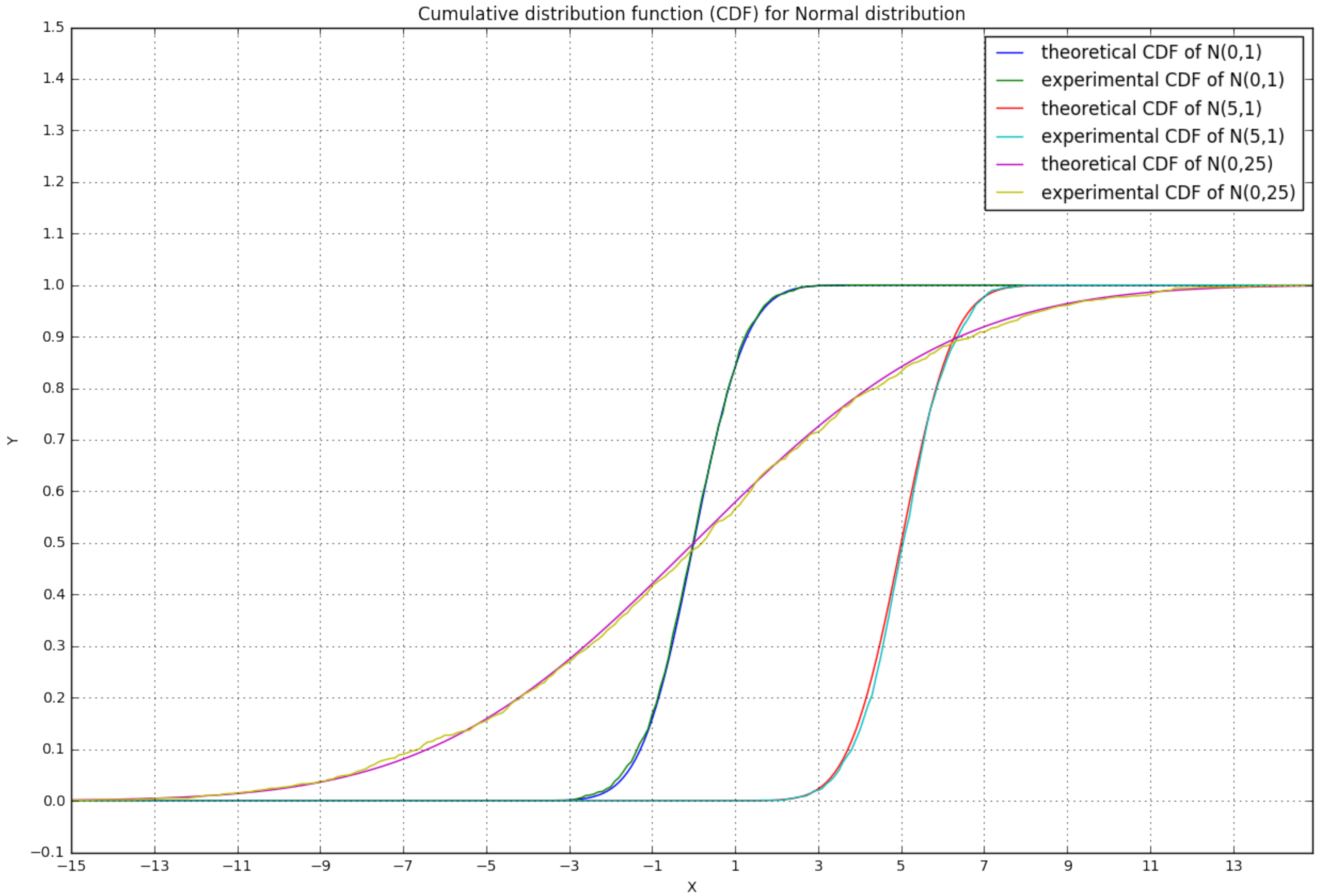
# Исследование нормального распределения

