“JOURNAL OF SCIENCE-INNOVATIVE RESEARCH IN UZBEKISTAN” JURNALI
VOLUME 3, ISSUE 03, 2025. MARCH
ResearchBib Impact Factor: 9.654/2024 ISSN 2992-8869

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ФИЗИЧЕСКИХ ПРОЦЕССОВ НА ЯЗЫКЕ PYTHON

D.A. Юсупова[1], С.З.Сироджиддинова[2], Ш. Акбарова[3]
Ферганский государственный университет, Фергана
[1.]Кандидат физико-математических наук, доцент кафедры физики,
[2] докторант кафедры физики, [3] студентка бакалавриата по направлению «Физика»

Аннотация. В данной статье рассматривается использование компьютерных технологий, в частности языка программирования Python, в преподавании физики в школе и вузах. Анализируются преимущества применения Python для моделирования физических процессов, математических расчетов и визуализации данных. Особое внимание уделяется библиотекам NumPy, SciPy, Matplotlib и SymPy, которые позволяют выполнять сложные вычисления и строить графики. Представлены примеры численного моделирования механических и электромагнитных явлений, а также анализа термодинамических процессов, что способствует углубленному изучению физических законов.

Обсуждаются образовательные выгоды цифровых инструментов, примеры их использования в школьной программе и перспективы интеграции программирования в физическое образование. Рассмотрены методические аспекты внедрения Python в учебный процесс, включая поэтапное обучение программированию, использование интерактивных сред разработки и проектной деятельности.

Применение предложенной методики в образовательном процессе показало, что использование Python значительно повышает эффективность преподавания физики, способствует развитию аналитического мышления и цифровых компетенций учащихся. Представленные примеры могут быть использованы для совершенствования учебных программ и подготовки методических материалов для преподавателей.

Ключевые слова: Python, преподавание физики, численное моделирование, механические колебания, движение в поле тяжести,

электромагнитные волны, заряд и разряд конденсатора, SciPy, Matplotlib, NumPy, SymPy, цифровые лаборатории, образовательное программирование, визуализация данных.

Введение

Современное школьное образование требует внедрения инновационных методик преподавания, которые помогают улучшить понимание сложных концепций и повышают интерес учеников к предмету. Одним из наиболее перспективных инструментов является язык программирования Python, который активно используется в образовательном процессе для изучения физики.

Благодаря простоте синтаксиса и мощному функционалу Python позволяет создавать интерактивные симуляции, моделировать физические явления и автоматизировать расчеты. Эти возможности делают обучение более наглядным, способствуют развитию аналитического мышления и формированию навыков работы с цифровыми инструментами.

Использование компьютерных технологий в преподавании физики особенно актуально в условиях ограниченного доступа к лабораторному оборудованию. Виртуальные эксперименты и численное моделирование помогают компенсировать этот недостаток, позволяя учащимся исследовать сложные явления, такие как механические колебания, электромагнитные процессы и термодинамические изменения. Таким образом, внедрение Python в школьный курс физики открывает новые горизонты для эффективного и современного обучения.

Методика

Использование языка программирования Python для моделирования физических процессов в образовательной практике требует тщательной методической проработки. В данной работе предлагается подход, основанный на поэтапном введении программирования в курс физики: на первых этапах учащиеся знакомятся с основами синтаксиса Python и его базовыми возможностями для вычислений.

Введение библиотеки NumPy для работы с массивами и математическими операциями позволяет автоматизировать расчёты, облегчая анализ физических задач. Применение Jupyter Notebook или Google Colab для написания и исполнения кода позволяет учащимся сразу видеть результаты

работы. Это способствует наглядному изучению физических законов и математических зависимостей.

Кроме того, изучение новых тем сопровождается практическими задачами, где учащиеся сами программируют модели физических явлений. Например, при изучении динамики можно предложить построить симуляцию движения тела в поле тяжести или моделировать колебания маятника. Использование готовых библиотек таких как, например SciPy для численного решения дифференциальных уравнений, Matplotlib для построения графиков и SymPy для символьных вычислений позволяет значительно упростить моделирование сложных процессов. Использование инструментов визуализации помогает учащимся интерпретировать полученные результаты, делая обучение более осмысленным.

