МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ФН
КАФЕДРА «ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И МАТЕМАТИЧЕСКАЯ ФИЗИКА»

Дисциплина: Теория случайных процессов

Отчет по выполнению домашнего задания №2.1
”Моделирование гауссовского процесса с данной автоковариационной функцией”

Группа: ФН11-62Б
Вариант 6

Студент: Ладыгина Л.В.
Преподаватель: Облакова Т.В

Москва, 2022

Задание

Моделирование гауссовского процесса с данной автоковариационной функцией

На отрезке $[0, T]$ с шагом h смоделируйте n траекторий гауссовского процесса ξ_t с заданным математическим ожиданием $m(t)$ и заданной автоковариационной функцией $K(t_1, t_2)$.

Выведите на печать две-три траектории вместе с графиком $m(t)$.

Выберите несколько пар сечений построенного процесса (для далеких значений t_1 и t_2, для близких, для соседних).

Для выбранных сечений ξ_{t_i} постройте гистограммы относительных частот, совмещенные с теоретической плотностью распределения СВ ξ_{t_i}.

Постройте для выбранных пар сечений диаграммы рассеяния, вычислите выборочные коэффициенты корреляции, постройте 95% доверительные интервалы (см. материал прошлого семестра). Сравните выборочные и теоретические значения коэффициентов корреляции.

Сформулируйте общие выводы.

6	$[0,6]$	0.05	150	$m(t) = 1 + e^{-t}$	$K(\tau) = \begin{cases} 2 - \lvert\tau\rvert, & \lvert\tau\rvert < 2 \\ 0, & \lvert\tau\rvert \geq 2 \end{cases}$

1. Функция математического ожидания и автоковариационная функция:

```
def m(t):
  return 1.0+math.exp(-t)

def K(t):
  if abs(t)<2:
    return 2-abs(t)
  else:
    return 0
```

Вектор математических ожиданий $y = m(t_i) = M_{ih}$:

```
h=0.05
n=150
a=0
b=6
N=(b-a)/h
#N=121
```

```
for i in range(N):
  x0=i*h
  x.append(x0)
  y.append(m(x0))
```

Вектор M_{ih}:

```
[2.0, 1.9512, 1.9048, 1.8607, 1.8187, 1.7788, 1.7408, 1.7047, 1.6703, 1.6376,
```

2. Ковариационная матрица $\Sigma(N * N)$:

```
sigma=[[K(h*(i-j)) for j in range(N)] for i in range(N)]
```

```
[2.0, 1.9512, 1.9048, 1.8607, 1.8187, 1.7788, 1.7408, 1.7047, 1.6703,
1.6376, 1.6065, 1.5769, 1.5488, 1.522, 1.4966, 1.4724, 1.4493, 1.4274,
1.4066, 1.3867, 1.3679, 1.3499, 1.3329, 1.3166, 1.3012, 1.2865, 1.2725,
1.2592, 1.2466, 1.2346, 1.2231, 1.2122, 1.2019, 1.192, 1.1827, 1.1738,
1.1653, 1.1572, 1.1496, 1.1423, 1.1353, 1.1287, 1.1225, 1.1165, 1.1108,
1.1054, 1.1003, 1.0954, 1.0907, 1.0863, 1.0821, 1.0781, 1.0743, 1.0707,
1.0672, 1.0639, 1.0608, 1.0578, 1.055, 1.0523, 1.0498, 1.0474, 1.045,
1.0429, 1.0408, 1.0388, 1.0369, 1.0351, 1.0334, 1.0317, 1.0302, 1.0287,
1.0273, 1.026, 1.0247, 1.0235, 1.0224, 1.0213, 1.0202, 1.0193, 1.0183,
1.0174, 1.0166, 1.0158, 1.015, 1.0143, 1.0136, 1.0129, 1.0123, 1.0117,
1.0111, 1.0106, 1.0101, 1.0096, 1.0091, 1.0087, 1.0082, 1.0078, 1.0074,
1.0071, 1.0067, 1.0064, 1.0061, 1.0058, 1.0055, 1.0052, 1.005, 1.0047,
1.0045, 1.0043, 1.0041, 1.0039, 1.0037, 1.0035, 1.0033, 1.0032, 1.003,
1.0029, 1.0027, 1.0026, 1.0025]
[2.0  1.95  1.9  1.85  1.8  1.75  1.7  1.65  1.6  1.55  1.5  1.45  1.4  1.35
1.2999999999999998  1.25  1.2  1.15  1.1  1.0499999999999998  1.0  0.95
0.8999999999999999  0.8499999999999999  0.7999999999999998  0.75  0.7
0.6499999999999999  0.5999999999999999  0.5499999999999998  0.5
0.44999999999999996  0.3999999999999999  0.34999999999999987
0.2999999999999998  0.25  0.19999999999999996  0.1499999999999999
0.09999999999999987 0.04999999999999982 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
```

```
[1.95 2.0 1.95 1.9 1.85 1.8 1.75 1.7 1.65 1.6 1.55 1.5 1.45 1.4 1.35
1.2999999999999998 1.25 1.2 1.15 1.1 1.0499999999999998 1.0 0.95
0.8999999999999999 0.8499999999999999 0.7999999999999998 0.75 0.7
0.6499999999999999 0.5999999999999999 0.5499999999999998 0.5
0.44999999999999996 0.3999999999999999 0.34999999999999987
0.2999999999999998 0.25 0.19999999999999996 0.1499999999999999
0.09999999999999987 0.04999999999999982 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
```