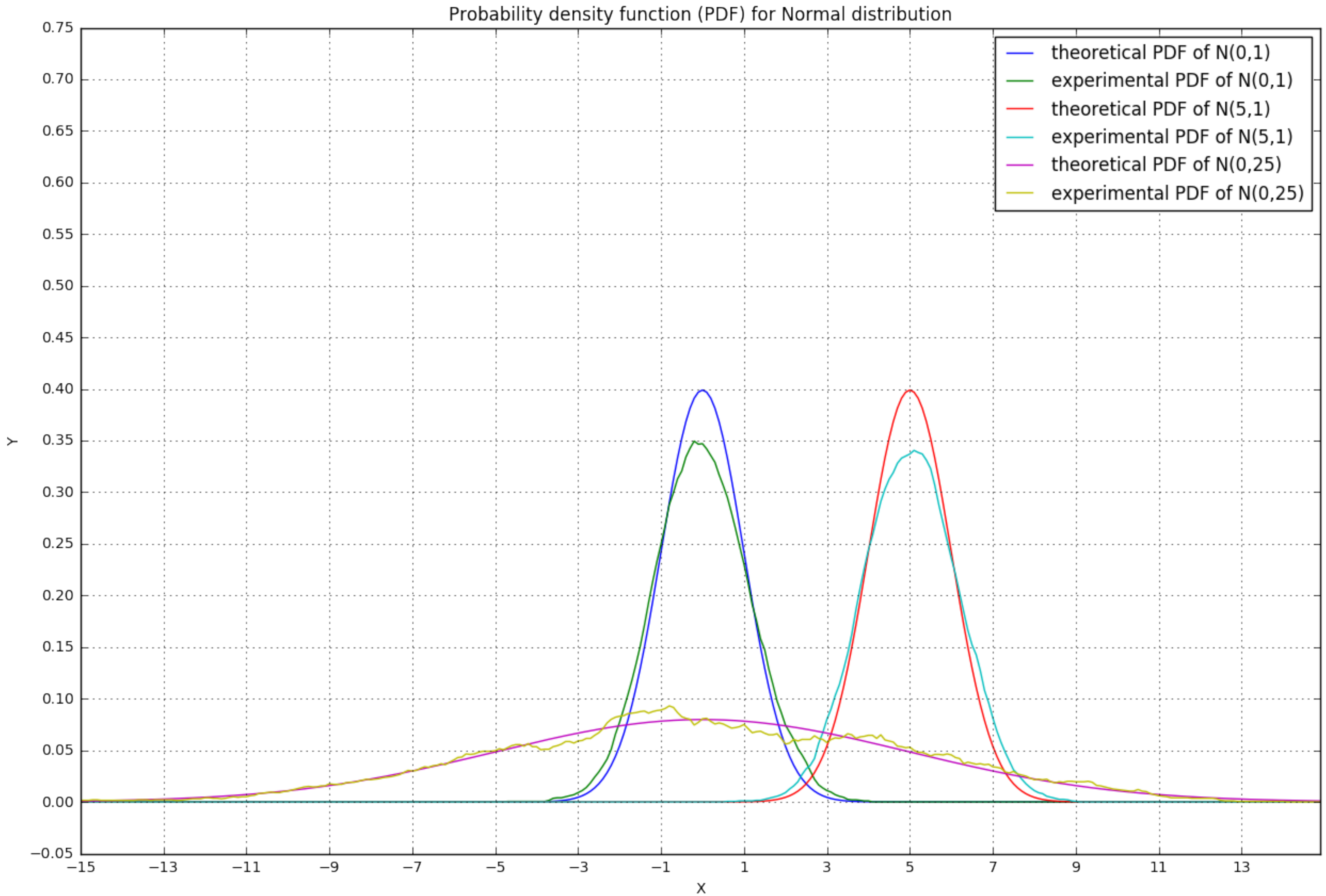
In [8]:

```
#!/usr/bin/env python

import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
from norm_lib import NormDistribution

X_MIN, X_MAX = -15, 15
Y_MIN, Y_MAX = float(.0), float(1.0)
X_FUNC_STEP = 0.1
X_FUNC = np.arange(X_MIN, X_MAX, X_FUNC_STEP)
IMG_X_SIZE, IMG_Y_SIZE = 15, 10
X_GRID_STEP = (X_MAX - X_MIN) / float(IMG_X_SIZE)
Y_GRID_STEP = (Y_MAX - Y_MIN) / float(IMG_Y_SIZE)
Y_GRID = np.arange(Y_MIN - Y_GRID_STEP, Y_MAX * 1.5 + Y_GRID_STEP, Y_GRID_STE
P)
X_GRID = np.arange(X_MIN, X_MAX, X_GRID_STEP)


fig = plt.figure(figsize=(IMG_X_SIZE, IMG_Y_SIZE))
plt.title("Cumulative distribution function (CDF) for Normal distribution")
for median, dispersion in [(0, 1), (5, 1), (0, 25)]:
    norm_distribution = NormDistribution(median, dispersion)
    plt.plot(X_FUNC, norm_distribution.cdf(X_FUNC), label="theoretical CDF of
%s" % norm_distribution)
    plt.plot(X_FUNC, norm_distribution.ecdf(X_FUNC), label="experimental CDF
of %s" % norm_distribution)
    plt.xticks(X_GRID)
    plt.yticks(Y_GRID)
    plt.legend()

    plt.xlabel("X")
    plt.ylabel("Y")
```

```
    plt.grid(True)
plt.show()
```



