data_prep

April 2, 2023

[427]:
```
%%javascript
IPython.OutputArea.prototype._should_scroll = function(lines) {
    return false;
}
```

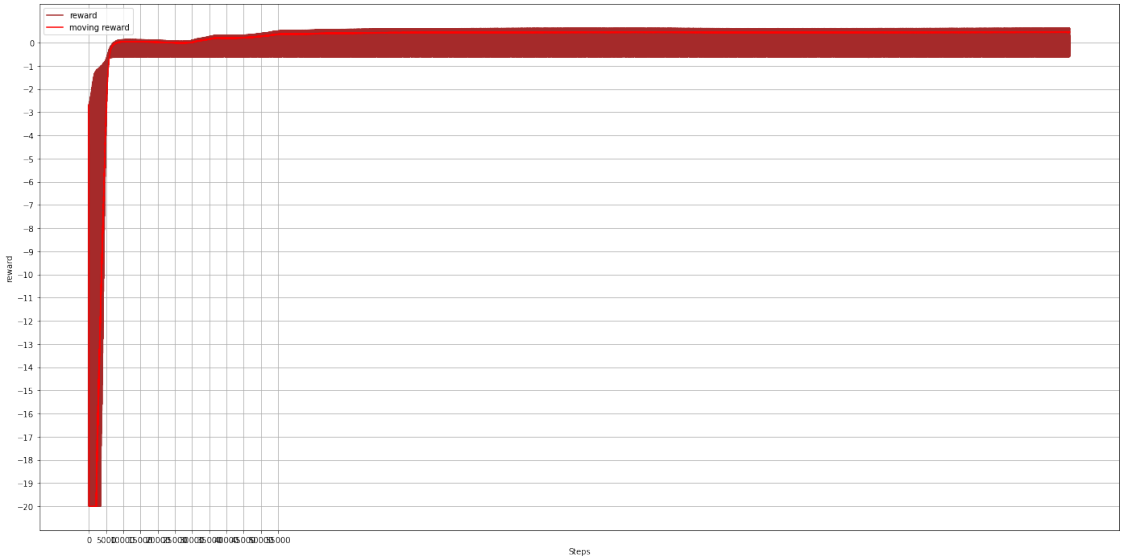
```
<IPython.core.display.Javascript object>
```

[428]:
```
logfile = "../logs/train-20220113-a2c-attempt2"
#logfile = "../logs/train-20220712-a2c-attempt2"
#logfile = "../logs/train-20220712+1-a2c-attempt2"
#logfile = "../logs/train-20220712+2-a2c-attempt2"
#logfile = "../logs/train-20220714-ddpg" #breach w64
#logfile = "../logs/train-20220714-w16-ddpg" #x dropped from 5 to 3.5k,␣
 ↪aoverdraft now and action is low
#logfile = "../logs/train-20220714-w32-ddp" #w16 didn't converge + actoc loss␣
 ↪climbing
#logfile = "../logs/train-20220714-w32-actual-ddpg" #breach
#logfile = "../logs/train-20220714-w24-ddpg" #breach
#logfile = "../logs/train-20220715-w20-ddpg"
#logfile = "../logs/train-20220715-w18-ddpg" #overdraft, didn't go to zero TOO␣
 ↪MUCH WEIGH?
#logfile = "../logs/train-20220716-w17-ddpg" #flipped into large overdraft,␣
 ↪action is low
#logfile = "../logs/train-20220720-w18-z.2-ddpg" #overdraft .03
#logfile = "../logs/train-20220722-w18-z.5-lr.4-ddpg" #x.2 - not converged model
#logfile = "../logs/train-20220723-w18-z.3-ddpg" #a went to 0
#logfile = "../logs/train-20220723-w18-z.4-ddpg" #a went to 0 - low start,␣
 ↪untrained model
logfile = "../logs/train-20220723-w18-z.5-ddpg" #overdrafts, maybe can go zero
#logfile = "../logs/train-20220724-w18-z1.-ddpg" #overdraft suppressed, waste␣
 ↪now
logfile = "../logs/train-20220724-w18-z.6-ddpg" #perfect
#logfile = "../logs/train-20220726-w18-z.6-2-ddpg" #went to waste
#logfile = "../logs/train-20220727-w18-z.6-3-ddpg" #went to waste
#logfile = "../logs/train-20220727-w24-z.6-ddpg" # too much x
#logfile = "../logs/train-20220727-w20-z.6-ddpg" #got ok then overdraft
#logfile = "../logs/train-20220801-w20-z.7-ddpg"
```

```
#logfile = "../logs/train-20230401-w20-z.7-ddpg" #action zero, breach
#logfile = "../logs/train-20230401-w19-z.7-ddpg"
#logfile = "../logs/train-20230402-w19.5-z.7-ddpg"
#logfile = "../logs/train-20230402-w19.5-z.7-ddpg-fix"
```

