In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from statistics import Rxxwf
import scipy as sp
import scipy.stats
from statistics import Rxx,Kxx

def transform_sig(X, H):
    """
    Расчет выходных сигналов на основе входных
    """
    outp = np.zeros(len(H))
    for j in range(len(H)):
        for i in range(j+1):
            outp[j] = outp[j] + X[i] *  H[j-i]
    return outp

def weight_functions():
 # Генерация сигнала вида sin(x)
    N = 500
    Xsin = np.array(range(N))
    Ysin = np.zeros(N)
    for i in range(N):
        Ysin[i] = np.sin(Xsin[i]*0.02)

    f = open('sin(x).txt', 'w')
    for item in Ysin:
        f.write("%f\n" % item)
    f.close()

    # Безинерционное звено
    N = 500
    X = np.array(range(N))
    Y = np.zeros(N)
    Y[0] = 1

    plt.subplot2grid((2, 4), (0, 1), colspan=3)
    plt.title('Безинерционное звено')
    plt.xlabel('t')
    plt.ylabel('h(t)')
    plt.grid(True)
    plt.plot(X, Y)

    plt.subplot2grid((2, 4), (0, 0))
    # Корреляционная функция безинерционного звена
    R, Q = Rxxwf(Y)
    plt.xlabel('q')
    plt.ylabel('Rxx')
    plt.grid(True)
    plt.fill_between(range(Q), R, np.zeros(Q))
    plt.plot(R)

    plt.subplot2grid((2, 4), (1, 0), colspan=2)
    plt.title('Входной сигнал')
    plt.xlabel('x')
    plt.ylabel('sin(x)')
    plt.grid(True)
    plt.plot(X, Ysin)

    outp = transform_sig(Ysin, Y)
    plt.subplot2grid((2, 4), (1, 2), colspan=2)
    plt.title('Выходной сигнал')
    plt.xlabel('x')
    plt.ylabel('sin(x)')
    plt.grid(True)
    plt.plot(X, outp)
    f = open('sin(x) - безинерционное звено.txt', 'w')
```

```
for item in outp:
    f.write("%f\n" % item)
f.close()

plt.tight_layout()
plt.savefig("h1.svg",type="svg",dpi=100)

# Звено с чистым запаздыванием
N = 500
X = (np.array(range(N)) - 0.2 * N)
Y = np.zeros(N)
Y[100] = 1

plt.subplot2grid((2, 4), (0, 1), colspan=3)
plt.title('Звено с чистым запаздыванием')
plt.xlabel('t')
plt.ylabel('h(t)')
plt.grid(True)
plt.plot(X, Y)

plt.subplot2grid((2, 4), (0, 0))
# Корреляционная функция звена с чистым запаздыванием
R, Q = Rxxwf(Y)
plt.xlabel('q')
plt.ylabel('Rxx')
plt.grid(True)
plt.fill_between(range(Q), R, np.zeros(Q))
plt.plot(R)

plt.subplot2grid((2, 4), (1, 0), colspan=2)
plt.title('Входной сигнал')
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.grid(True)
plt.plot(X, Ysin)

outp = transform_sig(Ysin, Y)
plt.subplot2grid((2, 4), (1, 2), colspan=2)
plt.title('Выходной сигнал')
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.grid(True)
plt.plot(X, outp)
f = open('sin(x) - звено с чистым запаздываением.txt', 'w')
for item in outp:
    f.write("%f\n" % item)
f.close()

plt.tight_layout()
plt.savefig("h2.svg",type="svg",dpi=100)

# Апериодическое звено 1-го порядка
T=15
N = 500
X = (np.array(range(N)))
Y = np.zeros(N)
for i in range(N):
    if X[i] >= 0:
        Y[i] = np.exp(-X[i]/T)/T

plt.subplot2grid((2, 4), (0, 1), colspan=3)
plt.title('Апериодическое звено 1-го порядка')
plt.xlabel('t')
plt.ylabel('h(t)')
plt.grid(True)
plt.plot(X, Y)

plt.subplot2grid((2, 4), (0, 0))
# Корреляционная функция апериодического звена 1-го порядка
R, Q = Rxxwf(Y)
plt.xlabel('q')
plt.ylabel('Rxx')
```