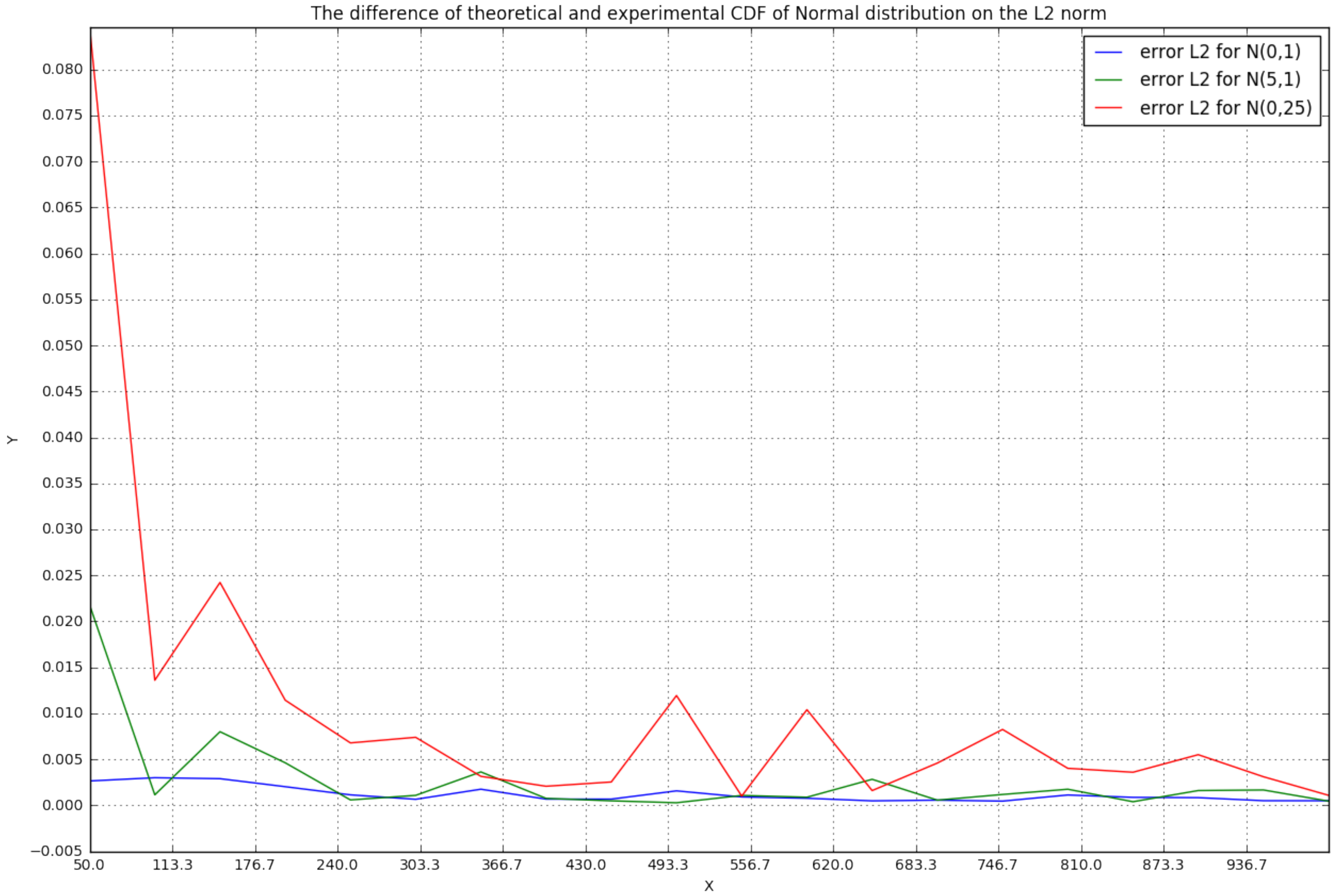
In [12]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
from norm_lib import NormDistribution

