



**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**  
**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**  
**«ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет автоматизации и информатики  
Кафедра автоматизированных систем управления

Индивидуальное домашнее задание  
по дисциплине “Основы теории управления”  
«Исследование системы управления нелинейным объектом»

Студент      АС-21-1      \_\_\_\_\_  
(подпись, дата)

Коновалов К.А

Руководитель  
старший преподаватель      \_\_\_\_\_  
(подпись, дата)

Болдырихин О.В

Липецк 2023

## Задание кафедры

Разработать программу для исследования заданной нелинейной системы.

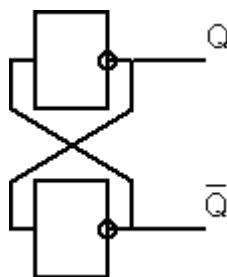
Программа должна выполнять следующие функции:

- определение положений равновесия системы (стационарных состояний, особых точек) и их типа в зависимости от значений параметров;
- задание значений параметров системы;
- задание начальных условий;
- задание возмущающих воздействий;
- задание управляющих воздействий;
- расчет характеристики векторного поля скоростей;
- моделирование возмущающих и управляющих воздействий;
- расчет состояния системы в заданные моменты времени при заданных параметрах системы, возмущающих и управляющих воздействий;
- вывод рассчитанных значений состояния в табличном и графическом виде;
- построение фазового портрета системы.

## 1 Теоретическая информация

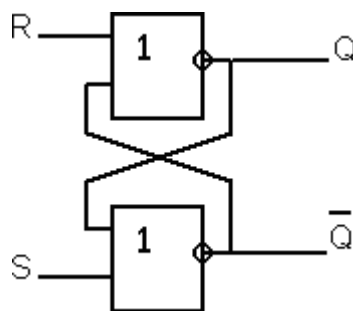
Триггер – схема (устройство) с двумя устойчивыми состояниями, переходящие из одного состояния в другое под действием внешнего сигнала и способное сохранить (сколь угодно долго) устойчивые состояния после снятия внешнего воздействия. Под термином “сколь угодно долго” подразумевается “до тех пор, пока включено питание устройства”.

В основе триггера лежит бистабильная ячейка – схема, которая может находиться только в двух устойчивых состояниях.



Два устойчивых состояния: первое  $Q=1$ ,  $\bar{Q}=0$ , второе  $Q=0$ ,  $\bar{Q}=1$ .

Триггер – управляемая бистабильная ячейка. Например на элементах ИЛИ-НЕ (хотя можно и на элементах И-НЕ):



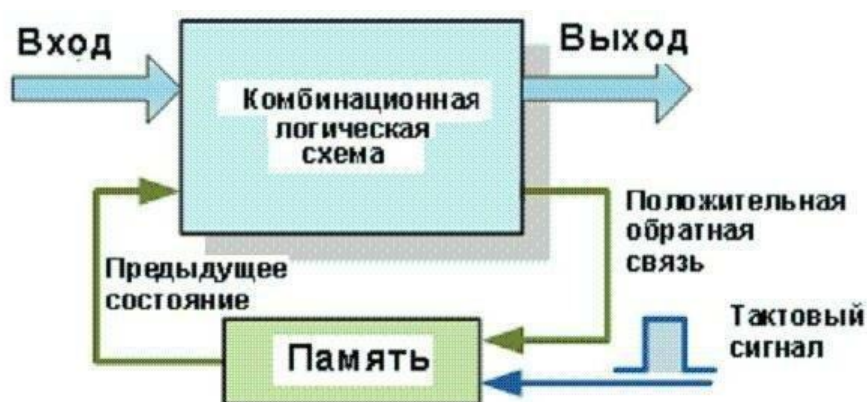
S – Set (установка), R – Reset (сброс), при  $S=R=0$  триггер превращается в не стабильную ячейку, сохраняющую своё состояние (это режим хранения информации).

Свойство бистабильности используется для запоминания двоичных цифровых сигналов в памяти компьютеров. Кроме того, триггеры применяют в качестве ключевых каскадов для включения и отключения каких-либо устройств, а также переключателей каналов (коммутаторов).

В динамических системах, бистабильность означает, что система имеет два стабильных состояния равновесия. Т.е. объект может находиться в любом из двух состояний.

Переключение триггера происходит скачком. В результате формируются импульсы прямоугольной формы с длительностью, равной интервалу между запускающими импульсами. Поэтому их часто используют в качестве формирователей или генераторов прямоугольных импульсов. Триггеры, как и импульсные генераторы различных типов, интенсивно развивались в радиотехнических и особенно импульсных радиолокационных системах. Сейчас они находят широкое применение в радиоэлектронной аппаратуре, измерительной и вычислительной технике, системах передачи информации, технологических установках и бытовой технике.

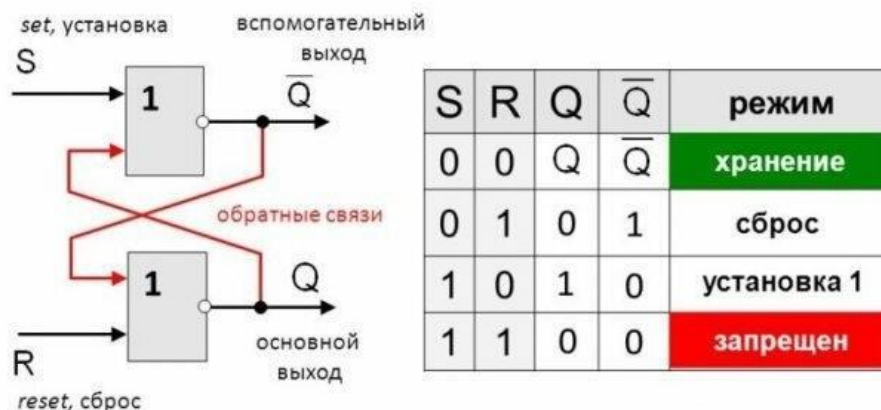
В отличие от комбинационных логических схем, которые изменяют состояние в зависимости от фактических сигналов, поданных на их входы в определенное время, последовательностные логические имеют некоторую форму присущей им встроенной «памяти», так что они могут учитывать как предыдущее, так и фактическое состояние их входов и выходов. Общая структурная схема последовательностного устройства показана ниже.

