Computational environment for the numerical modelling of hybrid superconductor / magnetic nanostructures

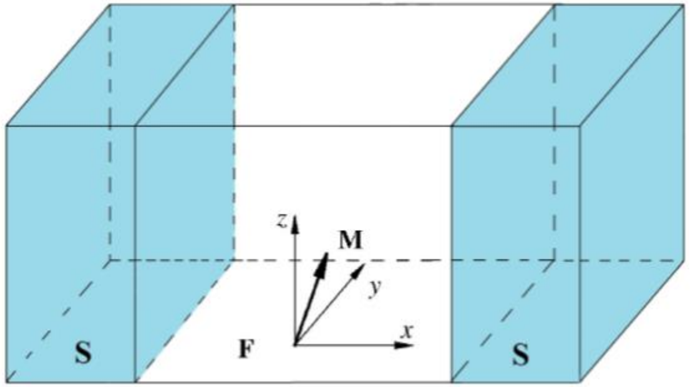
Andrey Nechaevskiy

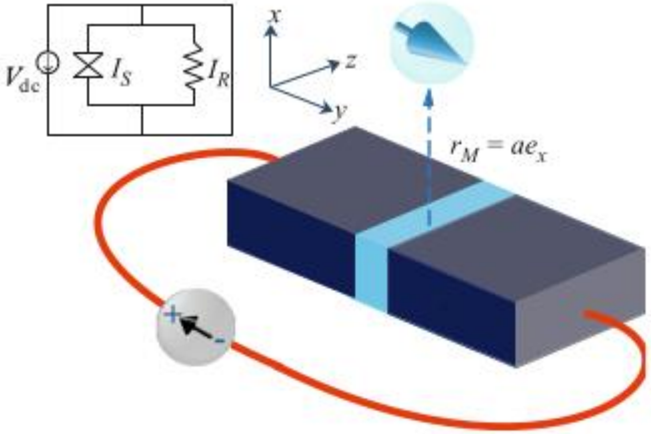
BLTP

This work was supported by Russian Science Foundation grant № 22-71-10022

JINR MLIT

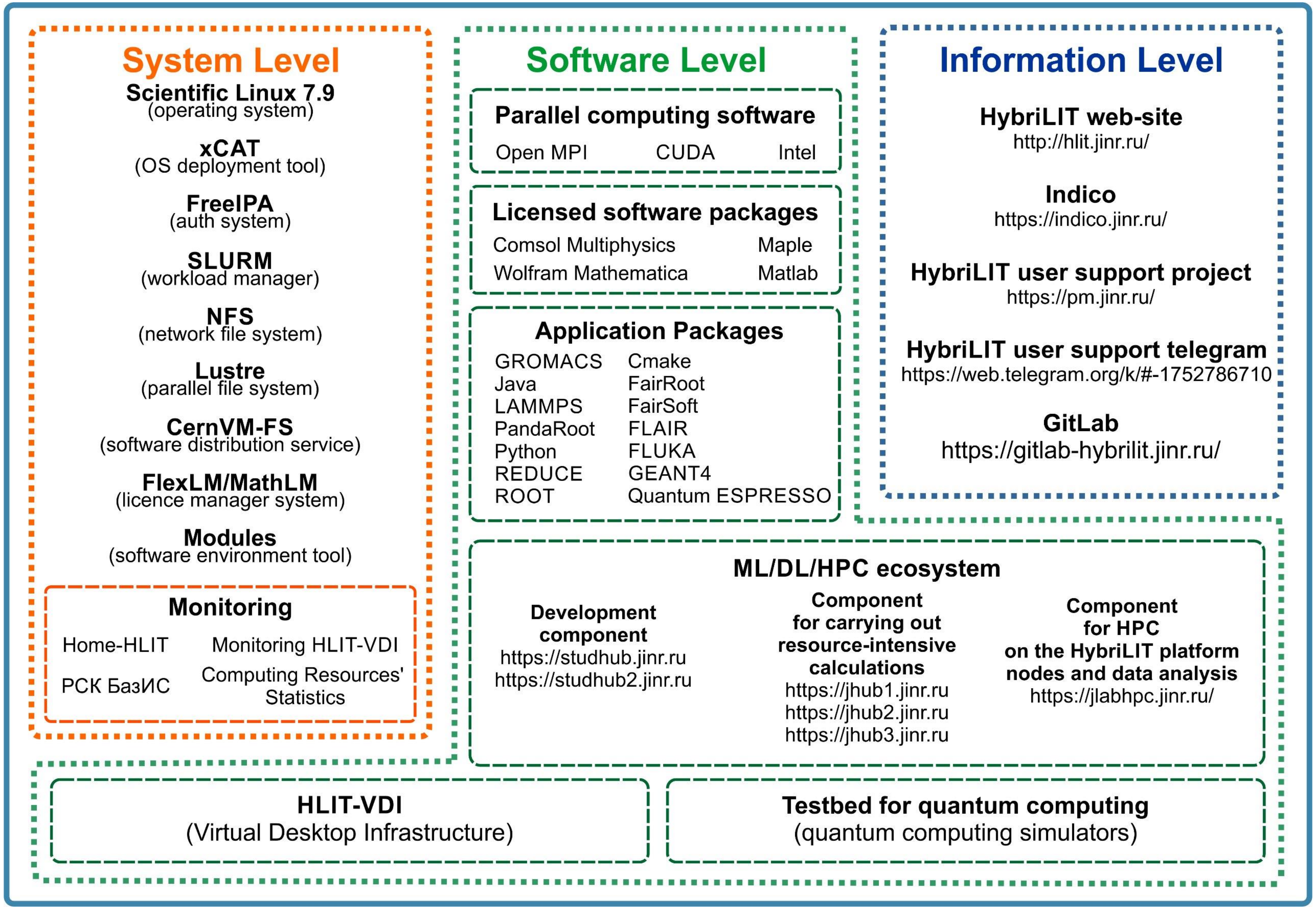
Mathematical modeling of superconducting nanostructures with a magnetic to study the possibilities of magnetization and magnetic excitations control using high-performance computing systems





The primary focus of modern nanotechnology research is the development, analysis and exploitation of novel materials and structures. The investigation of the physical properties of these objects requires the numerical solution of complex systems of non-linear differential equations, which requires a significant investment of time and computational resources. The transition to advanced digital technologies, such as high-performance hybrid computing (including parallel computing technologies on a cluster, on a graphics card, etc.), will facilitate the solution of this class of problems in a time-efficient manner, allowing the achievement of physically significant, world-class results. These results can then be used in the development of quantum computers and in solving problems in superconducting electronics and spintronics.

