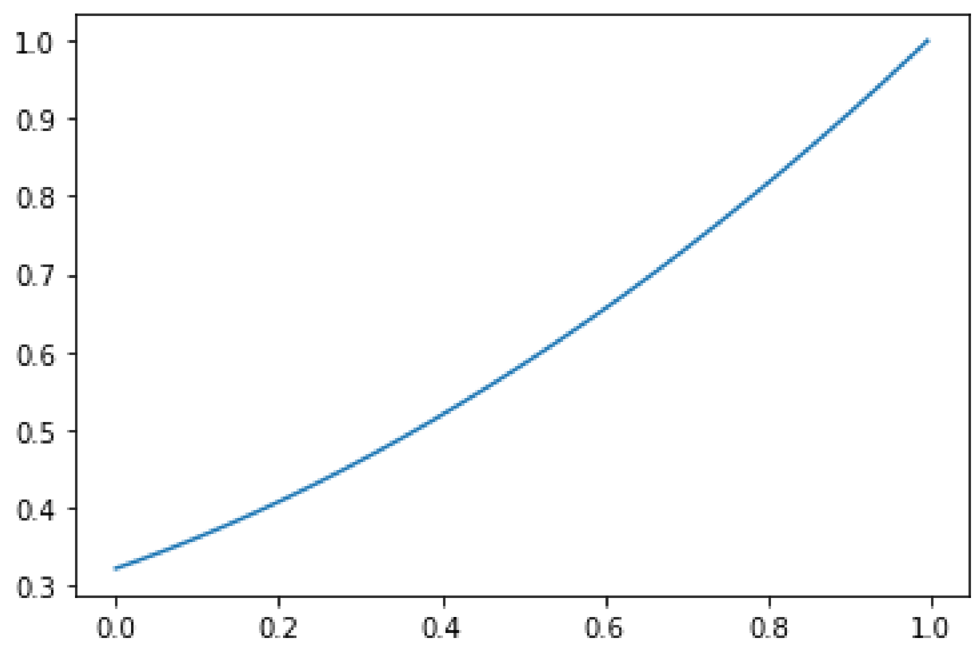
```
tensor([ 17, 232])
```



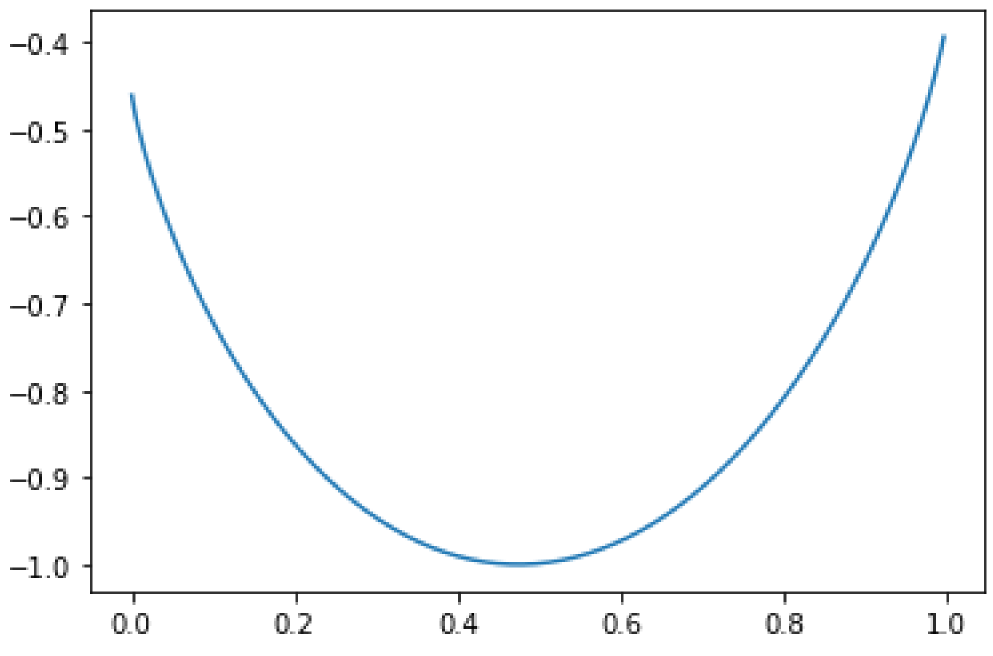
```
##########################
```



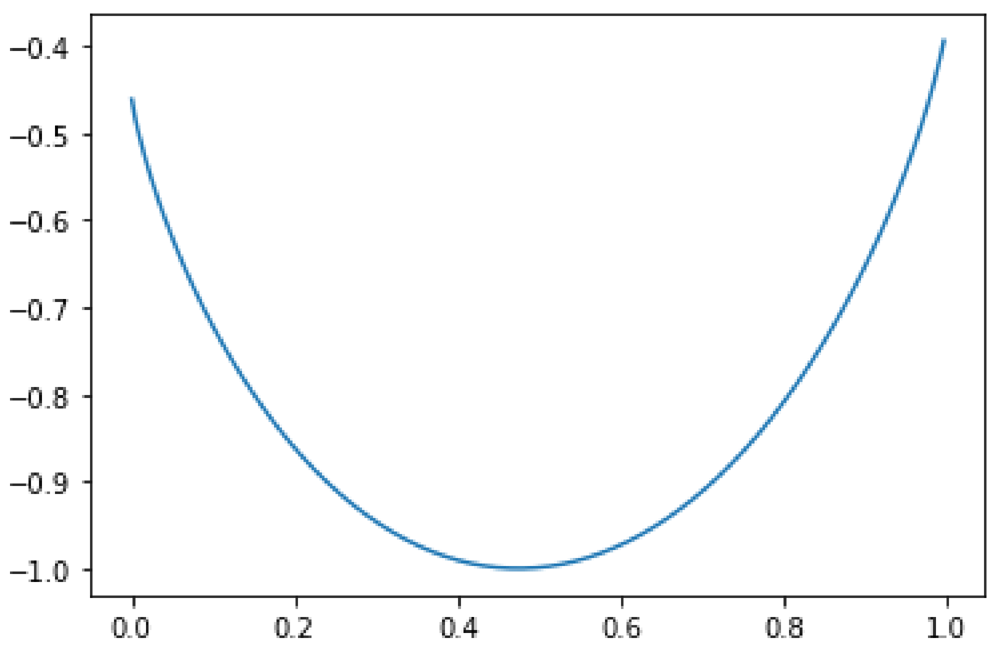
```