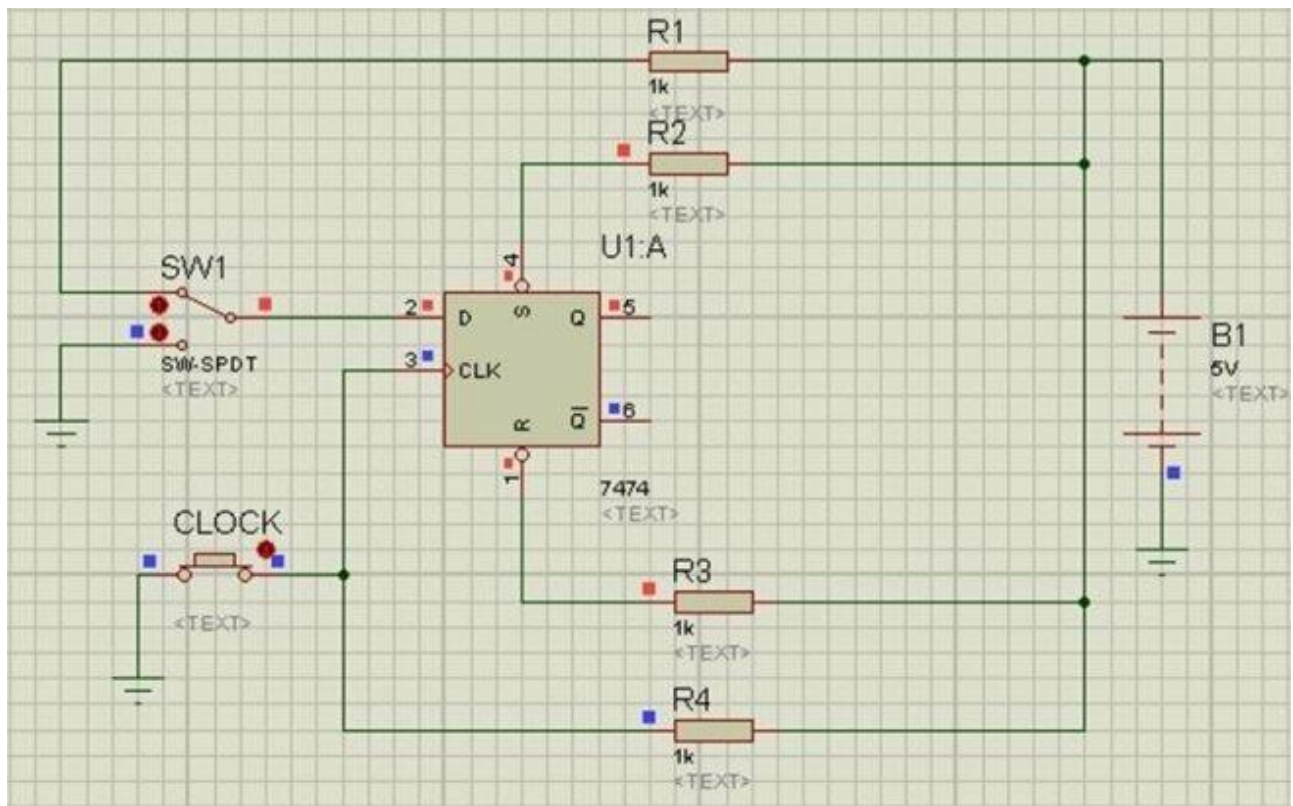


RS-триггер как цифровой управляющий автомат включает собственно память и комбинационную схему управления на типовых логических элементах, реализующую его входной логический алгоритм. Если рассматривать эту схему применительно к простейшим схемам триггеров, то

они не имеют структурно выделенной памяти в виде какой-то специализированной микросхемы или схемного узла. Память триггера существует на уровне функции, она словно встроена в алгоритм работы его комбинационной схемы управления. Проявлением этой «памяти» является так называемая бистабильность триггера, выходы которого могут находиться в одном из двух основных состояний: логической единицы (далее — 1) или логического нуля (далее — 0). Установившиеся значения своих выходов триггер запоминает («защелкивает» их) и сохраняет, пока не возникнет очередное изменение его входных сигналов.

**Триггер** – это логическая схема, способная хранить 1 бит информации (1 или 0). Строится на 2-х элементах **ИЛИ-НЕ** или на 2-х элементах **И-НЕ**.