MICC component: HybriLIT platform



ML/DL/HPC ecosystem

Development component

VM with JupyterHub
https://jhub.jinr.ru

jupyterhub

VM:
CPU: 24 Cores
RAM: 32 GB

Keras, TensorFlow, scikit learn, NumPy, pandas

Computation component

Servers with NVIDIA Volta & Intel Xeon Gold
https://jhub2.jinr.ru

Dell Volta specs:
GPU: 4x Nvidia Volta V100-SXM2 NVLink 32Gb HBM2
CPU: 2x Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz 20 Cores/40 Threads
RAM: 512 GB DDR4 2666MHz
SSD: 2*240 GB

HPClab component

VM with JupyterHub and SLURM
https://jlabhpc.jinr.ru/

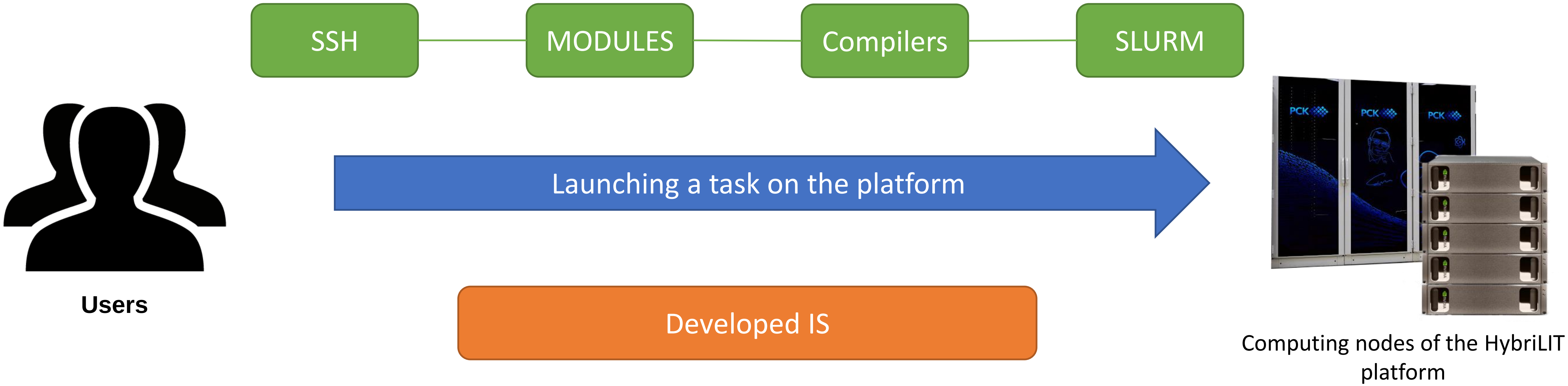
jupyterhub

VM:
CPU: 24 Cores
RAM: 64 GB

Keras, TensorFlow, scikit learn, NumPy, pandas

High Performance Computing

Development of information systems for theoretical and applied tasks on the HybriLIT platform



- The **main goal of the project** is to create a computational environment that will allow the efficient processing of the numerical modelling of hybrid S/F nanostructures.

- This environment will provide access to a range of tools and algorithms, as well as visualisation of computational results.

Tools and services for solving spintronics problems

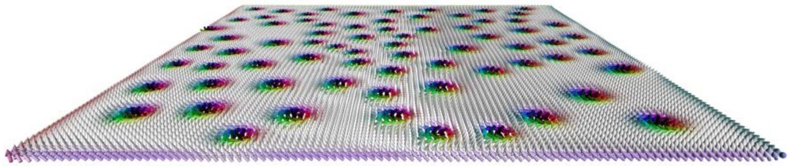
- Existing tools for modelling nanostructures have quite a wide functionality and are useful for modelling magnetic moment dynamics. But they do not involve the new types of interactions that can appear in hybrid nanostructures.

- For modelling hybrid nanostructures, there are currently no off-the-shelf modelling tools that can solve this problem.

- To solve this class of problems, we develop a computational environment for modelling hybrid superconductor/magnetic nanostructures, which provides access to the developed algorithms, computational resources and visualisation of the calculation results.

MATLAB SIMULINK®

COMSOL MULTIPHYSICS®

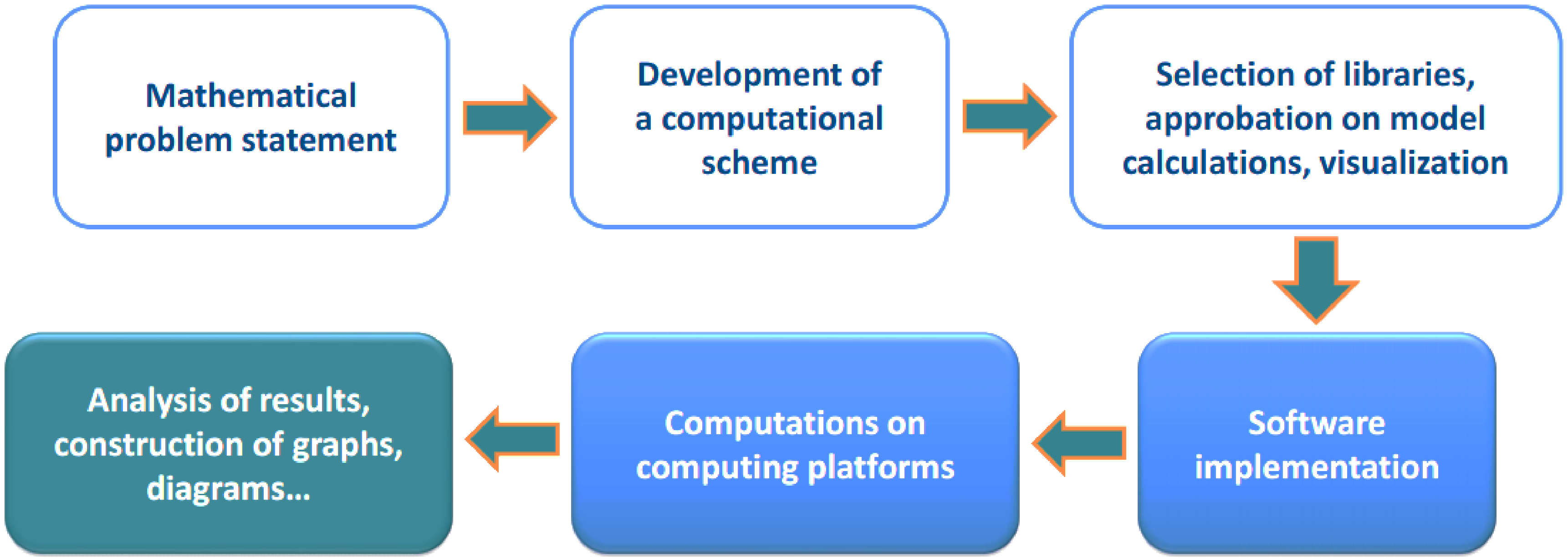


mumax^3
GPU-accelerated micromagnetism

spirit

OOMMF

Numerical Research Process



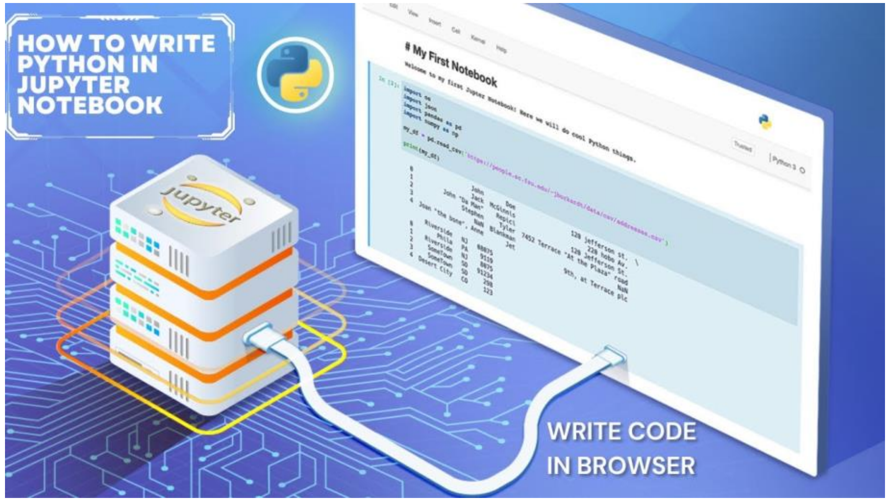
The creation of a toolkit that allows one to carry out computations, to visualize the results within a single application, and perform the most resource-intensive calculations in parallel is an urgent task.
The *Jupyter Notebook* environment provides this capability.