Таким образом, методика интеграции Python в процесс обучения физике, рассматриваемая в данной статье, позволяет сделать предмет более доступным и интересным для учащихся. Она способствует развитию цифровых компетенций, критического мышления и навыков программирования, которые востребованы в современной науке и инженерии. Примеры, приведенные в статье, демонстрируют, как моделирование физических явлений с помощью Python может повысить качество обучения и облегчить понимание сложных концепций.

Результаты

Применение данной методики в образовательном процессе показало, что использование Python значительно повышает эффективность преподавания физики. В ходе эксперимента учащиеся, использовавшие программирование для моделирования физических процессов, продемонстрировали более глубокое понимание теоретических основ и улучшенные навыки анализа данных.

Результаты моделирования подтвердили ожидаемые закономерности физических явлений, что позволило учащимся не только закрепить знания, но и на практике исследовать влияние различных параметров на динамику систем. Например, при изучении колебаний маятника учащиеся смогли визуально оценить отличие линейных и нелинейных колебаний, а при моделировании движения тела в поле тяжести – влияние угла броска на

“JOURNAL OF SCIENCE-INNOVATIVE RESEARCH IN UZBEKISTAN” JURNALI
VOLUME 3, ISSUE 03, 2025. MARCH
ResearchBib Impact Factor: 9.654/2024 ISSN 2992-8869

траекторию. Ниже приведем несколько примеров использования Python в моделировании физических процессов.

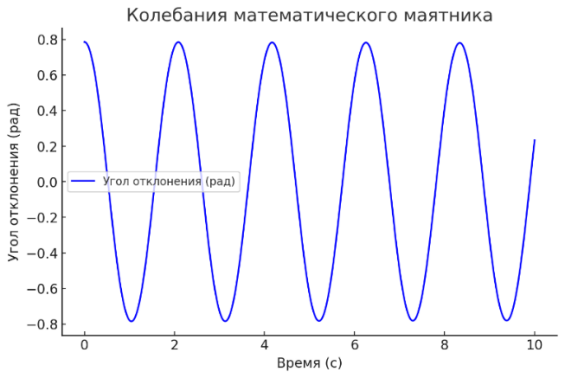
1.*Моделирование колебаний математического маятника.* Представленная программа на Python предназначена для симуляции колебаний математического маятника.

Математический маятник представляет собой систему, в которой масса подвешена на невесомой и нерастяжимой нити. Движение маятника описывается дифференциальным уравнением:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin(\theta) = 0 \quad ,$$

где: θ — угол отклонения маятника от вертикали, g — ускорение свободного падения (9.81 м/с²), L — длина маятника.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
# Параметры маятника
L = 1.0  # Длина маятника (м)
g = 9.81  # Ускорение свободного падения (м/с^2)
def pendulum_equations(t, y):
    theta, omega = y  # Угол и угловая скорость
    dydt = [omega, - (g / L) * np.sin(theta)]
    return dydt
# Начальные условия
theta0 = np.pi / 4  # Начальный угол (45 градусов)
omega0 = 0  # Начальная угловая скорость
```