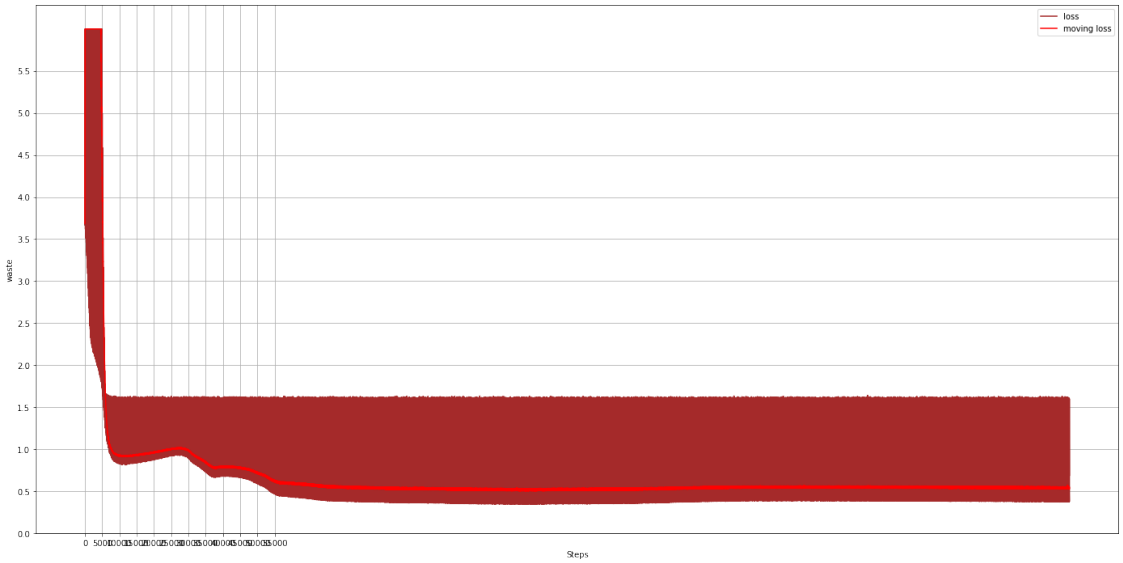
[429]:
```
#rewards: 1102 275 0.107610293

import matplotlib.pyplot as plt

import re
import numpy as np
from pathlib import Path

%matplotlib inline

def plot_loss(files):
    step = []
    loss = []
    mov_loss=[]
    mv = 0

    p = re.compile('^rewards:.*')

    for file in files:
        with open(file) as f:
            lines = f.readlines()
            for line in lines:
                if not p.match(line):
                    continue
                items = re.split(" +", line)
                step.append(int(items[1]))
                loss_value=float(items[3])
                loss.append(loss_value)

                if mv == 0:
                    mv = loss_value
                else:
                    mv = 0.005*loss_value + 0.995*mov_loss[-1]

                mov_loss.append(mv)

    p1, = plt.plot(step, np.clip(loss,-20,1), color='brown',label='reward')
    p2, = plt.plot(step, np.clip(mov_loss,-20,1), color='red',label='moving␣
 ↪reward')

    plt.xlabel("Steps")
    plt.ylabel("reward")
```

```
    plt.legend(handles=[p1,p2],labels=['reward','moving reward'],loc='best')
    plt.xticks(np.arange(0, 60000, 5000))
    plt.yticks(np.arange(-20, 1, step=1))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```



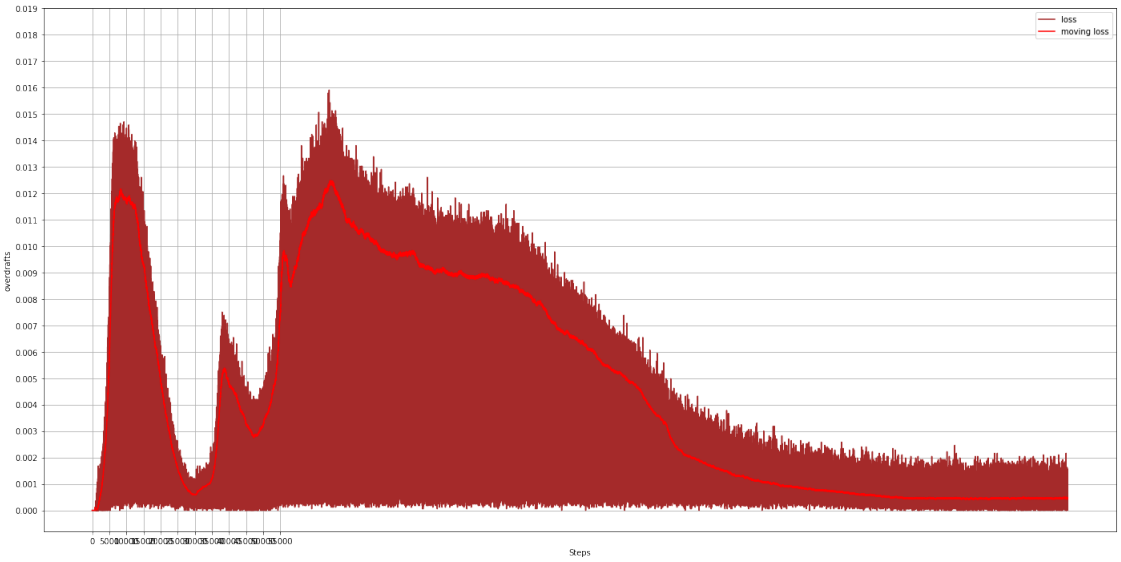
[430]:
```
#waste: 2083 520 0.058227174

import matplotlib.pyplot as plt

import re
import numpy as np
from pathlib import Path

%matplotlib inline

def plot_loss(files):
    step = []
    loss = []
    mov_loss=[]
    mv = 0

    p = re.compile('^waste:.*')
```

```
    for file in files:
        with open(file) as f:
            lines = f.readlines()
            for line in lines:
                if not p.match(line):
                    continue
                items = re.split(" +", line)
                step.append(int(items[1]))
                loss_value=float(items[3])
                loss.append(loss_value)

                if mv == 0:
                    mv = loss_value
                else:
                    mv = 0.005*loss_value + 0.995*mov_loss[-1]

                mov_loss.append(mv)

    p1, = plt.plot(step, np.clip(loss,0,6), color='brown',label='reward')
    p2, = plt.plot(step, np.clip(mov_loss,0,6), color='red',label='moving loss')

    plt.xlabel("Steps")
    plt.ylabel("waste")
    plt.legend(handles=[p1,p2],labels=['loss','moving loss'],loc='best')
    plt.xticks(np.arange(0, 60000, 5000))
    plt.yticks(np.arange(0., 6, step=0.5))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```