Tools and services for solving spintronics problems

Jupyter Notebook is a powerful code management tool.

It provides many opportunities for developers and researchers.

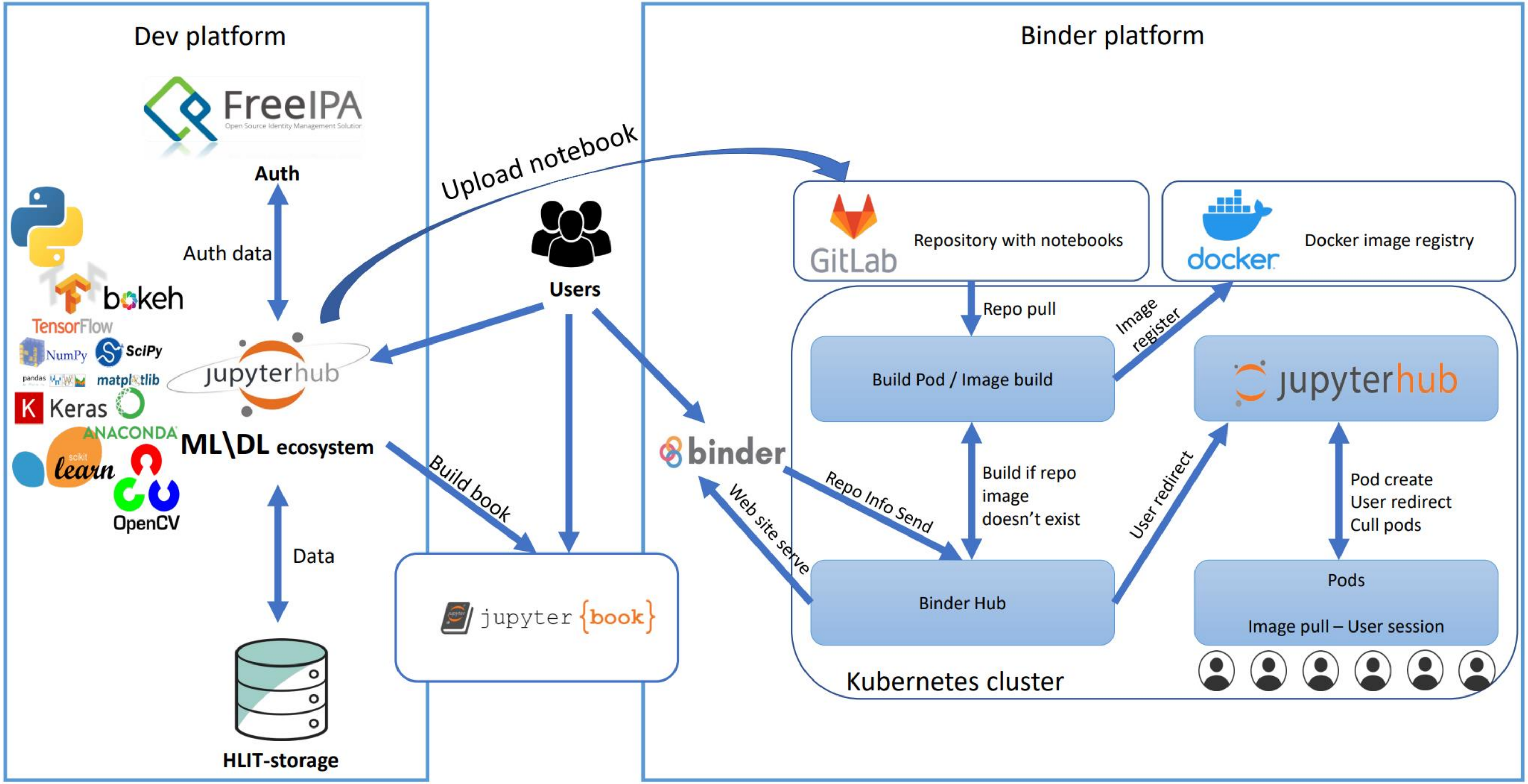
- An environment with syntax highlighting, error correction and other IDE features.
- The ability to run individual sections of code or the entire program.
- You can run any piece of code in any order, which makes testing and debugging easier.
- Inserting and outputting results right in the middle of the code.
- Code sharing and collaboration.
- Beautiful and clear document design.



Tools and services for solving spintronics problems

- The first step was the implementation of a model for the study of Josephson junctions in Python in the Jupyter Notebook environment.
- Integration of the Jupyter Binder platform, for users who do not have access to the HybriLIT platform, including for test calculations or training courses.
- Jupyter Book integration, allowing users to create books, in the form of Jupyter Notebooks, that can be viewed or downloaded to work on their computing resources.
- HybriLIT platform used for computation and active development.

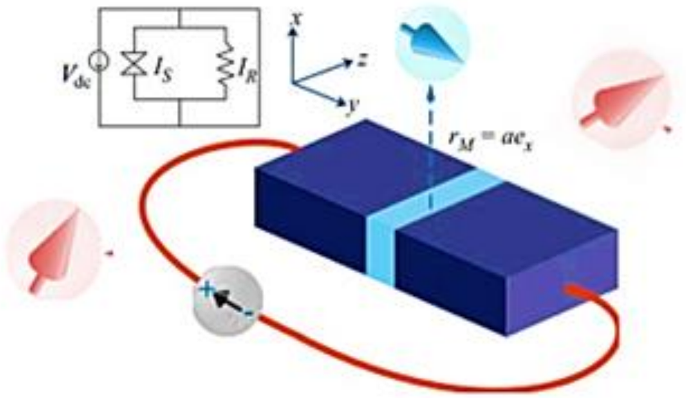
Computational environment for the numerical modelling



Development of information systems

jupyterhub

Symbolic computations block



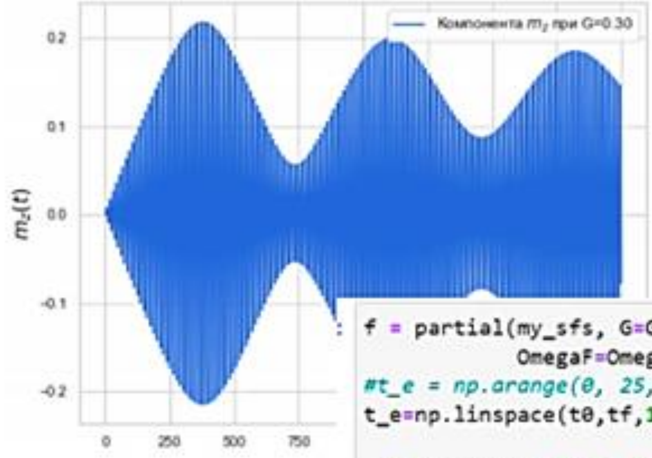
$$\gamma_{m_i} \quad = \quad = -\frac{\mu_0}{2\Phi_0}\int d\mathbf{r}_i \frac{\mathbf{M}_i \times \mathbf{r}_i}{r^3}$$

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

SymPy is a Python library for symbolic mathematics.

matplotlib is a main library for building graphs, diagrams in Python.

Block of numerical computations 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.

