



# Approaches to accelerating computations in Python

A.R. Rahmonova

The work was performed with the support of the Russian Science Foundation within the framework of project No. 22-71-10022

HybriLIT WORKSHOP 2025: Towards Efficient Scientific Computing

25-26 November



# Возможности Python для научных вычислений

## Ускорение многопараметрических расчетов

### Экосистема SciPY



**NumPy**-библиотека для научных вычислений в Python. Предоставляет поддержку многомерных массивов



SymPy

**SymPy** - библиотека для символьных вычислений



**Matplotlib**

библиотека для визуализации данных



**Joblib** - библиотека для Python, которая предоставляет инструменты для параллельных вычислений и эффективной обработки данных.

```
# подключение библиотеки Joblib
import joblib

from joblib import Parallel, delayed

# доступное количество CPU потоков
print(f"Number of CPU:
{joblib.cpu_count()}")

Out: Number of CPU: 40

rez =
Parallel(n_jobs=10)(delayed(funk_
parall)(k) for k in range(N * N))
```



**Numba** - JIT-компилятор с открытым исходным кодом, который транслирует подмножество кода Python и NumPy в быстрый машинный код.

*# подключение библиотеки numba*

```
import numba

from numba import njit

Директива @njit с опцией
parallel=True добавить перед
определением функции

@njit(parallel=True)

def
solution_numba_parallel(parame
ters)

...

Перед вызовом функции задается
количество потоков вычислений функцией

set_num_threads(80)
```



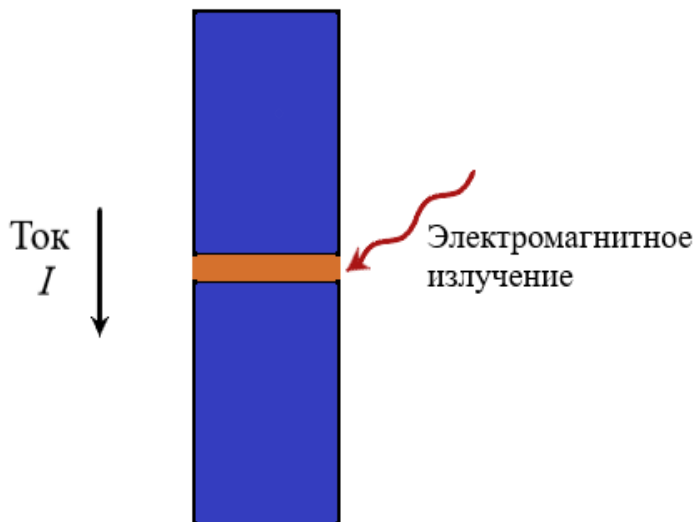
**Numba CUDA** - это инструмент для ускорения вычислений на графических процессорах с помощью платформы

*# подключение библиотеки numba cuda*

```
from numba import cuda
```

**@cuda.jit(device=True)** декоратор в Python для работы с технологией **CUDA** — это инструмент, который компилирует функции, выполняемые на графическом процессоре (GPU), с помощью библиотеки **Numba**.

# Влияние внешнего излучения на динамику джозефсоновского перехода



Под воздействием внешнего излучения, при условии **кратности частоты Джозефсона к частоте излучения** ( $n \omega_J = k \omega$ ), возникает ступенька постоянного напряжения на вольт-амперной характеристике (ВАХ) джозефсоновского перехода. Эта ступенька называется **ступенькой Шапиро**.

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

$V$  - напряжение

$I$  - внешний ток

$A$  - амплитуда излучения

$\varphi$  - разность фаз

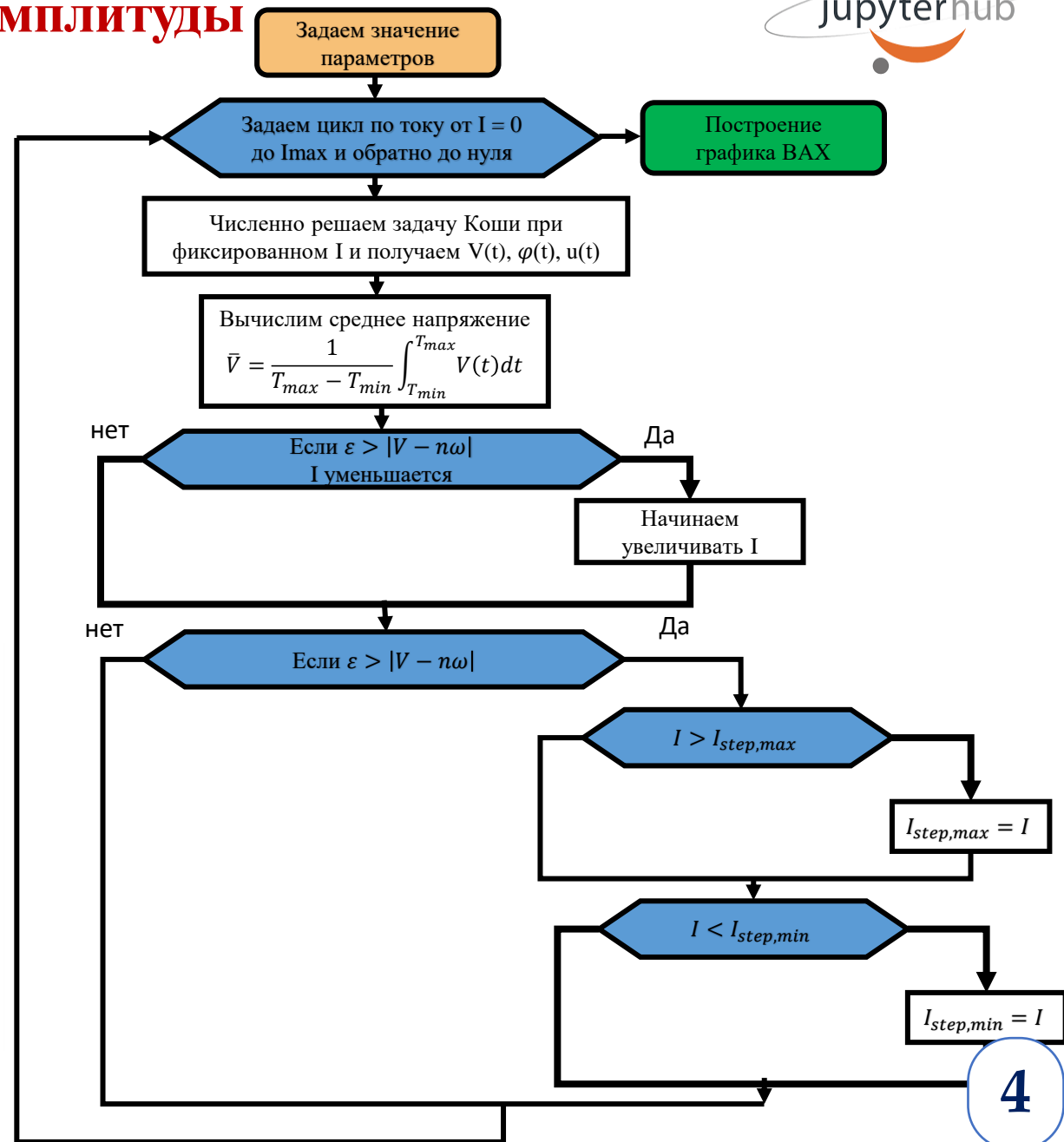
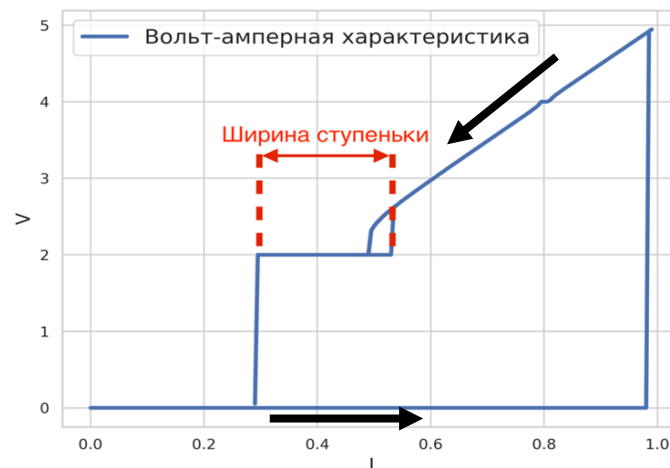
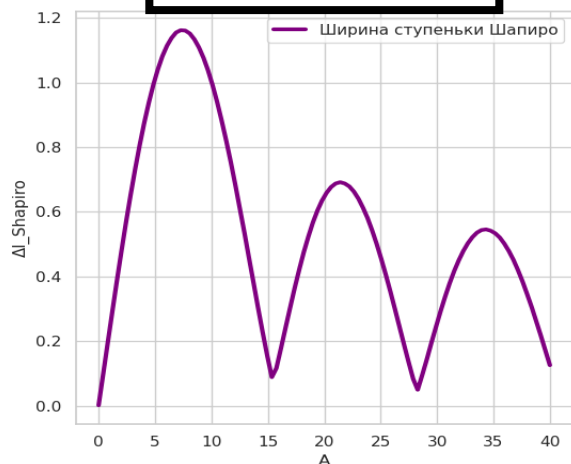
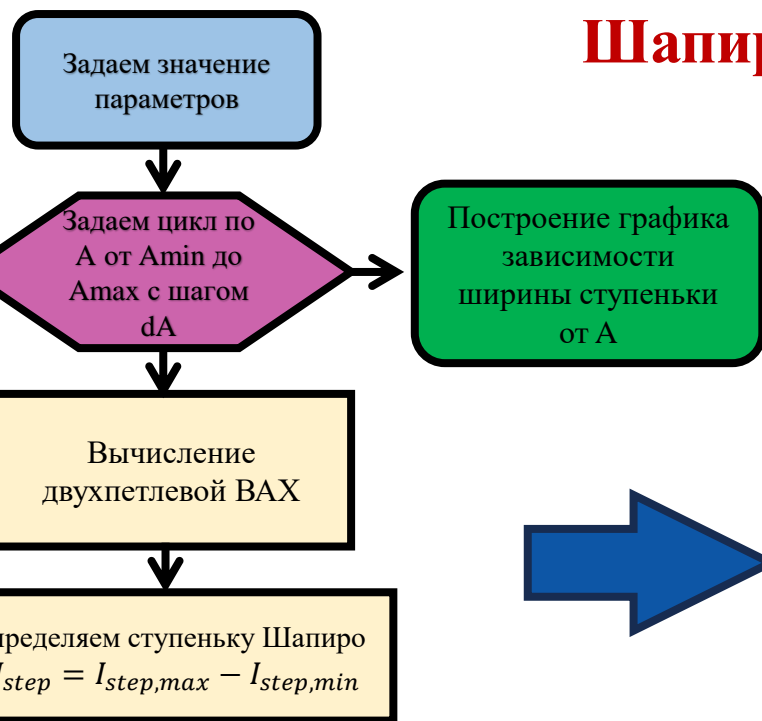
$\beta$  - параметр диссипации

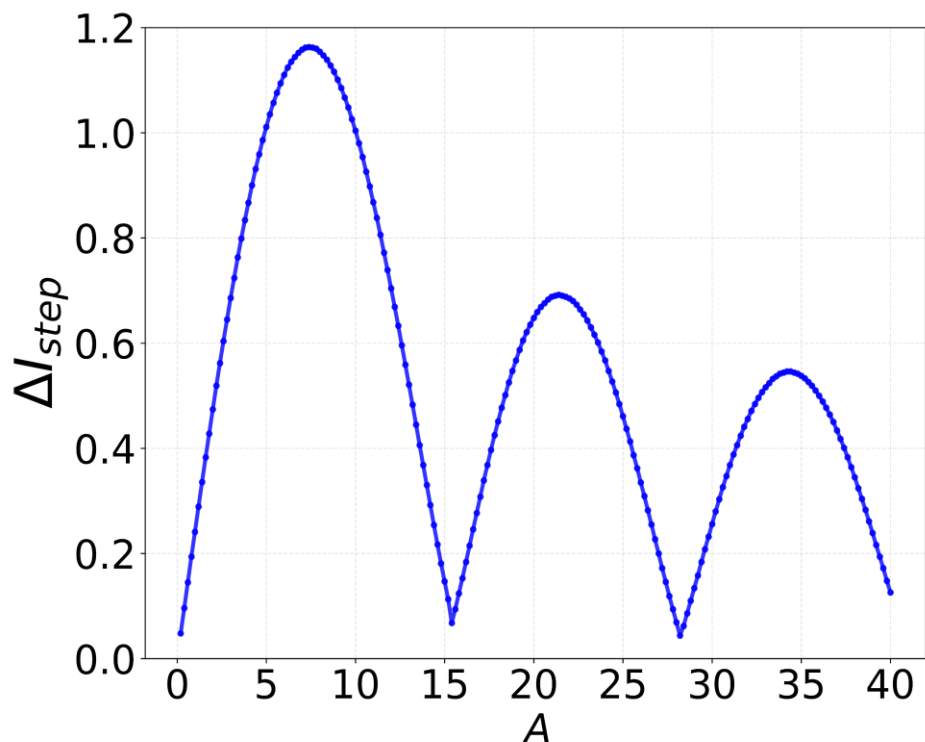
$\omega$  - частота излучения

Для вычисления вольт-амперной характеристики решается задача Коши в интервале времени  $[0, T_{\max}]$  при разных значениях тока в интервале  $[0, I_{\max}]$  с шагом  $\Delta I$  и усредняется полученная  $V$ .

При  $I=0$  задаются нулевые начальные условия:  $V = 0, \varphi = 0, u = 0$

При других значениях тока в качестве начальных условий используются значения  $V(T_{\max}), \varphi(T_{\max}), u(T_{\max})$  полученные при предыдущем шаге по току





Зависимость ширины ступеньки от амплитуды

Количество точек по амплитуде 200,  
 $A_{\min} = 0.2$ ,  $A_{\max} = 40$

**Joblib** при 50 параллельных потоков время расчета 4551 с

```
@njit(parallel=True)
def solution_numba_parallel(parameters)
    set_num_threads(1)
    время вычисления 1920 секунд

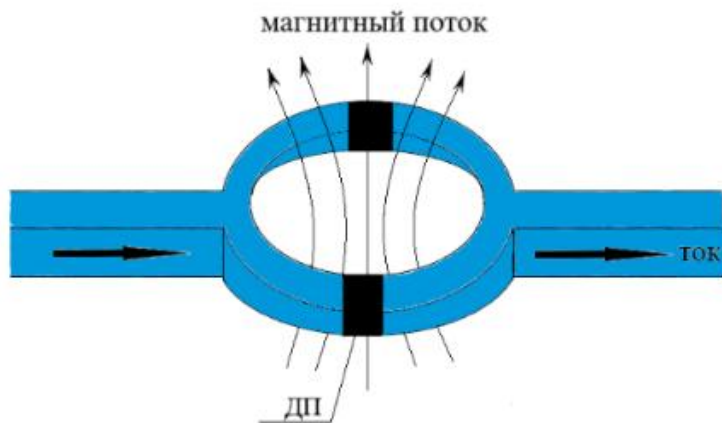
    set_num_threads(50)
    время вычисления 66 секунд
```

**Joblib** 50 потоков 4551 секунд

**Numba** 50 потоков 66 секунд

Достигнуто **ускорение в 68 раз.**

Вычисления проводились на компоненте ML/DL/HPC экосистемы гетерогенной платформы HybriLIT с характеристиками: 2x Intel Xeon Gold 6148 (2.4 ГГц, 20 ядер /40 потоков), 512 Гб оперативной памяти DDR4



Схематический вид СКВИД

Сверхпроводящий квантовый интерференционный прибор (СКВИД) представляет собой сверхпроводящее кольцо с двумя джозефсоновскими переходами. СКВИД может быть использован в качестве прибора для измерения слабых магнитных полей.

$$\begin{cases} \frac{dV_1}{dt} = \frac{1}{\beta_c} \left\{ \frac{I}{2} - \frac{d\varphi_1}{dt} - \sin(\varphi_1) + \frac{1}{2\beta_L} [\varphi_2 - \varphi_1 + 2\pi(n - \varphi_{ext})] \right\} \\ \frac{d\varphi_1}{dt} = V_1 \\ \frac{dV_2}{dt} = \frac{1}{\beta_c} \left\{ \frac{I}{2} - \frac{d\varphi_2}{dt} - \sin(\varphi_2) - \frac{1}{2\beta_L} [\varphi_2 - \varphi_1 + 2\pi(n - \varphi_{ext})] \right\} \\ \frac{d\varphi_2}{dt} = V_2 \end{cases}$$

### Искомые функции

$\varphi_1(t)$  - разность фаз первого ДП

$\varphi_2(t)$  - разность фаз второго ДП

$V_1(t)$  - напряжение на первом ДП

$V_2(t)$  - напряжение на втором ДП

### Параметры модели

$I$  - внешний электрический ток

$\beta_c$  - параметр МакКамбера

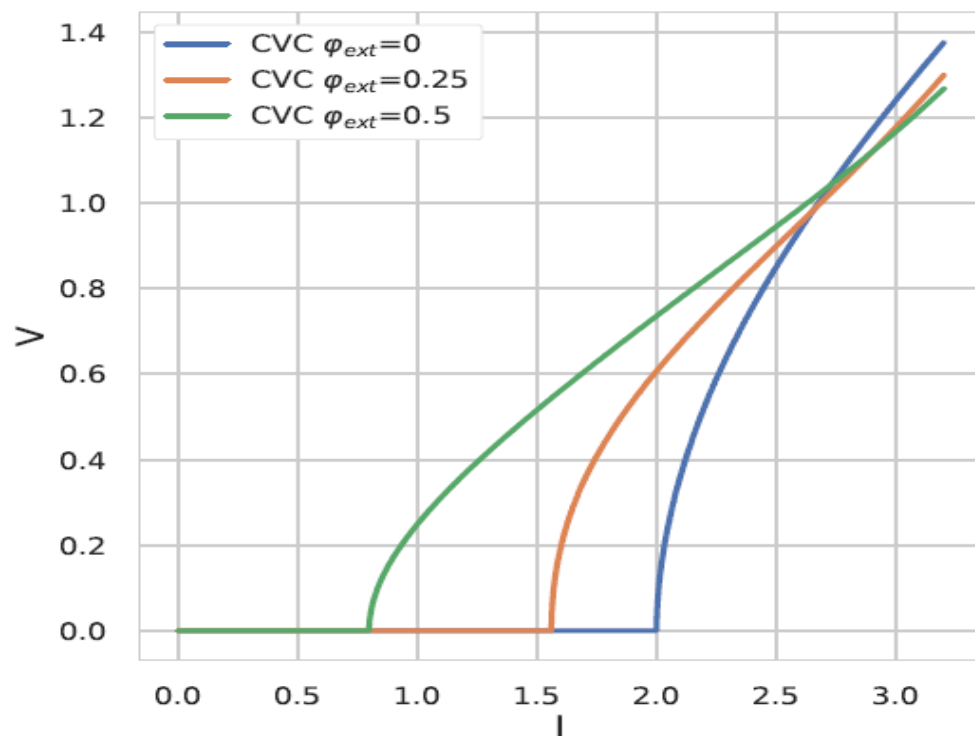
$\beta_L$  - нормированная индуктивность

$\varphi_{ext}$  - поток внешнего магнитного поля

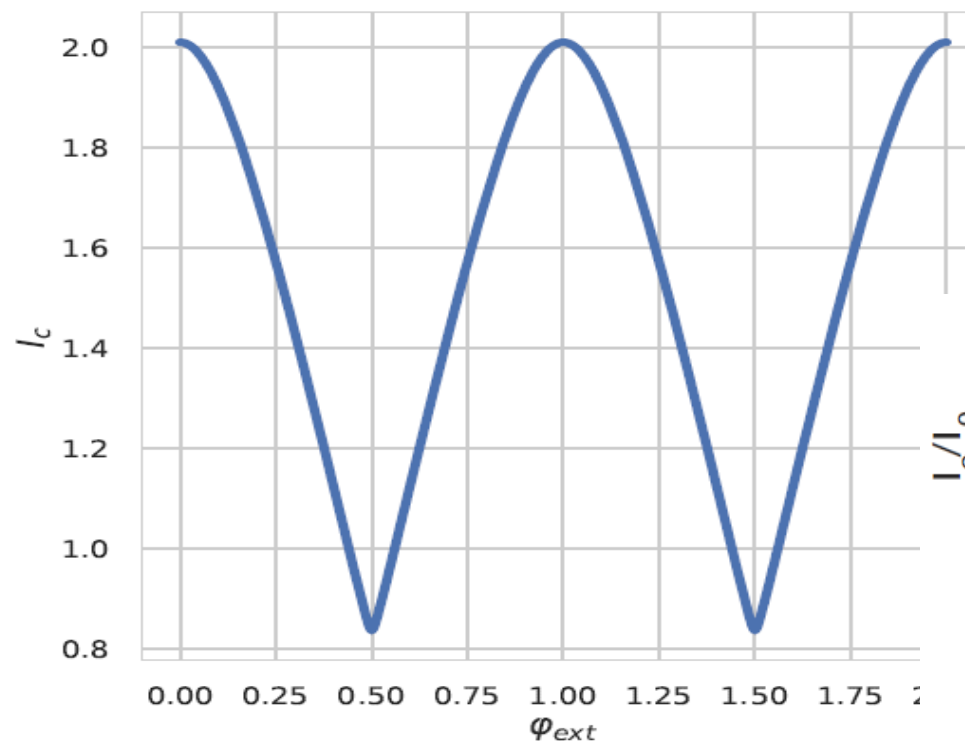
$n$  - число квантования магнитного потока

Для вычисления вольт-амперной характеристики решается задача Коши в интервале времени  $[0, T_{\max}]$  при разных значениях тока в интервале  $[0, I_{\max}]$  с шагом  $\Delta I$  и усредняются полученные  $V_1$  и  $V_2$ . При  $I=0$  задаются нулевые начальные условия:  $V_1=0$ ,  $V_2=0$ ,  $\varphi_1=0$ ,  $\varphi_2=0$ . При других значениях тока в качестве начальных условий используются значения  $V_1(T_{\max})$ ,  $V_2(T_{\max})$ ,  $\varphi_1(T_{\max})$ ,  $\varphi_2(T_{\max})$ , полученные при предыдущем шаге по току.

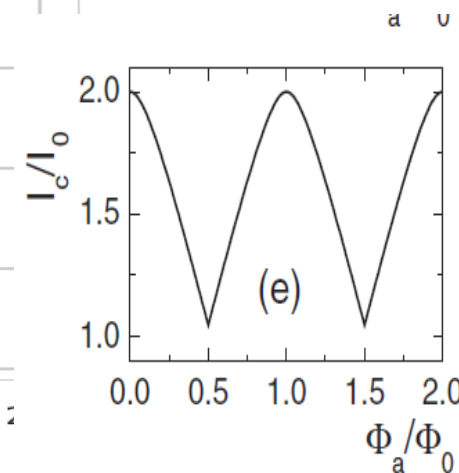




Рассчитанные ВАХ при значениях потока внешнего магнитного поля: 0, 0.25 и 0.5



Зависимость критического тока от величины потока внешнего магнитного поля



# Реализация на GPU

```
# Размер массивов
npoint = 2048

# Создаем массивы на CPU
ph_ext = np.arange(npoint, dtype=np.float64)
Ic_array = np.zeros(npoint, dtype=np.float64)

# Задаем интервал внешнего поля и определяем его шаг изменения
ph_extmin = 0
ph_extmax = 2.0
deltaph_ext = (ph_extmax - ph_extmin) / npoint

# Заполняем массив значений внешнего поля
for i in range(npoint):
    ph_ext[i] = ph_extmin + i * deltaph_ext

# Создаем массивы на GPU
ph_ext_gpu = cuda.to_device(ph_ext)
Ic_array_gpu = cuda.to_device(Ic_array)
```

```
start = time.time()

# Запускаем ядро CUDA
threads_per_block = 32

blocks_per_grid = (npoint + threads_per_block - 1) // threads_per_block
Vector_Ic_cuda[blocks_per_grid, threads_per_block](ph_ext_gpu, Ic_array_gpu)

end = time.time()
print(end - start)
```

1.3500266075134277

```
: #test block
stream = cuda.stream()
start = time.time()

# копирование
Ic_array_gpu.copy_to_host(Ic_array, stream = stream)

stream.synchronize()

end = time.time()
print(f"Copy time: {end - start:.3f} sec")
```

Copy time: 80.988 sec



2048 points

	Threads per block	Calculation time	Copy time from GPU to CPU
1	32	0.74530	80.98
2	64	0.76371	89.30
3	128	0.75718	83.060
4	256	0.63992	97.23
5	512	0.60015	127.78

4096 points

	Threads per block	Calculation time
1	32	0.476355
2	64	0.920817
3	128	0.425945
4	256	0.788543
5	512	0.876555

8192 points

	Threads per block	Calculation time
1	32	0.433546
2	64	0.386040
3	128	0.445090
4	256	0.591825
5	512	0.406400

	Points number	GPU (при оптимальном значении threads per block) сек	CPU (80 threads) сек
1	256	51	24
2	512	72	48
3	768	85	72
4	1024	84	95
5	1536	85	150
6	2048	84	199
7	3072	85	301
8	4096	84	403
9	5120	85	500
10	6144	84	595

	Points number	GPU (при оптимальном значении threads per block)сек	Numba CPU (80 threads) сек
11	7168	83	703
12	8192	82	815
13	9216	84	900
14	10240	84	1005
15	15360	93	1540
16	20480	96	1999
17	25600	109	2524
18	30720	106	3049
19	35840	126	3530
20	40960	124	4085



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-  
наноманит"



.ipynb  
.pdf

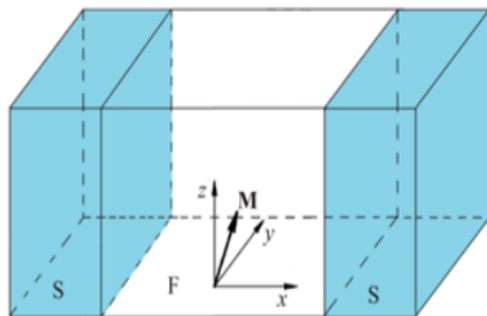
## Python-realization

Рассмотрим *Python*- реализацию программного кода, на основе которого получены результаты в статье:

[1] P. Kh. Atanasova, S.A. Panayotova, I.R. Rahmonov, Yu. M. Shukrinov, E.V. Zemlyanaya, and M. V. Bashashin.

**Periodicity in the Appearance of Intervals of the Reversal of the Magnetic Moment of a  $\phi_0$  Josephson Junction.**, JETP Letters, **2019**, Vol. 110, No. 11, pp. 722–726

Схема перехода  $\phi_0$ :



- (S) сверхпроводящие слои
- (F) ферромагнитный слой
- (M) магнитный момент ферромагнитного слоя, ось которого направлена вдоль оси  $z$ .

## Ключевые характеристики

Jupyter Book — проект с **открытым исходным кодом** для создания интерактивных и публикуемых онлайн-книг, документации и блогов на основе вычислительных материалов.

Jupyter Book поддерживает .ipynb, Markdown и reStructuredText, что позволяет объединять **текст с кодом, уравнениями, изображениями и интерактивными элементами**

Код в Jupyter Notebook можно выполнять во время создания книги, а результаты (выходные данные, графики) будут встроены в книгу.

Генерация различных типов материалов. Мы можем получить книгу в виде одностраничного или многостраничного сайта, а также **экспортировать её в PDF**.

# Jupyter Book для моделирования сверхпроводниковых/магнитных гибридных наноструктур

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

<http://studhub.jinr.ru:8080/books>

<http://studhub.jinr.ru:8080/itschool2024>

Ток-фазовое соотношение этого перехода задается выражением

$$j_s = j_c \sin(\varphi - \varphi_0), \quad (1)$$

где  $j_c$  - критический ток,  $\varphi$  - джозефсоновская разность фаз,  $\varphi_0 = \pi m_y$  - аномальный сдвиг фазы,  $m_y = M_y/M_s$  -  $y$  компонента намагниченности ( $M_y$ ) нормированная на намагниченность насыщения  $M_s$ ,  $r = 2d\hbar_{\text{ex}}/v_F$  - безразмерный параметр, определяющий величину спин-орбитального взаимодействия,  $d$  - толщина ферромагнитного барьера,  $\hbar_{\text{ex}}$  - обменное поле,  $v_F$  - скорость Ферми.

Отличительной чертой рассматриваемого джозефсоновского перехода является то, что критический ток сильно зависит от ориентации намагниченности, а именно, от  $x$  компоненты намагниченности в плоскости вдоль направления тока и задается выражением

$$j_c = j_b \int_{-\pi/2}^{\pi/2} \cos \varphi \exp\left(-\frac{\tilde{d}}{\cos \varphi}\right) \cos(\pi m_x \tan \varphi) d\varphi, \quad (2)$$

Создаем массивы для  $m_x$  и соответствующих значений интегралов

```
# массивы
Npoint = 1000
arr_mx = np.linspace(-1, 1, Npoint, endpoint=True)
arr_Ix = np.zeros(Npoint, dtype=np.float64)
arr_Iy = np.zeros(Npoint, dtype=np.float64)
arr_errx = np.zeros(Npoint, dtype=np.float64)
arr_erry = np.zeros(Npoint, dtype=np.float64)
# вычисление интегралов
for ind in range(Npoint):
    mx = arr_mx[ind]
    arr_Ix[ind], arr_errx[ind] = quad(func_Ix, -np.pi/2, np.pi/2,
                                     args=(mx, r, d))
    arr_Iy[ind], arr_erry[ind] = quad(func_Iy, -np.pi/2, np.pi/2,
                                     args=(mx, r, d))
```

Задаем начальные условия

```
mx0 = -0.5
mz0 = 0
my0 = np.sqrt(1-mx0*mx0-mz0*mz0)
```

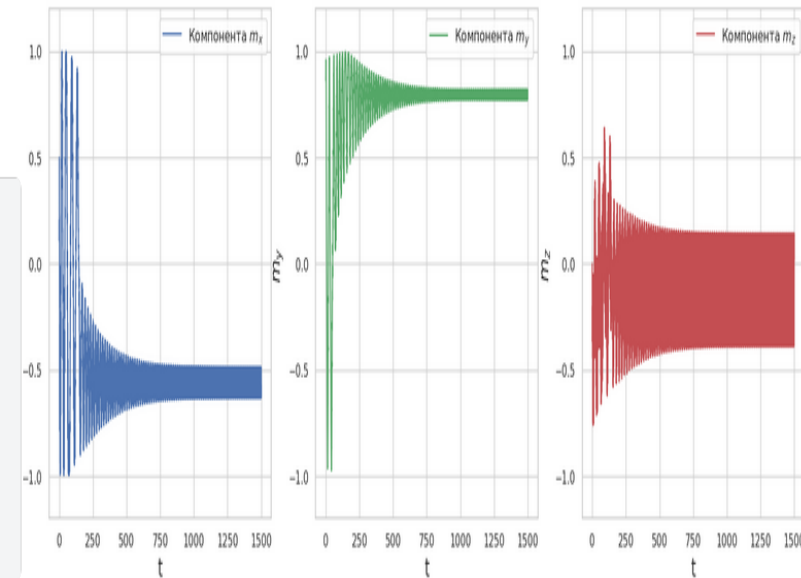
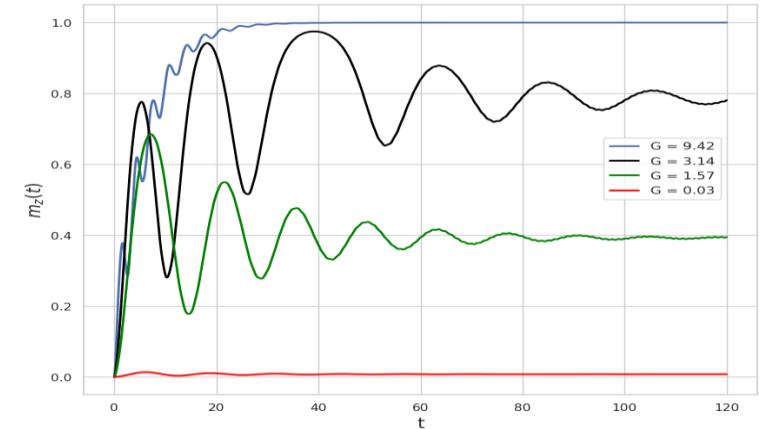
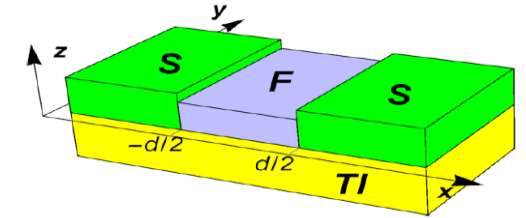
Численное решение системы уравнений

```
f = partial(my_sfs, G=G, r=r, alpha=alpha, wF=wF, d=d, V=V, J0=J0)
t_e = np.linspace(t0, tf, nt)
s0 = np.array([mx0, my0, mz0])
sol_1 = solve_ivp(f, [t0, tf], s0, t_eval=t_e, method='BDF',
                  rtol=1e-8, atol=1e-8)
```

Аппроксимация интегралов  $I_x$  и  $I_y$  многочленом девятой  $P_9$  и восьмой  $P_8$  степени

соответственно

```
# Using 9th order polynomial model
poly_model = odr.polynomial(9)
data = odr.Data(arr_mx, arr_Ix)
odr_obj = odr.ODR(data, poly_model)
# Running ODR fitting
output = odr_obj.run()
Ixfit = np.poly1d(output.beta[::-1])
```



### Welcome to HLIT Jupyter Book

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

Численное решение задачи Коши:  
библиотека SciPy

Задача 1: Линеаризованное  
уравнение на магнитный момент

Задача 2. Периодичность появления  
интервалов переворота  
намагниченности в  $\backslash(\phi\_0)$   
джозефсоновском переходе под  
воздействием импульса тока

# Welcome to HLIT Jupyter Book

Select the section you are interested in

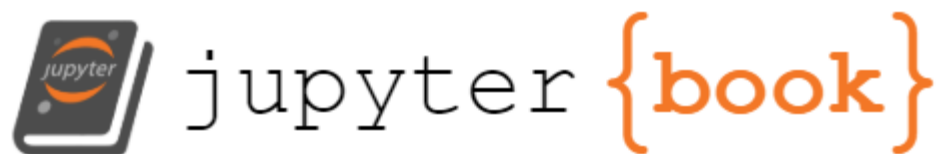
- Основы работы с Python: инструментарий на Python для решения научных и прикладных задач
- Численное решение задачи Коши: библиотека SciPy
- Задача 1: Линеаризованное уравнение на магнитный момент
- Задача 2. Периодичность появления интервалов переворота намагниченности в  $\backslash(\phi\_0)$  джозефсоновском переходе под воздействием импульса тока

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

By A.R. Rahmonova, A.S. Vorontsov, A.V. Nechaevskiy, I.R. Rahmonov, M.V. Bashashin, M.I. Zuev, O.I. Streltsova, Y.A. Butenko.  
© Copyright 2022.



# Инструментарий для численного моделирования систем, основанных на джозефсоновских переходах



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

<http://studhub.jinr.ru:8080/books>

<http://studhub.jinr.ru:8080/itschool2024>