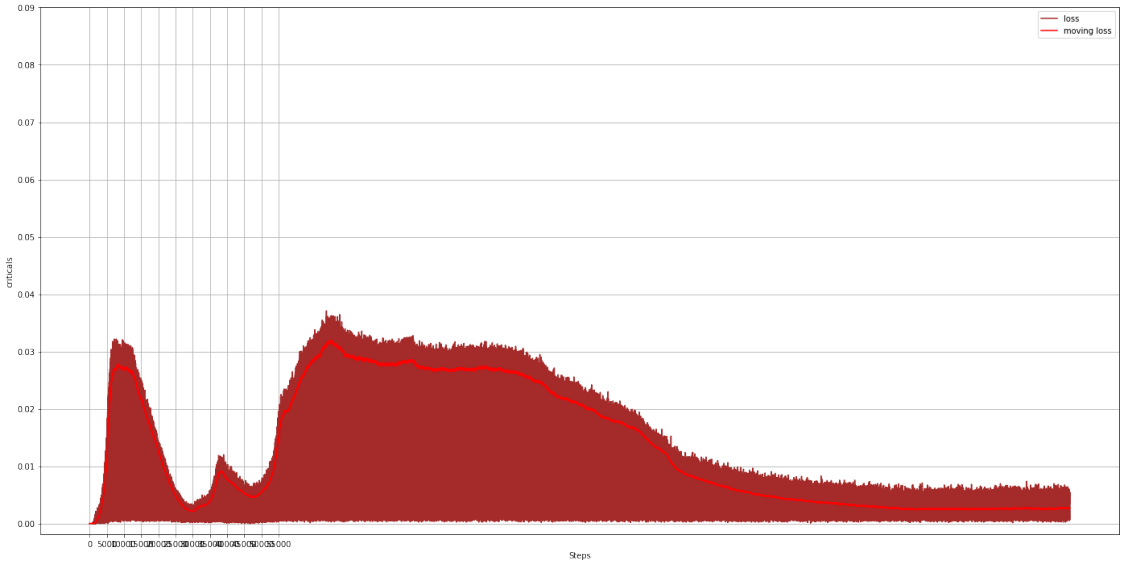
[431]:
```
#overdrafts: 2083 520 0

import matplotlib.pyplot as plt

import re
import numpy as np
from pathlib import Path

%matplotlib inline

def plot_loss(files):
    step = []
    loss = []
    mov_loss=[]
    mv = 0

    p = re.compile('^overdrafts:.*')

    for file in files:
        with open(file) as f:
            lines = f.readlines()
            for line in lines:
                if not p.match(line):
                    continue
                items = re.split(" +", line)
                step.append(int(items[1]))
                loss_value=float(items[3])
                loss.append(loss_value)
```

```
                if mv == 0:
                    mv = loss_value
                else:
                    mv = 0.005*loss_value + 0.995*mov_loss[-1]

                mov_loss.append(mv)

    p1, = plt.plot(step, np.clip(loss,0,0.02), color='brown',label='reward')
    p2, = plt.plot(step, np.clip(mov_loss,0,0.02), color='red',label='moving␣
 ↪loss')

    plt.xlabel("Steps")
    plt.ylabel("overdrafts")
    plt.legend(handles=[p1,p2],labels=['loss','moving loss'],loc='best')
    plt.xticks(np.arange(0, 60000, 5000))
    plt.yticks(np.arange(0, 0.02, step=0.001))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```



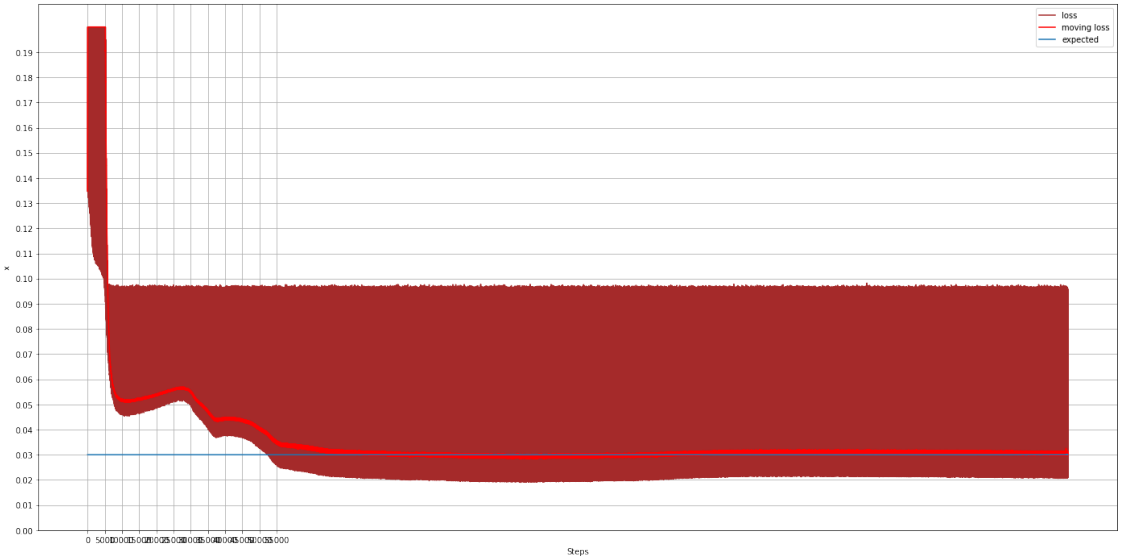
[432]:
```
#overdrafts: 2083 520 0

import matplotlib.pyplot as plt
```

```
import re
import numpy as np
from pathlib import Path

%matplotlib inline

def plot_loss(files):
    step = []
    loss = []
    mov_loss=[]
    mv = 0

    p = re.compile('^critical:.*')

    for file in files:
        with open(file) as f:
            lines = f.readlines()
            for line in lines:
                if not p.match(line):
                    continue
                items = re.split(" +", line)
                step.append(int(items[1]))
                loss_value=float(items[3])
                loss.append(loss_value)

                if mv == 0:
                    mv = loss_value
                else:
                    mv = 0.005*loss_value + 0.995*mov_loss[-1]

                mov_loss.append(mv)

    p1, = plt.plot(step, loss, color='brown',label='reward')
    p2, = plt.plot(step, mov_loss, color='red',label='moving loss')

    plt.xlabel("Steps")
    plt.ylabel("criticals")
    plt.legend(handles=[p1,p2],labels=['loss','moving loss'],loc='best')
    plt.xticks(np.arange(0, 60000, 5000))
    plt.yticks(np.arange(0, 0.1, step=0.01))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```



[433]:
```
#x    : 2083 520 0.297580898

import matplotlib.pyplot as plt

import re
import numpy as np
from pathlib import Path

%matplotlib inline

def plot_loss(files):
    step = []
    loss = []
    mov_loss=[]
    mv = 0

    p = re.compile('^x    :.*')

    for file in files:
        with open(file) as f:
            lines = f.readlines()
            for line in lines:
                if not p.match(line):
                    continue
                items = re.split(" +", line)
                step.append(int(items[2]))
                loss_value=float(items[4])
                loss.append(loss_value)
```

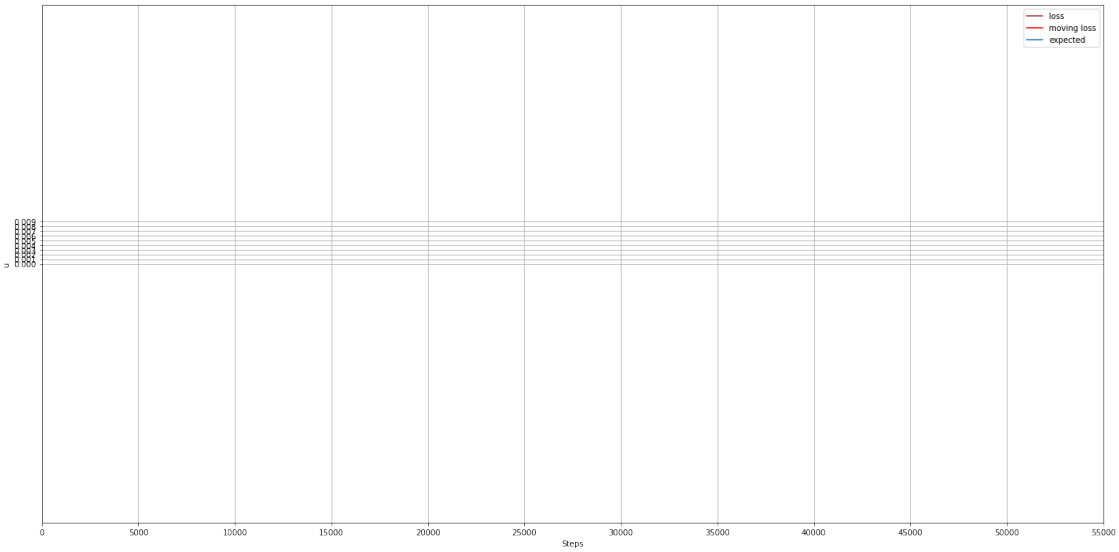
```
                if mv == 0:
                    mv = loss_value
                else:
                    mv = 0.005*loss_value + 0.995*mov_loss[-1]

                mov_loss.append(mv)

    p1, = plt.plot(step, np.clip(loss,0,0.2), color='brown',label='reward')
    p2, = plt.plot(step, np.clip(mov_loss,0,0.2), color='red',label='moving␣
↪loss')
    p11, = plt.plot(step, np.repeat(0.03, len(step)), label='expected')

    plt.xlabel("Steps")
    plt.ylabel("x")
    plt.legend(handles=[p1,p2,p11],labels=['loss','moving␣
↪loss','expected'],loc='best')
    plt.xticks(np.arange(0, 60000, 5000))
    plt.yticks(np.arange(0., 0.2, step=0.01))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```



[434]:
```
#u    : 2083 520 1
```