tensor([], dtype=torch.int64)
```



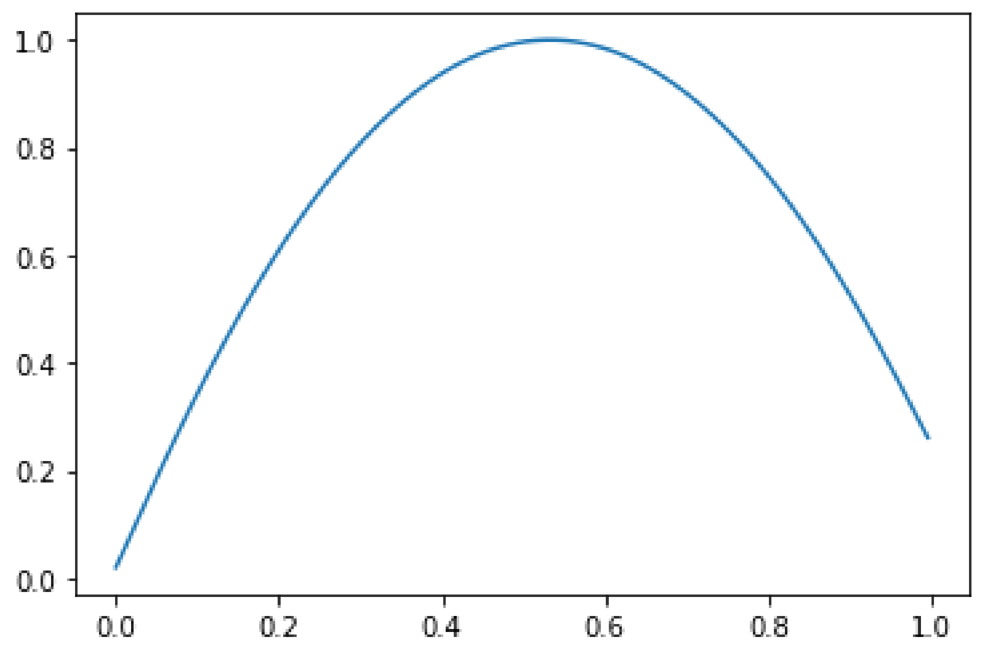
```
##########################
```



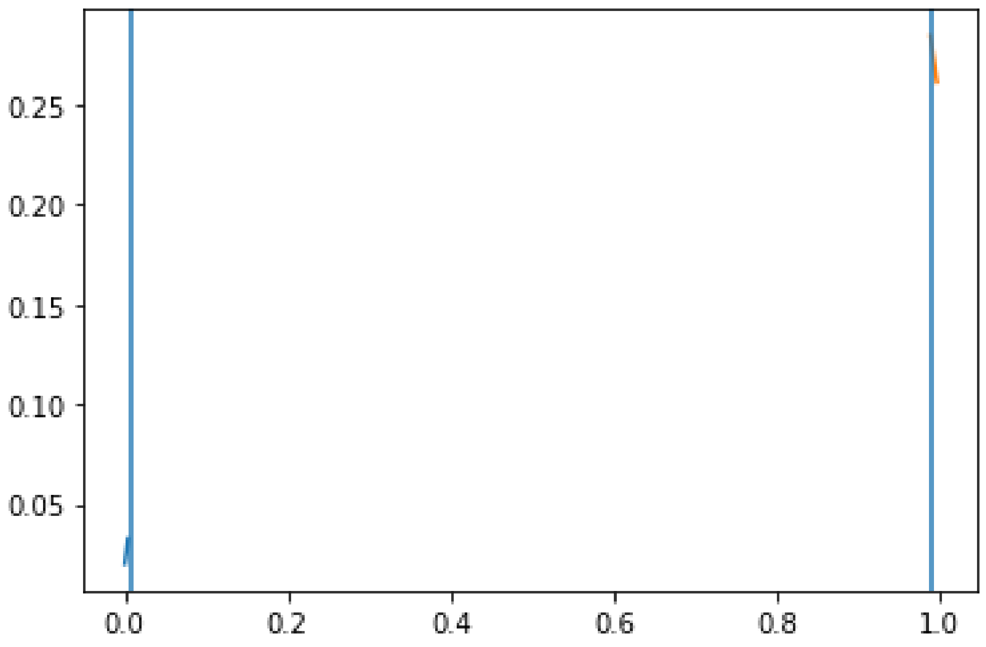
```
tensor([], dtype=torch.int64)
```


−0.4
−0.5
−0.6
−0.7
−0.8