Parallelization of multi-parameter computations

Joblib

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

Python implementation

Calculations for different values of parameters

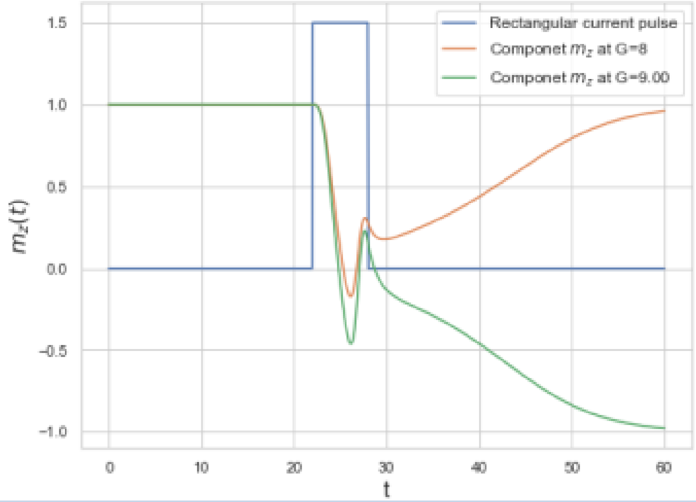
To analyze the possibility of reversing the magnetic moment of the ϕ_0-Josephson junction at different values of the parameters, we will carry out calculations for G=8.9.

```
from scipy.integrate import solve_ivp
from functools import partial
```

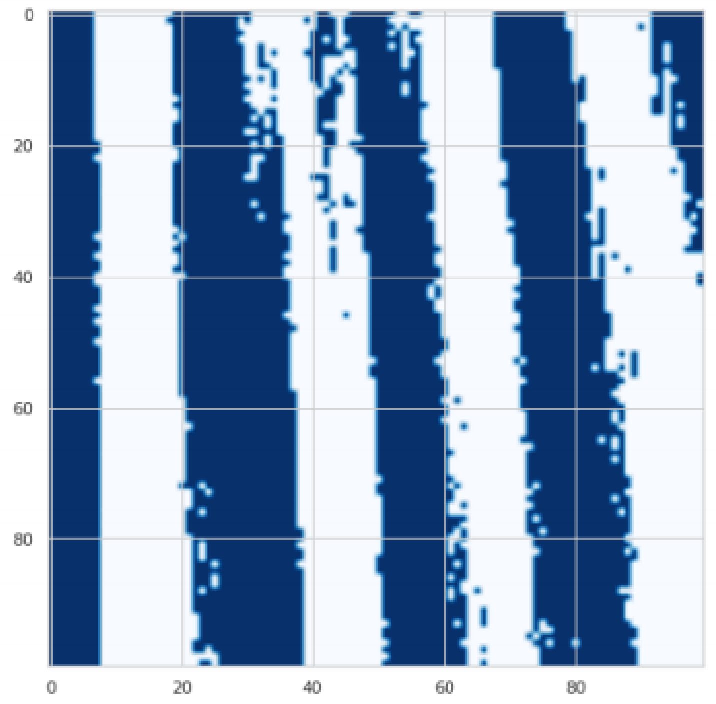
```
G=9
f = partial(my_sfs,  G=G, r=r, alpha=alpha, \
                As=As, t_s=t_s, delta_t=delta_t)
#t_e = np.arange(0, 25, 0.0001)
t_e=np.linspace(0,60,1000)

s0 = np.array([0, 0, 1, 0])
sol_2=solve_ivp(f,[0,60],s0, t_eval=t_e) # method = 'Radau'
```

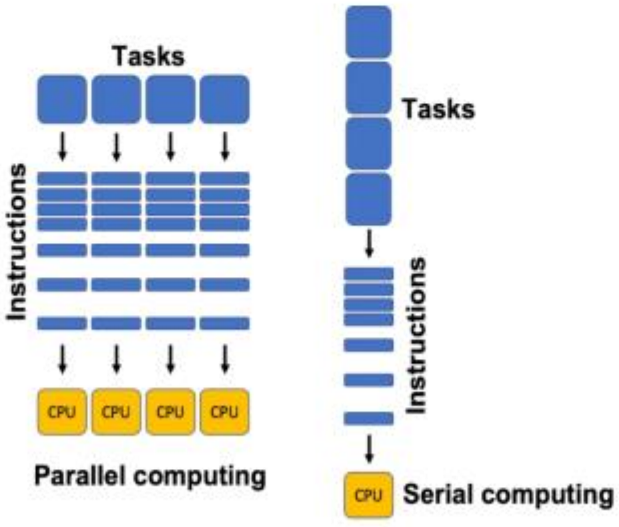
```
plt.figure(figsize = (8, 6))
plt.plot(t_e,y_I,  label= 'Rectangular current pulse')
plt.plot(sol_1.t, sol_1.y[2], label= 'Componet $m_z $ at G=8' )
plt.plot(sol_2.t, sol_2.y[2], label= 'Componet $m_z $ at G=%4.2f' %G)
plt.xlabel('t', size=16)
plt.ylabel('$m_z(t)$', size=16)
plt.legend(fontsize=12)
plt.show()
```




```
#plt.figure(figsize = (8, 6))
fig, ax1 = plt.subplots(figsize=(8, 8))
# mask out the negative and positve values, respectively
#Zpos = np.ma.masked_less(alpG[:,:,0], 0)
Z1 = Zc.reshape(N, N)
plt.imshow(Z1, interpolation='bilinear', cmap='Blues')
#plt.contourf(X, Y, Zc, 100)
#fig.colorbar(Zc, ax=ax1)
plt.show()
```



Parallel implementation with Python



Define a function called by each process

```
from joblib import Parallel, delayed
import numpy as np
```

```
def funk_parall(k):
    i=k%N
    j=k//N
    mz_sol=0
    G=G0+delta_G*i
    alpha=alpha0+delta_alpha*j
    f = partial(my_sfs, G=G, r=r, alpha=alpha, \
                As=As, t_s=t_s, delta_t=delta_t)
    t_e=np.linspace(0,60,1000)
    s0 = np.array([0, 0, 1, 0])
    sol_i=solve_ivp(f,[0,60],s0, t_eval=t_e) # method = 'Radau'
    if sol_i.y[2][999] < 0:
        mz_sol= -1
         # alpGxy[i+j*N,2] =- 1
    return mz_sol
```

Serial mode calculation

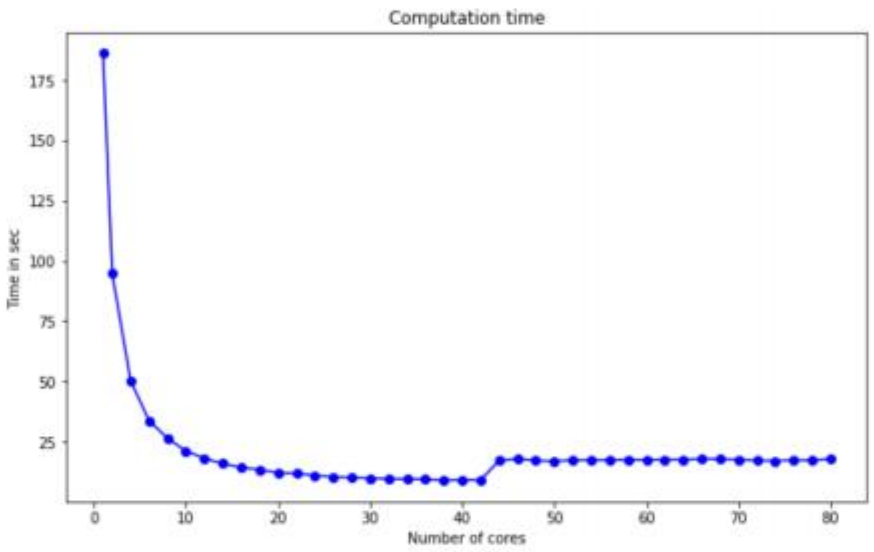
```
t0 = time.time()
rez= Parallel(n_jobs=1)\
    (delayed(funk_parall)(k) for k in range(N*N) )
t1 = time.time()
print(f'Execution time {t1 - t0} s')
```