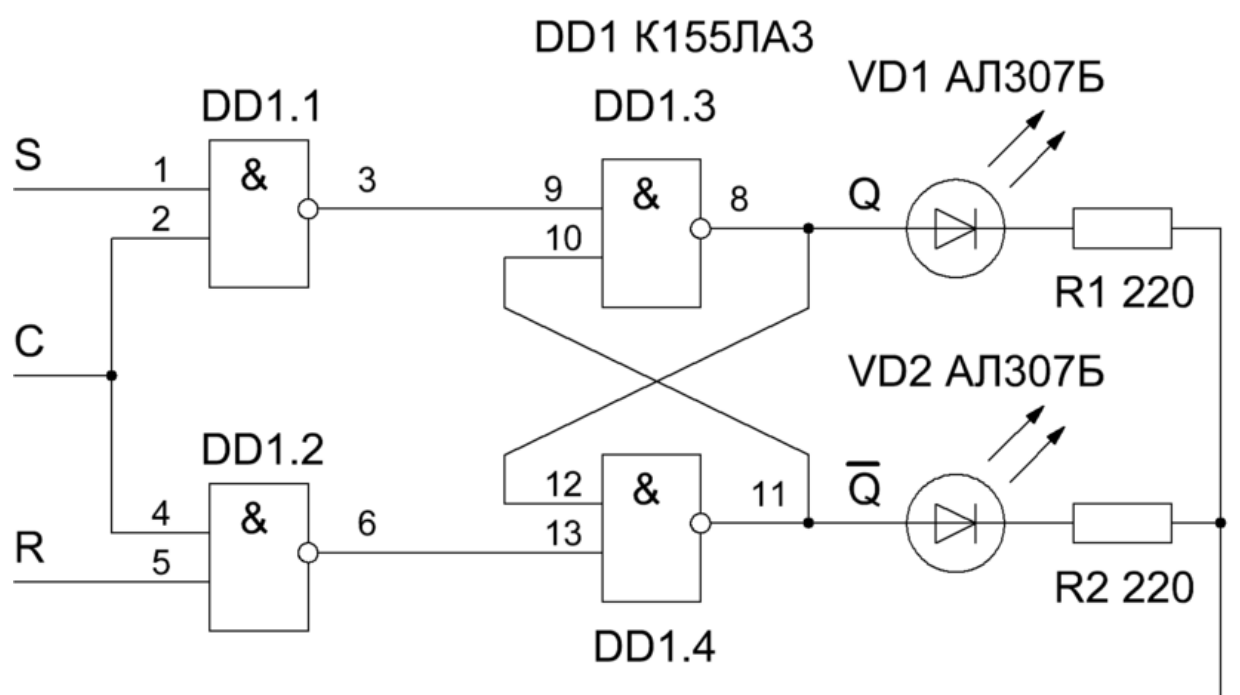


Если стандартные логические элементы являются строительными блоками комбинационных схем, бистабильные схемы, включая и RS-триггер, являются основными компонентами построения последовательностных логических устройств, таких, как регистры хранения данных, регистры сдвига, устройства памяти или счетчики. В любом случае рассматриваемые триггеры (разумеется, как и все последовательностные схемы) могут быть выполнены в виде следующих основных типов:

1. Асинхронный RS-триггер – схема, которая изменяет состояние сразу при изменении входных сигналов. Для рассматриваемого типа устройств ими являются сигналы на информационных входах R (сброс) и S (установка). Согласно установившейся практике, соответствующие входы называют так же, как и сигналы на них.
2. Синхронный RS-триггер, управляемый статически, работа которого синхронизирована с уровнем определенного тактового сигнала.

3. Триггер по п.2 с динамическим управлением, работа которого синхронизирована с моментами появления фронтов (или спадов) тактового сигнала.

Таким образом, если изменения состояния выходов происходят только при наличии тактового сигнала, который подается на отдельный тактовый вход С, то триггер является синхронным. В противном случае схема считается асинхронной. Чтобы сохранить свое текущее состояние, последовательностные схемы используют обратную связь, т. е. передачу части выходного сигнала на ее вход.



Рассмотрим, как происходит работа RS-триггера в этом состоянии, задаваемом значениями  $R = 0$  и  $S = 1$ . Поскольку на вход R элемента И-НЕ Y подан уровень 0, то  $\tilde{Q} = 1$  (логика И-НЕ). С выхода Y сигнал  $\tilde{Q}$  также подан обратно на элемент X (вход «А»). Поскольку  $S = A = 1$ , то  $Q = 0$ .

Если устанавливается  $R = 1$ , а вход  $S$  по-прежнему равен 1, то на входах  $Y$  имеем  $B = 0$  и  $R = 1$ , а его выход  $\tilde{Q} = 1$ , т. е. он не изменился. Итак, если  $S = 1$ , то RS-схема триггера «защелкивается» в состоянии «Установлен»  $Q = 0$  и  $\tilde{Q} = 1$ , а смена сигнала  $R$  его не изменяет.

Мы можем определить состояние сигналов  $Q$  и  $\tilde{Q}$  по следующей таблице истинности: следующей таблице истинности:

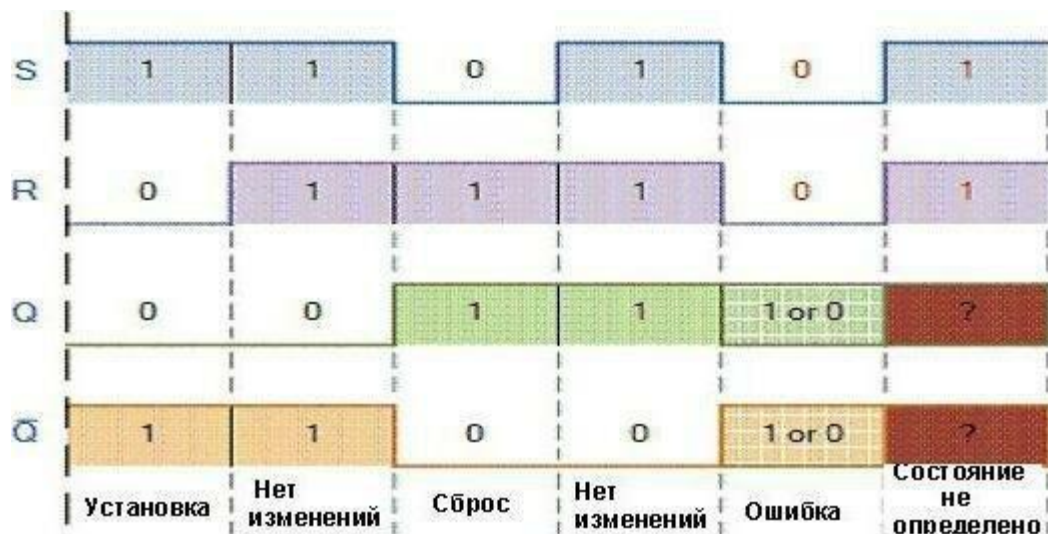
Состояние	S	R	Q	$\tilde{Q}$	Описание
Установка	1	0	0	1	Выход $\tilde{Q} = 1$
	1	1	0	1	без изменений
Сброс	0	1	1	0	Выход $\tilde{Q} = 0$
	1	1	1	1	без изменений
Недопустимое	0	0	1	1	состояние ошибки

Видно, что когда  $S = R = 1$ , то  $Q$  и  $\tilde{Q}$  могут быть равны как 1, так и 0 (но не одновременно!) в зависимости от уровней входов  $S$  или  $R$  перед возникновением данного состояния выходов. Таким образом, при условии  $S = R = 1$  нельзя изменить состояние выходов  $Q$  и  $\tilde{Q}$ . Оно может измениться только при смене уровня с 1 на 0 на одном из входов.