X_MIN, X_MAX = -15, 15
Y_MIN, Y_MAX = float(.0), float(.5)
X_FUNC_STEP = 0.1
X_FUNC = np.arange(X_MIN, X_MAX, X_FUNC_STEP)
IMG_X_SIZE, IMG_Y_SIZE = 15, 10
X_GRID_STEP = (X_MAX - X_MIN) / float(IMG_X_SIZE)
Y_GRID_STEP = (Y_MAX - Y_MIN) / float(IMG_Y_SIZE)
Y_GRID = np.arange(Y_MIN - Y_GRID_STEP, Y_MAX * 1.5 + Y_GRID_STEP, Y_GRID_STE
P)
X_GRID = np.arange(X_MIN, X_MAX, X_GRID_STEP)

fig = plt.figure(figsize=(IMG_X_SIZE, IMG_Y_SIZE))
plt.title("Probability density function (PDF) for Normal distribution")
for median, dispersion in [(0, 1), (5, 1), (0, 25)]:
    norm_distribution = NormDistribution(median, dispersion, None)
    plt.plot(X_FUNC, norm_distribution.pdf(X_FUNC), label="theoretical PDF of
%s" % norm_distribution)
    plt.plot(X_FUNC, norm_distribution.epdf(X_FUNC), label="experimental PDF
of %s" % norm_distribution)
    plt.xticks(X_GRID)
    plt.yticks(Y_GRID)
    plt.legend()

    plt.xlabel("X")
    plt.ylabel("Y")

    plt.grid(True)
plt.show()
```


Probability density function (PDF) for Normal distribution
theoretical PDF of N(0,1)
experimental PDF of N(0,1)
theoretical PDF of N(5,1)
experimental PDF of N(5,1)
theoretical PDF of N(0,25)
experimental PDF of N(0,25)
Y
X
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−15
−13
−11
−9
−7
−5
−3
−1
1
3
5
7
9
11
13

In [13]:

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
from scipy.integrate import quad
from norm_lib import NormDistribution

X_MIN, X_MAX = float(50), float(1000)
Y_MIN, Y_MAX = float(.0), float(.05)
X_FUNC_STEP = 50
X_FUNC = np.arange(X_MIN, X_MAX, X_FUNC_STEP)
IMG_X_SIZE, IMG_Y_SIZE = 15, 10
X_GRID_STEP = (X_MAX - X_MIN) / float(IMG_X_SIZE)
Y_GRID_STEP = (Y_MAX - Y_MIN) / float(IMG_Y_SIZE)
Y_GRID = np.arange(Y_MIN - Y_GRID_STEP, Y_MAX * 1.5 + Y_GRID_STEP, Y_GRID_STE
P)
X_GRID = np.arange(X_MIN, X_MAX, X_GRID_STEP)
X_START, X_END = -15, 15


def sample_size_func(distr, sample_size):
    sample = distr.gen_sample(sample_size)
    f1 = lambda x: norm_distribution.ecdf_in_point(x, sample)
    f2 = lambda x: norm_distribution.cdf_in_point(x)
    return quad(lambda x: (f1(x) - f2(x)) ** 2, X_START, X_END)[0]


fig = plt.figure(figsize=(IMG_X_SIZE, IMG_Y_SIZE))
plt.title("The difference of theoretical and experimental CDF of Normal distr
ibution on the L2 norm")
for median, dispersion in [(0, 1), (5, 1), (0, 25)]:
    norm_distribution = NormDistribution(median, dispersion, None)

    sample_size_array = np.arange(int(X_MIN), int(X_MAX + X_FUNC_STEP), int(X
```

```
_FUNC_STEP))
    vectorised_func = np.vectorize(lambda size: sample_size_func(norm_distrib
ution, size))
    distance_array = vectorised_func(sample_size_array)

    # print sample_size_dict
    plt.plot(sample_size_array, distance_array, label="error L2 for %s" % nor
m_distribution)
    # plt.plot(X_FUNC, norm_distribution.epdf(X_FUNC), color=experimental_col
or)
    plt.xticks(X_GRID)
    plt.yticks(Y_GRID)
    plt.legend()

    plt.xlabel("X")
    plt.ylabel("Y")

    plt.grid(True)
plt.show()
```