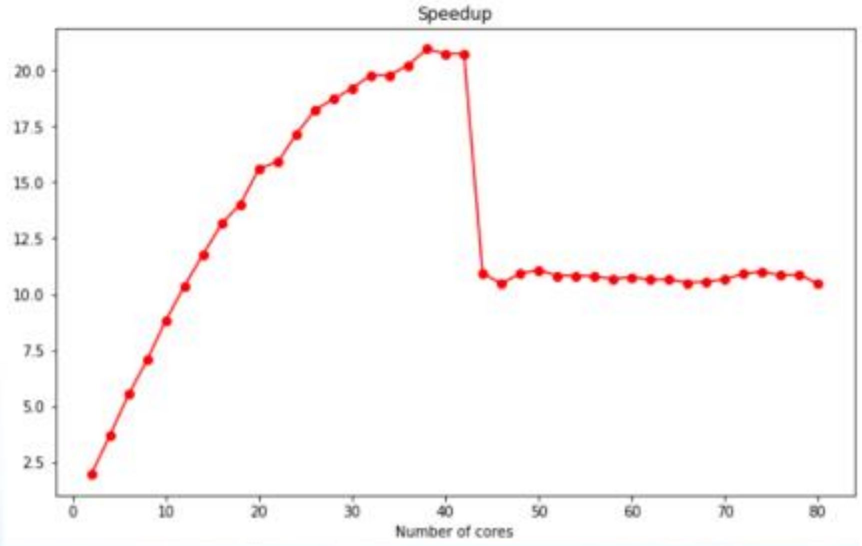
```
Execution time 159.9254457950592 s
```

Computing in Parallel Mode

```
t0 = time.time()
rez= Parallel(n_jobs=6)\
    (delayed(funk_parall)(k) for k in range(N*N) )
t1 = time.time()
print(f'Execution time {t1 - t0} s')
```

```
Execution time 34.51503801345825 s
```





Computational environment for the numerical modelling

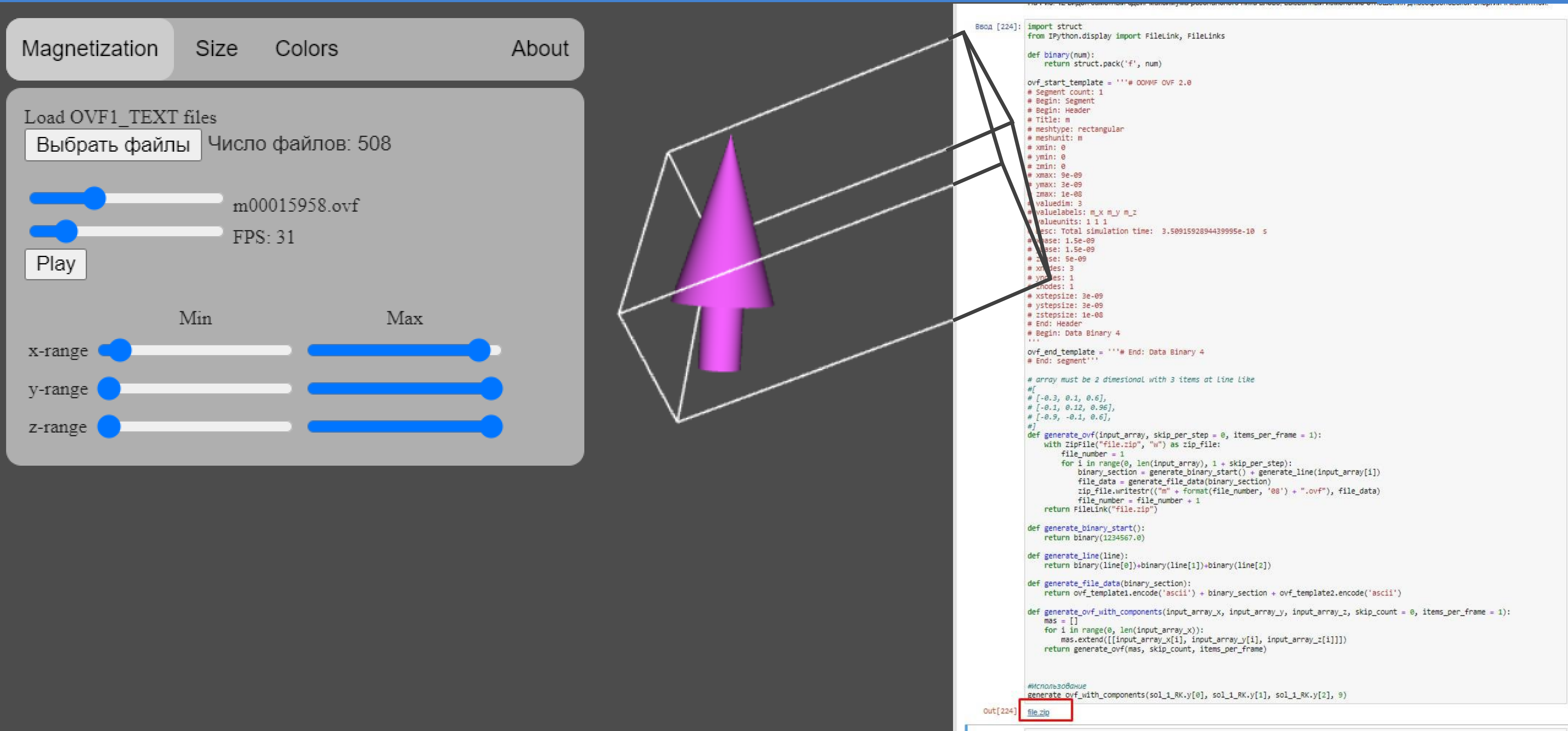
- Visualisation of simulation results

- Graphical interface for specifying input calculation parameters of models (geometric and physical parameters of models, numerical counting parameters)

- Implementation of the module for resource-intensive calculations on the GOVORUN supercomputer

- Debugging and testing of developed algorithms and modules

Computational environment for the numerical modelling



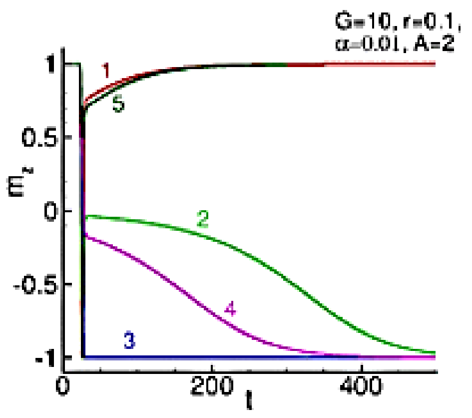
Computational environment for the numerical modelling

Numerical simulation of the magnetization reversal within the RF-SQUID model with φ0 junction depending on the external magnetic field pulse

M.V. Bashashin, A.R. Rahmonova E.V. Zemlyanaya, Yu. M. Shukrinov, I.R. Rahmonov

Magnetic reversal

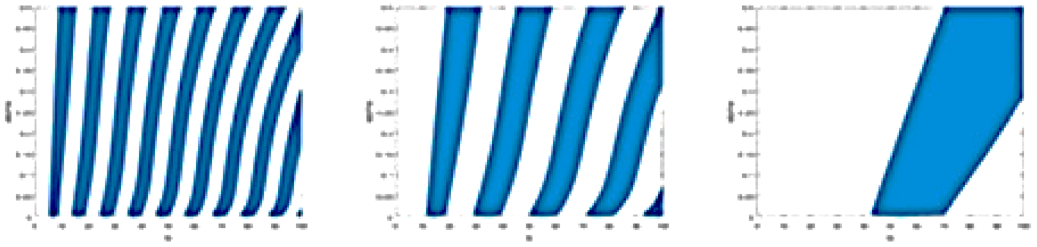
Magnetic reversal is an effect when m_z-component of the magnetic field changes the sign and takes the value -1 for a given initial value of +1.