```
import matplotlib.pyplot as plt

import re
import numpy as np
from pathlib import Path

%matplotlib inline

def plot_loss(files):
    step = []
    loss = []
    mov_loss=[]
    mv = 0

    p = re.compile('^p    :.*')

    for file in files:
        with open(file) as f:
            lines = f.readlines()
            for line in lines:
                if not p.match(line):
                    continue
                items = re.split(" +", line)
                step.append(int(items[2]))
                loss_value=float(items[4])
                loss.append(loss_value)

                if mv == 0:
                    mv = loss_value
                else:
                    mv = 0.005*loss_value + 0.995*mov_loss[-1]

                mov_loss.append(mv)

    p1, = plt.plot(step, np.clip(loss,0,0.01), color='brown',label='reward')
    p2, = plt.plot(step, np.clip(mov_loss,0,0.01), color='red',label='moving␣
↪loss')
    p11, = plt.plot(step, np.repeat(0.007, len(step)), label='expected')

    plt.xlabel("Steps")
    plt.ylabel("u")
    plt.legend(handles=[p1,p2,p11],labels=['loss','moving␣
↪loss','expected'],loc='best')
    plt.xticks(np.arange(0, 60000, 5000))
    plt.yticks(np.arange(0., 0.01, step=0.001))
    plt.grid(True, which='both')
    #plt.minorticks_on()
```

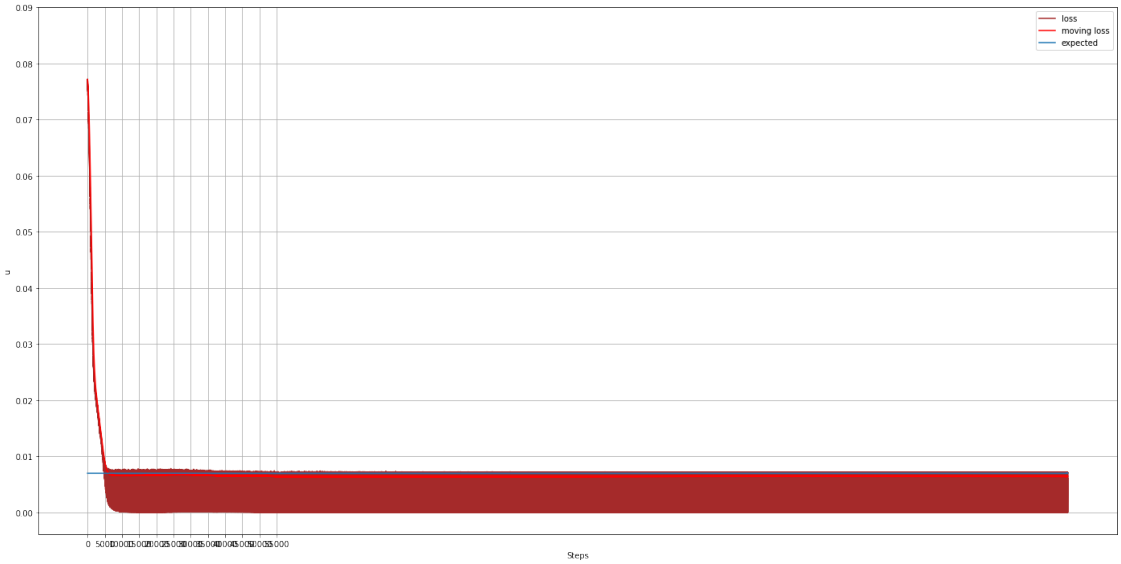
```
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```



[ ]:

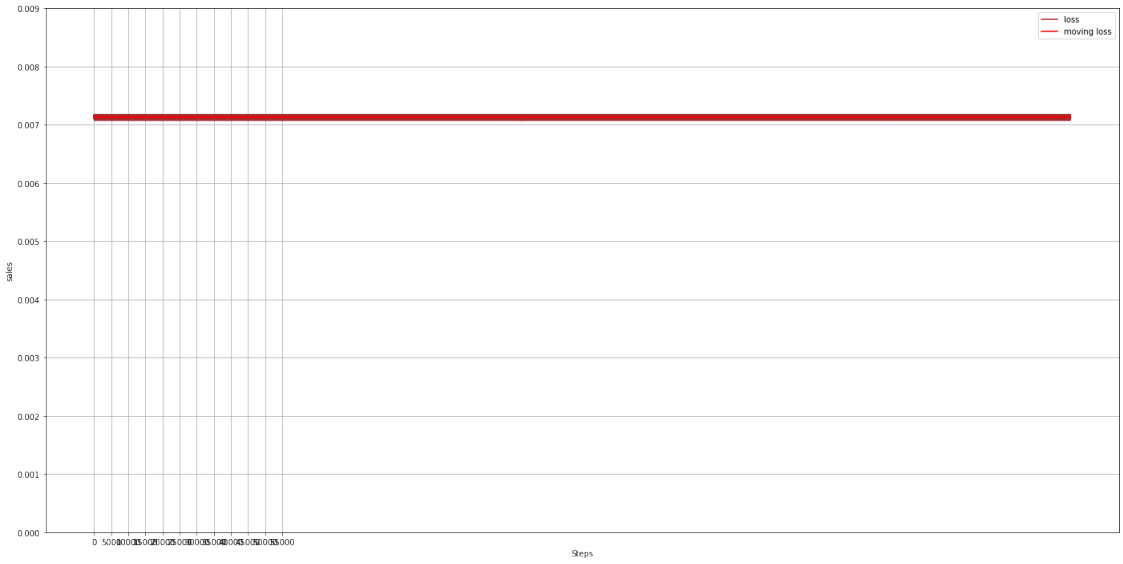
[435]:
```
#u    : 2083 520 1

import matplotlib.pyplot as plt

import re
import numpy as np
from pathlib import Path

%matplotlib inline

def plot_loss(files):
    step = []
    loss = []
    mov_loss=[]
    mv = 0

    p = re.compile('^a    :.*')

    for file in files:
        with open(file) as f:
            lines = f.readlines()
            for line in lines:
```

```
                if not p.match(line):
                    continue
                items = re.split(" +", line)
                step.append(int(items[2]))
                loss_value=float(items[4])
                loss.append(loss_value)

                if mv == 0:
                    mv = loss_value
                else:
                    mv = 0.005*loss_value + 0.995*mov_loss[-1]

                mov_loss.append(mv)

    p1, = plt.plot(step, np.clip(loss,0,0.1), color='brown',label='reward')
    p2, = plt.plot(step, np.clip(mov_loss,0,0.1), color='red',label='moving␣
↪loss')
    p11, = plt.plot(step, np.repeat(0.007, len(step)), label='expected')

    plt.xlabel("Steps")
    plt.ylabel("u")
    plt.legend(handles=[p1,p2,p11],labels=['loss','moving␣
↪loss','expected'],loc='best')
    plt.xticks(np.arange(0, 60000, 5000))
    plt.yticks(np.arange(0, 0.1, step=0.01))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```



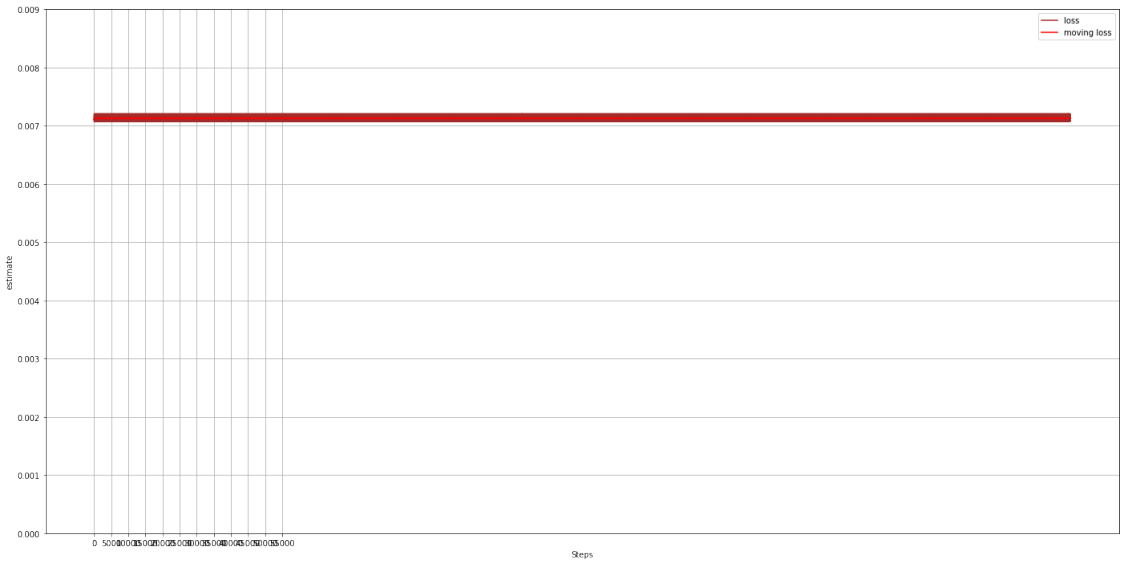
[436]:
```
#sales: 2082 520 0.702729702