−0.9
−1.0
0.0
0.2
0.4
0.6
0.8
1.0

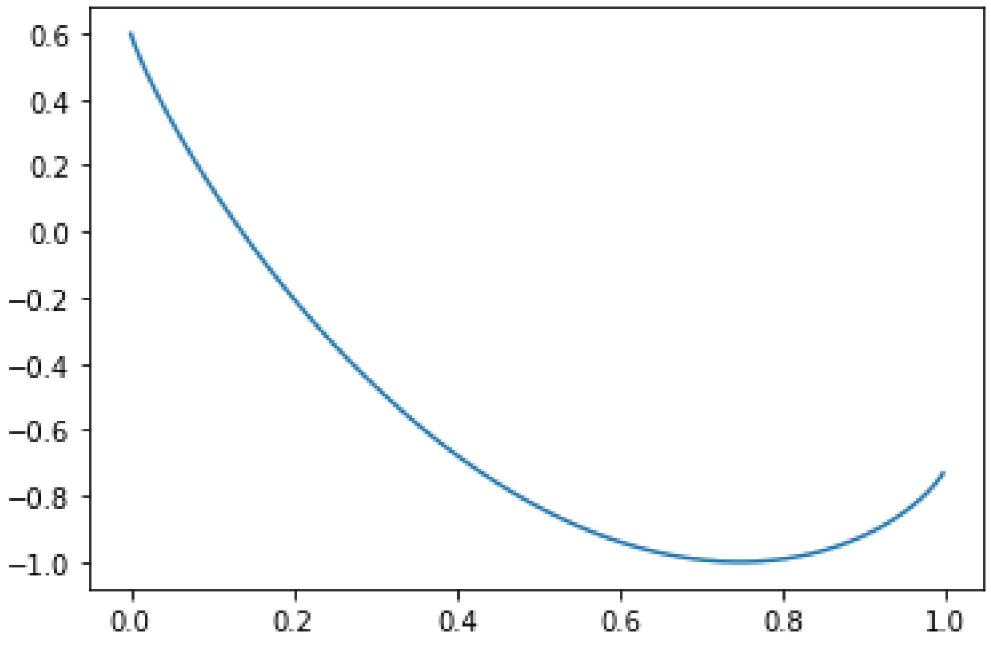
```
##########################
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

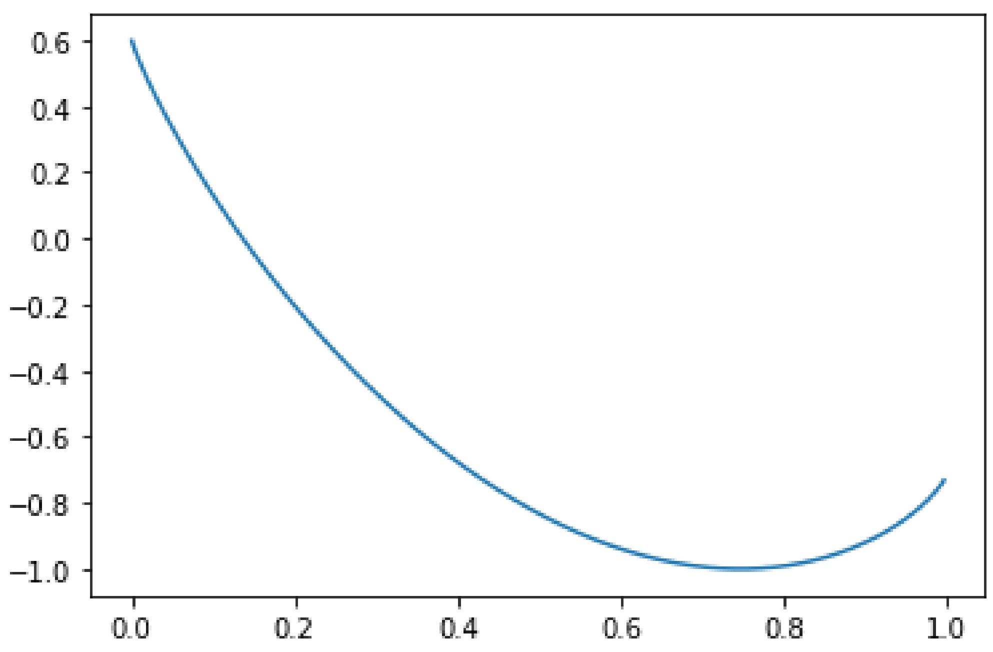
```
