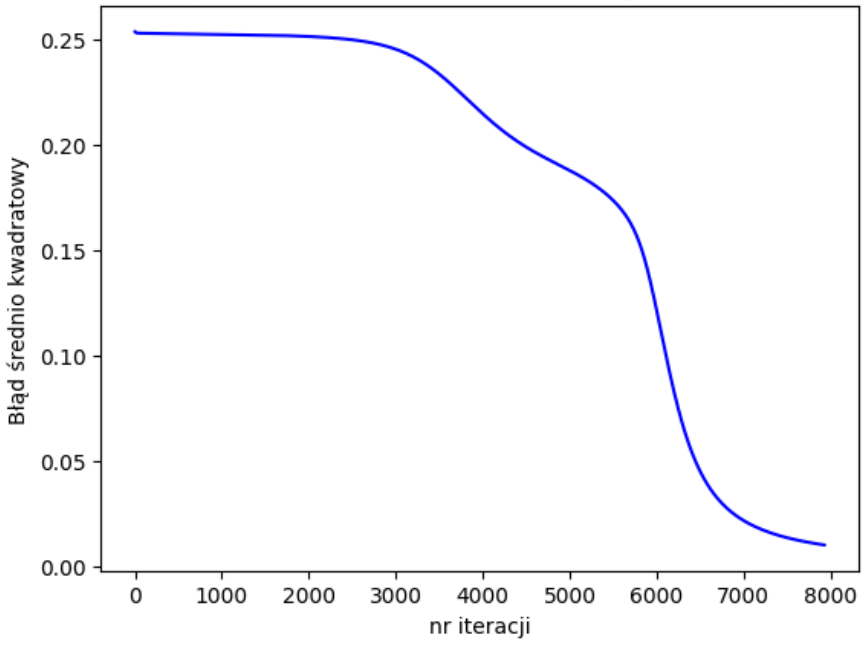
In [3]:
```
from os import truncate
import random
import math
import matplotlib.pyplot as plt
L=2
mi = 0.1
N = [2, 2, 1]

w = [
      [],
      [[],[0,0,0],[0,0,0]],
      [[],[0,0,0]]
    ]
s = [
      [],
      [None, 0, 0],
      [None, 0]
    ]
x = [
      [],
      [-1,0,0],
      [-1,0,0]
    ]

y = [
      [],
      [None, 0, 0],
      [None, 0]
    ]

e = [
      [],
      [None, 0, 0],
      [None, 0]
    ]
dlt = [
        [],
        [None, 0, 0],
        [None, 0]
      ]

def losuj_w():
  for k in range(1,L+1):
    for i in range(1, N[k]+1):
      for j in range(0,N[k-1]+1):
        w[k][i][j] = random.random()*2-1

def f(_s):
  return 1/(1+math.exp(-_s))

def f_poch(_s):
  return f(_s)*(1-f(_s))

def licz_y(_u):
  for k in range(1,L+1):
    for i in range(1,N[k-1]+1):
      if k==1:
        x[k][i] = _u[i-1]
```

```
      else:
        x[k][i] = y[k-1][i]
    for i in range(1,N[k]+1):
      s[k][i] = 0
      for j in range(0,N[k-1]+1):
        s[k][i] += w[k][i][j] * x[k][j]
      y[k][i] = f(s[k][i])
  return y[L][1]

def licz_e(d):
  for k in range(L, 1-1, -1):
    for i in range(1, N[k]+1):
      if k==L:
        e[k][i] = d - y[k][i]
      else:
        e[k][i] = 0;
        for j in range(1,N[k+1]+1):
          e[k][i] += dlt[k+1][j]*w[k+1][j][i]
      dlt[k][i] = e[k][i]*f_poch(s[k][i])
u = [
      [0,0],
      [0,1],
      [1,0],
      [1,1]
    ]
d = [0,
     1,
     1,
     0]
BK = []
it = []
def algorytm(_n):
  losuj_w()
  #epochs loop
  for ni in range(1,_n):
    #in vectors loop
    for ui in range(len(u)):
      result = licz_y(u[ui])
      licz_e(d[ui])
      for k in range(L, 0, -1):
        for i in range(1, N[k]+1, 1):
          for j in range(0,N[k-1]+1,1):
            w[k][i][j] = w[k][i][j] + mi*dlt[k][i]*x[k][j]

def algorytmBK(_n, warunek_stopu):
  losuj_w()
  #epochs loop
  for ni in range(1,_n):
    #in vectors loop
    _Q = 0.0;
    for ui in range(len(u)):
      result = licz_y(u[ui])
      licz_e(d[ui])
      for k in range(L, 0, -1):
        for i in range(1, N[k]+1, 1):
          for j in range(0,N[k-1]+1,1):
            w[k][i][j] = w[k][i][j] + mi*dlt[k][i]*x[k][j]
      _Q += pow(d[ui]-result,2)
    _Q = _Q *1/len(d)
    BK.append(_Q)
```

```
    it.append(ni+1)
    if(_Q <warunek_stopu):
      return True
  return False
stop = 0.01;
for ind in range(1,6):
  BK = []
  it = []
  done = algorytmBK(500000,stop)
  result = [0.0,0.0,0.0,0.0]
  rawresult = [0.0,0.0,0.0,0.0]
  for i in range(len(result)):
    rawresult[i] = licz_y(u[i])
    result[i]=round(rawresult[i])
  print("Próba nr:",ind)
  print("wyniki bez zaokrągleń:",rawresult)
  print("WYNIKI:",result)
  print("OCZEKIWANE:", d)
  if(done):
    print("Potrzebne było", it[len(it)-1], "iteracji aby osiągnąć błąd śr
  else:
    print("Mimo",it[len(it)-1]," iteracjom nie udało się osiągnać zakłada
  plt.figure(ind)
  plt.plot(it, BK, "b-")
  plt.xlabel("nr iteracji")
  plt.ylabel("Błąd średnio kwadratowy")
  title = "Próba nr: "+str(ind)+" Zmieniający się błąd średnio kwadratowy
  plt.title(title)
  plt.show()
```