import matplotlib.pyplot as plt

import re
import numpy as np
from pathlib import Path

%matplotlib inline

def plot_loss(files):
    step = []
    loss = []
    mov_loss=[]
    mv = 0

    p = re.compile('^debit:.*')

    for file in files:
        with open(file) as f:
            lines = f.readlines()
            for line in lines:
                if not p.match(line):
                    continue
                items = re.split(" +", line)
                step.append(int(items[1]))
                loss_value=float(items[3])
                loss.append(loss_value)
```

```
                if mv == 0:
                    mv = loss_value
                else:
                    mv = 0.005*loss_value + 0.995*mov_loss[-1]

                mov_loss.append(mv)

    p1, = plt.plot(step, np.clip(loss,0,0.01), color='brown',label='reward')
    p2, = plt.plot(step, np.clip(mov_loss,0,0.01), color='red',label='moving␣
 ↪loss')

    plt.xlabel("Steps")
    plt.ylabel("sales")
    plt.legend(handles=[p1,p2],labels=['loss','moving loss'],loc='best')
    plt.xticks(np.arange(0, 60000, 5000))
    plt.yticks(np.arange(0, 0.01, step=0.001))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```



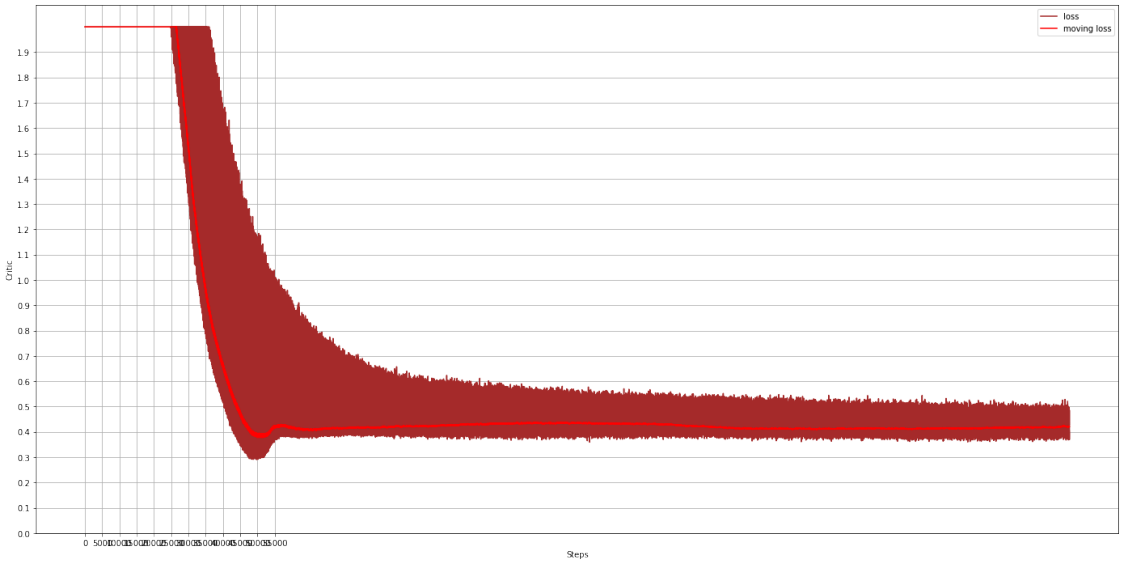
[437]:
```
#estimate: 2398 599 0.702729702

import matplotlib.pyplot as plt
```

```
import re
import numpy as np
from pathlib import Path

%matplotlib inline

def plot_loss(files):
    step = []
    loss = []
    mov_loss=[]
    mv = 0

    p = re.compile('^estimate:.*')

    for file in files:
        with open(file) as f:
            lines = f.readlines()
            for line in lines:
                if not p.match(line):
                    continue
                items = re.split(" +", line)
                step.append(int(items[1]))
                loss_value=float(items[3])
                loss.append(loss_value)

                if mv == 0:
                    mv = loss_value
                else:
                    mv = 0.005*loss_value + 0.995*mov_loss[-1]

                mov_loss.append(mv)

    p1, = plt.plot(step, np.clip(loss,0,0.01), color='brown',label='reward')
    p2, = plt.plot(step, np.clip(mov_loss,0,0.01), color='red',label='moving␣
 ↪loss')

    plt.xlabel("Steps")
    plt.ylabel("estimate")
    plt.legend(handles=[p1,p2],labels=['loss','moving loss'],loc='best')
    plt.xticks(np.arange(0, 60000, 5000))
    plt.yticks(np.arange(0, 0.01, step=0.001))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```



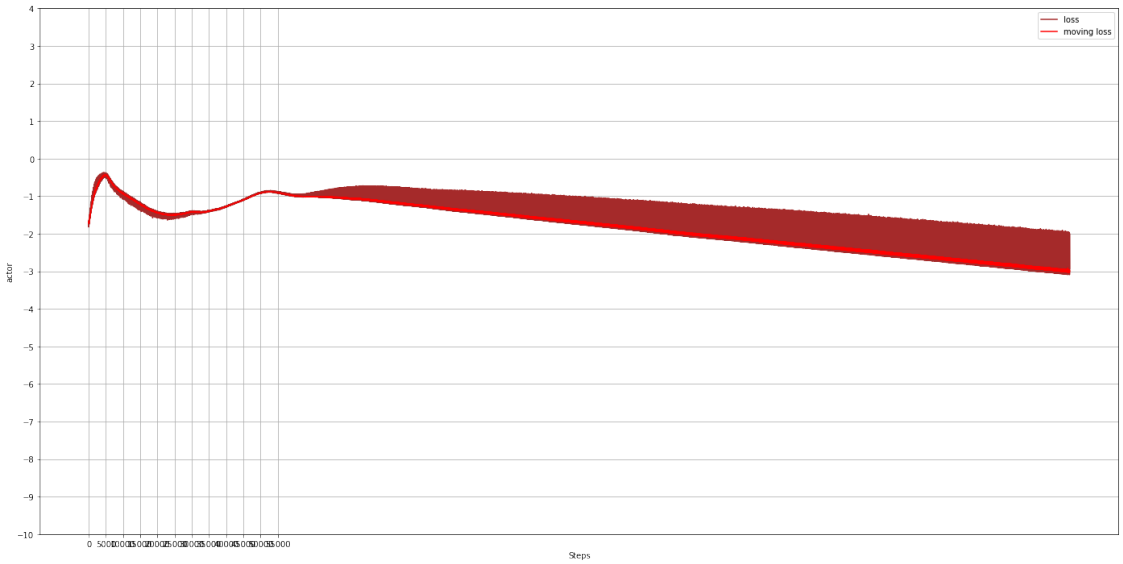
[438]:
```
#critic loss: 2082 520 0.127547532