```
    plt.grid(True)
    plt.fill_between(range(Q), R, np.zeros(Q))
    plt.plot(R)

    plt.subplot2grid((2, 4), (1, 0), colspan=2)
    plt.title('Входной сигнал')
    plt.xlabel('x')
    plt.ylabel('sin(x)')
    plt.grid(True)
    plt.plot(Xsin, Ysin)

    outp = transform_sig(Ysin, Y)
    plt.subplot2grid((2, 4), (1, 2), colspan=2)
    plt.title('Выходной сигнал')
    plt.xlabel('x')
    plt.ylabel('sin(x)')
    plt.grid(True)
    plt.plot(X, outp)
    f = open('sin(x) - апериодическое звено 1-го порядка.txt', 'w')
    for item in outp:
        f.write("%f\n" % item)
    f.close()

    plt.tight_layout()
    plt.savefig("h3.svg",type="svg",dpi=100)

    # Чёрный ящик
    N = 500
    X = np.array(range(N))
    Y = np.zeros(N)
    for i in range(N):
        if X[i] > 0 and X[i] < 100 * np.pi:
            Y[i] = np.sin(X[i]/100)/200

    plt.subplot2grid((2, 4), (0, 1), colspan=3)
    plt.title('Чёрный ящик')
    plt.xlabel('t')
    plt.ylabel('h(t)')
    plt.grid(True)
    plt.plot(X, Y)

    plt.subplot2grid((2, 4), (0, 0))
    # Корреляционная функция чёрного ящика
    R, Q = Rxxwf(Y)
    plt.xlabel('q')
    plt.ylabel('Rxx')
    plt.grid(True)
    plt.fill_between(range(Q), R, np.zeros(Q))
    plt.plot(R)

    plt.subplot2grid((2, 4), (1, 0), colspan=2)
    plt.title('Входной сигнал')
    plt.xlabel('x')
    plt.ylabel('sin(x)')
    plt.grid(True)
    plt.plot(Xsin, Ysin)

    outp = transform_sig(Ysin, Y)
    plt.subplot2grid((2, 4), (1, 2), colspan=2)
    plt.title('Выходной сигнал')
    plt.xlabel('x')
    plt.ylabel('sin(x)')
    plt.grid(True)
    plt.plot(X, outp)
    f = open('sin(x) - чёрный ящик.txt', 'w')
    for item in outp:
        f.write("%f\n" % item)
    f.close()

    plt.tight_layout()
    plt.savefig("h4.svg",type="svg",dpi=100)
```

In [2]:

```
def Rxx(X, D = 10):
    """
    Корреляционная функция
    """
    # Среднее значение X
    X_ = np.mean(X)
    N = len(X)
    Q = int(N/D)
    R = np.zeros(Q)
    for q in range(Q):
        for i in range(N - Q):
            R[q] = R[q] + (X[i] - X_) * (X[i + q] - X_)
    R = R / (N - Q)
    return R, Q

def Rxxwf(X, D = 5):
    """
    Корреляционная функция для весовых функций
    """
    N = len(X)
    Q = int(N/D)
    R = np.zeros(Q)
    for q in range(Q):
        for i in range(N - Q):
            R[q] = R[q] + X[i] * X[i + q]
    R = R / (N - Q)
    return R, Q

def Kxx(a):
    """
    Ковариационная функция
    """
    K = np.zeros(100)
    for q in range(100):
        K[q] = np.exp(-a * np.fabs(q))
    return K

def Kxxcn(X, t):
    """
    Ковариационная функция для окрашенного шума
    """
    X_ = np.mean(X)
    M = np.zeros(len(X)-t)
    for q in range(len(M)):
        M[q] = (X[q] - X_) * (X[q + t] - X_)
    return np.mean(M)
```

In [3]:

```
def mean_confidence_interval(data, confidence=0.95):
    """
    Вычисляет доверительный интервал для выборки data для доверительной вероятности confi
dence
    """
    n = len(data) #размер выборки
    m, se = np.mean(data), sp.stats.sem(data) #вычисляем среднее и стандартное отклонени
e
    h = se * sp.stats.t._ppf((1+confidence)/2., n-1) #вычисляем доверительный интервал
    return m, h

def mean_confidence_interval_colored_noise(data, confidence=0.95):
    """
    Вычисляет доверительный интервал для выборки data для доверительной вероятности confi
dence
    """
    n = len(data) #размер выборки
    m = np.mean(data)
    se = 0
    for p in range(n-1):
```

```
        se = se + (1 - (p + 1) / n) * np.exp(-(p+1)/50) * 0.05
    se = se * 2 / n + 1 / n
    h = se * sp.stats.t._ppf((1+confidence)/2., n-1) #вычисляем доверительный интервал
    return m, h
```

In [4]:

```
def covariance():
 T = 50
 k = 9.5

 plt.figure()
 # Ковариационные функции
 plt.title('Ковариационные функции сигналов')
 plt.xlabel('N')
 plt.ylabel('Kxx')
 plt.grid(True)

 a = -np.log(k*k/2/T)/T;
 K = Kxx(a);
 plt.plot(K, color='green', label = "Окрашенный шум")

 k = 9
 a = -np.log(k*k/2/T)/T
 K = Kxx(a)
 plt.plot(K, color='green')

 k = 8
 a = -np.log(k*k/2/T)/T
 K = Kxx(a)
 plt.plot(K, color='green')

 k = 7
 a = -np.log(k*k/2/T)/T
 K = Kxx(a)
 plt.plot(K, color='green')

 k = 5
 a = -np.log(k*k/2/T)/T
 K = Kxx(a)
 plt.plot(K, color='green')

 k = 1
 a = -np.log(k*k/2/T)/T
 K = Kxx(a)
 plt.plot(K, color='black', label = "Белый шум")
 plt.legend()

 plt.show()
```

In [5]:

```
def noises():
 n = 1250
 sq12 = np.sqrt(12)
 # Генерация белого шума
 X = (sum(np.random.rand(1000, n)) / 1000 - 0.5) * sq12

 T = 50
 k = np.sqrt(2 * T)
 Y = np.zeros(len(X)+1)
 # Генерация окрашенного шума
 for number in range(1249):
  Y[number+1] = (T-1)/T*Y[number]+k/T*X[number+1]

 X = X[:1000]
 Y = Y[-1000:]

 plt.subplot2grid((2, 4), (0, 0), colspan=3)
 # Белый шум
 plt.title('Белый шум')
```

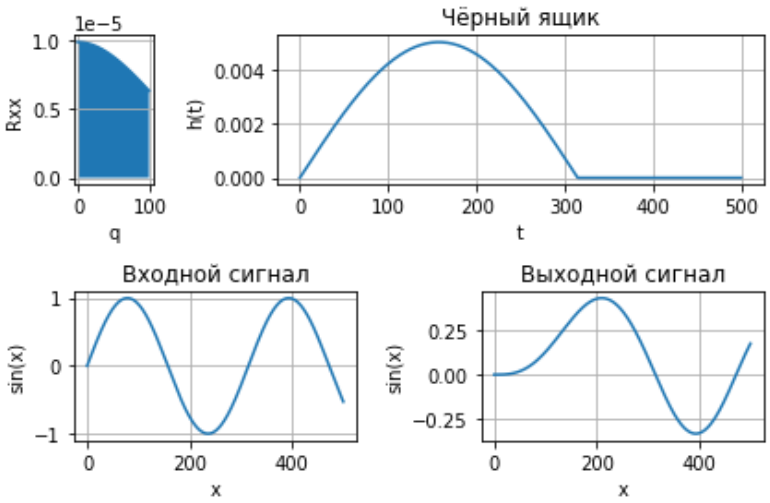
```
plt.xlabel('i')
plt.ylabel('x2[i]')
plt.grid(True)
plt.plot(X)
f = open('Белый шум.txt', 'w')
for item in X:
 f.write("%f\n" % item)
f.close()

