

ML/DL/HPC Ecosystem of the HybriLIT Heterogeneous Platform (MLIT JINR): New Opportunities for Applied Research

**Butenko Yu.¹, Ćosić M.², Nechaevskiy A.¹, Podgainy D.¹,
Rahmonov I.¹, Streltsova O.¹, Zuev M.¹**

¹ Joint Institute for Nuclear Research

² Vinča Institute of Nuclear Sciences – National Institute of the Republic of Serbia

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

The 6th International Workshop on Deep Learning in Computational Physics (DLCP-2022)

Dubna, JINR, 6-8 July 2022

Training courses, master classes and lectures

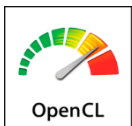
LIT staff and leading scientists
from JINR and its Member States

Leading manufacturers of
modern computing
architectures and software

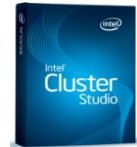
Parallel
programming
technologies

OpenMP

MPI



Tools for debugging and
profiling of parallel
applications



Work with applied software
packages

COMSOL
MULTIPHYSICS



ROOT
Data Analysis Framework



Wolfram Mathematica



GEANT4
Accelerating and Tracking



Frameworks and
tools for ML/DL
tasks

TensorFlow



scikit
learn



NumPy



Ecosystem for ML/DL/HPC tasks



Development component

VM with JupyterHub

<https://jhub.jinr.ru>



VM:
CPU: 24 Cores
RAM: 32 GB

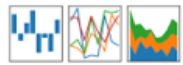
K Keras



scikit
learn



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/>



VM:
CPU: 24 Cores
RAM: 64 GB

K Keras



scikit
learn

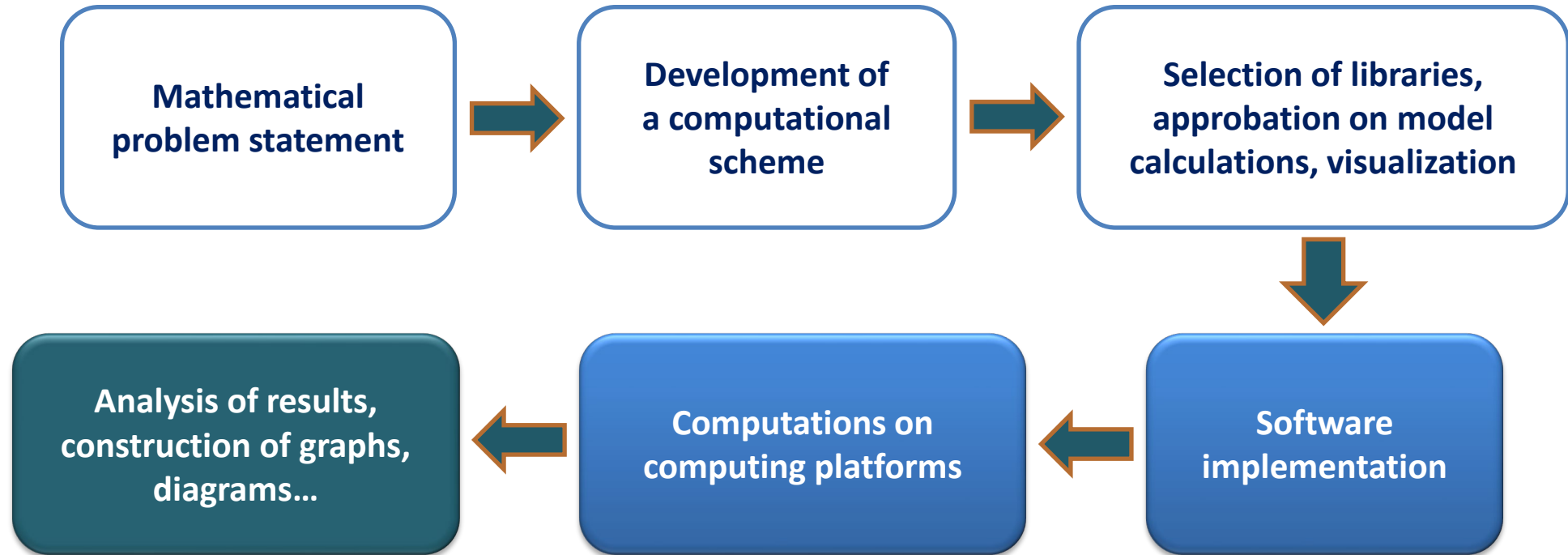


pandas



High Performance Computing

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.