tensor([  2, 247])
```



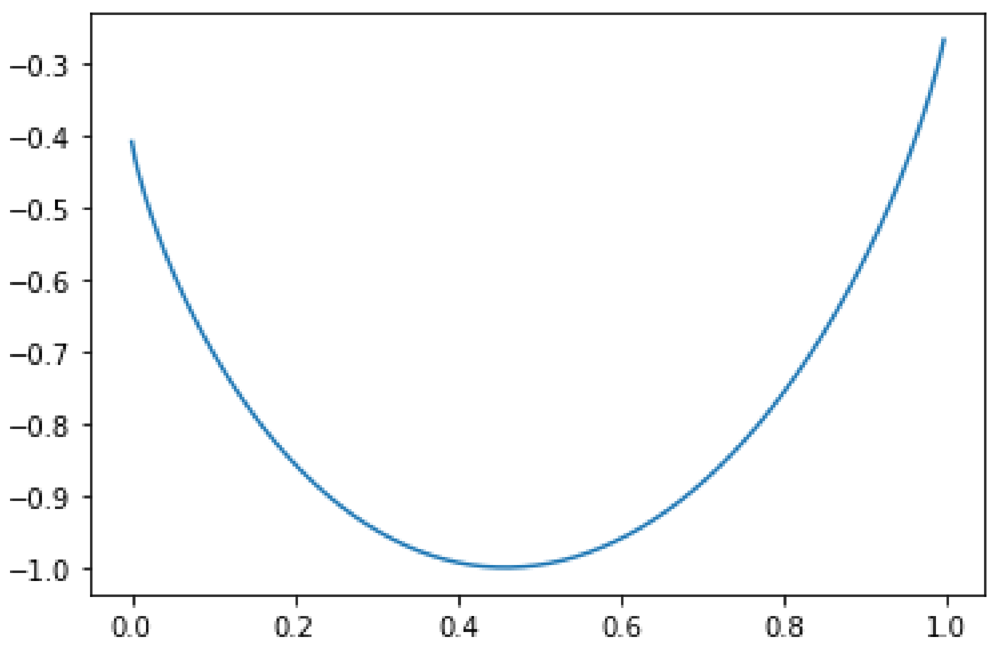
```
##########################
```



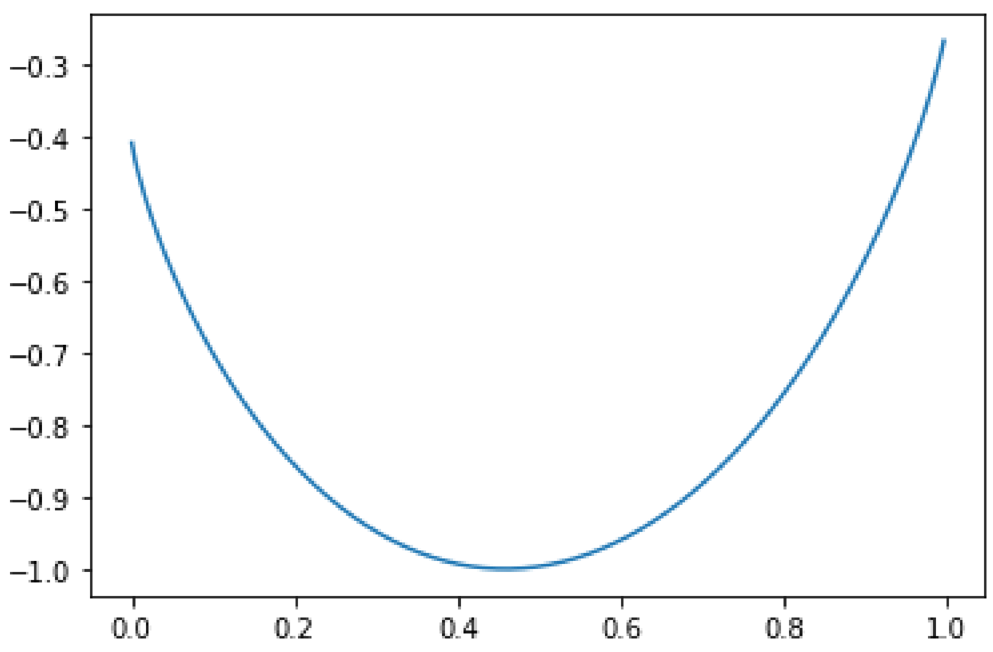
```
tensor([], dtype=torch.int64)
```



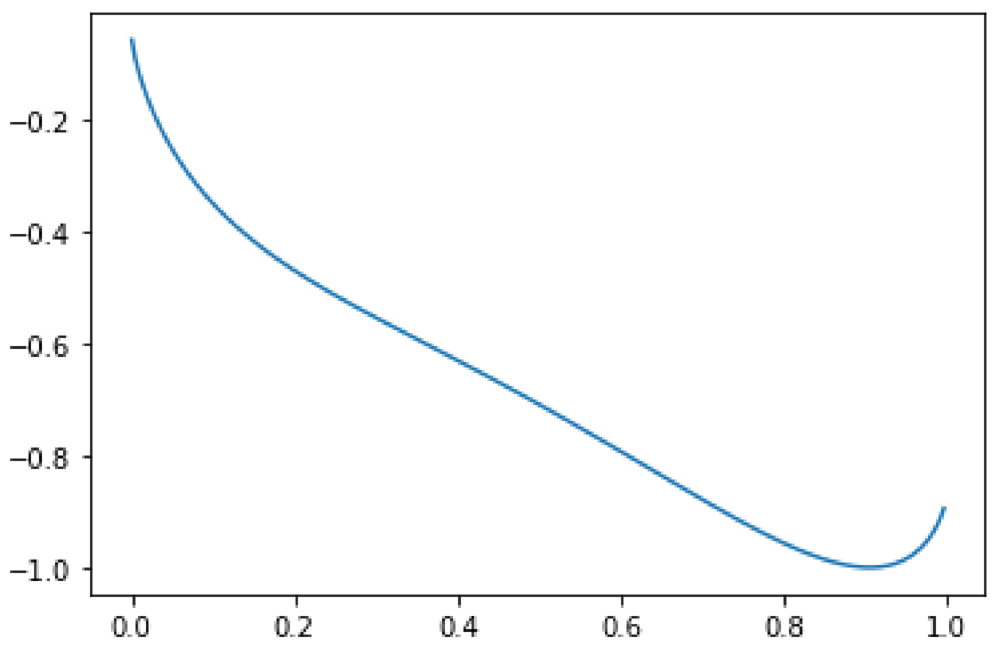