We analyze effect of the SQUID inductance L on MR. In figure the time d
= 1, 2, 3, 4, 5 fo
The numbers in
L.

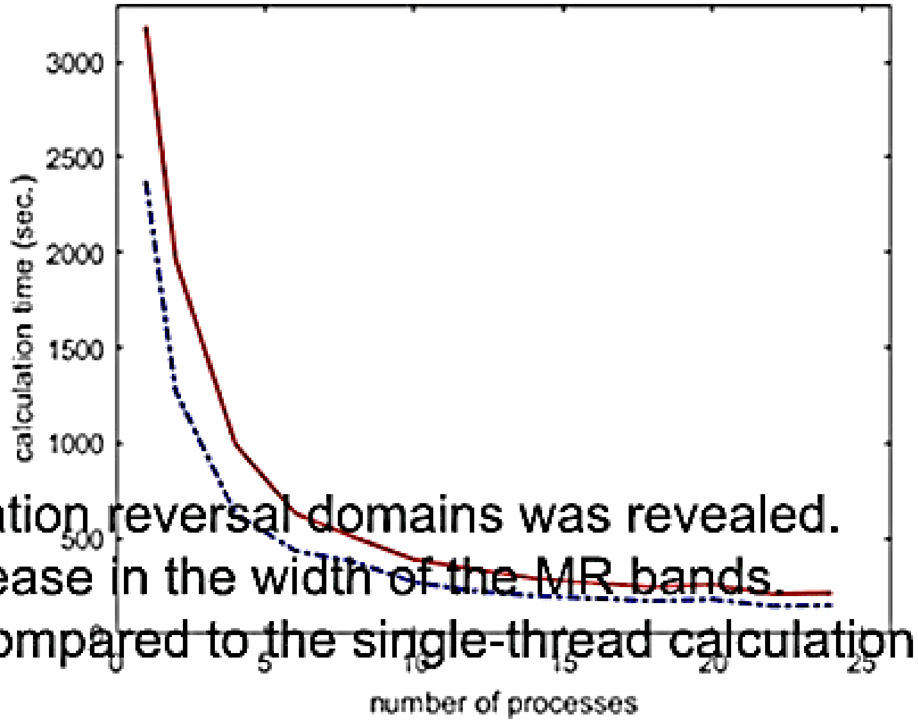
- In case of L = realized and c
- At L = 3 the fa acting of puls
- In case of the takes long tim
- At L = 5 again

The performed analysis allows us to conclude that MR has a period SQUID inductance.



Parallel implementation

- For the numerical solution of the system of method was used.
- The execution time of a serial C++ program of modeling ma the (G, α)-plane is 53 minutes (using Intel compiler).
- The parallelization process is based on the distribution of the plane between parallel threads. The values of G, α where the $|m_z(T_{max})+1| < \varepsilon$ is satisfied, are saved in output structure and file.
- Maximal speedup of MPI implementation is about 15.5 times.
- The same parallelization scheme was used in case of simulati planes.
- Also, using a parallel implementation, the influence of the AVX instructions built into the latest versions of Intel server proces processors are available on the Govorun supercomputer.



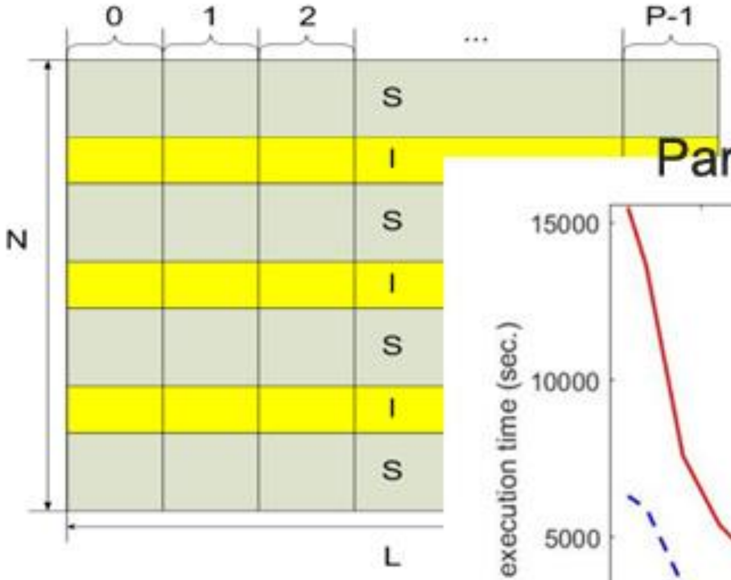
The influence of the inductance parameter on the width of magnetization reversal domains was revealed.
Shown that an increase in the inductance parameter leads to an increase in the width of the MR bands.
Max speedup of MPI + AVX-512 implementation is about 22 times compared to the single-thread calculation.

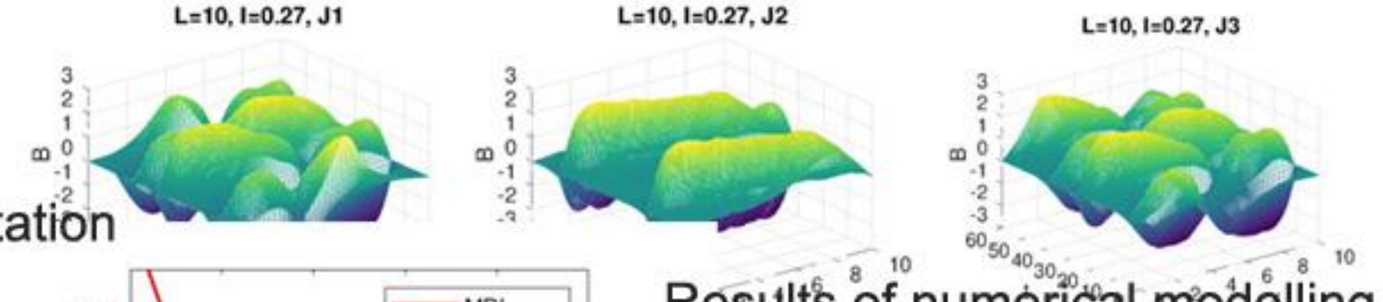
Computational environment for the numerical modelling

Numerical study of the influence of inductive and capacitive coupling on the volt-ampere characteristic in a system of long Josephson junctions