plt.subplot2grid((2, 4), (1, 0), colspan=3)
# Окрашенный шум
plt.title('Окрашенный шум')
plt.xlabel('i')
plt.ylabel('x3[i]')
plt.grid(True)
plt.plot(Y)
f = open('Окрашенный шум.txt', 'w')
for item in Y:
 f.write("%f\n" % item)
f.close()

plt.subplot2grid((2, 4), (0, 3))
# Гистограмма для белого шума
plt.hist(X, 20, density=True, facecolor='g', alpha=0.75,  orientation=u'horizontal')
plt.axis('off')

plt.subplot2grid((2, 4), (1, 3))
# Гистограмма для окрашенного шума
plt.hist(Y, 20, density=True, facecolor='g', alpha=0.75,  orientation=u'horizontal')
plt.axis('off')
plt.tight_layout()
plt.show(block=False)

plt.figure()
plt.subplot2grid((2, 3), (0, 0))
# Корреляционная функция белого шума
R, Q = Rxx(X)
plt.title('Белый шум')
plt.xlabel('q')
plt.ylabel('Rxx')
plt.grid(True)
plt.fill_between(range(Q), R, np.zeros(Q))
plt.plot(R)

plt.subplot2grid((2, 3), (1, 0))
# Корреляционная функция окрашенного шума
R, Q = Rxx(Y)
plt.title('Окрашенный шум')
plt.xlabel('q')
plt.ylabel('Rxx')
plt.grid(True)
plt.fill_between(range(Q), R, np.zeros(Q))
plt.plot(R)

# Оценка среднего от числа отсчётов для белого шума
Mx1 = np.zeros(len(X))
Sx1 = np.zeros(len(X))

for i in range(len(X-5)):
 Mx1[i], Sx1[i] = mean_confidence_interval(X[:i+6])

plt.subplot2grid((2, 3), (0, 1), colspan=2)
plt.title('Доверительные интервалы для среднего значения')
plt.xlabel('N')
plt.ylabel('M[x]')
plt.grid(True)
plt.plot(Mx1)
plt.plot(Sx1, color='red')
plt.plot(-Sx1,  color='red')
f = open('Белый шум средние значения.txt', 'w')
for item in Mx1:
 f.write("%f\n" % item)
```

```
 f.close()
 f = open('Белый шум доверительные интервалы.txt', 'w')
 for item in Sx1:
  f.write("%f\n" % item)
 f.close()

 # Оценка среднего от числа отсчётов для окрашенного шума
 Mx2 = np.zeros(len(Y))
 Sx2 = np.zeros(len(Y))

 for i in range(len(Y-5)):
  Mx2[i], Sx2[i] = mean_confidence_interval_colored_noise(Y[:i+6])

 plt.subplot2grid((2, 3), (1, 1), colspan=2)
 plt.xlabel('N')
 plt.ylabel('M[x]')
 plt.grid(True)
 plt.plot(Mx2)
 plt.plot(Sx2, color='red')
 plt.plot(-Sx2, color='red')
 f = open('Окрашенный шум средние значения.txt', 'w')
 for item in Mx2:
  f.write("%f\n" % item)
 f.close()
 f = open('Окрашенный шум доверительные интервалы.txt', 'w')
 for item in Sx2:
  f.write("%f\n" % item)
 f.close()