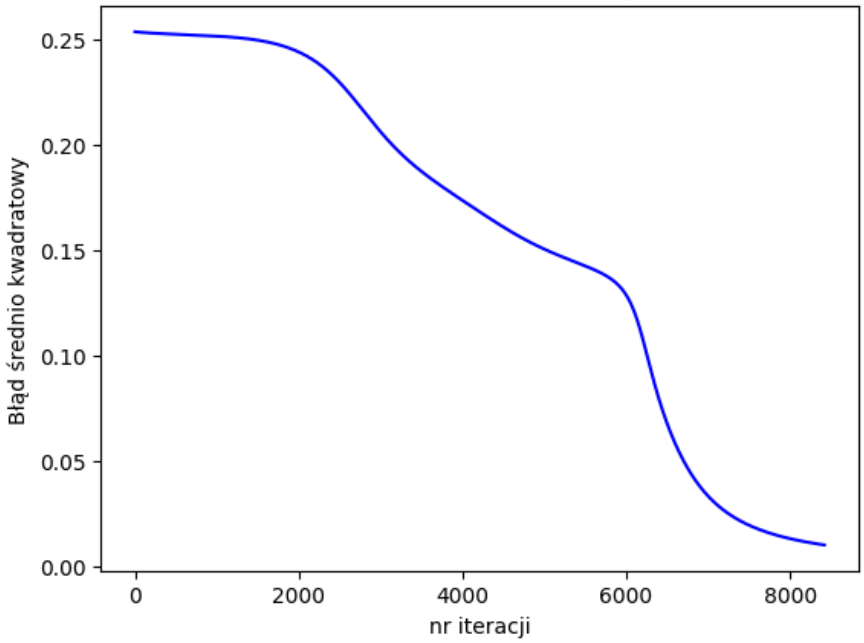
```
Próba nr: 1
wyniki bez zaokrągleń: [0.10331929435528953, 0.9013424036780845, 0.8931549
444999, 0.08972735513118568]
WYNIKI: [0, 1, 1, 0]
OCZEKIWANE: [0, 1, 1, 0]
Potrzebne było 7928 iteracji aby osiągnąć błąd średnio kwadratowy mniejszy
niż 0.01
```

Próba nr: 1 Zmieniający się błąd średnio kwadratowy względem kolejnych epok nauczania