```
#########################
```



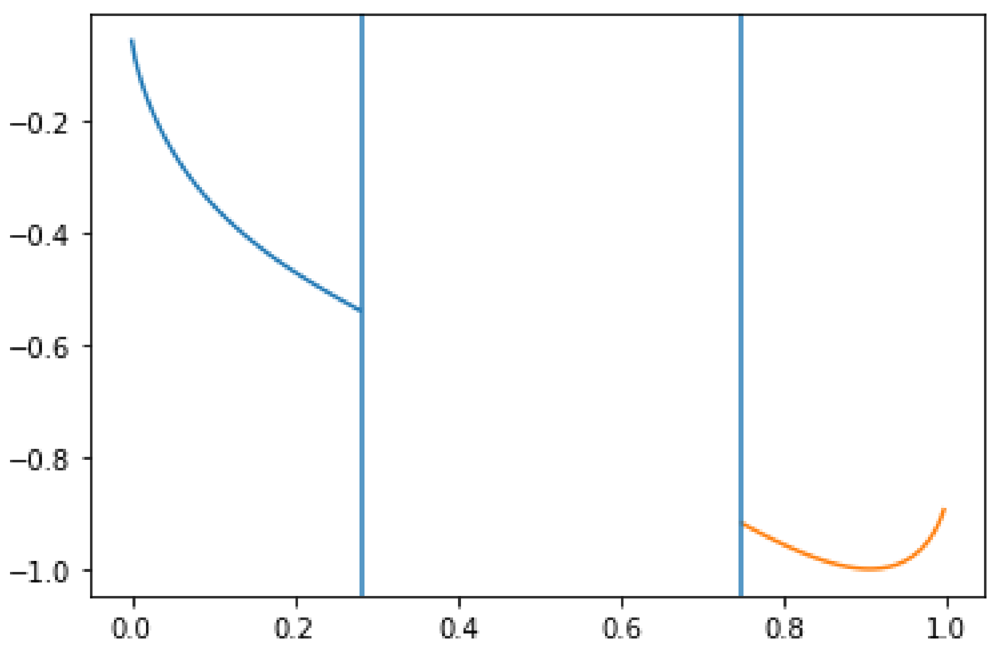
```
tensor([], dtype=torch.int64)
```



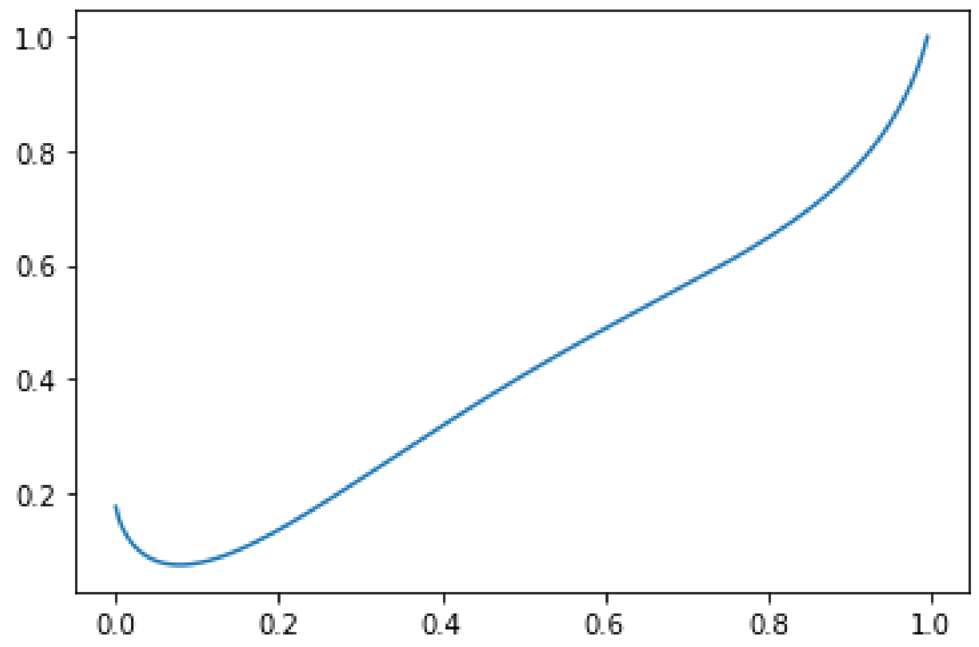
```
#########################
```



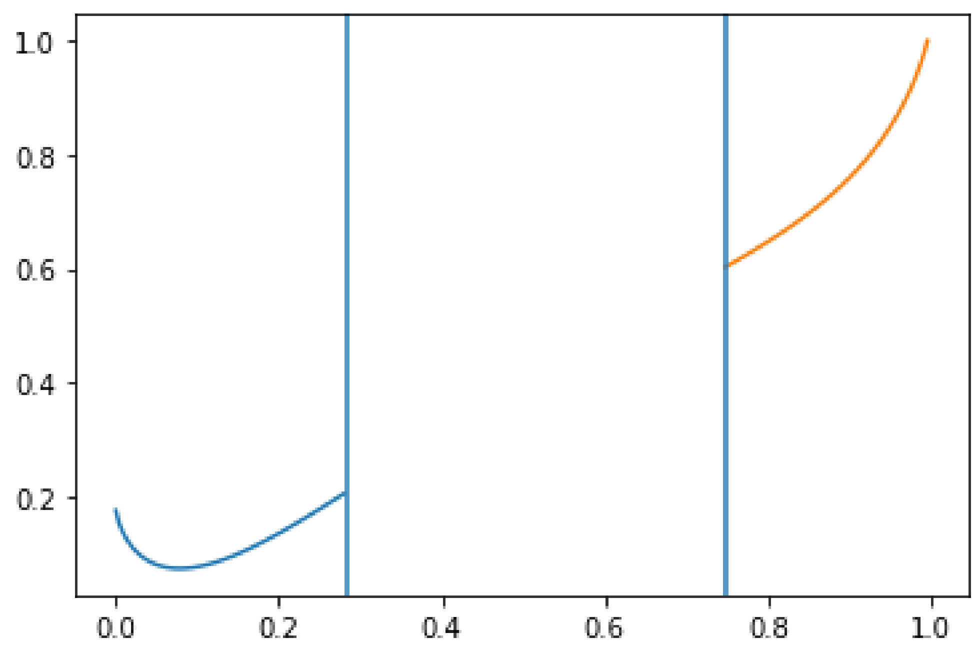
```
tensor([ 71, 187])
```



```
##########################
```

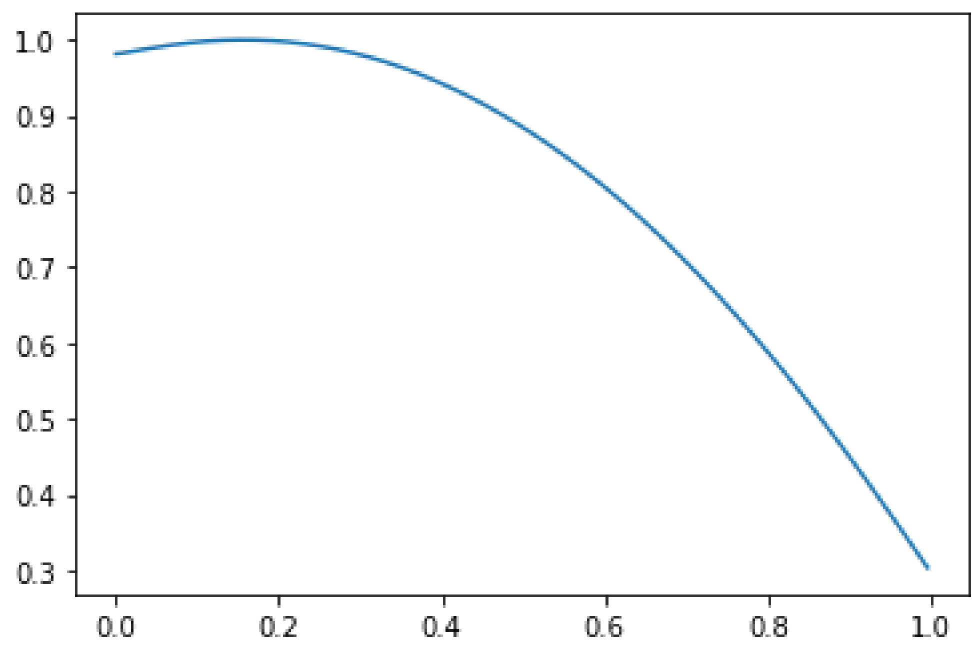


```
tensor([ 71, 187])
```



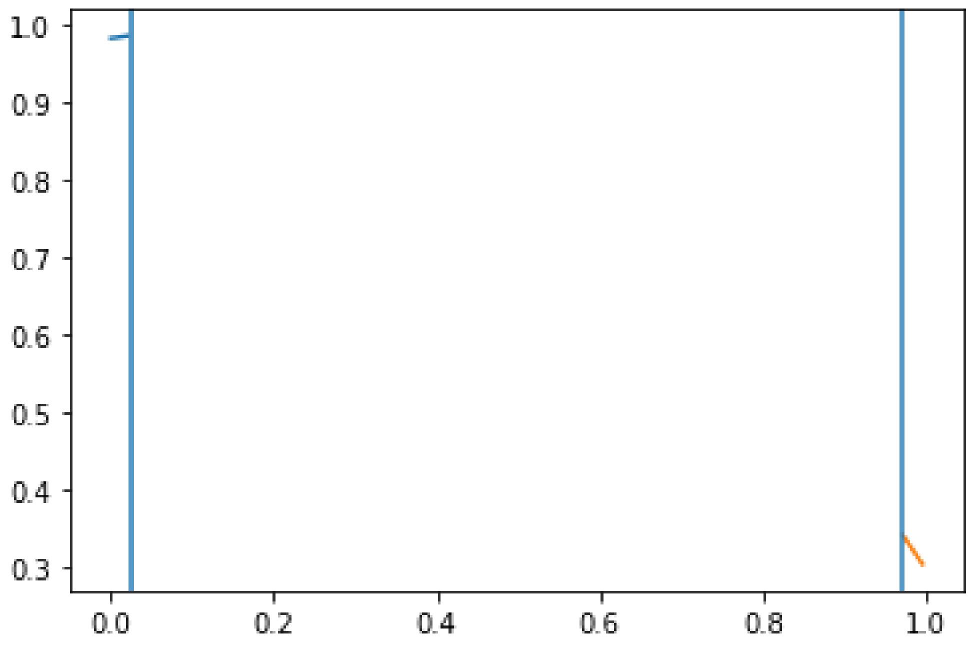