Значение  $S = R = 0$  является нежелательным или недопустимым состоянием, и его следует избегать. Состояние  $S = R = 0$  вызывает установку обоих выходов  $Q$  и  $\tilde{Q}$  на уровне 1, в то время как состояние  $\tilde{Q}$  всегда должно быть обратно  $Q$ . Результатом является то, что триггер теряет контроль над  $Q$  и  $\tilde{Q}$ , и если два входа теперь перейдут к состоянию 1, то схема становится неустойчивой и переключается в неопределенное состояние.

Сказанное в предыдущем разделе иллюстрирует следующая диаграмма переключения.





Как видно, при  $S = R = 0$  возникает дисбаланс (неопределенность) состояния выходов. Он может привести к переключению одного из выходов быстрее, чем другого, в результате чего произойдет переключение триггера в то или иное состояние, которое может не совпадать с требуемым, и данные будут повреждены. Это неустойчивое состояние обычно называют метастабильным.

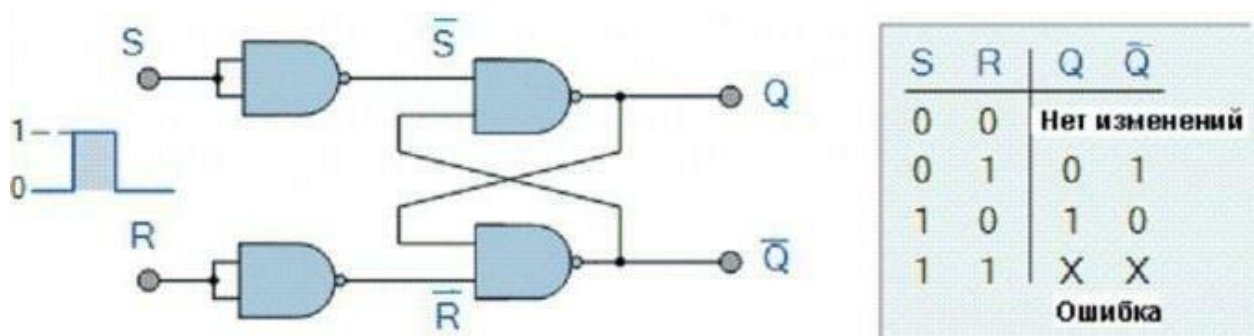
Таким образом, подобный триггер-защелка может быть переведен в состояние «Установлен» путем подачи 0 на его S-ввод (при наличии 1 на R-вводе) и переведен в состояние «Сброшен» подачей 0 на R-ввод (при наличии 1 на S-вводе). Триггер входит в неопределенное состояние (мета-стабильное), если на оба его входа одновременно подается уровень 0.

Переключение состояния выходов происходит с небольшой задержкой относительно изменения сигнала на одном из входов без использования тактового сигнала. Следовательно, рассмотренная выше схема представляет асинхронный RS-триггер.

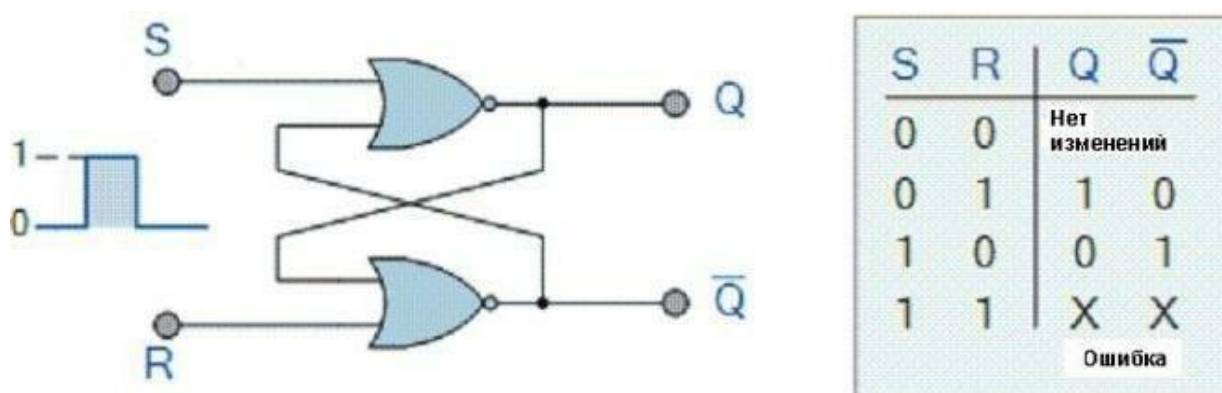
Как мы видели выше, базовые элементы И-НЕ рассмотренного RS-триггера работают так, что при его установке  $\tilde{Q} = 1$  и  $Q = 0$ , а при его сбросе  $\tilde{Q} = 0$  и  $Q = 1$ , хотя логичнее было бы в первом состоянии иметь  $Q = 1$ , а во втором —  $Q =$

0. При этом еще и получается, что смена состояний происходит при падении уровня сигнала с 1 до 0.

Таким образом, для правильной работы схемы триггера его входные сигналы нужно проинвертировать. Тогда переключения его состояний будут происходить при подаче положительных входных сигналов. Для этого в схему нужно добавить два дополнительных И-НЕ элемента, присоединенных как инверторы к  $\bar{S}$ - и  $\bar{R}$ -входам, как показано на рисунке ниже. Здесь на входах элементов И-НЕ уже представлены инверсные входные сигналы.



Так же, как и с использованием И-НЕ элементов, можно построить простой RS-триггер с использованием двух ИЛИ-НЕ элементов, соединенных по такой же схеме. Она будет работать аналогичным образом, как и рассмотренная выше схема И-НЕ. При этом активным является высокий уровень сигналов на входах, а недопустимое состояние возникает, когда на оба входа подан уровень логической «1», как это показано в таблице истинности на рисунке ниже.