import matplotlib.pyplot as plt

import re
import numpy as np
from pathlib import Path

%matplotlib inline

def plot_loss(files):
    step = []
    loss = []
    mov_loss=[]
    mv = 0

    p = re.compile('^critic loss:.*')

    for file in files:
        with open(file) as f:
            lines = f.readlines()
            for line in lines:
                if not p.match(line):
                    continue
                items = re.split(" +", line)
                step.append(int(items[2]))
                loss_value=float(items[4])
                loss.append(loss_value)
```

```
                if mv == 0:
                    mv = loss_value
                else:
                    mv = 0.005*loss_value + 0.995*mov_loss[-1]

                mov_loss.append(mv)

    p1, = plt.plot(step, np.clip(loss,0, 2), color='brown',label='reward')
    p2, = plt.plot(step, np.clip(mov_loss, 0, 2), color='red',label='moving␣
 ↪loss')

    plt.xlabel("Steps")
    plt.ylabel("Critic")
    plt.legend(handles=[p1,p2],labels=['loss','moving loss'],loc='best')
    plt.xticks(np.arange(0, 60000, 5000))
    plt.yticks(np.arange(0, 2, step=0.1))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```



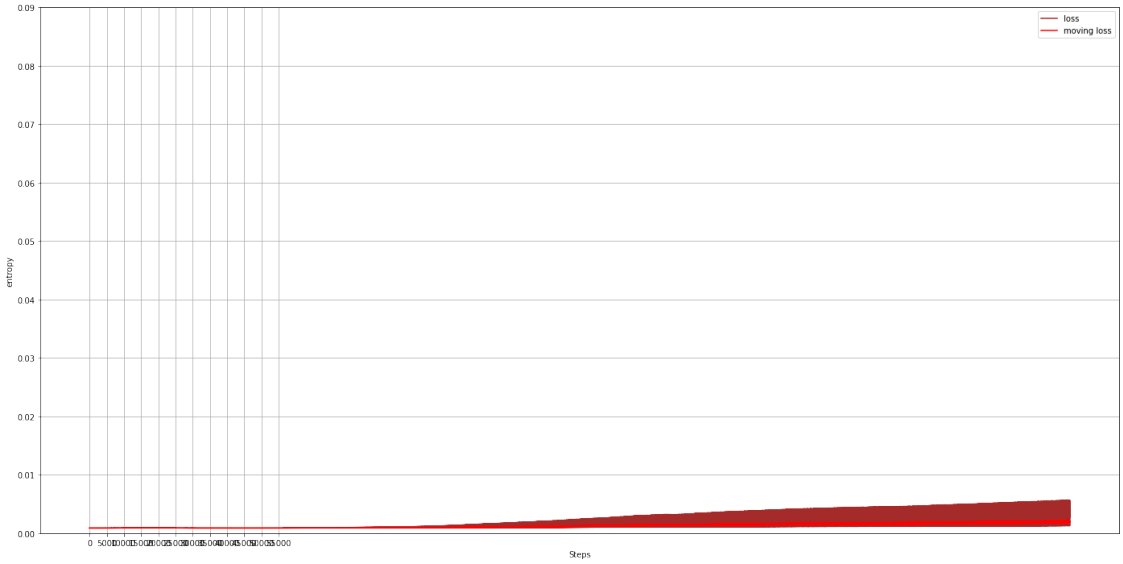
[439]:
```
#actor loss: 2082 520 0.0275358018

import matplotlib.pyplot as plt
```

```
import re
import numpy as np
from pathlib import Path

%matplotlib inline

def plot_loss(files):
    step = []
    loss = []
    mov_loss=[]
    mv = 0

    p = re.compile('^actor loss:.*')

    for file in files:
        with open(file) as f:
            lines = f.readlines()
            for line in lines:
                if not p.match(line):
                    continue
                items = re.split(" +", line)
                step.append(int(items[2]))
                loss_value=float(items[4])
                loss.append(loss_value)

                if mv == 0:
                    mv = loss_value
                else:
                    mv = 0.02*loss_value + 0.98*mov_loss[-1]

                mov_loss.append(mv)

    p1, = plt.plot(step, np.clip(loss,-10,5), color='brown',label='reward')
    p2, = plt.plot(step, np.clip(mov_loss,-10,5), color='red',label='moving␣
 ↪loss')

    plt.xlabel("Steps")
    plt.ylabel("actor")
    plt.legend(handles=[p1,p2],labels=['loss','moving loss'],loc='best')
    plt.xticks(np.arange(0, 60000, 5000))
    plt.yticks(np.arange(-10, 5, step=1))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```



[440]:
```
#entropy adjusted: 2082 520 0.000202997879

import matplotlib.pyplot as plt

import re
import numpy as np
from pathlib import Path

%matplotlib inline

def plot_loss(files):
    step = []
    loss = []
    mov_loss=[]
    mv = 0

    p = re.compile('^entropy adjusted:.*')

    for file in files:
        with open(file) as f:
            lines = f.readlines()
            for line in lines:
                if not p.match(line):
                    continue
                items = re.split(" +", line)
                step.append(int(items[2]))
                loss_value=float(items[4])
                loss.append(loss_value)
```