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

```
##########################
```

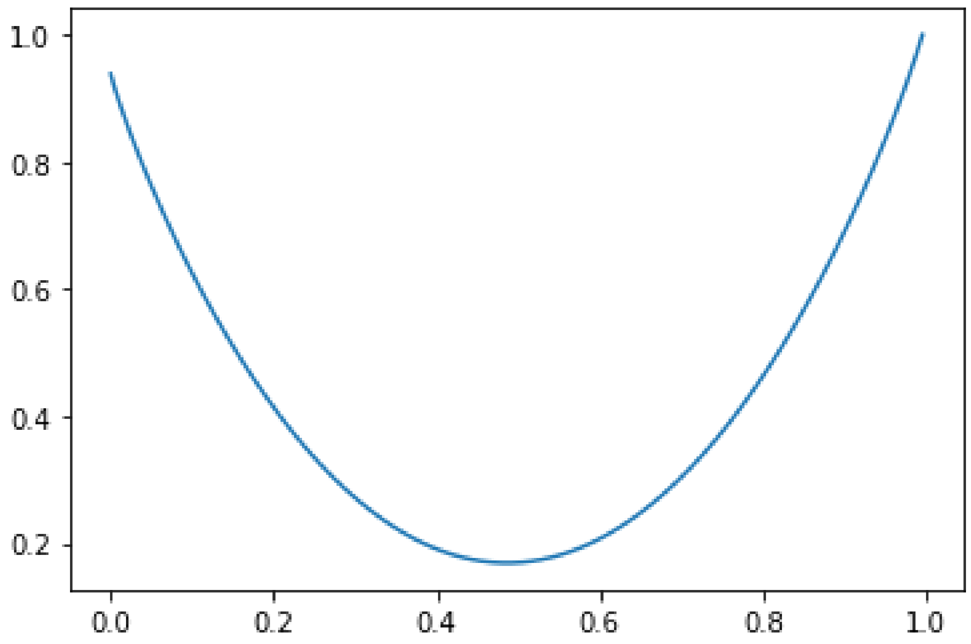


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0