```
t_span = (0, 10)  # Временной интервал (с)
t_eval = np.linspace(t_span[0], t_span[1], 1000)  # Дискретизация времени
# Решение дифференциального уравнения
sol = solve_ivp(pendulum_equations, t_span, [theta0, omega0], t_eval=t_eval, method='RK45')
# Визуализация результатов
plt.figure(figsize=(8, 5))
plt.plot(sol.t, sol.y[0], label='Угол отклонения (рад)', color='blue')
plt.xlabel('Время (с)')
plt.ylabel('Угол отклонения (рад)')
plt.title('Колебания математического маятника')
plt.legend()
plt.grid()
plt.show()
```

На графике представлена зависимость угла отклонения маятника от времени: Движение является гармоническим при малых углах (θ<15∘). При больших отклонениях маятник испытывает нелинейные колебания, отличающиеся от синусоидальной формы. Этот результат подтверждает классическую модель маятника и может быть использован для дальнейшего изучения динамики механических систем.

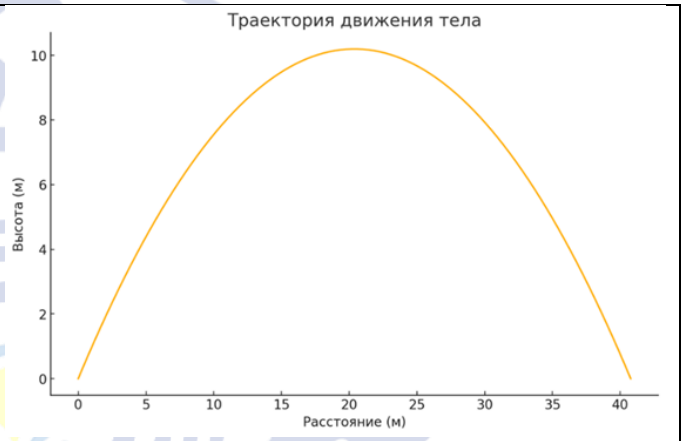
2."Моделирование движения тела в поле тяжести" или "Траектория движения снаряда, брошенного под углом к горизонту". Эта программа

рассчитывает и визуализирует траекторию движения объекта, брошенного под углом. График показывает траекторию движения тела, брошенного под углом 45 градусов с начальной скоростью 20 м/с. Движение представляет собой параболу, что соответствует классическому закону движения снаряда:

$$x = v_0 \cos(\theta)t = 0,$$

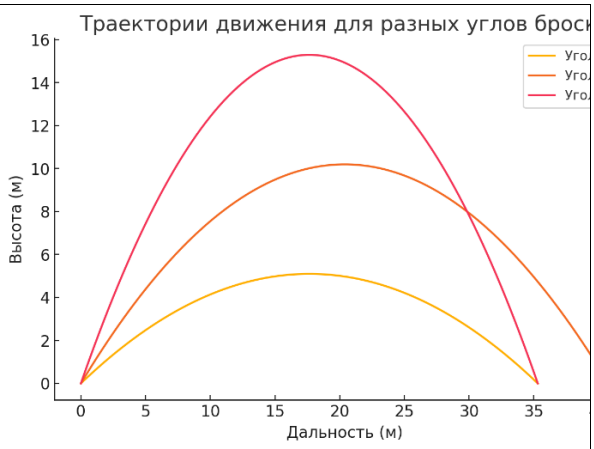
$$y = v_0 \sin(\theta)t - \frac{1}{2}gt^2$$

```
# Импорт библиотек
g = 9.81 # Ускорение свободного падения
v0 = 20 # Начальная скорость
theta = np.radians(45) # Угол (в радианах)
# Временная шкала
t = np.linspace(0, 2 * v0 * np.sin(theta) / g, num=100)
# Координаты
x = v0 * np.cos(theta) * t
y = v0 * np.sin(theta) * t - 0.5 * g * t**2
# График траектории
plt.plot(x, y)
plt.xlabel("Расстояние (м)")
plt.ylabel("Высота (м)")
plt.title("Траектория движения тела")
plt.grid()
plt.show()
```



Для моделирования траектории для разных углов броска, например, **30°, 45°, 60°** на одном графике, достаточно **изменить код** и **добавить несколько значений угла**.