```
                if mv == 0:
                    mv = loss_value
                else:
                    mv = 0.02*loss_value + 0.98*mov_loss[-1]

                mov_loss.append(mv)

    p1, = plt.plot(step, loss, color='brown',label='reward')
    p2, = plt.plot(step, mov_loss, color='red',label='moving loss')

    plt.xlabel("Steps")
    plt.ylabel("entropy")
    plt.legend(handles=[p1,p2],labels=['loss','moving loss'],loc='best')
    plt.xticks(np.arange(0, 60000, 5000))
    plt.yticks(np.arange(0.00, 0.1, step=0.01))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```

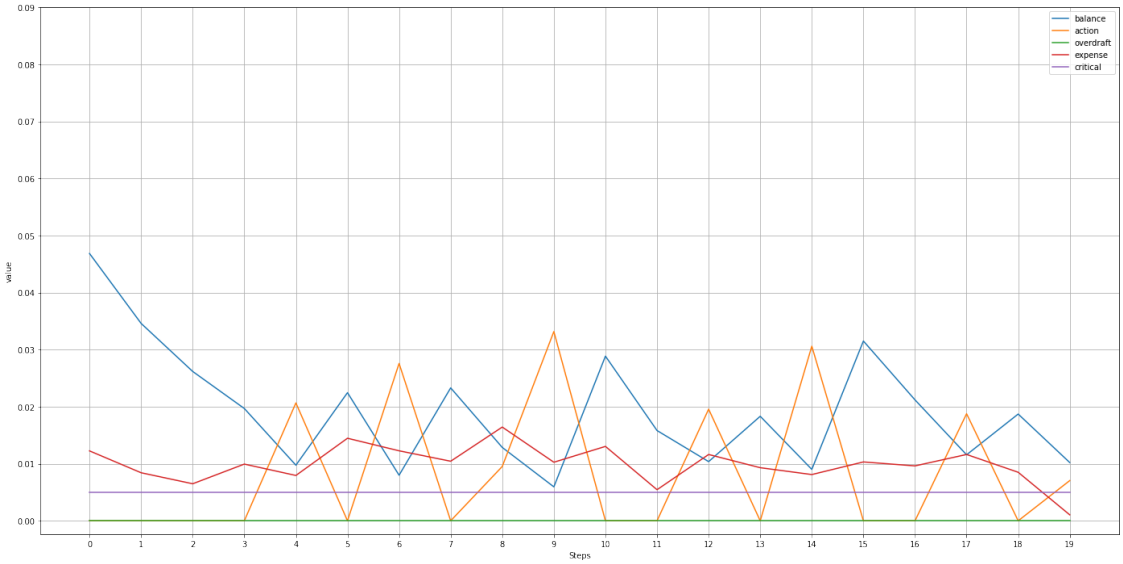


[441]:
```
import matplotlib.pyplot as plt

import re
import numpy as np
from pathlib import Path
```

```
%matplotlib inline

logfile = "../output.csv"

def plot_loss(files):
    step = []
    stock = []
    action = []
    overdraft = []
    stock_he = []
    action_he = []
    overdraft_he = []
    expense_estimate = []
    expense = []

    mov_loss=[]
    mv = 0

    i = 0
    for file in files:
        with open(file) as f:
            lines = f.readlines()

            it = iter(lines)

            while i < 50:
                try:
                  line = next(it)
                  stock.append(float(line.split(':')[1].split(',')[201]))
                  line = next(it)
                  action.append(float(line.split(':')[1].split(',')[201]))
                  line = next(it)
                  overdraft.append(float(line.split(':')[1].split(',')[201]))

                  line = next(it)
                  stock_he.append(float(line.split(':')[1].split(',')[201]))
                  line = next(it)
                  action_he.append(float(line.split(':')[1].split(',')[201]))
                  line = next(it)
                  overdraft_he.append(float(line.split(':')[1].split(',')[201]))

                  line = next(it)
                  expense_estimate.append(float(line.split(':')[1].
↪split(',')[201]))
                  line = next(it)
                  expense.append(float(line.split(':')[1].split(',')[201]))
```

```
                step.append(i)

                i = i + 1
              except:
                break

    p1, = plt.plot(step, stock, label='stock')
    p2, = plt.plot(step, action, label='action')
    p4, = plt.plot(step, overdraft,label='overdraft')

    #p5, = plt.plot(step, stock_he, label='stock he')
    #p6, = plt.plot(step, action_he, color='green',label='action he')
    #p8, = plt.plot(step, overdraft_he,label='overdraft he')

    #p9, = plt.plot(step, expense_estimate, color='yellow', label='estimate')
    p10, = plt.plot(step, expense, label='expense')
    p11, = plt.plot(step, np.repeat(0.005, len(step)), label='critical')

    plt.xlabel("Steps")
    plt.ylabel("value")
    #plt.legend(handles=[p1,p2],labels=['loss','moving loss'],loc='best')
    #plt.legend(handles=[p5,p6,p8,p9,p10],labels=[stock he','overdraft he',␣
↪'capacity'],loc='best')
    plt.
↪legend(handles=[p1,p2,p4,p10,p11],labels=['balance','action','overdraft','expense','critica
    #plt.legend(handles=[p5,p6,p8,p9],labels=['stock he','action he','overstock␣
↪he','overdraft he','sales'],loc='best')
    #plt.
↪legend(handles=[p1,p2,p4,p5,p6,p8,p9],labels=['stock','action','overstock','overdraft','sto
↪he','action he','overstock he','overdraft he','sales'],loc='best')
    #plt.legend(handles=[p2],labels=['actor'],loc='best')
    plt.xticks(np.arange(0, len(step), 1))
    plt.yticks(np.arange(0, 0.1, step=0.01))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    print (np.mean(stock))
    plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```

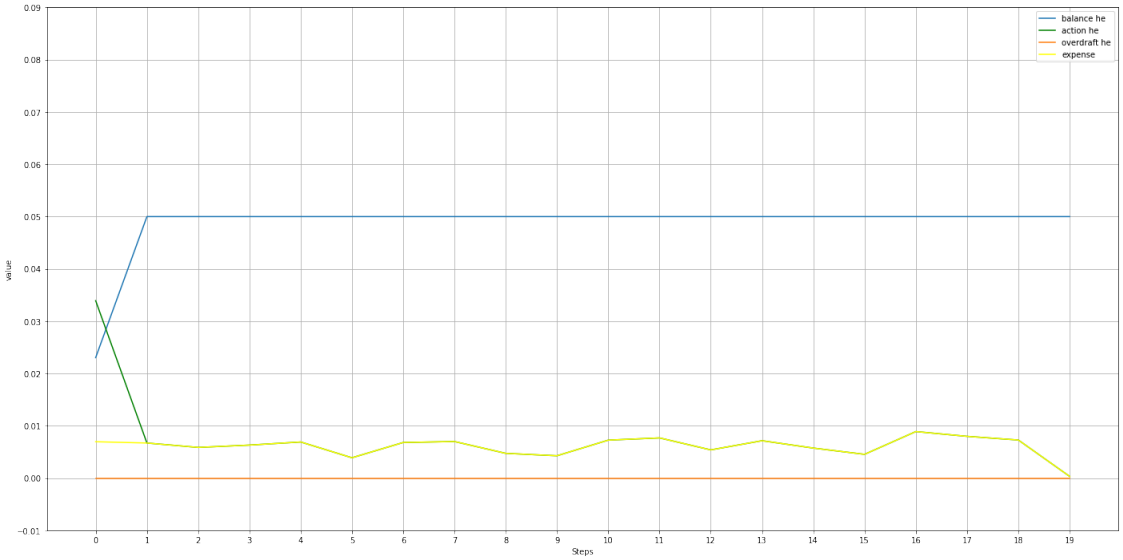
```
0.019263824549999996
```



[442]:
```
###### import matplotlib.pyplot as plt