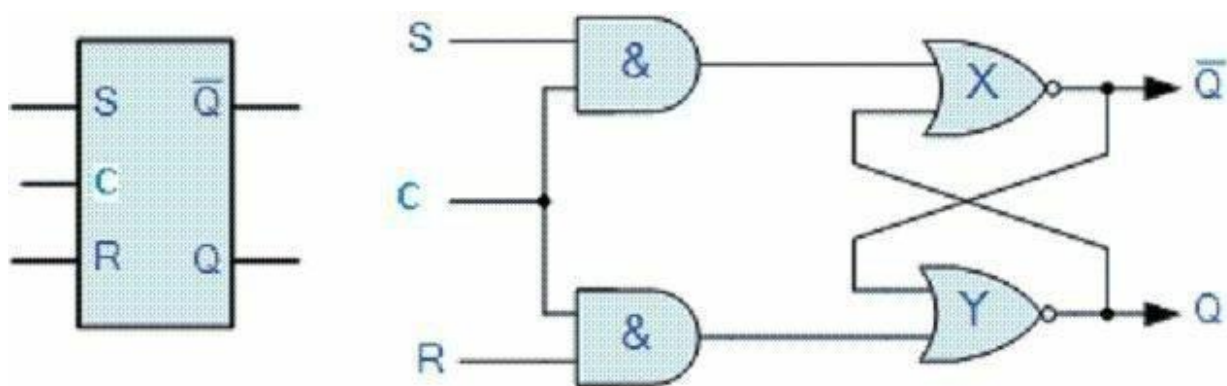




Для повышения мощности подключаемой нагрузки можно собрать триггер с применением тиристоров. К управляющему электроду присоединяют вход S, к затвору – R. Для поддержания постоянного напряжения на аноде подойдет транзистор, включенный в соответствующую цепь.

Несмотря на общие тенденции миниатюризации, вполне допустимо создать функциональный триггер из реле. Подобные решения, в частности, применяют для защиты цепей питания при включении мощных электроприводов.

Иногда желательно в последовательностных логических схемах иметь бистабильный триггер, изменяющий свое состояние, когда соблюдены определенные условия, независимо от состояния S- или R-входов. Такая схема может быть создана подключением двухвходового элемента И последовательно с каждым входом триггера. Объединив два входа элементов И, получим новый вход триггера. Добавление его означает, что выходы Q и  $\bar{Q}$  изменяют состояние, когда сигнал на нем является высоким, и, следовательно, он может быть использован в качестве тактового C-ввода, как показано на рисунке ниже.



Когда сигнал на C-входе находится на уровне 0, то выходы двух элементов И — также на уровне 0 (логика элемента И), независимо от состояния двух входов S и R, а два выхода Q и  $\bar{Q}$  «защелкнуты» в последнем установившемся состоянии. Когда сигнал на C-входе изменяется на уровень 1, то схема отвечает

как обычный бистабильный триггер, становясь прозрачной для установки и сброса состояний.

Этот дополнительный С-вход также может быть подключен к выходу генератора тактовой частоты синхронизации, образуя тогда синхронный RS-триггер. Таким образом, данная схема работает как стандартная бистабильная триггерная «защелка», но выходы активируются только тогда, когда уровень 1 подан на С-вход, и отключаются при появлении уровня логического нуля.

Выяснив, что значит триггер, несложно использовать полученные знания для решения практических задач. С помощью логических элементов:

автоматизируют работу систем освещения;

обеспечивают безопасное подключение станков и других мощных нагрузок;

предотвращают опасные режимы с использованием сигналов от внешних датчиков.

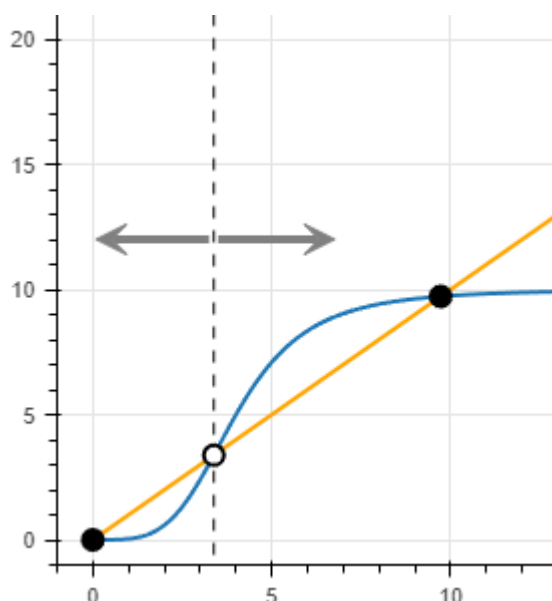
В данной работе, я показал, как положительная авторегуляция может при некоторых условиях генерировать в системе сбистабильностью.

Мы можем представить положительную схему используя функцией Хилла:

$$\frac{dx}{dt} = \beta \frac{(x/k)^{n1}}{1 + (x/k)^n} - \gamma x.$$

Мы можем получить некоторое представление о поведении этой системы, не погружаясь сразу в решение ODE или даже нахождение ее устойчивых состояний. Во-первых, мы замечаем, что  $dx/dt$  исчезает для  $x=0$  независимо от значений параметров, поэтому  $x=0$  всегда находится в устойчивом состоянии. Если активатор отсутствует, он остается отсутствующим.

Чтобы пойти дальше, мы обратимся к графическим для различных значений  $x$ .



Значение  $dx/dt$  при определенном значении  $x$  определяется разницей между производительностью и скоростью разложения для этого значения  $x$ . Производная по времени обращается в нуль ( $dx/dt=0$ ), и мы имеем фиксированную точку.

Обратите внимание, что приведенный выше график, в дополнение к сообщению нам местоположения неподвижных точек, также сообщает нам направление, в котором система будет двигаться при любом заданном значении  $x$ . Если при этом значении скорость производства выше скорости разложения, то  $dx/dt > 0$ , поэтому  $x$  увеличится. Соответственно, если скорость разложения выше, чем скорость производства при определенном значении  $x$ , то  $dx/dt < 0$  и  $x$  будет уменьшаться.