```
import numpy as np
import matplotlib.pyplot as plt
# Начальные параметры
v0 = 20 # Начальная скорость (м/с)
angles = [30, 45, 60] # Углы броска в градусах
g = 9.81 # Ускорение свободного падения (м/с^2)
plt.figure(figsize=(8, 5))
for theta in angles:
theta_rad = np.radians(theta) # Перевод угла в радианы
vx = v0 * np.cos(theta_rad) # Горизонтальная скорость
vy = v0 * np.sin(theta_rad) # Вертикальная скорость
# Время полета
T = 2 * vy / g # Полное время полета
t = np.linspace(0, T, num=100) # Разбиение времени
# Вычисление координат
x = vx * t
y = vy * t - 0.5 * g * t**2
# Добавление траектории на график
plt.plot(x, y, label=f'Угол {theta}°')
# Визуализация графика
plt.xlabel('Дальность (м)')
```



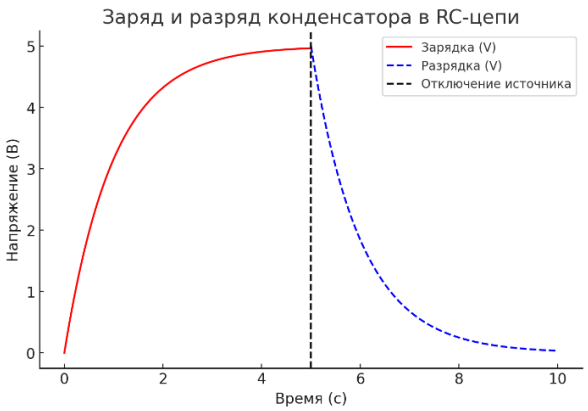
```
    plt.ylabel('Высота (м)')
    plt.title('Траектории движения для разных углов броска')
    plt.legend()
    plt.grid()
    plt.show()
```

Горизонтальная составляющая скорости остается постоянной на всём пути, а вертикальная составляющая сначала убывает из-за гравитации, достигая нуля в верхней точке, а затем увеличивается в противоположном направлении. Горизонтальное перемещение определяется, так как нет внешних сил, влияющих на него. Вертикальное перемещение определяется с учетом ускорения свободного падения, что приводит к квадратичной зависимости высоты от времени. В результате траектория представляет собой параболу. Этот метод позволяет моделировать баллистику и является основой для изучения движения тел в гравитационном поле.

3.*Моделирование заряда и разряда конденсатора в RC-цепи*

Ниже представлена программа на Python для симуляции заряда и разряда конденсатора в RC-цепи и визуализации изменения напряжения во времени

```
    import numpy as np
    import matplotlib.pyplot as plt
    # Параметры цепи
    R = 1000  # Сопротивление (Ом)
    C = 0.001  # Емкость (Фарады)
    V0 = 5  # Напряжение источника (Вольты)
    # Функции для расчета заряда и разряда конденсатора
    def capacitor_charging(t, R, C, V0):
```



```
        return V0 * (1 - np.exp(-t / (R * C)))
    def capacitor_discharging(t, R, C, V0):
        return V0 * np.exp(-t / (R * C))
    # Временной диапазон
    t_charge = np.linspace(0, 5 * R * C, 500)  # Зарядка конденсатора до 5τ
    t_discharge = np.linspace(0, 5 * R * C, 500)  # Разрядка после отключения источника
    V_charge = capacitor_charging(t_charge, R, C, V0)
    V_discharge= capacitor_discharging(t_discharge, R, C, V0)
    # Визуализация заряда и разряда конденсатора
    plt.figure(figsize=(8, 5))
    plt.plot(t_charge, V_charge, label='Зарядка (V)', color='red')
    plt.plot(t_discharge + t_charge[-1], V_discharge, label='Разрядка (V)', color='blue', linestyle='dashed')
    plt.axvline(x=t_charge[-1], color='black', linestyle='--', label='Отключение источника')
    plt.xlabel('Время (c)')
    plt.ylabel('Напряжение (В)')
```