…

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0.04999999999999982 0.09999999999999987
0.1499999999999999 0.19999999999999996 0.25 0.2999999999999998
0.34999999999999987 0.3999999999999999 0.44999999999999996 0.5
0.5499999999999998 0.5999999999999999 0.6499999999999999 0.7 0.75
0.7999999999999998 0.8499999999999999 0.8999999999999999 0.95 1.0
1.0499999999999998 1.1 1.15 1.2 1.25 1.2999999999999998 1.35 1.4 1.45
1.5 1.55 1.6 1.65 1.7 1.75 1.8 1.85 1.9 1.95 2.0 1.95 ]
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0.04999999999999982 0.09999999999999987
0.1499999999999999 0.19999999999999996 0.25 0.2999999999999998
0.34999999999999987 0.3999999999999999 0.44999999999999996 0.5
0.5499999999999998 0.5999999999999999 0.6499999999999999 0.7 0.75
0.7999999999999998 0.8499999999999999 0.8999999999999999 0.95 1.0
1.0499999999999998 1.1 1.15 1.2 1.25 1.2999999999999998 1.35 1.4 1.45
1.5 1.55 1.6 1.65 1.7 1.75 1.8 1.85 1.9 1.95 2.0]
```

с помощью встроенных функций Python находим корень Холецкого:

```
L = np.linalg.cholesky(sigma)
```

Матрица L:

```
[[1.41421356 0.         0.         ... 0.         0.         0.        ]
 [1.37885822 0.31424513 0.         ... 0.         0.         0.        ]
 [1.34350288 0.31026734 0.31421995 ... 0.         0.         0.        ]
 ...
 [0.         0.         0.         ... 0.25714716 0.         0.        ]
 [0.         0.         0.         ... 0.2550394  0.25713852 0.        ]
 [0.         0.         0.         ... 0.25293163 0.25501341 0.25712974]]
```

3. Генерируем вектор n случайных стандартных гауссовских векторов ε:

```
for j in range(n):
  epsilon=[]
  M=[]
  for i in range(N):
    e=sps.norm(loc=0, scale=1).rvs()
    epsilon.append(e)
```

С помощью встроенных функций находим вектор $\eta = L\varepsilon$:

```
eta=np.dot(L,epsilon)
```

Находим вектор $\xi_{ih} = \eta_i + M_{ih}$:

```
for i in range(N):
  M.append(m(i*h))
ksi=eta+M
Ksi.append(ksi)
```

вектор ξ_{ih}:

траектория ksi1:

```
[  2.28838739    2.16032352    2.47399357    2.29883507    2.05844345
1.629037581.54689109   2.19336509   2.5937307    2.10031589   2.0347652
1.453939831.51508701   1.41905542   1.28190165   0.99886936   1.22954732
1.165870281.17555918   1.207106     1.20955847   1.3091439    1.48755875
2.099131772.29640998   2.99066524   2.56804107   3.36155712   2.89911326
3.045291271.97216218   2.27723923   2.54555067   3.00428889   2.79275152
2.5754522  2.49749765   2.62816516   2.66951492   3.17201206   3.30865402
3.36161546  2.74978889   2.55562684   2.59950651   2.79952812   2.66328794
2.65745268  2.41090266   2.21924087   2.08475183   2.58298637   2.48851228
2.27308689  1.95650761   1.81321519   1.11340606   0.99274423   1.05462322
0.78724827  1.11354655   0.95770339   0.58891121   0.42880328   0.23520616
-0.24698577 -0.02938684 -0.59844396 -0.27919869 -0.17369773   0.01223981
-0.12509178 -0.69240435 -0.57326522 -0.31367881   0.40691563   0.34994302
0.14750997  0.68267652   0.31855098   0.1957092    0.16006182   0.1809464
0.65595874  0.74625019   0.86602126   0.98709362   0.78910658   0.91194579
1.14674289  1.23108456   0.61757523   0.90723905   0.89044926   0.99078965
```

```
1.29606983 1.5452946    1.4472585    1.24553038   1.16483243   1.39404702
1.1755908  1.10850716   1.32204419   1.12680807   1.55231534   1.56972795
1.62440529 1.54459541   1.09837912   1.14719497   1.11816313   1.13371214
0.97488776 0.84077897   0.27478581   0.53418818   0.83742948   0.57965329
0.43292694 0.74463119]
```

траектория ksi2:

```
  [  2.27925978    2.260433       2.47005531    2.39248196    2.68300206
2.73048198 2.6537449    2.7565186    2.27915727   2.0751046    2.5279258
2.21518534 2.524814     2.56777462   2.61087944   2.35005077   2.38432363
2.42674384   2.62275678   2.17628384   2.31846362   2.07840156   2.34569327
2.67365475 2.94850295   3.17602693   2.99705559   3.255559     3.23907582
3.36906729 3.39085578   3.64934876   3.27413337   3.46501671   3.48814286
3.44534696 3.35066863   2.93947227   2.92626458   2.98591601   3.49184591
3.88986043 3.55690361   3.57156554   3.26623575   3.47344172   3.38536284
3.36264323 3.10039806   3.07865895   3.18081734   3.30957903   2.91521451
2.89214461 2.67956379   3.0440964    2.53201718   2.55994477   2.92218246
3.27926079 3.39787069   3.68809978   3.46155692   3.45332856   3.11330872
3.01870578 2.95568651   2.73255051   2.91849613   3.00572981   2.94996347
2.97701117 3.24242783   2.98937797   2.51179322   2.4509987    2.41006996
2.476146  2.94778272   3.0572079    2.876398     2.02906595   2.37423204
2.40567812 2.51917027   2.55972557   2.15894741   2.27206741   2.47957871
2.46532168 2.2250152    2.3355914    2.38704061   2.3629412    2.32247362
1.82657499 1.75503276   1.80324472   1.46258299   1.3252845    1.17504419
0.8763877  0.95772762   0.84181735   0.87638271   0.48680626   0.29976867
0.20081963  0.18139147   0.63907809   0.14483453 -0.07041513 -0.17461431
0.27657108 0.19592792   0.1456472   -0.02122441   0.14448142 -0.37534989
-0.58010143 -0.9400103 ]
```

траектория ksi3:

```
  [  3.04551754    3.52810501    3.32206592    3.48392984    3.863215
3.70598341 3.16948477   3.07651407   2.51372347   2.2351845    1.86295497
1.82615786 1.99835299   1.75344048   1.96561978   1.65825995   1.27222774
1.04026809 1.12523643   1.16444481   0.94036436   0.58558989   0.51446592
0.57965155 0.79009847   0.63320259   0.82450448   0.72249103   0.67316824
0.52252515  0.40001254   0.20271579   0.45143187   0.29247189 -0.3138654
0.07604756 0.38099807   0.45350042   0.32728209   0.26070064 -0.14407758
-0.90500504 -0.38656652 -0.25932155 -0.41886627   0.29008     0.08461008
-0.31768894 -0.28901761 -0.09862706 -0.02919098 -0.06595229 -0.34432819
-0.18279525 -0.33873366 -0.47785329 -0.64137714 -0.21660816 -0.13353594
-0.48002243 -0.04001135 -0.35728959 -0.35247515 -0.10817194  0.07649949
0.68685414 0.68507735   0.79784882   0.16037027   0.45748585   0.53246724
0.7999095  0.85532614   1.17869901   1.68827141   1.48773236   1.43225391
1.34762084 1.0874623    0.89882646   1.04869875   1.31093063   1.48815737
1.21704295 0.99753391   0.63745778   0.62470142   1.3444289    1.57616317
1.83599473 2.1467554    2.31579225   2.58308047   1.96538242   1.50045691
1.47230046 1.33612731   1.33096156   1.12727668   1.34791221   1.20516573
1.40908076 1.50557981   1.7286048    1.32805696   1.11856488   0.82268104
```

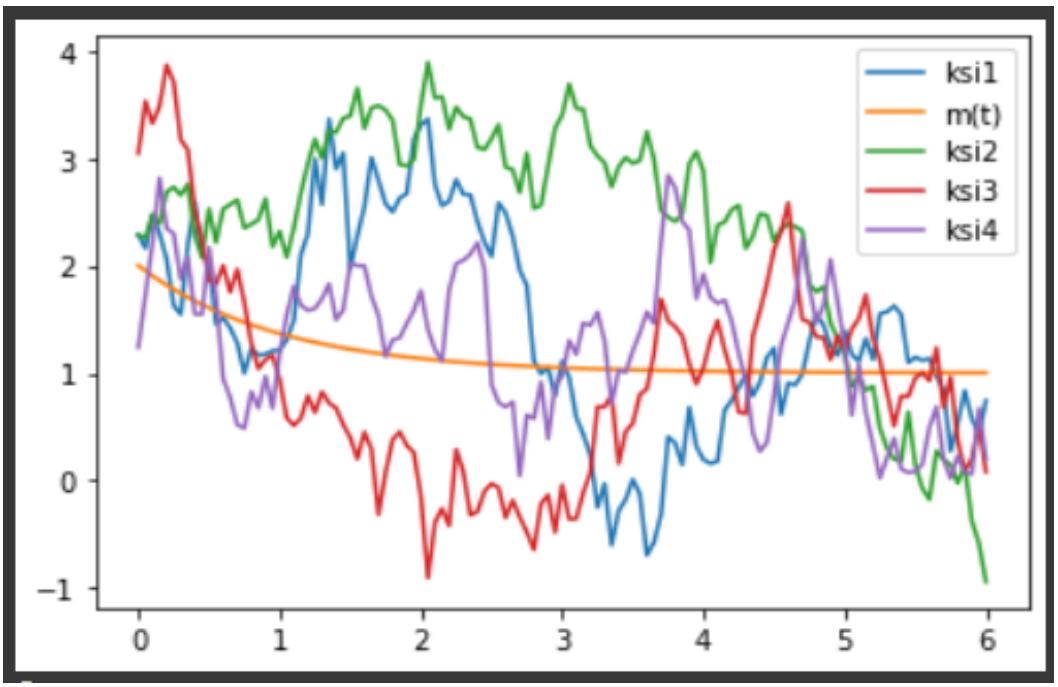
```
0.51048396  0.78638684   0.78234464   0.9558435    1.00341015   0.92994237
1.23460748  0.67968166   0.96136589   0.37545569   0.08685694   0.20604259
0.51748483 0.08150512]
```

траектория ksi4:

```
 [1.23979886 1.72812859 2.21246299 2.81333824 2.34550736 2.29716239
 1.87343481 2.07693463 1.54855586 1.54562362 2.1661523  1.7130828
 0.94829941 0.7729385  0.52167863 0.48564061 0.82719845 0.67546748
 0.96956805 0.67214506 1.18461058 1.53849101 1.80587127 1.6334523
 1.5820235  1.60146228 1.69219992 1.82937032 1.49659017 1.58670774
 2.02095471 1.99929996 1.997019   1.71495211 1.53461293 1.15515902
 1.30875194 1.32723989 1.46227734 1.58341975 1.76655947 1.40486189
 1.21740619 1.11268138 1.77317457 2.0185741  2.04904067 2.10484779
 2.21194237 1.96817491 0.89405893 0.72596619 0.684941   0.73182978
 0.04513155 0.60918157 0.57738245 0.91725581 0.3911458  0.89408163
 0.9689552  1.30232925 1.17449859 1.466311   1.44472461 1.56582608
 1.30226159 0.71499034 1.02607218 1.00832009 1.20705573 1.37363368
 1.5685561  1.46136936 2.22264329 2.83825299 2.72275113 2.41067679
 2.32895439 1.69320722 1.91926909 1.70536173 1.64909104 1.67965368
 1.48277057 1.18397518 0.92752469 0.44295618 0.27202289 0.3561741
 0.79536848 1.25777174 1.47087786 1.68026699 2.25077969 1.7936068
 1.47594527 1.62850467 2.05520243 1.66199352 1.25550264 0.6083492
 1.08357442 0.6298107  0.33298179 0.02432538 0.21442707 0.39339649
 0.10849679 0.07451995 0.08582911 0.15216386 0.49603872 0.68723751
 0.20003594 0.02264378 0.23118304 0.07872797 0.0617592  0.67110356
 0.19739225]
```

…

Построенные траектории вместе с функцией математического ожидания:



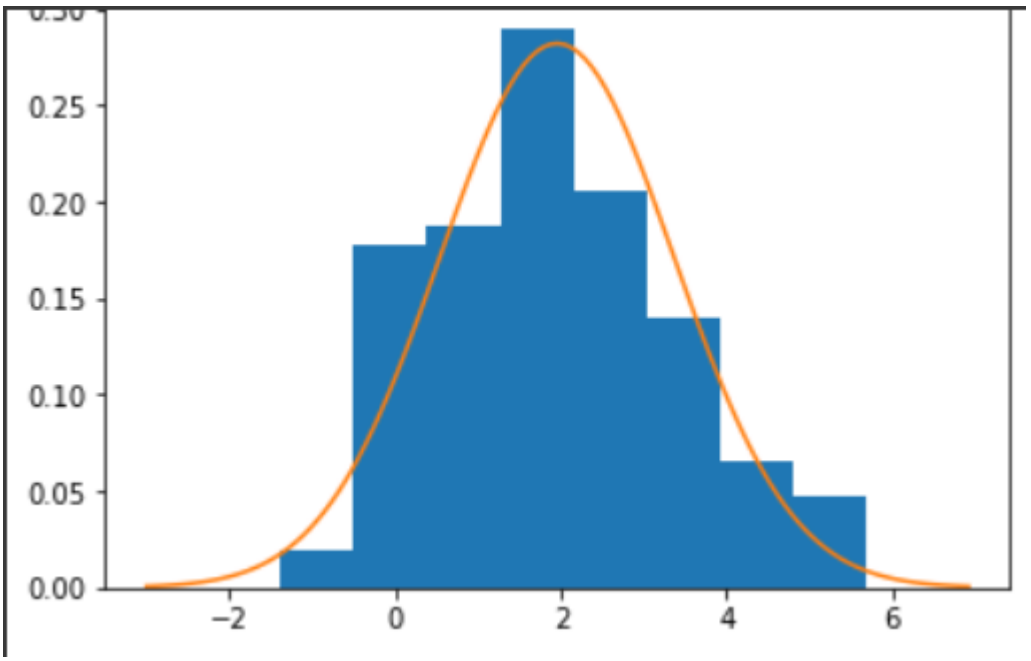
Возьмём три сечения: $(t_1 = 0.05\,,\ t_2 = 0.1), (t_1 = 0.05,\ t_2 = 0.4), (t_1 = 0.05,\ t_2 = 2.5)$

Для этого запоминаем все состояния в моменты $t = h,\ t = 2h,\ t = 8h,\ t = 50h$:

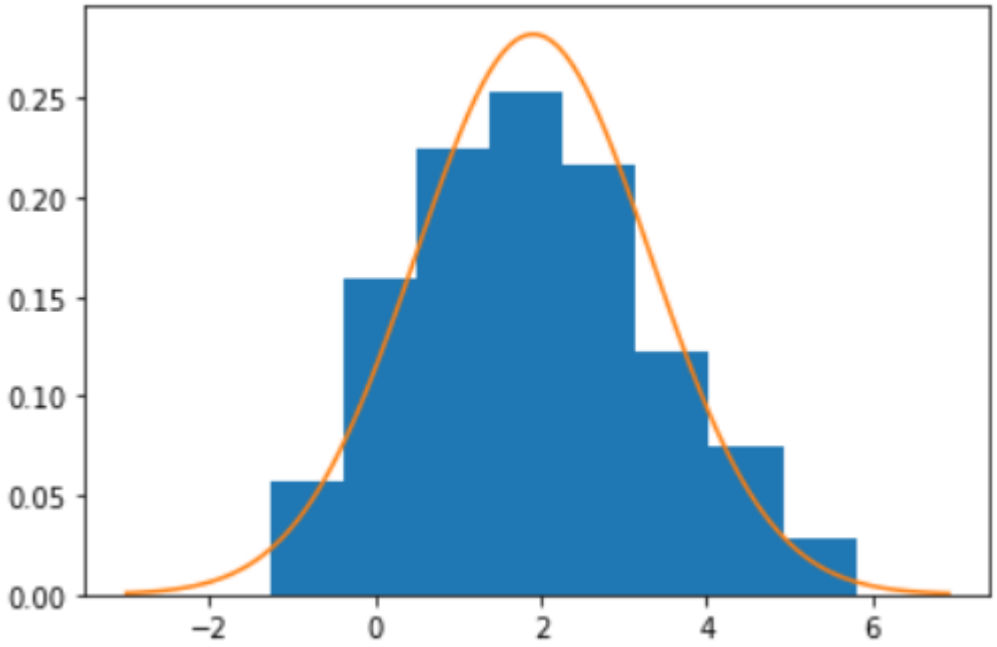
```
T_ksi1=[]
T_ksi2=[]
T_ksi8=[]
T_ksi50=[]
for i in range(N):
  for j in range(N):
  #print(Ksi,"\n")
    if j==1 :
      T_ksi1.append(Ksi[i][j])
    if j==2 :
      T_ksi2.append(Ksi[i][j])
    if j==8 :
      T_ksi8.append(Ksi[i][j])
    if j==50 :
      T_ksi50.append(Ksi[i][j])
```

Найденные сечения:

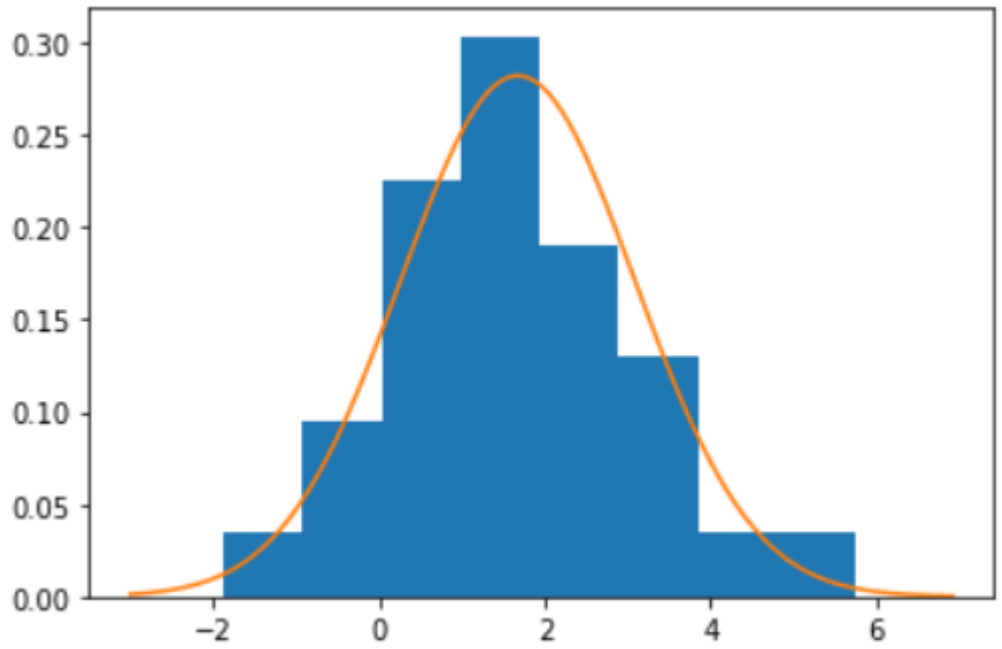
	T1	T2	T8	T50
0	3.634290	3.204326	3.508347	1.922787
1	2.160324	2.473994	2.593731	2.084752
2	2.260433	2.470055	2.279157	3.180817
3	3.528105	3.322066	2.513723	-0.029191
4	1.728129	2.212463	1.548556	0.894059
..	...	...	...	...
116	1.363009	1.594339	1.378801	0.404912
117	2.592536	2.780391	1.613023	1.248494
118	4.419468	4.151058	4.172846	2.286180
119	5.446438	5.794336	3.415104	0.299405
120	5.124690	5.254702	5.746193	-0.075022



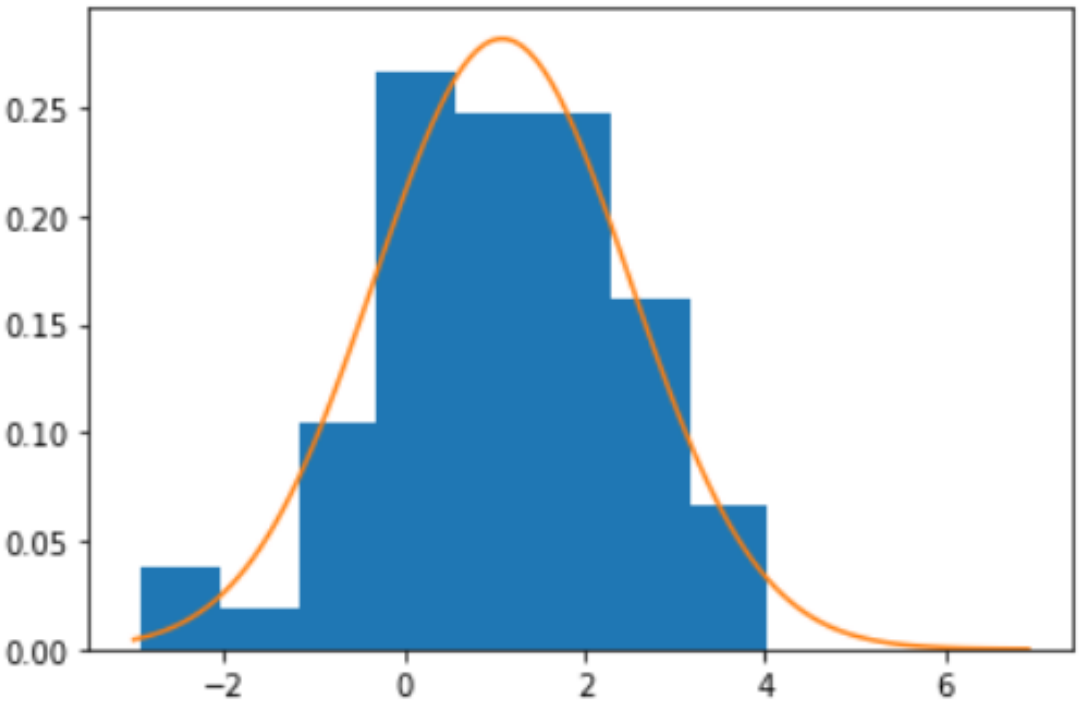
Гистограмма относительных частот и теоретическая плотность распределения для t= 0.05



Гистограмма относительных частот и теоретическая плотность распределения для t= 0.1



Гистограмма относительных частот и теоретическая плотность распределения для t= 0.4



Гистограмма относительных частот и теоретическая плотность распределения для t= 2.5

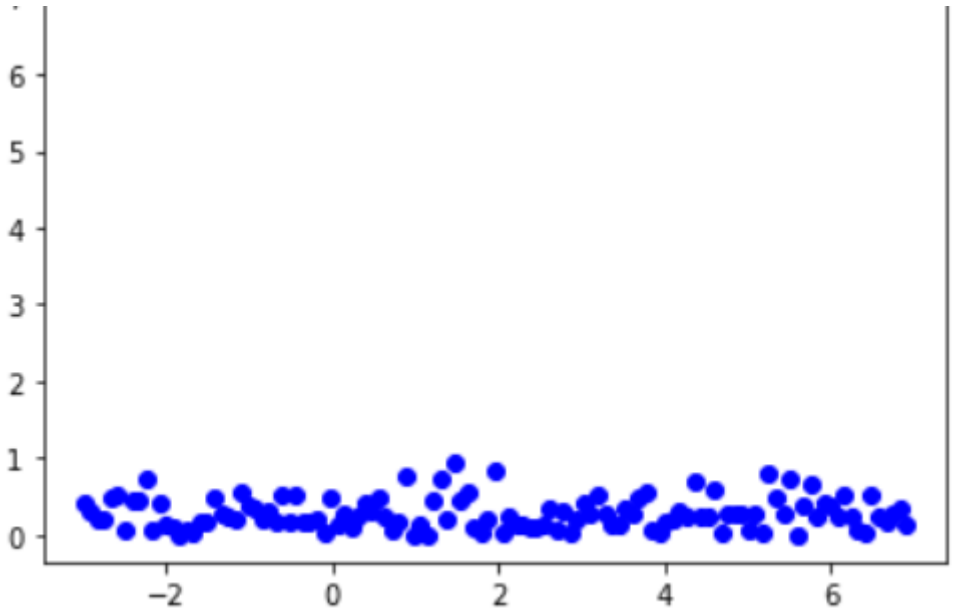


Диаграмма рассеяния для соседних t

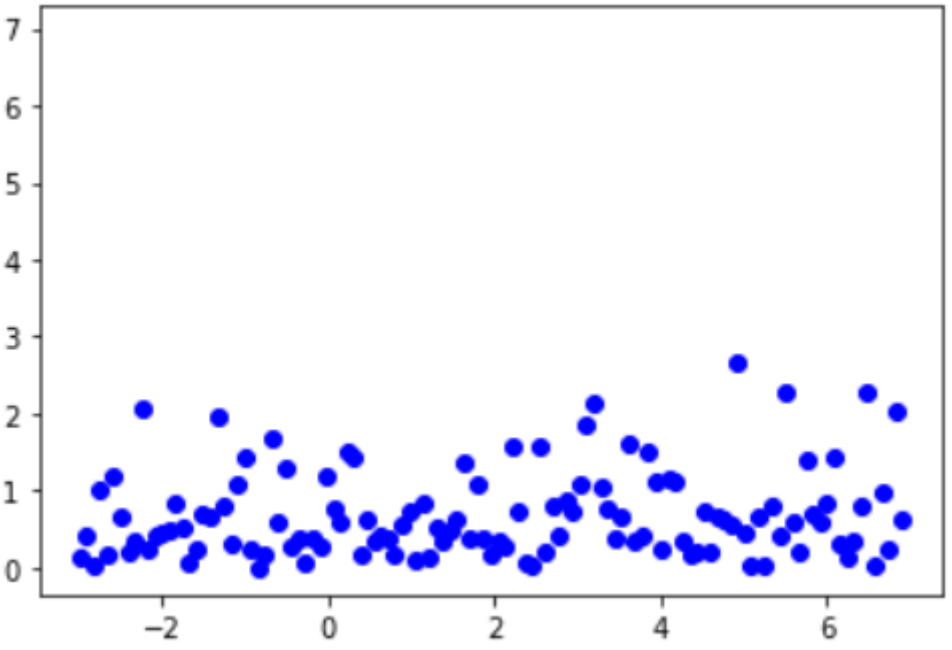


Диаграмма рассеяния для близких t

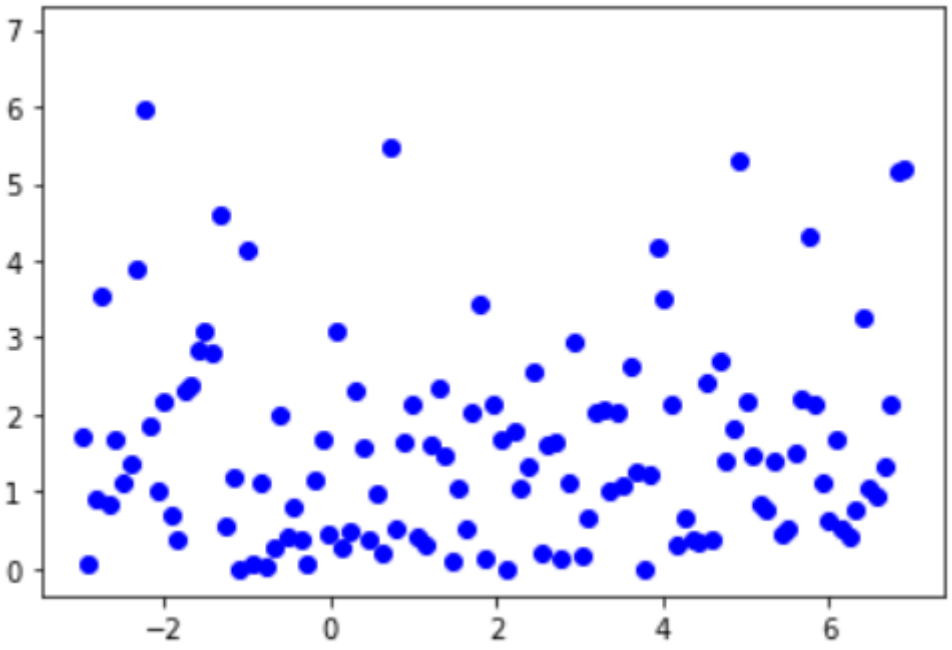


Диаграмма рассеяния для далеких t

3 Вычислим выборочные коэффициенты корреляции и доверительные интервалы для каждого сечения при α = 0,05:

формула вычисления выборочных коэффициентов корреляции:

$$\widehat{r_{\xi_t \xi_\tau}} = \frac{\sum_i (\xi_t - \overline{\xi_t})(\xi_\tau - \overline{\xi_\tau})}{\sqrt{D_{\xi_t}}\sqrt{D_{\xi_\tau}}}$$

Средние значения $\overline{\xi_{t1}}$, $\overline{\xi_{t2}}$, $\overline{\xi_{t8}}$, $\overline{\xi_{t50}}$:

```
avg_ksi1=mean(T_ksi1)
avg_ksi2=mean(T_ksi2)
avg_ksi8=mean(T_ksi8)
avg_ksi50=mean(T_ksi50)
```

Выборочные дисперсии и ковариации $St\_\tau = \frac{\sum_i (\xi_t - \overline{\xi_t})(\xi_\tau - \overline{\xi_\tau})}{N}$

```
D1=0
D2=0
D8=0
D50=0
D1_2=0
D1_8=0
D1_50=0



for i in range(N):
  D1=D1+(T_ksi1[i]-avg_ksi1)**2
  D2=D2+(T_ksi2[i]-avg_ksi2)**2
  D8=D8+(T_ksi8[i]-avg_ksi8)**2
  D50=D50+(T_ksi50[i]-avg_ksi50)**2

  S1_2=S1_2+(T_ksi1[i]-avg_ksi1)*(T_ksi2[i]-avg_ksi2)
  S1_8=S1_8+(T_ksi1[i]-avg_ksi1)*(T_ksi8[i]-avg_ksi8)
  S1_50=S1_50+(T_ksi1[i]-avg_ksi1)*(T_ksi50[i]-avg_ksi50)

D1=float(D1)/N
D2=float(D2)/N
D8=float(D8)/N
D50=float(D50)/N
S1_2=float(S1_2)/N
S1_8=float(S1_8)/N
S1_50=float(S1_50)/N
```

Выборочные коэффициенты корреляции:

```
r1_2=float(S1_2)/math.sqrt(D1*D2)
r1_8=float(S1_8)/math.sqrt(D1*D8)
r1_50=float(S1_50)/math.sqrt(D1*D50)
```

Теоретические коэффициенты корреляции:

$\sigma = K(0) = \sqrt{2}$

```
r1_2t=round(float(K(h*(2-1 ))/sigma**2),4)
r1_8t=round(float(K(h*(8-1 ))/sigma**2),4)
r1_50t=round(float(K(h*(50-1 ))/sigma**2),4)
```

Выборочный коэффициент корреляции	Теоретический коэффициент корреляции	Погрешность
r1_2=0.971	r1_2t=0.975	0.004
r1_8=0.8289	r1_8t=0.825	0.0039
r1_50=0.1152	r1_50t=0.0	0.1152

Доверительные интервалы для выборочных коэффициентов корреляции:

Используем формулу:

$$\text{th}\left(Arth\, r_в - \frac{r_в}{2(n-1)} - \frac{u_{1-\frac{\alpha}{2}}}{\sqrt{n-3}}\right) < r < \text{th}\left(Arth\, r_в - \frac{r_в}{2(n-1)} + \frac{u_{1-\frac{\alpha}{2}}}{\sqrt{n-3}}\right)$$

используя встроенные функции находим обратные функции нормального распределения и верхние и нижние границы интервалов:

```
alpha=0.05
r12_low=math.tanh(math.atanh(r1_2)-float(r1_2)/(2*(n-1))-sps.norm.ppf(1-float(alpha)/2)/math.sqrt(n-3))
r12_up=math.tanh(math.atanh(r1_2)-float(r1_2)/(2*(n-1))+sps.norm.ppf(1-float(alpha)/2)/math.sqrt(n-3))
print(r12_low,"<r1_2<",r12_up)


r18_low=math.tanh(math.atanh(r1_8)-float(r1_8)/(2*(n-1))-sps.norm.ppf(1-float(alpha)/2)/math.sqrt(n-3))
r18_up=math.tanh(math.atanh(r1_8)-float(r1_8)/(2*(n-1))+sps.norm.ppf(1-float(alpha)/2)/math.sqrt(n-3))
print(r18_low,"<r1_8<",r18_up)


r150_low=math.tanh(math.atanh(r1_50)-float(r1_50)/(2*(n-1))-sps.norm.ppf(1-float(alpha)/2)/math.sqrt(n-3))
r150_up=math.tanh(math.atanh(r1_50)-float(r1_50)/(2*(n-1))+sps.norm.ppf(1-float(alpha)/2)/math.sqrt(n-3))
print(r150_low,"<r1_50<",r150_up)
```

Полученные доверительные интервалы:

```
0.9598627593041124 <r1_2< 0.9787727326700663
0.7699049838695395 <r1_8< 0.8724875697374868
-0.0463085967962962 <r1_50< 0.270097013556242
```

Выводы:

В результате выполнения домашнего задания были построены траектории гауссовского процесса и изучены далёкие, близкие и соседние сечения смоделированного гауссовского процесса. Видно, что коэффициент корреляции

уменьшается с увеличением $t2 - t1$. Показано, что выборочные и теоретические коэффициенты корреляции близки при малых $t2 - t1$ и разница между ними возрастает с увеличением $t2 - t1$. Также с возрастанием разности между сечениями увеличивается ширина доверительного интервала коэффициента корреляции.