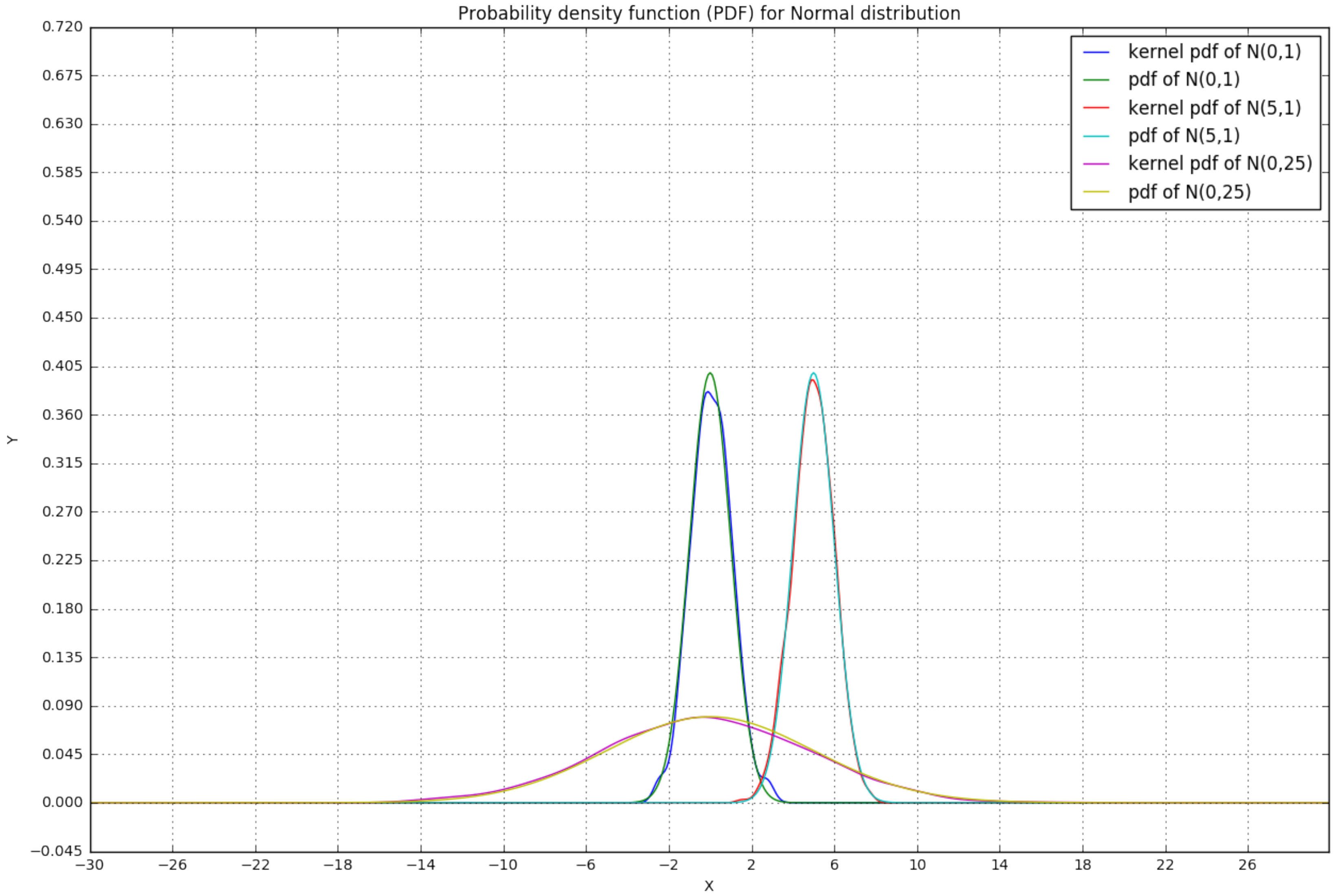


The difference of theoretical and experimental CDF of Normal distribution on the L2 norm
error L2 for N(0,1)
error L2 for N(5,1)
error L2 for N(0,25)
Y
0.080
0.075
0.070
0.065
0.060
0.055
0.050
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
−0.005
50.0
113.3
176.7
240.0
303.3
366.7
430.0
493.3
556.7
620.0
683.3
746.7
810.0
873.3
936.7
X

In [14]:

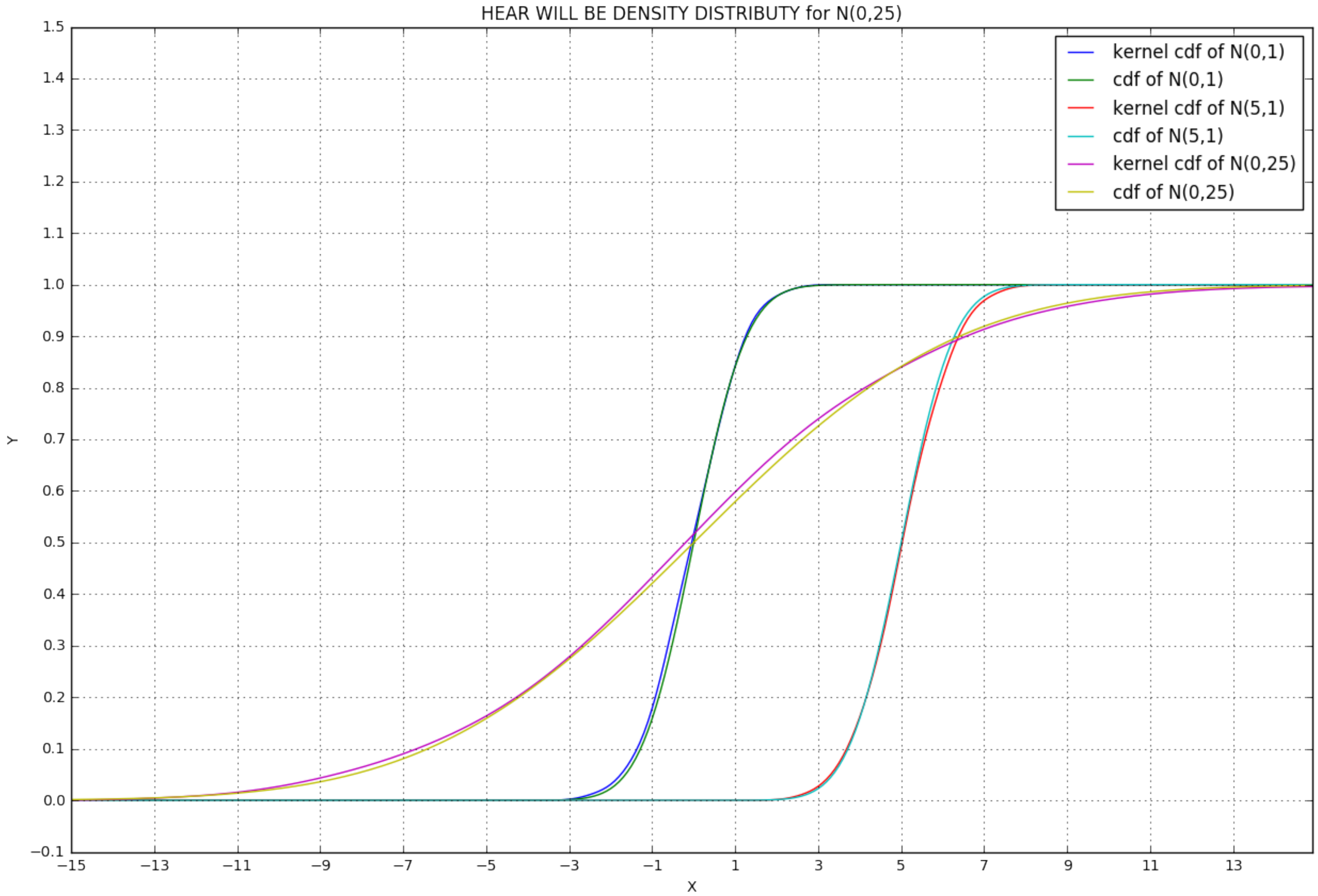
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
from scipy.integrate import quad
from norm_lib import NormDistribution

X_MIN, X_MAX = -30, 30
Y_MIN, Y_MAX = float(.0), float(.45)
X_FUNC_STEP = .1
X_FUNC = np.arange(X_MIN, X_MAX, X_FUNC_STEP)
IMG_X_SIZE, IMG_Y_SIZE = 15, 10
X_GRID_STEP = (X_MAX - X_MIN) / float(IMG_X_SIZE)
Y_GRID_STEP = (Y_MAX - Y_MIN) / float(IMG_Y_SIZE)
Y_GRID = np.arange(Y_MIN - Y_GRID_STEP, Y_MAX * 1.5 + Y_GRID_STEP, Y_GRID_STE
P)
X_GRID = np.arange(X_MIN, X_MAX, X_GRID_STEP)


# print "integral = %s" % quad(lambda x: x ** 2, -5., 5.)
fig = plt.figure(figsize=(IMG_X_SIZE, IMG_Y_SIZE))
plt.title("Probability density function (PDF) for Normal distribution")
# for median, dispersion in [(0, 1)]:
for median, dispersion in [(0, 1), (5, 1), (0, 25)]:
    norm_distribution = NormDistribution(median, dispersion, None)
    sample = norm_distribution.gen_sample()
    kernel_pdf = norm_distribution.gauss_kernel(sample)
    pdf_array = kernel_pdf(X_FUNC)
    # print f(X_FUNC)
    # print sample_size_dict
    plt.plot(X_FUNC, pdf_array, label="kernel pdf of " + str(norm_distributio
n))
    plt.plot(X_FUNC, norm_distribution.pdf(X_FUNC), label="pdf of " + str(nor
m_distribution))
```