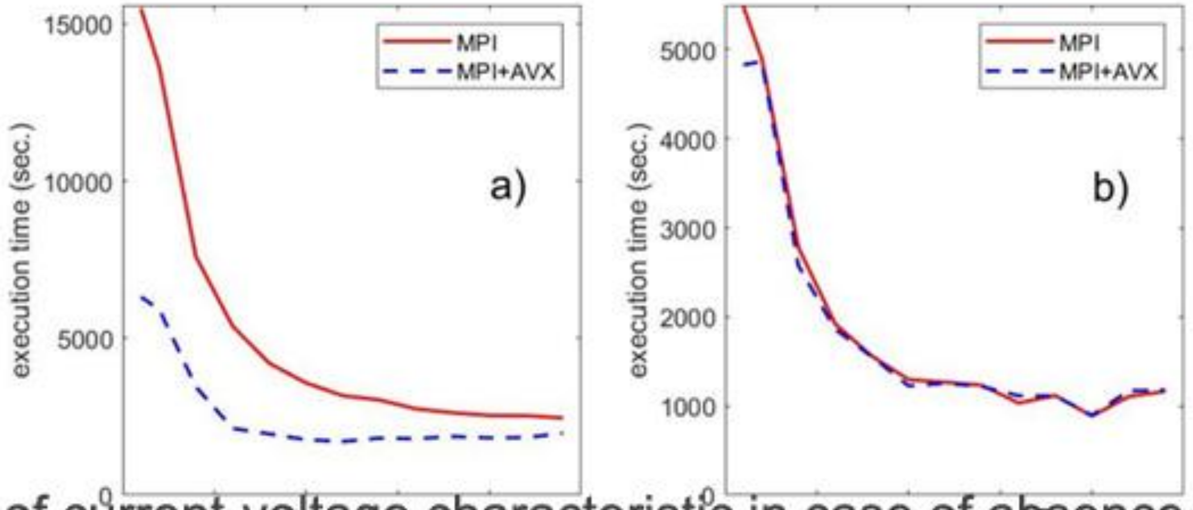
M.V. Bashashin, E.V. Zemlyanaya, I.R. Rahmonov

Parallelisation scheme

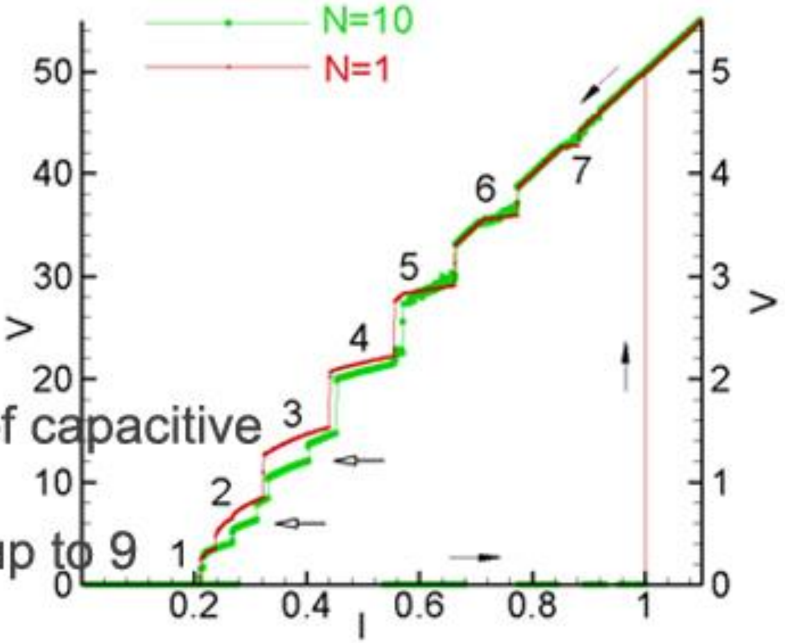




Parallel implementation



Results of numerical modelling



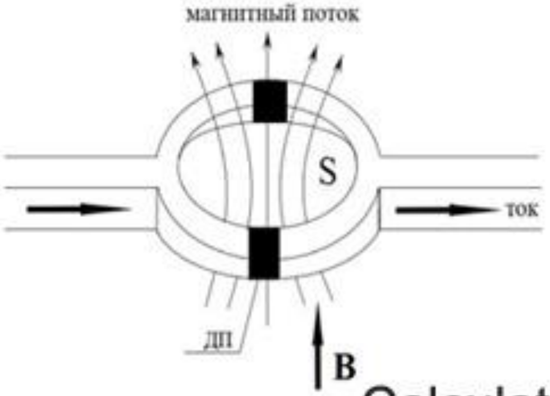
Computer modelling of current-voltage characteristic in case of absence of capacitive coupling in the presence of only inductive coupling has been carried out. Test calculations have shown the possibility ofreduction of counting time up to 9 times at calculations in parallel mode.

Computational environment for the numerical modelling

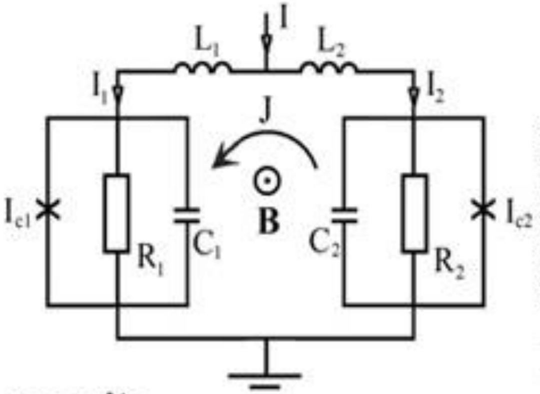
Simulation of the dynamics of a superconducting quantum interferometer with two Josephson junctions based on Python in the Jupyter Book environment

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

Mathematical model

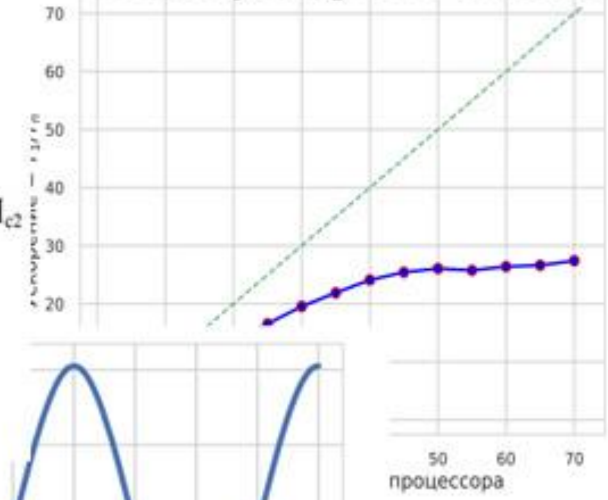


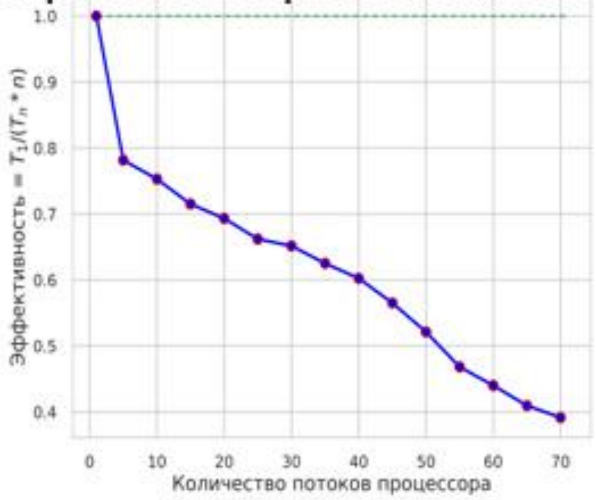
(a)