```
    plt.title('Заряд и разряд конденсатора в RC-цепи')
    plt.legend()
    plt.grid()
    plt.show()
```

График показывает зарядку и разрядку конденсатора во времени в RC-цепи. Зарядка конденсатора: когда конденсатор подключен к источнику питания, напряжение на нем экспоненциально возрастает, приближаясь к V_0 = 5В. Процесс описывается уравнением:

$$V(t) = V_0(1 - e^{-\frac{t}{RC}})$$

Разрядка конденсатора: после отключения источника напряжение на конденсаторе экспоненциально уменьшается. Разряд описывается уравнением:

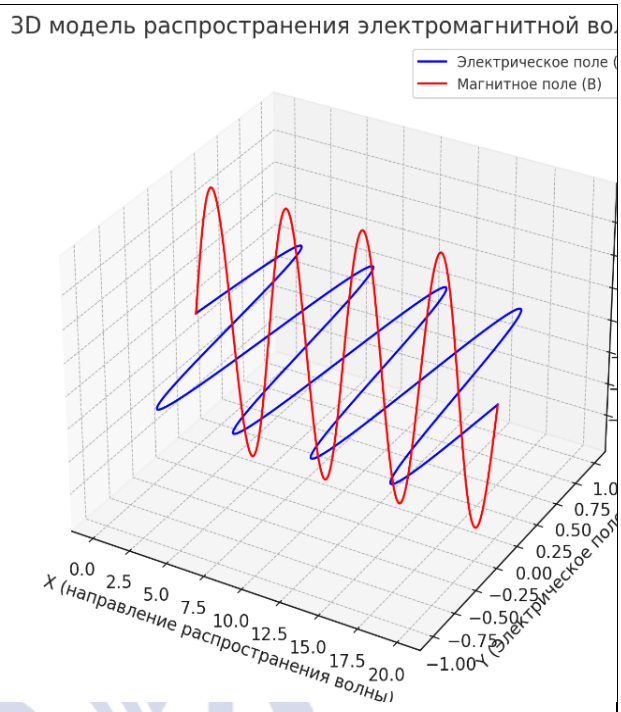
$$V(t) = V_0 e^{-\frac{t}{RC}}$$

Черная пунктирная линия на графике обозначает момент отключения источника питания, после чего начинается процесс разряда.

4. *3D модель распространения электромагнитной волны с взаимно перпендикулярными полями.*

В данной программе моделируется распространение плоской электромагнитной волны, где электрическое и магнитное поля колеблются взаимно перпендикулярно друг другу и направлению распространения.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
# Параметры волны
wavelength = 2  # Длина волны
k = 2 * np.pi / wavelength  # Волновое число
omega = 2 * np.pi  # Угловая частота
t = 0  # Начальный момент времени
# Создание сетки значений
x = np.linspace(0, 10, 500)  # Распространение вдоль оси X
E = np.sin(k * x - omega * t)  # Электрическое поле (вдоль оси Y)
B = np.sin(k * x - omega * t)  # Магнитное поле (вдоль оси Z)
# 3D визуализация
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
# Построение электрического и магнитного полей
ax.plot(x, E, np.zeros_like(x),
```



```
label='Электрическое поле (E)', color='blue')
        ax.plot(x, np.zeros_like(x), B, label='Магнитное поле (B)', color='red')
        # Настройки осей
        ax.set_xlabel('Распространение волны (X)')
        ax.set_ylabel('Электрическое поле (E)')
        ax.set_zlabel('Магнитное поле (B)')
        ax.set_title('3D модель распространения электромагнитной волны')
        # Добавление легенды
        ax.legend()
        plt.show()
```

Полученная 3D-модель показывает взаимно перпендикулярные колебания электрического (E) и магнитного (B) полей, распространяющиеся вдоль оси X. Электрическое поле направлено вдоль оси Y, а магнитное поле – вдоль оси Z. Оба поля изменяются синусоидально во времени и соответствуют классической модели поперечной электромагнитной волны.

Используется дискретизированная сетка значений **x** для моделирования распространения волны. Уравнения для электрического $E(x,t)=E_0\sin(kx-\omega t)$ и магнитного $B(x, t) = B_0 \sin(kx - \omega t)$ полей рассчитываются в каждой точке.

Электрическое поле визуализируется вдоль оси Y (синяя линия), а магнитное поле строится вдоль оси Z (красная линия). Оба поля изменяются синусоидально и остаются Данный результат подтверждает модель Максвелла и описывает поведение света и радиоволн в вакууме. Эта модель позволяет анализировать электромагнитное излучение и может быть расширена для изучения более сложных эффектов, таких как поляризация и интерференция волн.