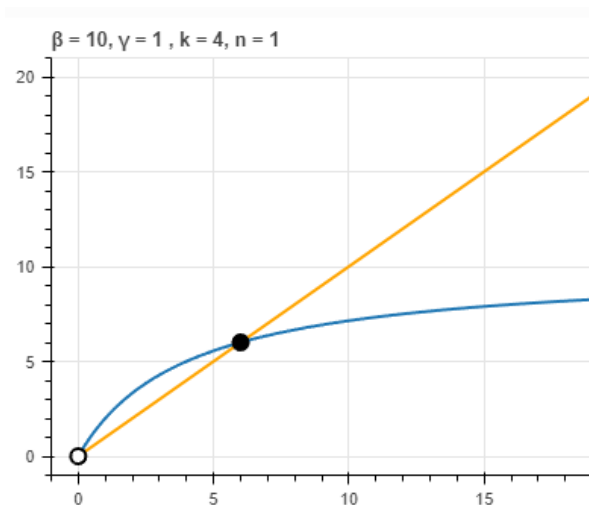
Глядя на эти стрелки, мы видим, что система движется к некоторым фиксированным точкам и удаляется от других. Когда система вблизи фиксированной точки стремится двигаться к этой фиксированной точке с любого

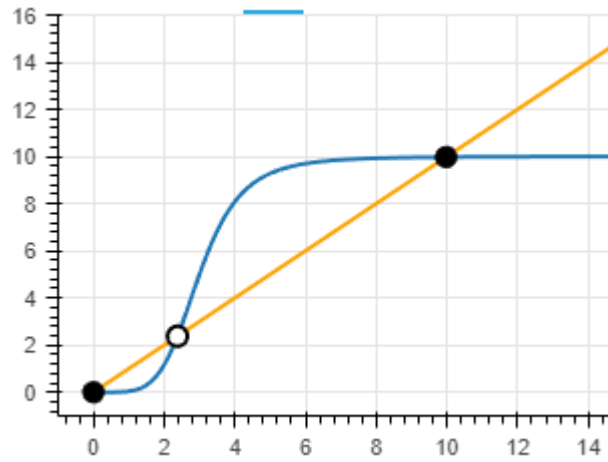


направления, тогда мы называем эту фиксированную точку стабильной. В противном случае это называется нестабильной неподвижной точкой. Если начальное условие этой системы не помещает ее точно в нестабильную фиксированную точку, то по мере изменения системы с течением времени она будет сходиться к одной из двух стабильных фиксированных точек. К какой фиксированной точке он сходится, определяется начальным значением  $x$ ; если система начинается ниже нестабильной фиксированной точки (слева от пунктирной линии), то она будет сходиться к нижней стабильной фиксированной точке; если она начинается выше нестабильной фиксированной точки, она будет сходиться к более высокой стабильной фиксированной точке. Таким образом, каждая устойчивая неподвижная точка имеет область притяжения, набор значений  $x$ , при которых система будет сходиться к неподвижной точке. На приведенном выше графике область притяжения для более высокой фиксированной точки - это все значения  $x$  справа от пунктирной линии, а для нижней фиксированной точки - все значения  $x$  слева от пунктирной линии (включая, конечно, только неотрицательный  $x$ ).

Поскольку эта система имеет две стабильные неподвижные точки, мы можем описать ее как бистабильную.

Если мы уменьшим коэффициент до 1, сохранив другие параметры неизменными, вы увидите, что теперь у нас есть только одна стабильная фиксированная точка (и одна нестабильная фиксированная точка при  $x = 0$ ).





По мере увеличения  $\gamma$  две верхние фиксированные точки становятся все ближе и ближе друг к другу, в конечном счете исчезая, поскольку система становится моностабильной, всегда сходящейся к единственному значению  $x = 0$ .

Диапазон значений параметра, который находится вблизи этой точки исчезновения, важен, потому что очень небольшое изменение параметра может привести к очень резкому изменению поведения системы. Такое явление называется бифуркацией. В этом случае мы можем видеть, что система переходит от моностабильности к бистабильности, и в критической области пространства параметров вблизи этой бифуркации даже незначительного изменения значения параметра достаточно, чтобы подтолкнуть систему к переходу в тот или иной режим.

Вспомним наше дифференциальное уравнение. Здесь мы можем идентифицировать две переменные —  $x$  и  $t$  — и четыре параметра —  $\beta$ ,  $\gamma$ ,  $k$  и  $n$ . Понимание того, как схема ведет себя при изменении четырех параметров, является сложной задачей. Однако динамика не зависит от всех четырех параметров независимо. Процедура безразмерности позволяет нам уменьшить количество независимых параметров.

$$x = \left[ \frac{N}{L^3} \right]$$

$$t = t_d \tilde{t},$$

$$x = x_d \tilde{x}$$



Подставляя их в уравнение, мы получаем

$$\frac{x_d}{t_d} \frac{d\tilde{x}}{d\tilde{t}} = \beta \frac{(x_d \tilde{x}/k)^n}{1 + (x_d \tilde{x}/k)^n} - \gamma x_d \tilde{x}$$

Теперь мы можем разделить обе части нашего уравнения на  $k / t_d$ , чтобы получить

$$\frac{d\tilde{x}}{d\tilde{t}} = \frac{\beta t_d}{k} \frac{\tilde{x}^n}{1 + \tilde{x}^n} - \gamma t_d \tilde{x}$$

Из нашего анализа, приведенного выше, мы обнаружили, что система обладает следующими свойствами.

- Система всегда имеет фиксированную точку при  $x=0$ .
- Система имеет либо ноль, одну, либо две фиксированные точки с  $x>0$ .
- Если система имеет ноль фиксированных точек с  $x>0$ , она является моностабильной при  $x=0$ .
- Если система имеет две неподвижные точки с  $x>0$ , то большая из двух стабильна, как и неподвижная точка, равная нулю, при этом средняя неподвижная точка нестабильна. Это бистабильный режим.

## Вывод

В ходе выполнения индивидуального домашнего задания была исследована нелинейная стохастическая динамическая система с бистабильностью.

Было выяснено как параметры системы влияют на точки устойчивости и их количество. Наличие случайных процессов накладывает дополнительные ограничения на эти параметры, т.к. необходимо, чтобы система сохраняла устойчивое состояние до момента подачи управляющего воздействия.