```
    plt.xticks(X_GRID)
    plt.yticks(Y_GRID)
    plt.legend()

    plt.xlabel("X")
    plt.ylabel("Y")

    plt.grid(True)
plt.show()
```


Probability density function (PDF) for Normal distribution
kernel pdf of N(0,1)
pdf of N(0,1)
kernel pdf of N(5,1)
pdf of N(5,1)
kernel pdf of N(0,25)
pdf of N(0,25)
Y
X
0.720
0.675
0.630
0.585
0.540
0.495
0.450
0.405
0.360
0.315
0.270
0.225
0.180
0.135
0.090
0.045
0.000
−0.045
−30
−26
−22
−18
−14
−10
−6
−2
2
6
10
14
18
22
26

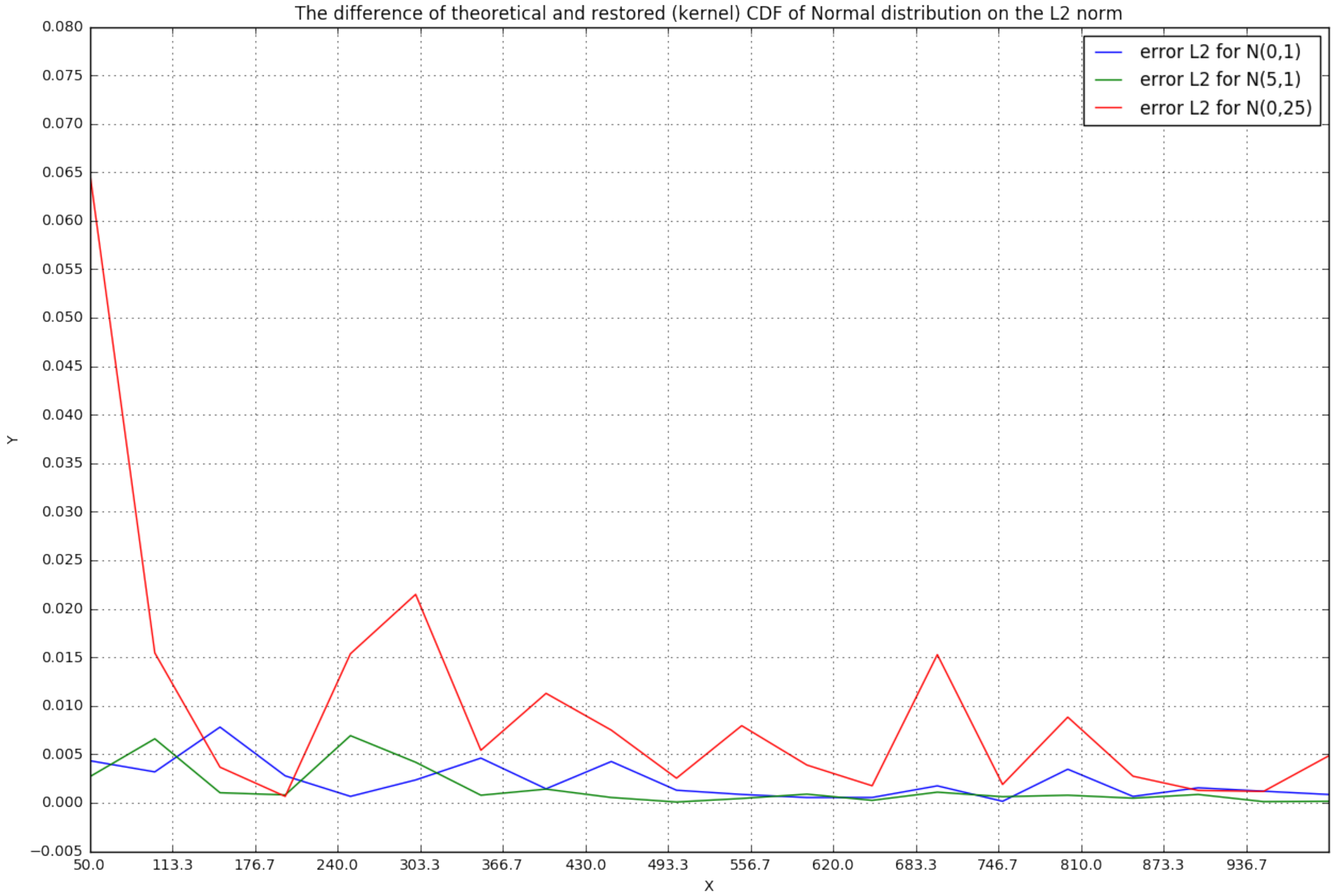
In [5]:
```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
from scipy.integrate import quad
from norm_lib import NormDistribution, integral

X_MIN, X_MAX = float(-15), float(15)
Y_MIN, Y_MAX = float(.0), float(1.)

