



# Development of Python-based tools for modeling the dynamics of systems based on Josephson junctions

A.R. Rahmonova, O.I. Streltsova, M.I. Zuev, I.R. Rahmonov

Meshcheryakov Laboratory of Information Technologies, JINR  
Bogoliubov Laboratory of Theoretical Physics, JINR

**Joint Institute for Nuclear Research**

Dubna 2024





# MLIT - BLTP Joint projects



## *Main directions*

*Meshcheryakov  
Laboratory of  
Information  
Technologies, JINR*

*Development of algorithms for spintronics problems  
and high-performance computing*

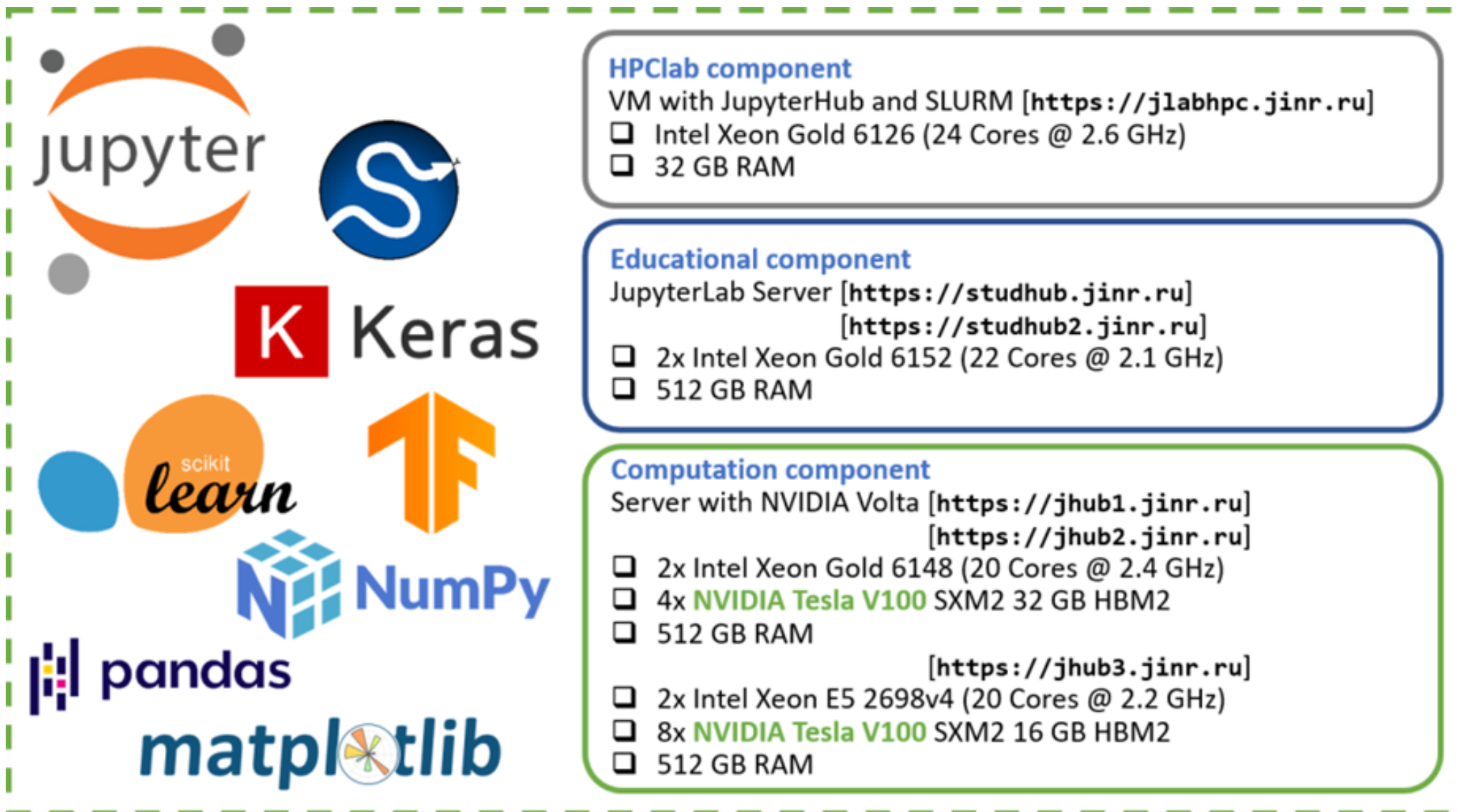
*Bogoliubov  
Laboratory of  
Theoretical Physics,  
JINR*

*Rahmonova A. R.  
Bashahin M. V.  
Balashov N. A.  
Zemlyanaya E. V.  
Zuev M. I.  
Nechaevskiy A. V.  
Streltsova O. I.  
Sokolov I. A.*

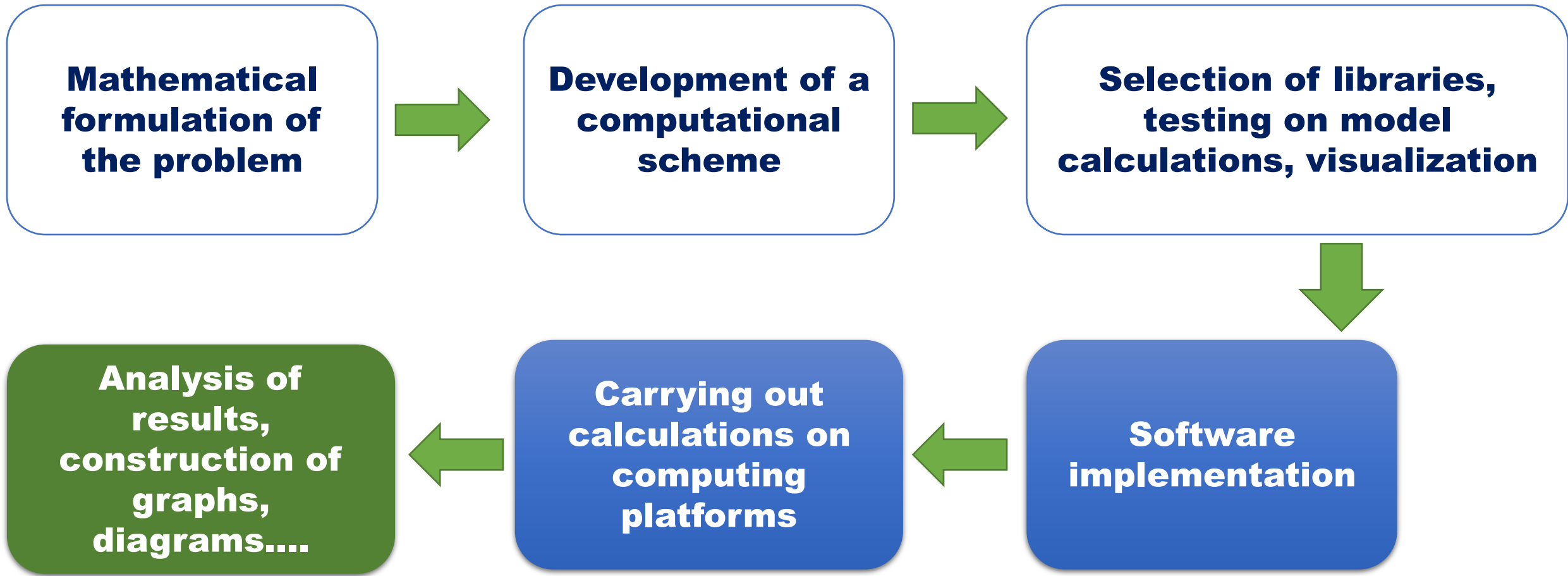
*Development of cloud services for mathematical  
modeling of a system of coupled Josephson  
junctions*

*Shukrinov Yu. M.  
Abdelgani M. A.  
Kulikov K. B.  
Mazanik A.A.  
Rahmonov I.R.*

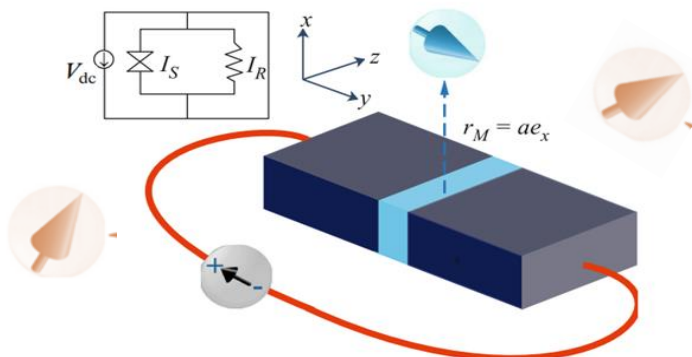
*Development of Python tools for mathematical  
modeling of Josephson structures*



# Process of numerical research



## Block of Symbolic calculations



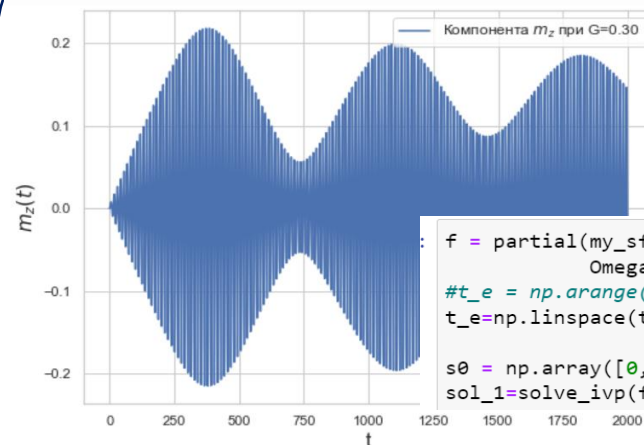
$$\gamma_{m_i} = -\frac{\mu_0}{2\Phi_0} \int d\mathbf{r}_i \frac{\mathbf{M}_i \times \mathbf{r}_i}{r^3}$$

$$B_{12}(r_{12}, m_1) = \frac{\mu_0}{4\pi} \left( \frac{3(m_1 \cdot \hat{r})\hat{r}}{b^5} - \frac{m_1}{b^3} \right)$$



**SymPy** is a Python library for symbolic mathematics.

## Block of numerical calculations and analysis



```
f = partial(my_sfs, G=G, alpha=alpha, k=k, \
            OmegaF=OmegaF, V=V)
#t_e = np.arange(0, 25, 0.0001)
t_e=np.linspace(t0,tf,100000)

s0 = np.array([0, 1, 0])
sol_1=solve_ivp(f,[t0,tf],s0, t_eval=t_e, method='RK45')
```



**SciPy** is an open-source software for mathematics, science, and engineering.

## Ускорение многопараметрических расчетов

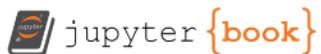


**Joblib** is a set of tools to provide lightweight pipelining in Python



**Numba** is an open source JIT compiler that translates a subset of Python and NumPy code into fast machine code.

# Various possibilities of Jupyter Book



Welcome to HLIT Jupyter Book

Основы работы с Python: инструментарий на Python для решения научных и прикладных задач

Численное решение задачи Коши: библиотека SciPy

Параллельные вычисления на Python

Задача 1: Линеаризованное уравнение на магнитный момент

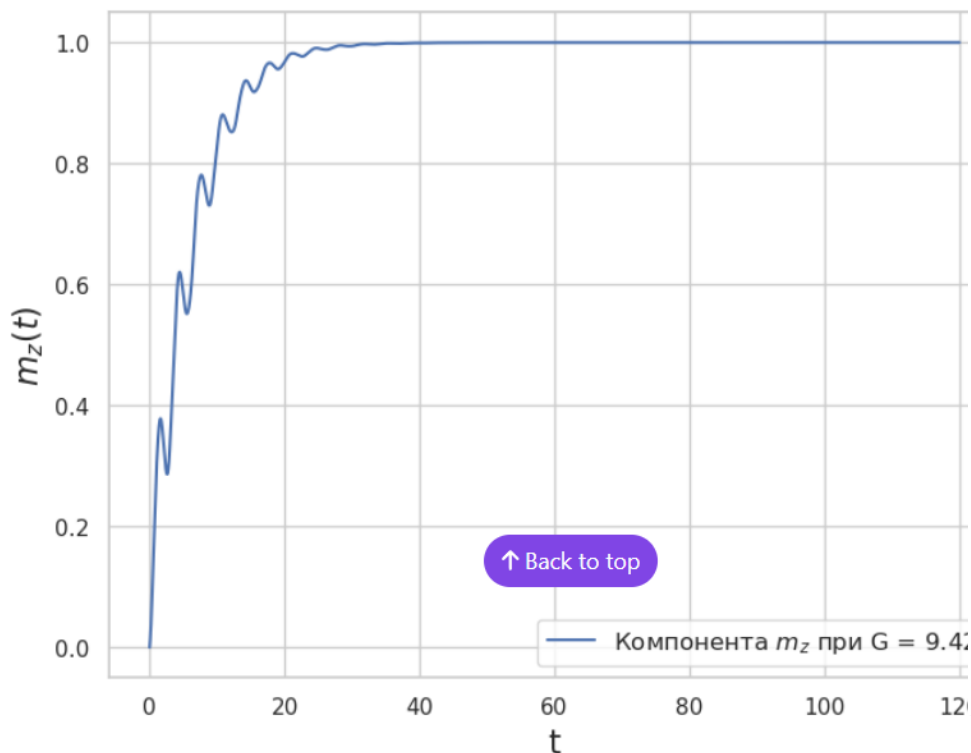
Задача 2. Периодичность появления интервалов переворота намагниченности в  $\Phi 0$  джозефсоновском переходе под воздействием импульса тока

Задача 3. Ферромагнитный резонанс и динамика магнитного момента в системе "джозефсоновский переход-наномангнит"

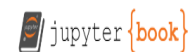
```
f = partial(my_sfs, G=G3, alpha=alpha, k=k, OmegaF=OmegaF, V=V)
t_e = np.linspace(t0, tf, nt)

s0 = np.array([0, 1, 0])
sol_G3 = solve_ivp(f, [t0, tf], s0, t_eval=t_e, method='RK45',
                  rtol=1e-8, atol=1e-9)
```

```
plt.figure(figsize=(8, 6))
plt.plot(sol_G3.t, sol_G3.y[2], label='Компонента $m_z$ при G = %4.2f' % G3)
plt.xlabel('t', size=16)
plt.ylabel('$m_z(t)$', size=16)
plt.legend(fontsize=12)
plt.show()
```



.ipynb  
.pdf



Welcome to HLIT Jupyter Book

Основы работы с Python: инструментарий на Python для решения научных и прикладных задач

Численное решение задачи Коши: библиотека SciPy

Параллельные вычисления на Python

Задача 1: Линеаризованное уравнение на магнитный момент

Задача 2. Периодичность появления интервалов переворота намагниченности в  $\Phi 0$  джозефсоновском переходе под воздействием импульса тока

Задача 3. Ферромагнитный резонанс и динамика магнитного момента в системе "джозефсоновский переход-наномангнит"

Математическая постановка задачи

Инструментарий для численного моделирования

Численное моделирование: исследование переориентации намагниченности в системе

Численное моделирование: проявление ФМР на зависимости



Contents

Пример 1: Численно решить задачу Коши:

Коши:

Пример 2: Численно решить задачу Коши:

Коши:

## Численное решение задачи Коши: библиотека SciPy

Материалы этой части основаны на книге (JBook):

Qingkai Kong, Timmy Siau, Alexandre M. Bayen. «Python Programming And Numerical Methods: A Guide For Engineers And Scientists»: <https://pythonnumericalmethods.berkeley.edu/notebooks/Index.html>

**Задача Коши:** Рассмотрим решение начальной задачи (*Initial value problem*) для системы обыкновенных дифференциальных уравнений первого порядка, разрешенных относительно производной:

$$\begin{cases} \frac{dy(t)}{dt} = f(t, y(t)), & (1) \\ y|_{t=t_0} = y_0, \end{cases}$$

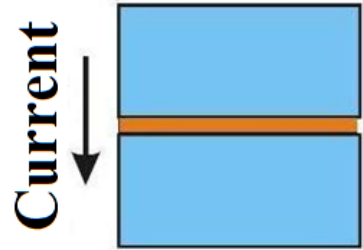
где  $y = (y_1, \dots, y_n)^T$  - вектор-функция.

### Пример 1: Численно решить задачу Коши:

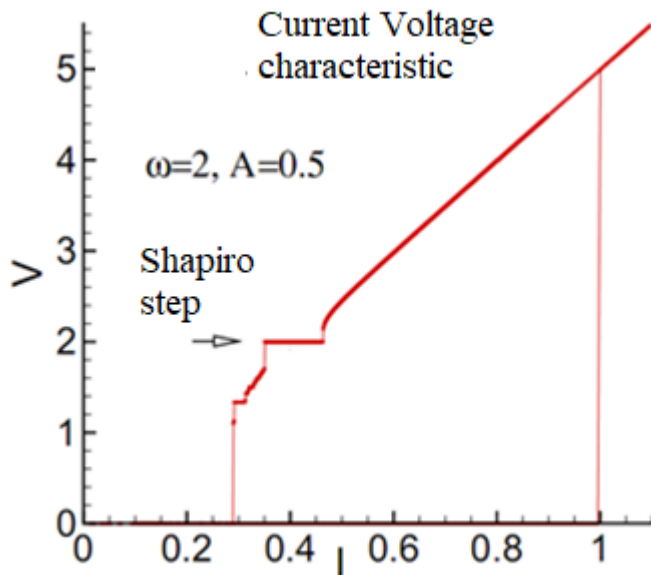
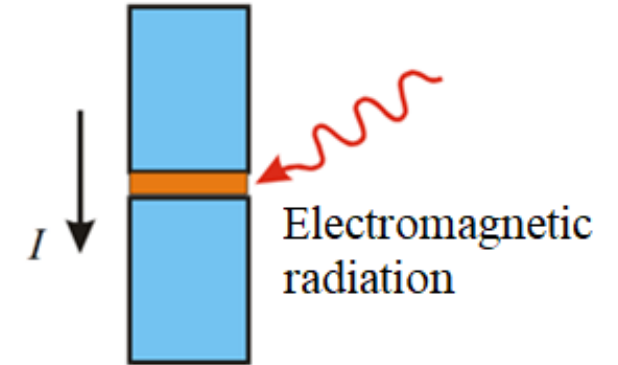
$$\begin{cases} \frac{dy}{dt} = y \cos(t), & (2) \\ y(0) = y_0. \end{cases}$$

Для сравнения приведем аналитическое решение задачи (2):

# Influence of external radiation on the dynamics of the Josephson junction

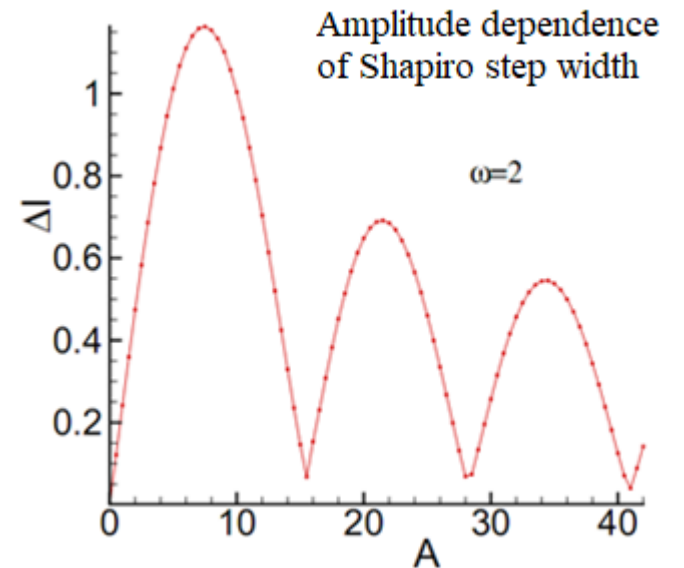


The coupling of two superconducting layers through a thin non-superconducting barrier forms a Josephson junction.



Under the influence of external radiation, provided that the Josephson frequency is a multiple of the radiation frequency ( $n \omega_J = k \omega$ ), a constant voltage step appears on the current-voltage characteristic of the Josephson junction. This step is called the Shapiro step.

The width of the Shapiro step depends on the amplitude and frequency of the radiation.



# Dynamical equations for describing of the Josephson junction under the influence of radiation

$$\begin{aligned}\frac{dV}{dt} &= I - \beta V - \sin \varphi + A \sin(u) \\ \frac{d\varphi}{dt} &= V \\ \frac{du}{dt} &= \omega\end{aligned}$$

Initial condition

$$t=0, V=0, \varphi=0, u=0$$

## Tasks

- Calculation of the current-voltage characteristic of a Josephson junction under the influence of external radiation.
- Calculation of the amplitude dependence of the width of the Shapiro step

HLIT Jupyter book

[Welcome to HLIT Jupyter Book](#)

Python-инструментарий для моделирования динамики джозефсоновского перехода под воздействием внешнего излучения

## Вычисляем ВАХ

### Задаем значения параметров для вычисления ВАХ

Отметим, что при вычислении необходимо согласовать все временных характеристики с периодом внешнего излучения во избежании накоплении ошибок при усреднении. Для этого нужно вычислить период внешнего излучения  $T = 2\pi/\omega$ . Из построенных выше графиков видно, что решения стабилизируется после  $T_{\min} = 60$  (для  $\omega = 2$ ), это соответствует примерно  $T_{\min} = 20T$  (начало интервала для усреднения). Для вычисления ВАХ если выберем временной интервал  $T_{\max} = 250$  это будет соответствовать примерно  $T_{\max} = 80T$  (максимальное значение времени) и, соответственно, шаг по времени  $\Delta t = T/50$ .

```
T = 2 * np.pi/omega # Период внешнего излучение
Tmin = 20 * T # Начало интервала для интегрирования для усреднения
Tmax = 80 * T # Максимальное значение времени
deltat = T/50 # шаг по времени
ntmin = int(Tmin/deltat)
nt = int(Tmax/deltat)

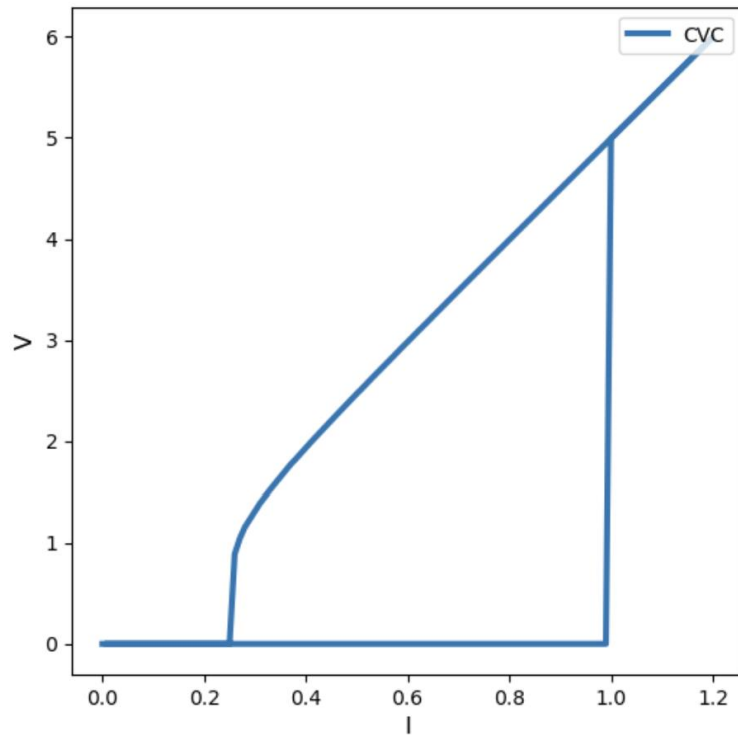
deltaIext = 0.01
Iext = 0.0
a = 1.0
Iext_max = 1.2
A = 0.5
Vplot = []
Iplot = []
s0 = np.array([0, 0, 0])
```

Введем параметр `Tlimit` ограничивающий интервал изменения по току для избежания заикливания

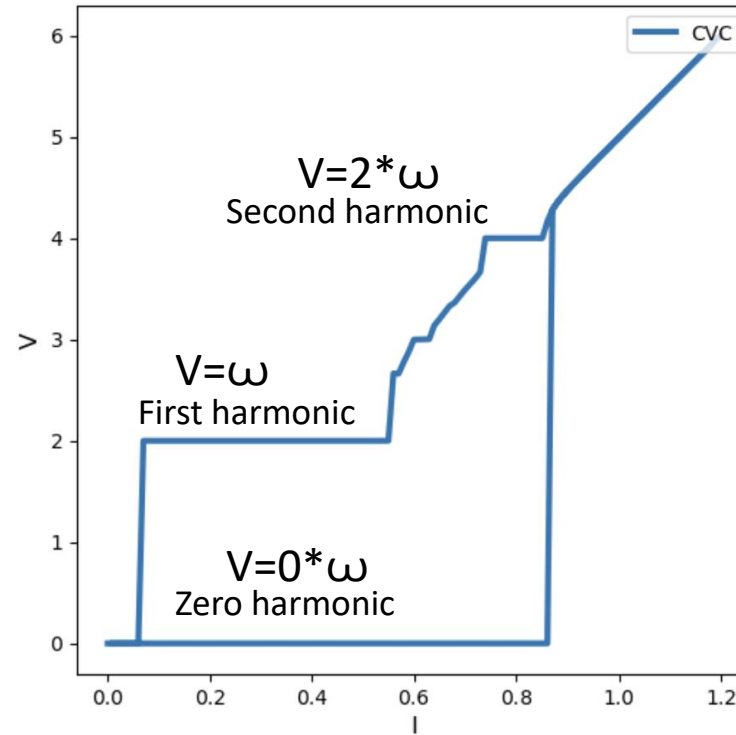
<http://studhub.jinr.ru:8080/jjbook/intro.html>



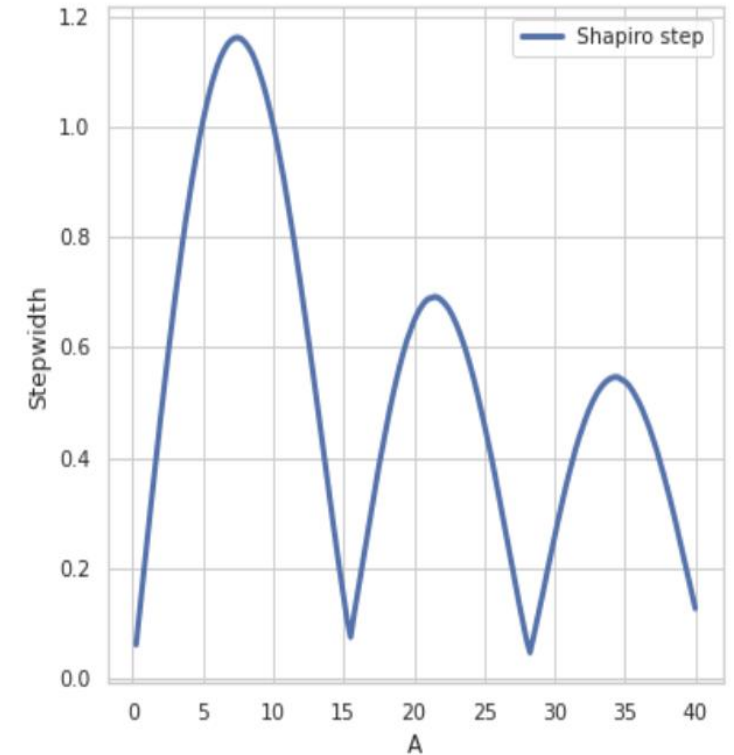
Calculated current-voltage characteristics at  $\omega=2$  and amplitude values:  $A=0, A=3$



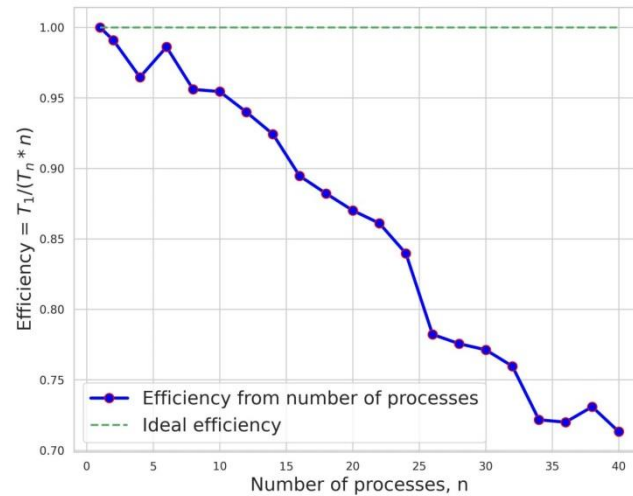
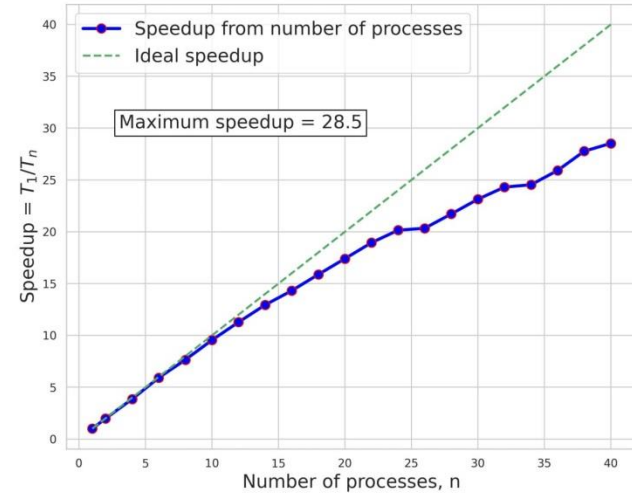
CVC without radiation



CVC with radiation at  $\omega=2$  and  $A=3$



Amplitude dependence of Shapiro step

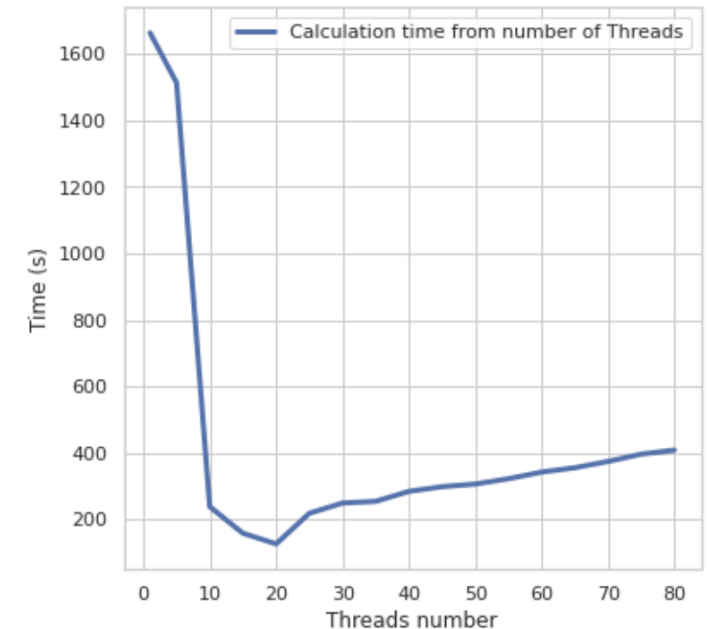


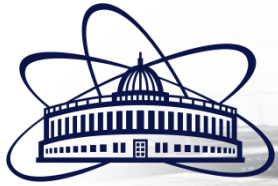
For 160 values of amplitude with stepsize  $\Delta A = 0.25$  the duration of the calculation in serial mode was **29 hours**. Parallel calculations were carried out using the **Joblib** library. An acceleration of about 28.5 times was obtained when using 40 threads and the calculation time was reduced to **1 hour**.

## Numba results

Calculations were carried out for the same values as on the previous result, i.e. for 160 amplitude values with a stepsize of  $\Delta A = 0.25$  and calculation duration in serial mode was **5 min**.


In parallel mode using **20 threads**, the calculation time was **26 seconds**. Acceleration of calculations - **11 times**. Also, if we compare calculation times using **Numba**, a **70 time** acceleration was achieved





**Thank you for your attention**

**Toolkit based on Python libraries and Jupyter ecosystems for solution scientific and applied problems**

 `jupyter {book}`