 plt.tight_layout()
 plt.show(block=False)
```

In [6]:

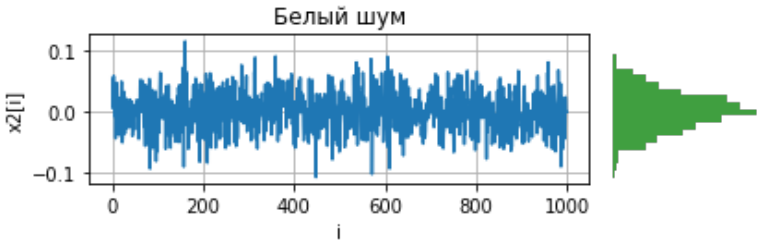
```
weight_functions()
```

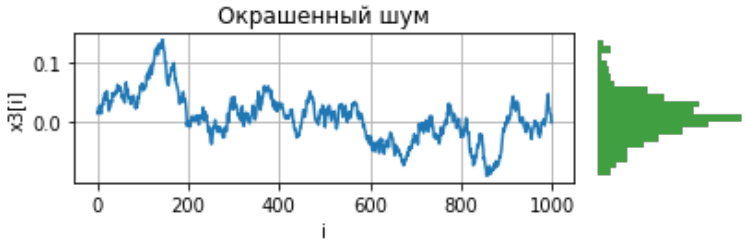
```
<ipython-input-1-3e276bfbab5c>:74: MatplotlibDeprecationWarning: savefig() got unexpected
keyword argument "type" which is no longer supported as of 3.3 and will become an error t
wo minor releases later
  plt.savefig("h1.svg",type="svg",dpi=100)
<ipython-input-1-3e276bfbab5c>:118: MatplotlibDeprecationWarning: savefig() got unexpecte
d keyword argument "type" which is no longer supported as of 3.3 and will become an error
two minor releases later
  plt.savefig("h2.svg",type="svg",dpi=100)
<ipython-input-1-3e276bfbab5c>:165: MatplotlibDeprecationWarning: savefig() got unexpecte
d keyword argument "type" which is no longer supported as of 3.3 and will become an error
two minor releases later
  plt.savefig("h3.svg",type="svg",dpi=100)
<ipython-input-1-3e276bfbab5c>:211: MatplotlibDeprecationWarning: savefig() got unexpecte
d keyword argument "type" which is no longer supported as of 3.3 and will become an error
two minor releases later
  plt.savefig("h4.svg",type="svg",dpi=100)
```

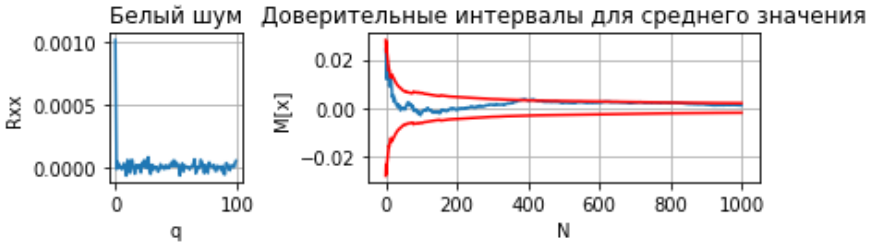


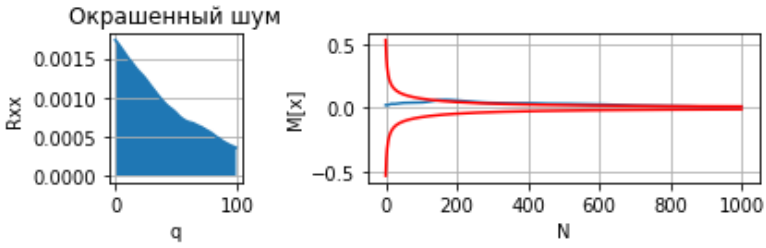
In [7]:

```
noises()
```



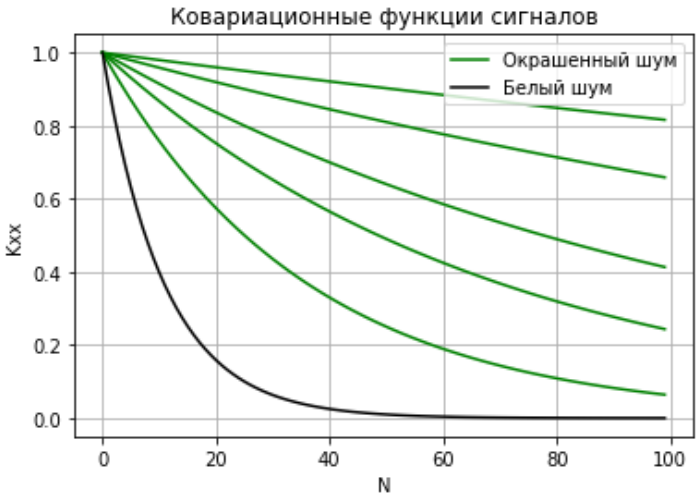






In [8]:

```
covariance()
```



In [ ]:

```
```