Бистабильность широко используется в устройствах цифровой электроники для хранения двоичных данных. Это основная характеристика триггера, схемы, которая является фундаментальным строительным блоком компьютеров и некоторых типов полупроводниковой памяти. Бистабильное устройство может хранить один бит двоичных данных, при этом одно состояние представляет «0», а другое состояние - «1».

## Текст программы для исследования

```
import os, sys, subprocess
if "google.colab" in sys.modules:
    cmd = "pip install --upgrade biocircuits watermark"
    process = subprocess.Popen(cmd.split(), stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    stdout, stderr = process.communicate()
    data_path = "https://biocircuits.github.io/chapters/data/"
else:
    data_path = "data/"
# -----

import numpy as np
import pandas as pd
import scipy.integrate
import scipy.optimize

import biocircuits.jsplots

import bokeh.io
import bokeh.models
import bokeh.plotting

bokeh.io.output_notebook()
# Parameters
beta = 10
gamma = 1
k = 4
n = 4

# Theroetical curves
x = np.linspace(0, 20, 400)
prod = beta * (x / k) ** n / (1 + (x / k) ** n)
removal = gamma * x

# Fixed points
fp_rhs = lambda x: beta / gamma * (x / k) ** n / (1 + (x / k) ** n)
fixed_points = [
    0,
    float(scipy.optimize.fixed_point(fp_rhs, 3)),
    float(scipy.optimize.fixed_point(fp_rhs, 5)),
]

# Build plot
p = bokeh.plotting.figure(
    frame_height=275,
    frame_width=375,
    x_axis_label="x",
    y_axis_label="production or removal rate",
    title=f" $\beta = \{beta\}$ ,  $\gamma = \{gamma\}$ ,  $k = \{k\}$ ,  $n = \{n\}$ ",
    x_range=[-1, 20],
)

# Plot production and removal rates
p.line(x, prod, line_width=2, color="#1f77b4")
p.line(x, removal, line_width=2, color="orange")

# Plot fixed points
for i, fp in enumerate(fixed_points):
    if i % 2:
        p.add_layout(
            bokeh.models.Span(
                location=fp,
                level="underlay",
                dimension="height",
                line_color="black",
```

```

        line_dash="dashed",
    )
    )
    fill_color = "white"
else:
    fill_color = "black"

p.circle(
    [fp], [gamma * fp], color="black", size=10, line_width=2, fill_color=fill_color,
)

# Annotate
p.add_layout(
    bokeh.models.Arrow(
        end=bokeh.models.VeeHead(size=15, fill_color="gray", line_color="gray"),
        line_width=4,
        x_start=3.25,
        y_start=12,
        x_end=0.1,
        y_end=12,
        line_color="gray",
    )
)
p.add_layout(
    bokeh.models.Arrow(
        end=bokeh.models.VeeHead(size=15, fill_color="gray", line_color="gray"),
        line_width=4,
        x_start=3.5,
        y_start=12,
        x_end=6.75,
        y_end=12,
        line_color="gray",
    )
)
p.add_layout(
    bokeh.models.Arrow(
        end=bokeh.models.VeeHead(size=15, fill_color="gray", line_color="gray"),
        line_width=4,
        x_start=20,
        y_start=12,
        x_end=13,
        y_end=12,
        line_color="gray",
    )
)
p.text(
    x=[14],
    y=[9.933],
    text=["production rate"],
    text_color="#1f77b4",
    text_font_size="10pt",
    text_align="left",
    text_baseline="top",
)
p.text(
    x=[14],
    y=[14],
    text=["removal rate"],
    text_color="orange",
    text_font_size="10pt",
    text_align="left",
    angle=0.6,
)

bokeh.io.show(p)
# Reduce n to 1
n = 1

# Recompute production curve

```

```

prod = beta * (x/k) ** n / (1 + (x/k) ** n)

# Fixed points
fp_rhs = lambda x: beta / gamma * (x/k) ** n / (1 + (x/k) ** n)
fixed_points = [0, float(scipy.optimize.fixed_point(fp_rhs, 3))]

# Build plot
p = bokeh.plotting.figure(
    frame_height=275,
    frame_width=375,
    x_axis_label="x",
    y_axis_label="production or removal rate",
    title=f"β = {beta}, γ = {gamma} , k = {k}, n = {n}",
    x_range=[-1, 20],
)

# Plot production and removal rates
p.line(x, prod, line_width=2, color="#1f77b4")
p.line(x, removal, line_width=2, color="orange")

# Plot fixed points
for i, fp in enumerate(fixed_points):
    fill_color = "black" if i % 2 else "white"
    p.circle(
        [fp], [gamma * fp], color="black", size=10, line_width=2, fill_color=fill_color,
    )

bokeh.io.show(p)
def g(x, beta, n):
    return x ** n - beta * x ** (n - 1) + 1

x0 = -3
x1 = 0.6
x_left = np.logspace(-4, x0, 100)
x_right = np.logspace(x1, 1, 100)
x_mid = np.logspace(x0, x1, 400)

# Build plot
p = bokeh.plotting.figure(
    frame_height=275,
    frame_width=275,
    x_axis_label="x",
    y_axis_label="g(x)",
    x_range=[1e-4, 1e1],
    y_range=[-1, 15],
    x_axis_type="log",
)

# Emphasize x axis
p.line([1e-6, 1e2], [0, 0], line_width=1, color="black")

# Plot lines
p.line(x_mid, g(x_mid, 1.5, 2), line_width=2, color="#6baed6")
p.line(
    np.append(x_mid, 7), g(np.append(x_mid, 7), 2.2, 2), line_width=2, color="#6baed6",
)
p.line(x_left, g(x_left, 1.5, 2), line_width=2, color="#08519c")
p.line(x_right, g(x_right, 1.5, 2), line_width=2, color="#08519c")

# Add labels
p.text(
    x=[1.3e-4],
    y=[1.2],
    text=["g(x) → 1 as x → 0"],
    text_align="left",
    text_baseline="bottom",
    text_font_size='8pt',
    text_color="#08519c"
)