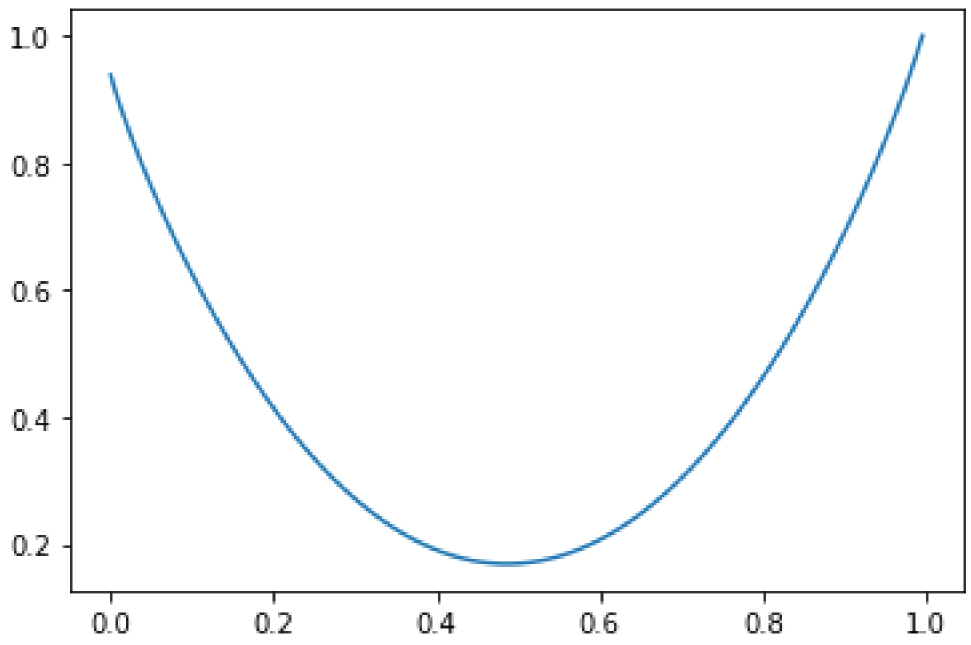
```
tensor([  6, 243])
```



```
##########################
```



```
tensor([], dtype=torch.int64)
```


1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

```
[ ]:
```