Таким образом, предложенная методика не только улучшает качество обучения, но и способствует развитию исследовательских навыков, обеспечивая возможность применения численного моделирования в решении физических задач. Представленные примеры, такие как моделирование колебаний маятника, движения тела в поле тяжести, заряда и разряда конденсатора, а также распространения электромагнитных волн, позволяют учащимся самостоятельно анализировать физические явления, проводить численные эксперименты и визуализировать результаты. Благодаря этому учащиеся могут не только изучать теоретические аспекты, но и проводить вычислительные эксперименты, сравнивать результаты моделирования с теоретическими предсказаниями, а также исследовать влияние различных параметров на физические процессы. Например, при моделировании колебаний маятника можно анализировать влияние длины подвеса на период колебаний, а в задаче о движении тела в поле тяжести — оценивать траекторию при разных начальных скоростях и углах броска. Включение подобных примеров в образовательный процесс способствует формированию у учащихся навыков анализа данных, построения математических моделей и их программной реализации. Это делает методику перспективной для дальнейшего внедрения в школьное и вузовское образование, особенно в курсах, ориентированных на междисциплинарный подход и цифровые технологии.

Обсуждение

Анализ показал, что применение Python в преподавании физики способствует улучшению понимания сложных физических явлений за счет визуализации и моделирования, повышению интереса учащихся благодаря интерактивному взаимодействию с кодом и данными, развитию вычислительных навыков и программирования, что становится все более востребованным в современных научных и инженерных дисциплинах, возможности проведения лабораторных работ без необходимости дорогостоящего оборудования. Кроме того, внедрение Python в школьный курс физики позволило сократить время на проведение расчетов и анализа данных, а также способствовало развитию у учащихся навыков работы с цифровыми инструментами, необходимыми для решения прикладных задач.

Результаты исследования подтверждают, что использование компьютерных технологий, в частности Python, значительно повышает эффективность преподавания физики в школе. Виртуальные лаборатории и моделирование физических процессов позволяют учащимся проводить эксперименты, которые в традиционных условиях были бы сложны или невозможны. Программирование становится не только инструментом расчетов, но и средством для изучения реальных физических явлений.

Тем не менее, существуют определенные вызовы, связанные с внедрением Python в образовательный процесс. Одним из них является необходимость подготовки учителей, так как не все преподаватели физики обладают достаточными навыками программирования. Решением данной проблемы может стать организация курсов повышения квалификации, а также использование готовых учебных ресурсов и скриптов для работы с Python.

Заключение

Использование компьютерных технологий, таких как Python, в преподавании физики в школе открывает новые возможности для изучения предмета. Программирование позволяет не только автоматизировать расчеты, но и моделировать физические явления, создавая интерактивные и наглядные учебные материалы. Это способствует развитию у учащихся аналитического мышления, навыков работы с данными и подготовке к дальнейшему изучению технических наук. Для успешного внедрения Python в школьное образование необходимо обеспечить методическую поддержку преподавателей, а также создать доступные образовательные ресурсы, позволяющие использовать данный инструмент в учебном процессе.

Список литературы:

1. Python Software Foundation. (2023). Python Documentation. https://docs.python.org/3/
2. Oliphant, T. E. (2006). Guide to NumPy. Trelgol Publishing.
3. Virtanen, P., et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261–272.
4. Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90–95.
5. Eric Matthes. (2019). Python Crash Course: A Hands-On, Project-Based Introduction to Programming. No Starch Press.

6. Feynman, R. P., Leighton, R. B., & Sands, M. (2010). The Feynman Lectures on Physics. Basic Books.
7. Downey, A. (2016). Think Python: How to Think Like a Computer Scientist. O'Reilly Media.
8. Langtangen, H. P. (2016). A Primer on Scientific Programming with Python. Springer.
9. Beazley, D. (2009). Python Essential Reference. Addison-Wesley.
10. Geron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.