X_FUNC_STEP = .1
X_FUNC = np.arange(X_MIN, X_MAX, X_FUNC_STEP)
IMG_X_SIZE, IMG_Y_SIZE = 15, 10
X_GRID_STEP = (X_MAX - X_MIN) / float(IMG_X_SIZE)
Y_GRID_STEP = (Y_MAX - Y_MIN) / float(IMG_Y_SIZE)
Y_GRID = np.arange(Y_MIN - Y_GRID_STEP, Y_MAX * 1.5 + Y_GRID_STEP, Y_GRID_STE
P)
X_GRID = np.arange(X_MIN, X_MAX, X_GRID_STEP)


fig = plt.figure(figsize=(IMG_X_SIZE, IMG_Y_SIZE))
plt.title("Cumulative distribution function (CDF) for Normal distribution")
for median, dispersion in [(0, 1), (5, 1), (0, 25)]:
    norm_distribution = NormDistribution(median, dispersion, None)
    sample = norm_distribution.gen_sample()

    kernel_pdf = norm_distribution.gauss_kernel(sample)
    pdf_array = kernel_pdf(X_FUNC)
    kernel_cdf = np.vectorize(lambda x: quad(kernel_pdf, X_MIN, float(x))[0])
    cdf_array = kernel_cdf(X_FUNC)

    plt.plot(X_FUNC, cdf_array, label="kernel cdf of " + str(norm_distributio
n))
    plt.plot(X_FUNC, norm_distribution.cdf(X_FUNC), label="cdf of " + str(nor
```

```
m_distribution))

    plt.xticks(X_GRID)
    plt.yticks(Y_GRID)
    plt.legend()

    plt.xlabel("X")
    plt.ylabel("Y")
    plt.grid(True)
plt.show()
```


HEAR WILL BE DENSITY DISTRIBUTY for N(0,25)
kernel cdf of N(0,1)
cdf of N(0,1)
kernel cdf of N(5,1)
cdf of N(5,1)
kernel cdf of N(0,25)
cdf of N(0,25)
Y
X
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−15
−13
−11
−9
−7
−5
−3
−1
1
3
5
7
9
11
13

In [3]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
from scipy.integrate import quad
from norm_lib import NormDistribution

X_MIN, X_MAX = float(50), float(1000)
Y_MIN, Y_MAX = float(.0), float(0.05)

X_FUNC_STEP = 50
X_FUNC = np.arange(X_MIN, X_MAX, X_FUNC_STEP)
IMG_X_SIZE, IMG_Y_SIZE = 15, 10
X_GRID_STEP = (X_MAX - X_MIN) / float(IMG_X_SIZE)
Y_GRID_STEP = (Y_MAX - Y_MIN) / float(IMG_Y_SIZE)
Y_GRID = np.arange(Y_MIN - Y_GRID_STEP, Y_MAX * 1.5 + Y_GRID_STEP, Y_GRID_STE
P)
X_GRID = np.arange(X_MIN, X_MAX, X_GRID_STEP)

X_START, X_END = -15, 15


def sample_size_func(distr, sample_size):
    sample = distr.gen_sample(sample_size)
    kernel_pdf = norm_distribution.gauss_kernel(sample)
    f1 = lambda x: quad(kernel_pdf, X_START, float(x))[0]
    f2 = lambda x: norm_distribution.cdf_in_point(x)
    return quad(lambda x: (f1(x) - f2(x)) ** 2, X_START, X_END)[0]


fig = plt.figure(figsize=(IMG_X_SIZE, IMG_Y_SIZE))
plt.title("The difference of theoretical and restored (kernel) CDF of Normal
distribution on the L2 norm")
for median, dispersion in [(0, 1), (5, 1), (0, 25)]:
```

```
    norm_distribution = NormDistribution(median, dispersion, None)
    sample_size_array = np.arange(int(X_MIN), int(X_MAX + X_FUNC_STEP), int(X
_FUNC_STEP))
    vectorised_func = np.vectorize(lambda size: sample_size_func(norm_distrib
ution, size))
    distance_array = vectorised_func(sample_size_array)
    plt.plot(sample_size_array, distance_array, label="error L2 for %s" % nor
m_distribution)

    plt.xticks(X_GRID)
    plt.yticks(Y_GRID)
    plt.legend()

    plt.xlabel("X")
    plt.ylabel("Y")
    plt.grid(True)
plt.show()
```




# Выводы

По графикам зависимости погрешности от размера выборки можно сделать следующие выводы:

- ядровое восстановление функции нормального распределения имеет погрешность сравнимую с погрешностью эмпирической функцией распределения, полученной методом Монте-Карло;
- нормальное распределение с большей диспресией имеет в среднем большую погрешность.

```
In [ ]:
```