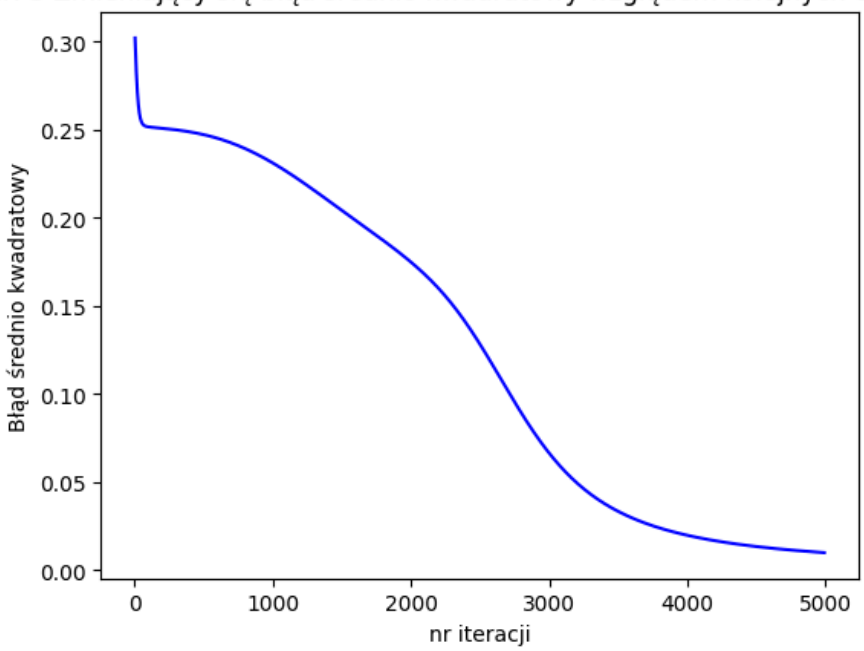
```
Próba nr: 2
wyniki bez zaokrągleń: [0.0732894723345218, 0.9050872418596462, 0.90665989
97915444, 0.12945993179162493]
WYNIKI: [0, 1, 1, 0]
OCZEKIWANE: [0, 1, 1, 0]
Potrzebne było 8418 iteracji aby osiągnąć błąd średnio kwadratowy mniejszy
niż 0.01
```

Próba nr: 2 Zmieniający się błąd średnio kwadratowy względem kolejnych epok nauczania

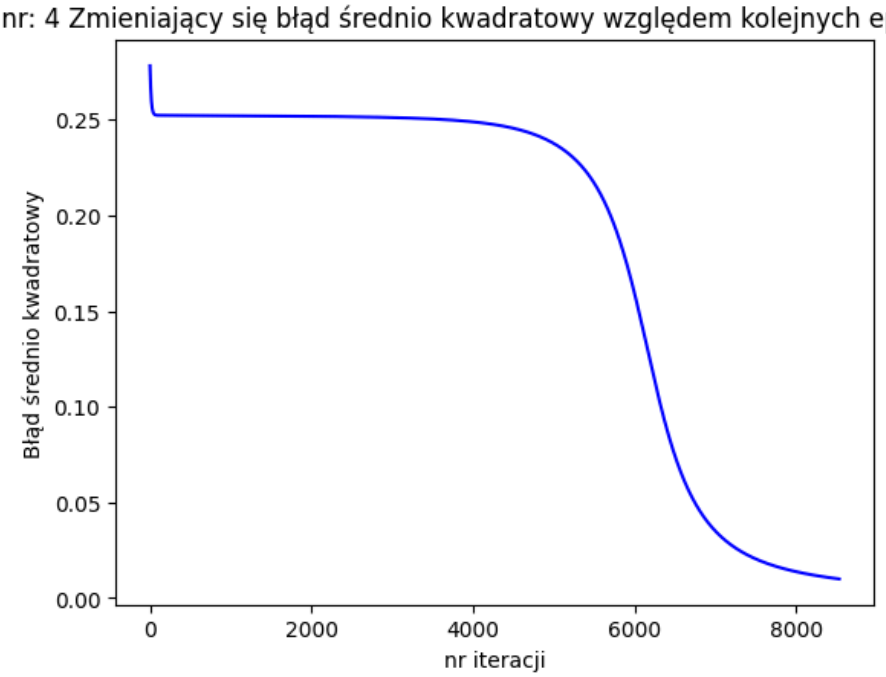



```
Próba nr: 3
wyniki bez zaokrągleń: [0.08230960126791742, 0.896253243390597, 0.89631267
19280484, 0.10765695373818361]
WYNIKI: [0, 1, 1, 0]
OCZEKIWANE: [0, 1, 1, 0]
Potrzebne było 4990 iteracji aby osiągnąć błąd średnio kwadratowy mniejszy
niż 0.01
```

Próba nr: 3 Zmieniający się błąd średnio kwadratowy względem kolejnych epok nauczania



```
Próba nr: 4
wyniki bez zaokrągleń: [0.11079786777828139, 0.9043716408306153, 0.9041798
061153694, 0.09640208246903746]
WYNIKI: [0, 1, 1, 0]
OCZEKIWANE: [0, 1, 1, 0]
Potrzebne było 8532 iteracji aby osiągnąć błąd średnio kwadratowy mniejszy
niż 0.01
```




```
Próba nr: 5
wyniki bez zaokrągleń: [0.11101291232271833, 0.9132616282850746, 0.8965463
260263653, 0.09676503757250966]
WYNIKI: [0, 1, 1, 0]
OCZEKIWANE: [0, 1, 1, 0]
Potrzebne było 6959 iteracji aby osiągnąć błąd średnio kwadratowy mniejszy
niż 0.01
```