```

```

)

p.text(
    x=[3e0],
    y=[14],
    text=[" $g(x) \rightarrow \infty$  as  $x \rightarrow \infty$ "],
    text_align="right",
    text_baseline="bottom",
    text_font_size='8pt',
    text_color="#08519c"
)

p.text(
    x=[2e0],
    y=[4],
    text=["Does  $g(x)$  cross 0?"],
    text_align="right",
    text_baseline="bottom",
    text_color="#6baed6",
    text_font_size='8pt',
)

bokeh.io.show(p)
# Get bifurcation line
n = np.linspace(1, 10, 400)
beta = n * (n - 1) ** ((1 - n) / n)

# Build the plot
p = bokeh.plotting.figure(
    height=300,
    width=400,
    x_axis_label="n",
    y_axis_label="nondimensional  $\beta$ ",
    x_range=[0, 10],
    y_range=[0, 3],
    tools=["save"],
)
p.patch(np.append(n, [n[-1], 1]), np.append(beta, [0, 0]), color="lightgray", alpha=0.7)
p.patch([0, 0, 1, 1], [0, 10, 10, 0], color="lightgray", alpha=0.7)
p.patch(np.append([n[-1], 1], n), np.append([3, 3], beta), color="gray", alpha=0.7)
p.line(n, beta, line_width=2, color="black")
p.line([1, 1], [1, 5], line_width=2, color="black")
p.text(x=4.5, y=2.2, text=["bistable"])
p.text(x=3.2, y=0.8, text=["monostable"])

bokeh.io.show(p)
# Parameters
beta = 10
gamma = 1
k = 4
n = 4

# Theoretical curves
x = np.linspace(0, 20, 400)
prod = beta * (x/k) ** n / (1 + (x/k) ** n)
removal = gamma * x

# Fixed points
fp_rhs = lambda x: beta / gamma * (x/k) ** n / (1 + (x/k) ** n)
fixed_points = [
    0,
    float(scipy.optimize.fixed_point(fp_rhs, 3)),
    float(scipy.optimize.fixed_point(fp_rhs, 5)),
]
fp3 = fixed_points[2]

# Build plot
p1 = bokeh.plotting.figure(
    frame_height=150,

```

```

    frame_width=375,
    x_axis_label="x",
    y_axis_label="production or removal rate",
    title=f"Initial environment.  $\beta = \{\text{beta}\}$ ,  $\gamma = \{\text{gamma}\}$ ,  $k = \{k\}$ ,  $n = \{n\}$ ",
    x_range=[-1, 20],
    y_range=[-1.5, 15],
)
p3 = bokeh.plotting.figure(
    frame_height=150,
    frame_width=375,
    x_axis_label="x",
    y_axis_label="production or removal rate",
    title=f"Return to initial environment.  $\beta = \{\text{beta}\}$ ,  $\gamma = \{\text{gamma}\}$ ,  $k = \{k\}$ ,  $n = \{n\}$ ",
    x_range=[-1, 20],
    y_range=[-1.5, 15],
)

# Plot production and removal rates
p1.line(x, prod, line_width=2, color="#1f77b4")
p1.line(x, removal, line_width=2, color="orange")
p3.line(x, prod, line_width=2, color="#1f77b4")
p3.line(x, removal, line_width=2, color="orange")

# Plot fixed points
for i, fp in enumerate(fixed_points):
    fill_color = "white" if i % 2 else "black"
    p1.circle(
        [fp], [gamma * fp], color="black", size=10, line_width=2, fill_color=fill_color,
    )
    p3.circle(
        [fp], [gamma * fp], color="black", size=10, line_width=2, fill_color=fill_color,
    )

# Recalculate plot for disturbance condition
gamma = 2.5 # new value

prod = beta * (x/k) ** n / (1 + (x/k) ** n)
removal = gamma * x

# Build plot
p2 = bokeh.plotting.figure(
    frame_height=150,
    frame_width=375,
    x_axis_label="x",
    y_axis_label="production or removal rate",
    title=f"Perturbed environment.  $\beta = \{\text{beta}\}$ ,  $\gamma = \{\text{gamma}\}$ ,  $k = \{k\}$ ,  $n = \{n\}$ ",
    x_range=[-1, 20],
    y_range=[-1.5, 15],
)

# Plot production and removal rates
p2.line(x, prod, line_width=2, color="#1f77b4")
p2.line(x, removal, line_width=2, color="orange")

# Plot fixed points
p2.circle(
    [0], [0], color="black", size=10, line_width=2, fill_color="black",
)

# Add commentary
p1.circle(
    [fp3], [fp3], color="gray", size=25, line_width=3, fill_color=None,
)
p1.text(
    x=[10],
    y=[7],
    text=["system is here"],
    text_color="gray",
    text_font_size="10pt",

```

```

        text_align="left",
        angle=0,
    )
    p2.add_layout(
        bokeh.models.Arrow(
            end=bokeh.models.VeeHead(size=15, fill_color="gray", line_color="gray"),
            line_width=4,
            x_start=fp3,
            y_start=fp3 - 0.5,
            x_end=1,
            y_end=1,
            line_color="gray",
        )
    )
    p2.text(
        x=[5.5],
        y=[3.5],
        text=["system moves here"],
        text_color="gray",
        text_font_size="10pt",
        text_align="left",
        angle=0,
    )
    p2.circle(
        [0], [0], color="gray", size=25, line_width=3, fill_color=None,
    )
    p3.text(
        x=[1],
        y=[-1.5],
        text=["system remains here"],
        text_color="gray",
        text_font_size="10pt",
        text_align="left",
        angle=0,
    )
    p3.circle(
        [0], [0], color="gray", size=25, line_width=3, fill_color=None,
    )

# Build layout
hysteresis_layout = bokeh.layouts.gridplot(
    [
        p1,
        bokeh.layouts.Spacer(height=10),
        p2,
        bokeh.layouts.Spacer(height=10),
        p3,
        bokeh.layouts.Spacer(height=10),
    ],
    ncols=1,
)

bokeh.io.show(hysteresis_layout)

```