import re
import numpy as np
from pathlib import Path

%matplotlib inline

logfile = "../output.csv"

def plot_loss(files):
    step = []
    stock = []
    action = []
    overdraft = []
    stock_he = []
    action_he = []
    overdraft_he = []
    expense_estimate = []
    expense = []

    mov_loss=[]
    mv = 0

    i = 0
    for file in files:
        with open(file) as f:
            lines = f.readlines()
```

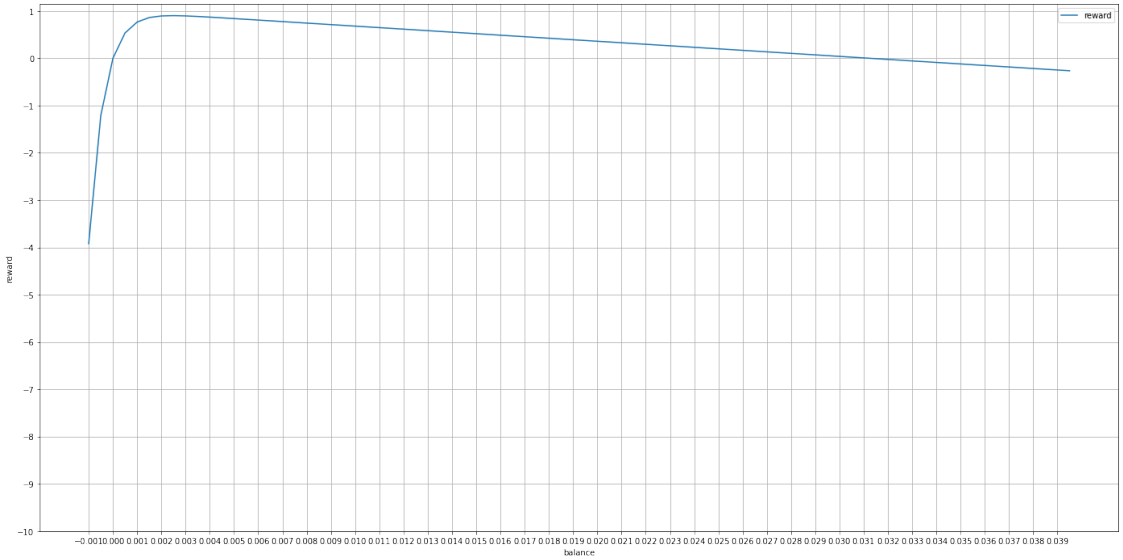
```
            it = iter(lines)

            while i < 50:
                try:
                  line = next(it)
                  stock.append(float(line.split(':')[1].split(',')[7]))
                  line = next(it)
                  action.append(float(line.split(':')[1].split(',')[7]))
                  line = next(it)
                  overdraft.append(float(line.split(':')[1].split(',')[7]))

                  line = next(it)
                  stock_he.append(float(line.split(':')[1].split(',')[7]))
                  line = next(it)
                  action_he.append(float(line.split(':')[1].split(',')[7]))
                  line = next(it)
                  overdraft_he.append(float(line.split(':')[1].split(',')[7]))

                  line = next(it)
                  expense_estimate.append(float(line.split(':')[1].
↪split(',')[7]))
                  line = next(it)
                  expense.append(float(line.split(':')[1].split(',')[7]))

                  step.append(i)

                  i = i + 1
                except:
                  break

    #p1, = plt.plot(step, stock, label='stock')
    #p2, = plt.plot(step, action, label='action')
    #p4, = plt.plot(step, overdraft,label='overdraft')

    p5, = plt.plot(step, stock_he, label='balance he')
    p6, = plt.plot(step, action_he, color='green',label='action he')
    p8, = plt.plot(step, overdraft_he,label='overdraft he')

    #p9, = plt.plot(step, expense_estimate, color='yellow', label='estimate')
    p10, = plt.plot(step, expense, color='yellow', label='expense')

    plt.xlabel("Steps")
    plt.ylabel("value")
    #plt.legend(handles=[p1,p2],labels=['loss','moving loss'],loc='best')
    #plt.legend(handles=[p5,p6,p8,p9,p10],labels=['stock he','overdraft he',␣
↪'capacity'],loc='best')
```

```
    #plt.
↪legend(handles=[p1,p2,p4,p9],labels=['stock','action','overdraft','sales'],loc='best')
    plt.legend(handles=[p5,p6,p8,p10],labels=['balance he','action␣
↪he','overdraft he','expense'],loc='best')
    #plt.
↪legend(handles=[p1,p2,p4,p5,p6,p8,p9],labels=['stock','action','overstock','overdraft','sto
↪he','action he','overstock he','overdraft he','sales'],loc='best')
    #plt.legend(handles=[p2],labels=['actor'],loc='best')
    plt.xticks(np.arange(0, len(step), 1))
    plt.yticks(np.arange(-0.01, 0.1, step=0.01))
    plt.grid(True, which='both')
    #plt.minorticks_on()
    #plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss([logfile])
```



[443]:
```
###### import matplotlib.pyplot as plt

import numpy as np

%matplotlib inline

def plot_loss():
  step = []
  r = []

  def zero(i):
```

```
    if i <= 0:
      return 1
    return 0

  def critical(i):
    if i <= 0.005:
      return 1
    return 0

  k1 = 1
  k2 = 0.1
  k3 = 32
  k4 = 1600

  #for i in np.arange(0, 0.1, step=0.01):
  for i in np.arange(-0.001, 0.04, step=0.0005):
    step.append(i)
    #r(s) = 1 - k1*zero(i) - k2*critical - k3*balance
    #r.append(k1*zero(i) + k2*critical(i) + k3*i)
    r.append(1 - (np.exp(-k4*i)+k3*i))
    #r.append(k3*i)

  plt.xlabel("balance")
  plt.ylabel("reward")

  p1, = plt.plot(step, r, label='reward')
  plt.legend(handles=[p1],labels=['reward'],loc='best')

  plt.xticks(np.arange(-0.001, 0.04, step=0.001))
  plt.yticks(np.arange(-10, 2, step=1))
  plt.grid(True, which='both')
  #plt.minorticks_on()
  plt.show()

fig = plt.figure(figsize=(24,12))
plot_loss()
```


reward
1
0
−1
−2
−3
−4
−5
−6
−7
−8
−9
−10
reward
balance

[ ]: