The 0-1 Test

Gottwald & Melbourne (2016) applied to the Rössler System

Ava Bauer

July 2, 2024

1 Introduction

A large part of climate science is concerned with estimating and predicting (parts of) the state of our planet based on a limited set of data, a complicated endeavour involving billions of parameters. What further complicates matters is that often the models we use to describe anything from weather and atmosphere to turbulence are inherently unpredictable. From the 1960s onwards, the many scientists realised that chaos erupts in many areas of climate studies, and many simple examples of such systems were studied, such as the famous Lorenz 63 model [Phy24]. These chaotic systems are characterized by their sensitivity to initial conditions, though we give no formal definition here.

While many of these systems are deterministic, the chaos in their dynamics implies that other tools might be needed to predict these systems over time, or that they are simply unpredictable. In analysing our models, it is therefore important to distinguish between regular and chaotic dynamics. A commonly used method was to compute the largest Lyapunov exponent of a system, which can be computed from its equations or by making estimates based on the data. In most cases, this requires expensive phase-space reconstructions. In 2004, George Gottwald and Ian Melbourne developed a new test, the 0-1 test, which only makes use of the time series of the data and outputs a 0 or a 1, indicating regular or regular dynamics respectively [GM04]. Though the test has its limitations and sensitivities, it has a wide range of applications both inside and outside of mathematics. In this report, we shortly describe the method developed by Gottwald and Melbourne as described in their reviewed [GM16], and apply it to the Rössler system. We shortly encounter one of the subtilities in the test, which lies in the resampling frequency of a continuous time series.

2 Method

(i) Description of the algorithm

As the input of the 0-1 test, we consider the time series $\phi(n)$, $n = 1, 2, \ldots, N$ (often an observable from the system we are considering), from which we compose the 2-dimensional system

$$\begin{aligned} p(n+1) &= p(n) + \phi(n)\cos(cn) \\ q(n+1) &= q(n) + \phi(n)\sin(cn), \end{aligned} \tag{1}$$

where $c \in (0, \pi)$ is a fixed parameter. Solving this system results in the series

$$p_c(n) = \sum_{j=1}^{n} \phi(j) \cos(jc), \quad q_c(n) = \sum_{j=1}^{n} \phi(j) \sin(jc), \tag{2}$$

We then define the time-averaged mean square displacement by

$$M_c(n) = \lim_{N\to\infty} \sum_{j=1}^{N} [p_c(j+n) - p_c(j)]^2 + [q_c(j+n) - q_c(j)]^2 \tag{3}$$

and the growth rate of the sequence M_c as

$$K_c = \lim_{n\to\infty} \frac{\log M_c(n)}{\log n}. \tag{4}$$

To analyse K_c numerically as asymptotic quality, we consider $n \leq N_0$ where $N_0 \ll N$. Here, we take $N_0 \approx \frac{N}{10}$.

The value of M_c oscillates as n grows, which is why in some analysis, it can be useful to instead estimate K_c from the modified mean square displacement

$$D_c(n) = M_c(n) - V_{\text{osc}}(c, n), \tag{5}$$

which has the same growth rate as M_c. The correction term is given by

$$V_{\text{osc}}(c, n) = (E\phi)^2 \frac{1 - \cos(nc)}{1 - \cos(c)},$$

where $E\phi$ can be computed by taking the mean of the time series.

The next step in the method is to estimate K_c, which can be done through the regression or correlation method. In this report, we use the correlation method, which measures the strength of $M_c(n)$ with linear growth.

To compute the correlation coefficient, we write

$$\xi = (1, 2, \ldots, N_0), \quad \Delta = (M_c(1), M_c(2), \ldots, M_c(N_0))$$

and define the correlation coefficient as

$$K_c = \text{corr}(\xi, \Delta) = \frac{\text{cov}(\xi, \Delta)}{\sqrt{\text{var}(\xi)\text{var}(\Delta)}} \in [-1, 1]. \tag{6}$$

After computing the value of K_c for various choices of c, we compute K as the median of K_c. In case $K \approx 0$, the test indicates regular dynamics, while if $K \approx 1$, we have chaotic dynamics.

(ii) Mathematical justification

The idea of the test is to distinguish the trajectories of the system 1. If the system has (quasi)-periodic dynamics, the trajectories are typically bounded, while for chaotic dynamics, the trajectories behave like a two-dimensional Brownian motion. For a bounded trajectory, the mean square displacement is also bounded, resulting in a growth rate K of 0, while for the chaotic case, M grows linearly, so K is 1. The rigorous justification of these ideas can be found in the theory for systems with Euclidean symmetry, which considers p and q as translation

variables for an unknown dynamical system $f\colon X \to X$ for some state space X. For details, we refer to [GM04, Section 5].

The interval of c is chosen such the V_{osc} is well-defined, as well as to avoid so-called resonances. These resonances are especially relevant for the periodic dynamics. If a periodic component has frequency ω, then the Fourier decomposition of the time series ϕ contains a term proportional to $\exp(-i\omega k)$. Then if $c = \omega$, the series $p_c(n)$ is proportional to n, which results in the mean square displacement being of order n^2. Irrespective of the dynamics, this results in 'spikes' of K_c around 1, even when the dynamics are regular, as we also encounter in Figure 4 later.

Another important note is that the test depends on the length of the time series, as it depends on the asymptotic behaviour of M_c. Hence N should be chosen large enough such that the 'typical' behaviour of p and q becomes dominant.

3 Application: the Rössler equations

As a numerical example for the 0-1 test, we consider the 3-dimensional Rössler system given by

$$\begin{cases} \dot{x} &= -y - z \\ \dot{y} &= x + ay \\ \dot{z} &= b + z(x - d). \end{cases} \tag{7}$$

This continuous time system first studied by Otto Rössler in the 1970s, and exhibits both periodic and chaotic behaviour for different values of the parameters a, b and d. As an example of chaotic behaviour, we investigate the parameters $a = b = 0.2$ and $d = 5.7$, which where the parameters originally researched by Rössler in [Rös76]. As an example of periodic behaviour, we consider $a = b = 0.1$ and $d = 8.5$. The parameters used by [GM16] were $a = 0.432, b = 2$ and $d = 4$. To give a sense of the dynamics, a plot of the (x, y)-plane of the Rössler system is shown in Figure 1.

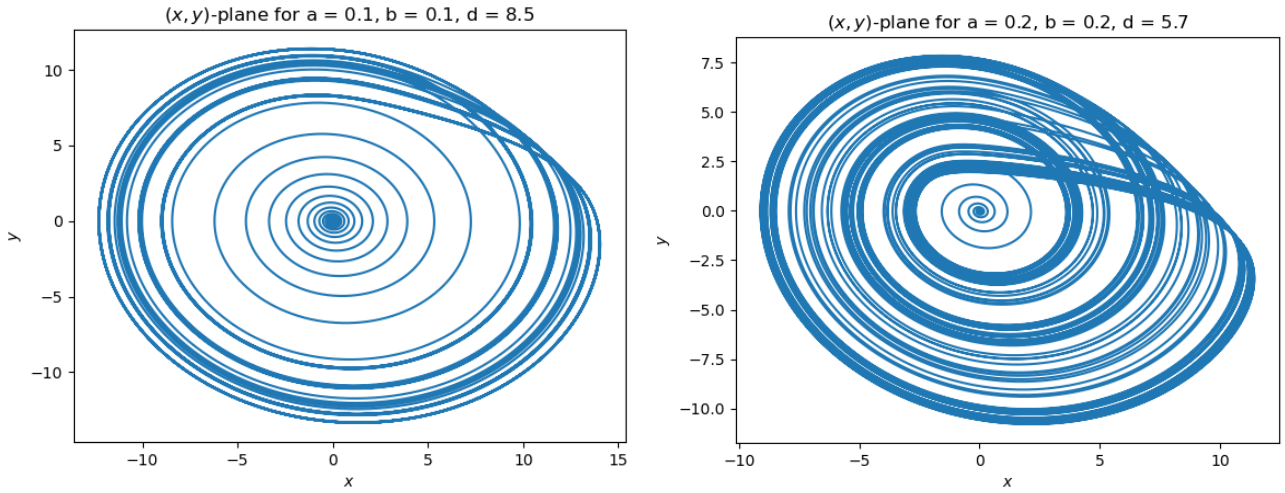

Figure 1: Plot of the regular dynamics (left) and chaotic dynamics of the Rössler system. Only the first 50.000 time steps are shown to produce a clearer image. For 500.000 time steps, the regular dynamics look the same due to the periodicity of the map, while for the regular dynamics, most of the plane is covered by the orbits of the map.

We use the fourth order Runge-Kutta method to numerically integrate the system up to time T such that $\Delta t = 0.01$ and we have $N = 500.000$ data points. As observable, we consider the

time series ϕ to be the x-component of the numerically integrated solution, for which we take discrete time points $\phi(t_1), \phi(t_2), \ldots, \phi(t_N)$. We choose these time points by taking a sampling time τ_s and setting $t_j = j\tau_s$. As explained by [GM16, Section 5.2], the Rössler system is an example of a system where we run into the problem of oversampling if τ_s is too small. We do not treat this issue in-depth, but base the choice of τ_s on the analysis by [GM16] and on numerical experiments for different parameters and data size. It turns out that for our applications, choosing $\tau_s = 1$ produces good results. The sampling of the timeseries ϕ is shown in Figure 2.

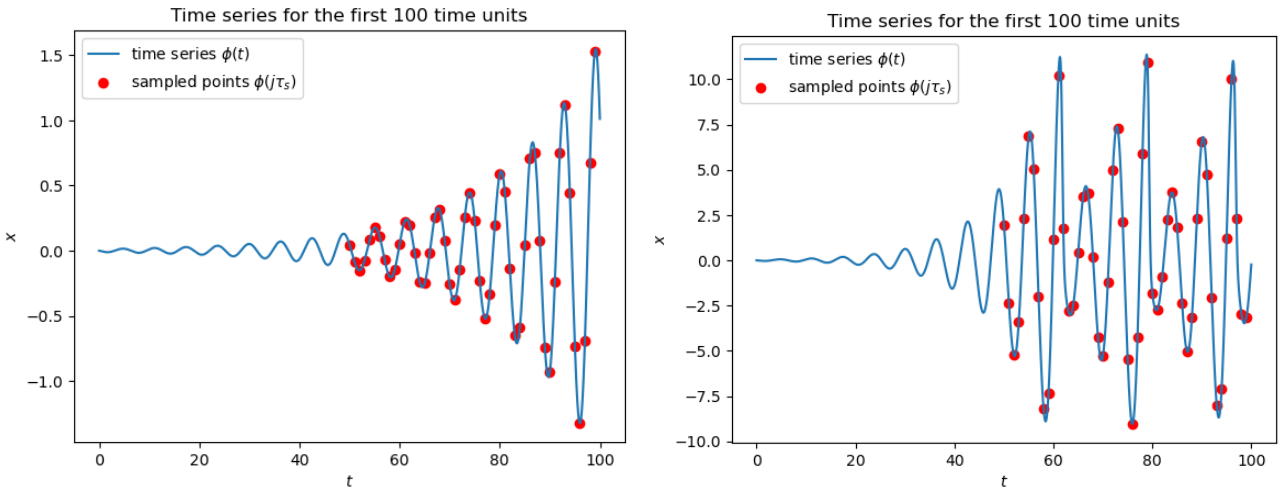

Figure 2: Plot of the sampling method of the time series ϕ, the x-variable of the Rössler system. We start sampling after 5000 time steps, in order to disregard transient behaviour.

As first step in the 0-1 test, we compute the values p_c and q_c. For c, we sample 100 uniformly random values in the interval $(\frac{\pi}{5}, \frac{4\pi}{5})$, which ensures we minimize the number of resonances. For an arbitrary value of c, we show the dynamics of the (p_c, q_c)-plane in Figure 3. These figures illustrate the distinction between the regular and chaotic dynamics of the translation variables p and q.

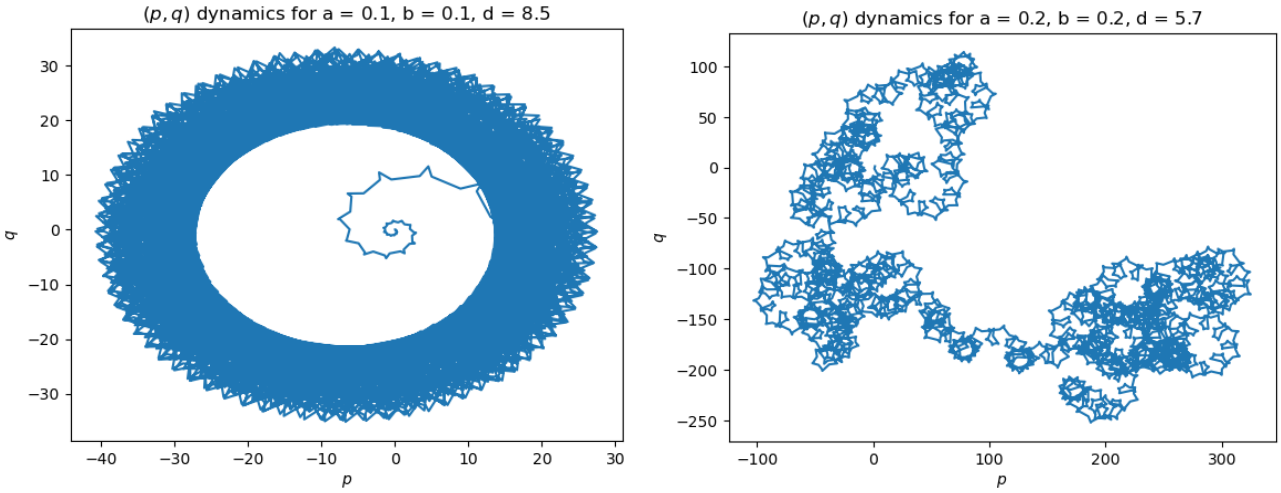

Figure 3: Plot of the regular dynamics (left) and chaotic dynamics of the Rössler system. On the left, $c \approx 0.822$, and on the right, $c \approx 0.675$.

For these 100 values of c, we then compute K_c for the regular and chaotic case. In Figure 4, we show K_c plotted against the values of c. For the regular case, $K \approx 0.0037$, and for the chaotic

case $K \approx 0.94$ demonstrate that the 0-1 test clearly distinguishes the behaviour between these dynamics.

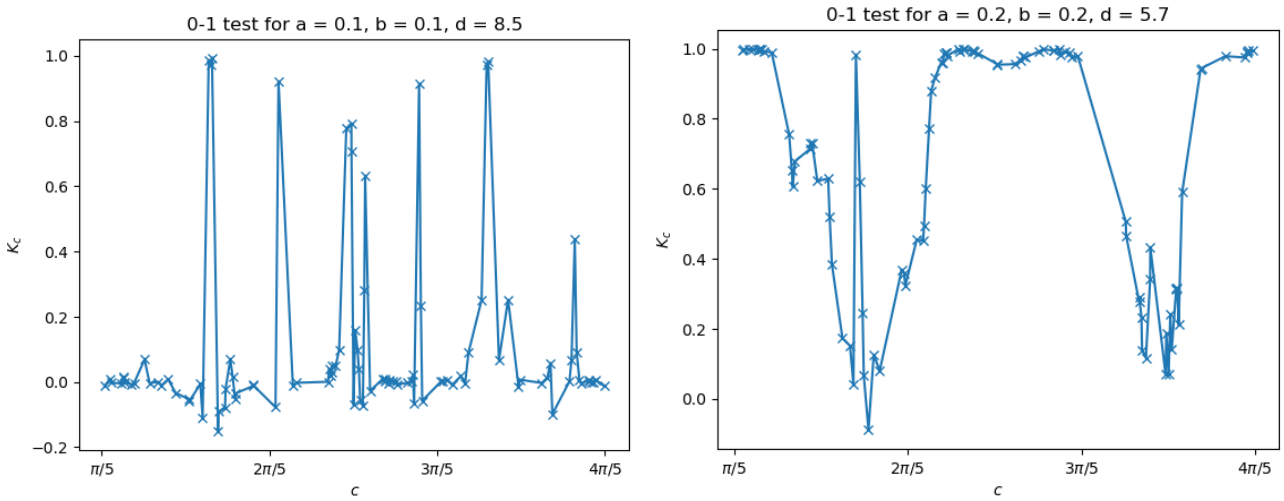

Figure 4: Plot of K_c for the regular dynamics (left) with median $K \approx 0.003733$ and chaotic dynamics (right) with $K \approx 0.9435$.

In this elementary example, we have only used direct analysis of M_c to compute K_c, while for systems containing more oscillating terms, it might be necessary to use D_c instead. The code used was also tested for the parameters of the logistics map used as an example in [GM16], to assert the implementation is correct.

References

[Git] Naereen @ Github. *Runge-Kutta methods for ODE integration in Python*. https://perso.crans.org/besson/publis/notebooks/Runge-Kutta_methods_for_ODE_integration_in_Python.html. Online; accessed 18 June 2024.

[GM04] Georg A Gottwald and Ian Melbourne. "A new test for chaos in deterministic systems". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 460.2042 (2004), pp. 603–611.

[GM16] Georg A Gottwald and Ian Melbourne. "The 0-1 test for chaos: A review". In: *Chaos detection and predictability* (2016), pp. 221–247.

[Phy24] Spencer Weart American Institute of Physics. *Chaos in the Atmosphere*. https://history.aip.org/climate/chaos.htm. Online; accessed 2 July 2024. 2024.

[Rös76] Otto E Rössler. "An equation for continuous chaos". In: *Physics Letters A* 57.5 (1976), pp. 397–398.

A Code

The implementation of the Runge Kutta method was taken from [Git].

```
# %% [markdown]
# # Implementation of the 0-1 test for chaos

# %%
```

```
import numpy as np
import scipy as sc
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
import statistics

# %% [markdown]
# ### Functions

# %%
# Runge-Kutta
def rungekutta4(f, y0, t, args=()):
    n = len(t)
    y = np.zeros((n, len(y0)))
    y[0] = y0
    print(y[0])
    print(np.shape(y))
    for i in range(n - 1):
        h = t[i+1] - t[i]
        k1 = f(y[i], t[i], *args)
        k2 = f(y[i] + k1 * h / 2., t[i] + h / 2., *args)
        k3 = f(y[i] + k2 * h / 2., t[i] + h / 2., *args)
        k4 = f(y[i] + k3 * h, t[i] + h, *args)
        y[i+1] = y[i] + (h / 6.) * (k1 + 2*k2 + 2*k3 + k4)
    return y

# %%
def Rossler(U, t, a, b, d):
    dxdt = -U[1] - U[2] #dx/dt = -y - z
    dydt = U[0] + a*U[1] #dy/dt = x + ay
    dzdt = b + U[2]*(U[0]-d) #dz/dt = b + z(x-d)

    return np.array([dxdt, dydt, dzdt])

# %%
def f_log(x,mu):
    return mu*x*(1-x)

# %% [markdown]
# ### Parameters

# %%
# parameters time series
a, b, d = 0.1, 0.1, 8.5 #regular; chaos: 0.2, 0.2, 5.7 #(G&M): 0.432, 2, 3
datapoints = 500000
dt = 0.01
T = datapoints*dt
t = np.arange(0,T+dt,dt)
U0 = np.zeros(3) #initial condition

```

```
# parameters 0-1 test
size_c = 100
c = np.random.uniform(np.pi/5, 4*np.pi/5, size_c) #[0.9]
c.sort()

# test parameters logistics map
mu_reg = 3.55
mu_chaos = 3.97

# %% [markdown]
# ### Generate a time series

# %%
# time series rössler map
sol = rungekutta4(Rossler, U0, t, args=(a,b,d))
samp_par = 1 #sampling parameter
start_time = 50 #disregard first 50 time units (of 1000 total)
indices = np.arange(int(start_time/dt), len(t), int(samp_par/dt)) #indices to sample
phi = sol[:,0] #full time series is x coordinate
samp_t = t[indices]
samp_phi = phi[indices] #sampled time series according to sampling parameter

N = len(samp_phi)
print(N)
#N0 = N/10
mean_phi = statistics.mean(samp_phi)
print(mean_phi)


# %%
# logistics map test series
N_log = 5000
phi_log = np.zeros(N_log)
phi_log[0] = 0.5 #starting value x_0 of the logistics map
for n in range(N_log-1):
    phi_log[n+1] = f_log(phi_log[n], mu_chaos)

print(phi_log)
mean_log = statistics.mean(phi_log)
print(mean_log)

# %%
plt.plot(t,phi)
plt.scatter(samp_t, samp_phi, marker = 'o', color = 'red')
plt.show()

# Plot the first 100 values
plt.plot(t[:10000], phi[:10000], label = r'time series $\phi(t)$')
plt.scatter(samp_t[samp_t < t[10000]], samp_phi[samp_t < t[10000]], marker='o', color='
plt.legend()
```

```
plt.xlabel("$t$")
plt.ylabel("$x$")
plt.title("Time series for the first 100 time units")
plt.show()

plt.plot(sol[:50000,0], sol[:50000,1]) #!!! up to 50000
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.title(f"$(x,y)$-plane for a = {a}, b = {b}, d = {d}")
plt.show()

# %% [markdown]
# ### From time series to least squares

# %%
def create_p_q(c,N,phi):
    p = np.zeros((len(c),N))
    q = np.zeros((len(c),N))
    for i in range(len(c)):
        p[i,0] = phi[0]*np.cos(c[i])
        q[i,0] = phi[0]*np.sin(c[i])
        for n in range(N-1):
            p[i,n+1] = p[i,n] + phi[n+1]*np.cos((n+2)*c[i])
            q[i,n+1] = q[i,n] + phi[n+1]*np.sin((n+2)*c[i])

    return p,q

def create_M(c,p,q,N):
    N0 = int(N/10)
    M = np.zeros((len(c),N0))
    for i in range(len(c)):
        for n in range(N0):
            for j in range(N-N0):
                M[i,n] += (p[i,(j+n+1)]-p[i,j])**2 + (q[i,(j+n+1)]-q[i,j])**2
            M[i,n] = (1/(N-N0))*M[i,n]
    return M, N0

def create_D(c,M,N,mean):
    N0 = int(N/10)
    V_osc = np.zeros((len(c),N0))
    for i in range(len(c)):
        for n in range(N0):
            V_osc[i,n] = (mean**2)*((1-np.cos((n+1)*c[i]))/(1 - np.cos(c[i])))
    return M - V_osc, V_osc

# %%
p, q = create_p_q(c,N,samp_phi)
M_r, N0_r = create_M(c,p,q,N)
D_r, V_oscr = create_D(c,M_r, N, mean_phi)

```

```
# %%
p_log, q_log = create_p_q(c,N_log,phi_log)
M_log, N0_log = create_M(c,p_log,q_log,N_log)
D_log, V_osclog = create_D(c,M_log,N_log, mean_log)

# %%
c_index = 10
print(c[c_index])
plt.plot(p[c_index,:], q[c_index,:])
plt.title(f"$(p, q)$ dynamics for a = {a}, b = {b}, d = {d}")
plt.xlabel('$p$')
plt.ylabel('$q$')
plt.show()

plt.plot(p_log[0,:], q_log[0,:], color = 'magenta', linewidth = 0.5)
plt.show()

n = np.arange(0,N0_log)
plt.plot(n,M_log[0,:], label = 'M_c')
plt.plot(n,D_log[0,:], label = 'D_c')
plt.plot(n,V_osclog[0,:], label = 'V_osc')
plt.title("logistics map")
plt.legend()
plt.show()

n_r = np.arange(0,N0_r)
plt.plot(n_r,M_r[0,:], label = 'M_c')
plt.plot(n_r,D_r[0,:], label = 'D_c')
plt.plot(n_r,V_oscr[0,:], label = 'V_osc')
plt.title("Rössler system")
plt.legend()
plt.show()

# %% [markdown]
# ### Correlation method

# %%
def correlation_method(N0, c, M):
    zeta = np.arange(1,N0+1)
    K_seq = np.zeros(len(c))
    for index in range(len(c)):
        K_seq[index],_ = pearsonr(zeta,M[index,:])
    K = statistics.median(K_seq)
    return K_seq, K

# %%
#K_seq_log, K_log = correlation_method(N0_log, c, M_log)
K_seq_r, K_r = correlation_method(N0_r, c, M_r)
print(K_r)
#print(K_log)
```

```
205
206 #plt.plot(c,K_seq_log, marker = 'x')
207 #plt.show()
208
209 #K_r = 0.9434730037124811 voor N=500 000, tau = 1, a, b, d = 0,2, 0,2, 5,7
210
211 ticks = [np.pi / 5, 2 * np.pi / 5, 3 * np.pi / 5, 4 * np.pi / 5]
212 tick_labels = [r'$\pi/5$',r'$2\pi/5$', r'$3\pi/5$', r'$4\pi/5$']
213 plt.plot(c,K_seq_r, marker = 'x')
214 plt.xlabel('$c$')
215 plt.ylabel('$K_c$')
216 plt.title(f"0-1 test for a = {a}, b = {b}, d = {d}")
217 plt.xticks(ticks, tick_labels)
218 plt.show()
```