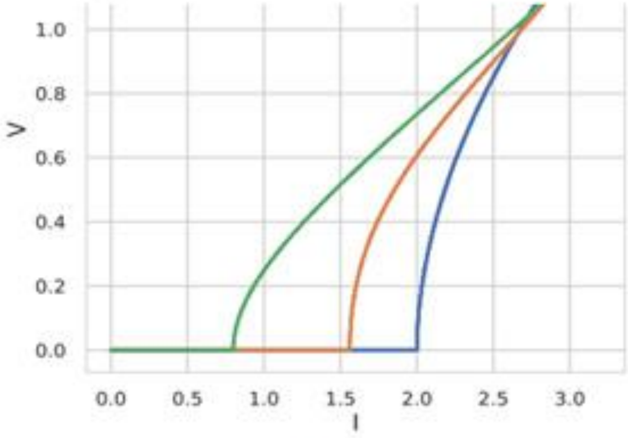
(b)

Analysing the efficiency of parallel implementation

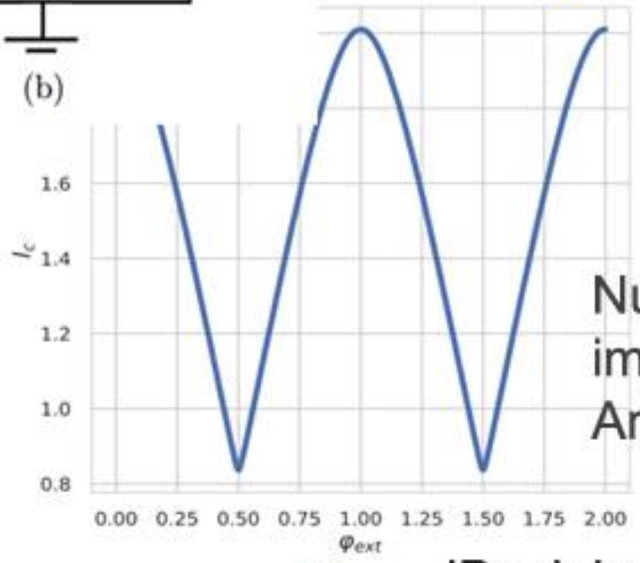




Calculation results



(a)

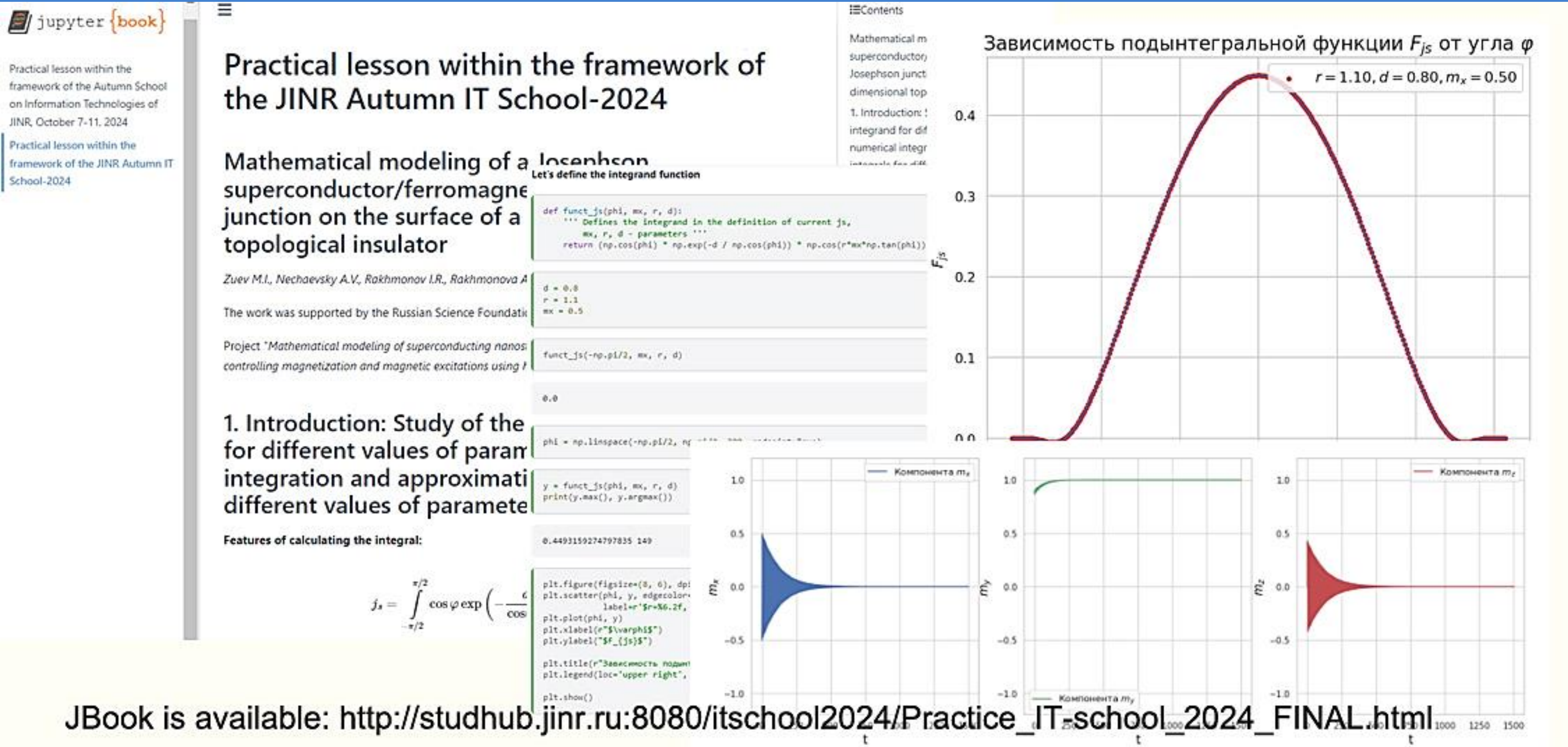


(b)

Numba lib was used in the software implementation of the developed algorithms. An acceleration of 27 times was achieved.

JBook is available: http://studhub.jinr.ru:8080/squid

JINR Autumn School on Information Technologies 2024



JBook is available: http://studhub.jinr.ru:8080/itschool2024/Practice_IT-school_2024_FINAL.html

Welcome to HLIT Jupyter Book

Select the section you are interested in

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.

Conclusion

- Jupyter Notebook is an environment that enables users to create and execute Jupyter Notebooks.
- Notebooks are documents comprising live code, equations, visualisations and descriptive text.
- Notebooks can be executed in a multitude of programming languages, including Python, R, and Julia.
- Feasible to incorporate libraries and packages, including NumPy, Pandas, Matplotlib, and numerous others.
- Jupyter Binder enables users to run Jupyter notebooks in a web browser without requiring authorisation or the installation of software on the local machine.
- The developed platform allows users to perform sophisticated data analysis, numerical calculations and visualisation tasks with minimal effort on the HybriLIT platform.
- Project is create new opportunities for the study of hybrid nanostructures.

Project Team

This work was supported by Russian Science Foundation grant № 22-71-10022

Head of the project
Dr. Kirill Kulikov
(BLTP JINR)

Dr. Yuriy M. Shukrinov
(BLTP JINR)

Dr. Nashaat Majid
(BLTP JINR)

Dr. Oksana I. Streltsova
(MLIT JINR)

Dr. Andrey V. Nechaevskiy
(MLIT JINR)

Maksim V. Bashashin
(MLIT JINR)

Maksim I. Zuev
(MLIT JINR)

Adiba Rakhmonova
(MLIT JINR)

Questions?