Developed services



saas.jinr.ru

Long Josephson junctions stack simulation
Superconductor-Ferromagnetic-Superconductor Josephson junction simulation
Annular Array of JJs average
Long Josephson junction coupled with the ferromagnetic thin film
Stack of short JJ with LC shunting
Stack of short JJ

$$\begin{cases} \frac{\partial \varphi}{\partial t} = V, \\ \frac{\partial V}{\partial t} = \frac{\partial^2 \varphi}{\partial x^2} - \sin \varphi - \beta V + I. \end{cases}$$

$$\varphi(x, t)|_{x=0} = 0, \quad \frac{\partial \varphi(x, t)}{\partial t} \Big|_{t=0} = 0,$$

граничные условия

$$\frac{\partial \varphi(x, t)}{\partial x} \Big|_{x=0} = H_{ext}, \quad \frac{\partial \varphi(x, t)}{\partial x} \Big|_{x=L} = H_{ext}$$

JINR cloud SaaS

Manual

This work is supported by the Russian Science Foundation under grant #18-71-10095

JINR CLOUD SERVICE
for scientific and
engineering computations

Sign in via JINR Single Sign-On

Demo

This work is supported by the Russian Science Foundation under
grant #18-71-10095

Job parameters

Physical parameters

B_c:
25

G:
0.1

r:
0.1

a:
0.1

ω_F:
0.5

Parameters of external radiation ☐

Calculational parameters

T_{max}:
1000

T_c:
100

Δt:
0.005

I₀:
0.01

I_{max}:
1.5

I_{min}:
0

I_{break}:
0

I_{noise}:
0

I_{more1}:
0.1

I_{more2}:
0.9

ΔI₀:
0.005

ΔI₁:
0.0005

Resources

JINR cloud

HybridLIT cluster

Number of VMs: 10/20

CPU per VM: 15/22

RAM per VM (GB): 27/41

1

1

1

../	02-Apr-2021 11:04	32K
Idisp.dat	02-Apr-2021 11:04	32K
Iq.dat	02-Apr-2021 11:04	32K
Is.dat	02-Apr-2021 11:04	32K
Is_w_down.dat	02-Apr-2021 11:04	32K
Is_w_up.dat	02-Apr-2021 11:03	0
Isum.dat	02-Apr-2021 11:04	32K
Voltage.dat	02-Apr-2021 11:04	32K
initial_cond.dat	02-Apr-2021 11:05	184K
m Tf.dat	02-Apr-2021 11:05	48K
m_max_w_down.dat	02-Apr-2021 11:05	75K
m_max_w_up.dat	02-Apr-2021 11:03	8192
mx_max_I_down.dat	02-Apr-2021 11:05	52K
mx_max_I_up.dat	02-Apr-2021 11:03	8192
my_max_I_down.dat	02-Apr-2021 11:05	52K
my_max_I_up.dat	02-Apr-2021 11:03	8192
mz_max_I_down.dat	02-Apr-2021 11:05	50K
mz_max_I_up.dat	02-Apr-2021 11:03	8192
sfsjj.836337.err	02-Apr-2021 11:02	99
sfsjj.836337.log	02-Apr-2021 11:05	1594
sfsjj.836337.out	02-Apr-2021 11:05	36K

Developed services



sconduct.jinr.ru

Главная | Демо модели | Публикации | Войти

Расчет временной динамики сверхпроводник/ферромагнит/сверхпроводник

Справочные материалы

Уравнения Лондо-Фейнмана-Гинзбурга для вектора намагниченности

$$\frac{\partial \mathbf{M}}{\partial t} = -\gamma \mathbf{M} \times \mathbf{H}_{\text{eff}} + \frac{\gamma}{M_0} (\mathbf{M} \times \frac{\partial \mathbf{M}}{\partial t}) \quad (1)$$

γ - гиромагнитное отношение, α - коэффициент демплинга
M₀ - модуль вектора намагниченности в состоянии равновесия
H_{eff} - эффективное поле в направлении осевой симметрии

$$\mathbf{H}_{\text{eff}} = \frac{K}{M_0} \left[G \sin^2 \left(\varphi - \frac{M_z}{M_0} \right) \hat{\varphi} + \frac{M_z}{M_0} \hat{z} \right] \quad (2)$$

G = E₂(K)/E₁(K) - отношение дивергентности энергии к энергии магнитной индукции
K - постоянная анизотропии
U - объем ферромагнитного слоя
φ - параметр угла ориентации намагниченности, z' - проекция фаз

Уравнение релаксации намагниченности

$$I = \frac{8}{\pi} \left(\frac{d_0}{d} + \frac{r}{M_0} \frac{\partial M_z}{\partial t} \right) + I_0 \sin \left(\varphi - \frac{M_z}{M_0} \right) \quad (3)$$

I₀ - критический ток, I_c - критический ток, I_c - критический ток

Уравнения Лондо-Фейнмана-Гинзбурга для вектора намагниченности в скалярной форме в безразмерных переменных

$$\frac{dM_z}{dt} = \frac{1}{1 + \alpha^2} \left[\alpha M_z H_{\text{eff},z} - \alpha M_z H_{\text{eff},z} + \alpha M_z H_{\text{eff},z} + \alpha M_z H_{\text{eff},z} - H_{\text{eff},z} M_z^2 \right]$$

$$\frac{dM_z}{dt} = \frac{1}{1 + \alpha^2} \left[\alpha M_z H_{\text{eff},z} - \alpha M_z H_{\text{eff},z} + \alpha M_z H_{\text{eff},z} + \alpha M_z H_{\text{eff},z} - H_{\text{eff},z} M_z^2 \right]$$

Константы магнитного момента m_B, магнетон Бора μ_B (i = x, y, z), намагниченность эффективная H_{eff,z}, поле, намагниченность m_B, μ_B, которые определяются параметрами, указанными в файле my_time.dat

Время t, безразмерное поле H_{eff,z}, намагниченность M_z, параметр φ

$$\begin{aligned} H_{\text{eff},z}(t) &= 0, \\ M_z(t) &= G \sin^2(t) - m_z(t), \\ M_z(t) &= m_z(t). \end{aligned}$$

Параметры модели

α

G

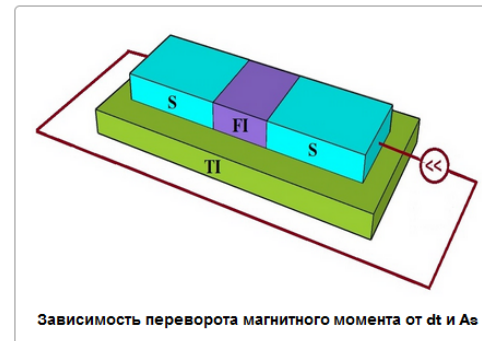
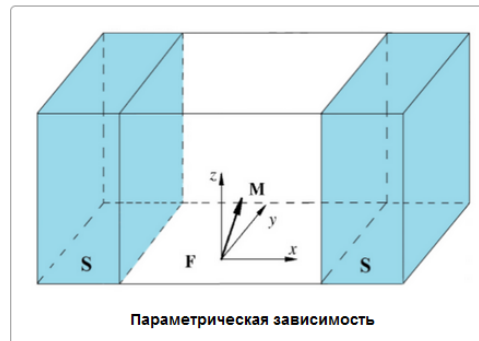
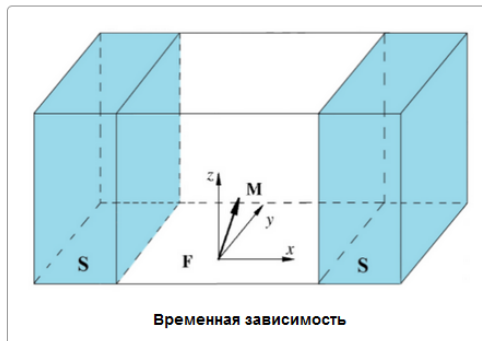
r

ω

As

dt

Примеры реализованных алгоритмов:

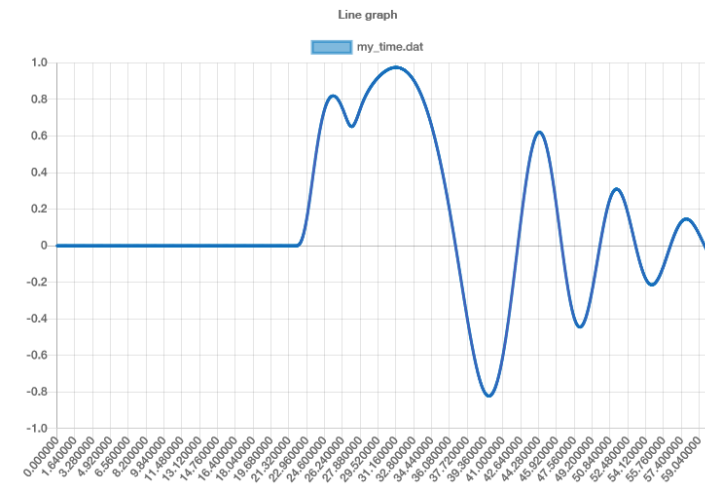


Выбранный файл: my_time.dat

0.000000	0.0000000000000000
0.010000	0.0000000000000000
0.020000	0.0000000000000000
0.030000	0.0000000000000000
0.040000	0.0000000000000000
0.050000	0.0000000000000000
0.060000	0.0000000000000000
0.070000	0.0000000000000000
0.080000	0.0000000000000000
0.090000	0.0000000000000000
0.100000	0.0000000000000000
0.110000	0.0000000000000000
0.120000	0.0000000000000000
0.130000	0.0000000000000000
0.140000	0.0000000000000000
0.150000	0.0000000000000000
0.160000	0.0000000000000000
0.170000	0.0000000000000000
0.180000	0.0000000000000000
0.190000	0.0000000000000000
0.200000	0.0000000000000000
0.210000	0.0000000000000000
0.220000	0.0000000000000000
0.230000	0.0000000000000000
0.240000	0.0000000000000000
0.250000	0.0000000000000000
0.260000	0.0000000000000000
0.270000	0.0000000000000000
0.280000	0.0000000000000000
0.290000	0.0000000000000000
0.300000	0.0000000000000000
0.310000	0.0000000000000000
0.320000	0.0000000000000000
0.330000	0.0000000000000000
0.340000	0.0000000000000000
0.350000	0.0000000000000000
0.360000	0.0000000000000000
0.370000	0.0000000000000000
0.380000	0.0000000000000000

Полученные файлы:

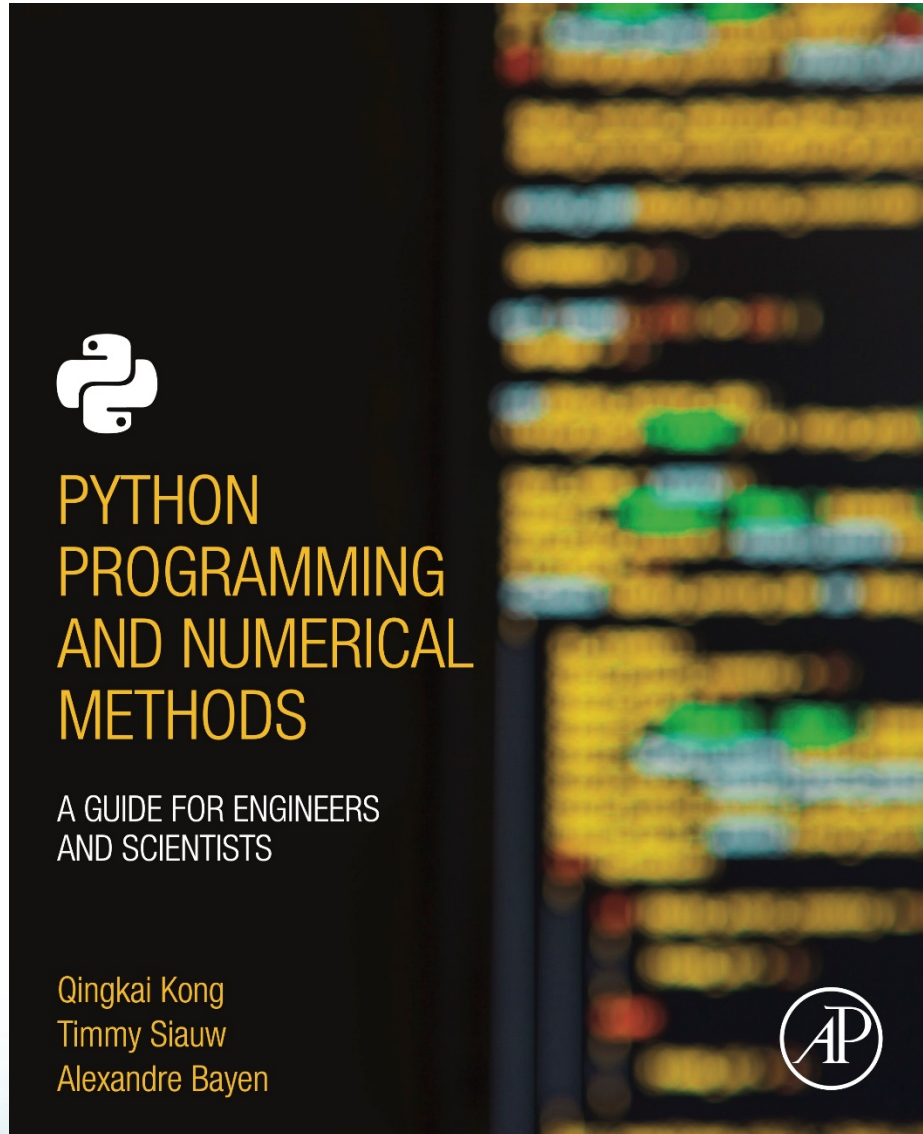
Выбранный файл: my_time.dat



Python Numerical Methods



pythonnumericalmethods.berkeley.edu



jupyter {book}



matplotlib

pandas



K Keras

NumPy

seaborn



Example 1. Problem to study the dynamics of magnetization in a Phi-0 Josephson Junction (SFS structure)



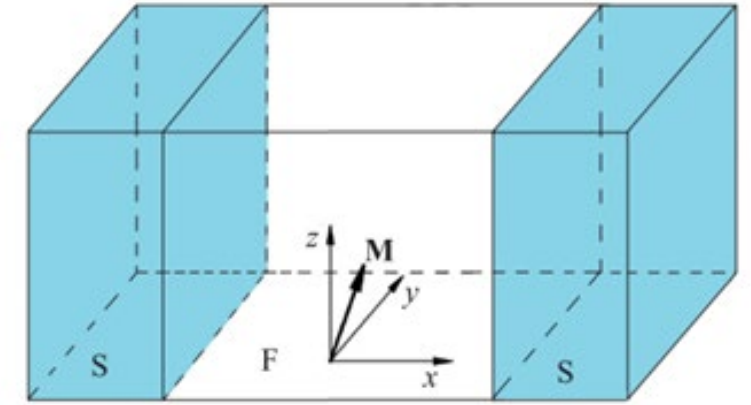
Collaboration with Ilhom Rahmonov (Bogoliubov Laboratory of Theoretical Physics, JINR)

The dynamics of the magnetic moment M of the system under consideration is described by the Landau-Lifshitz-Gilbert equation:

$$\begin{aligned}\frac{dm_x}{dt} &= -\frac{1}{1 + M^2 \alpha^2} \{m_y H_z - m_z H_y + \alpha[m_x(M, H) - H_x]\}, \\ \frac{dm_y}{dt} &= -\frac{1}{1 + M^2 \alpha^2} \{m_z H_x - m_x H_z + \alpha[m_y(M, H) - H_y]\}, \\ \frac{dm_z}{dt} &= -\frac{1}{1 + M^2 \alpha^2} \{m_x H_y - m_y H_x + \alpha[m_z(M, H) - H_z]\},\end{aligned}$$

$M = [m_x, m_y, m_z]$ are the magnetic moment components; the effective field components $H = [H_x, H_y, H_z]$ depend on the Josephson phase difference ϕ and are defined as follows:

$$\begin{aligned}H_x(t) &= 0, \\ H_y &= Gr \sin(\phi(t) - tm_y(t)), \\ H_z(t) &= m_z(t).\end{aligned}$$



Model parameters:

G – ratio of the Josephson energy to the magnetic anisotropy energy;
 r – spin-orbit interaction constant;
 α – Hilbert dissipation parameter;
in this study $w = 1$.

The equation for the Josephson phase difference $\phi(t)$ is determined from the equation for the electric current I flowing through the Josephson junction, measured in units of the critical current I_c :

$$\frac{d\phi}{dt} = -\frac{1}{w} \left(\sin(\phi - rm_y) + r \frac{dm_y}{dt} \right) + \frac{1}{w} I,$$

Example 1. Python implementation



Calculations for different values of parameters

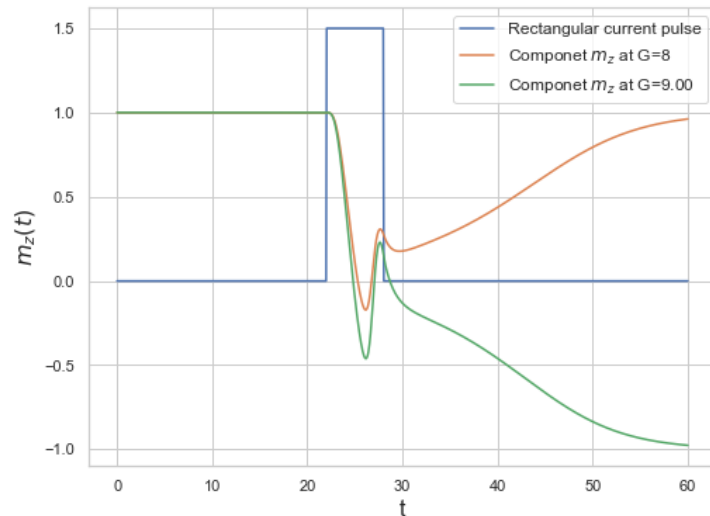
To analyze the possibility of reversing the magnetic moment of the ϕ_0 -Josephson junction at different values of the parameters, we will carry out calculations for $G=8.9$.

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

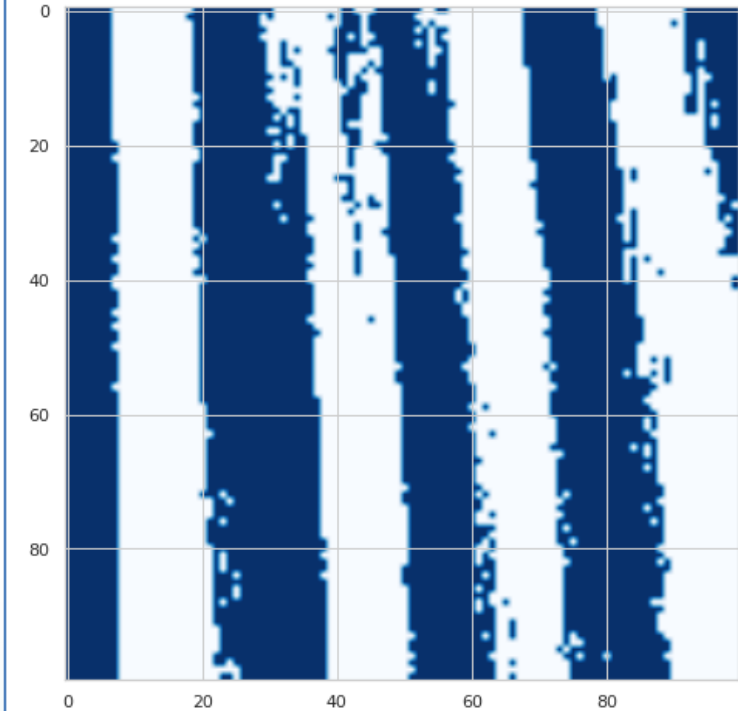
```
G=9
f = partial(my_sfs, G=G, r=r, alpha=alpha, \
            As=As, t_s=t_s, delta_t=delta_t)
#t_e = np.arange(0, 25, 0.0001)
t_e=np.linspace(0,60,1000)

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

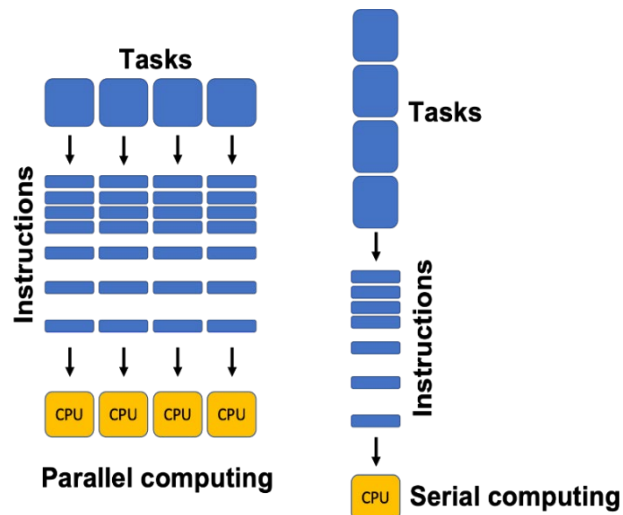
```
plt.figure(figsize = (8, 6))
plt.plot(t_e,y_I, label= 'Rectangular current pulse')
plt.plot(sol_1.t, sol_1.y[2], label= 'Componet $m_z$ $ at G=8' )
plt.plot(sol_2.t, sol_2.y[2], label= 'Componet $m_z$ $ at G=%4.2f' %G)
plt.xlabel('t', size=16)
plt.ylabel('$m_z(t)$', size=16)
plt.legend(fontsize=12)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



```
#plt.figure(figsize = (8, 6))
fig, ax1 = plt.subplots(figsize=(8, 8))
# mask out the negative and positive 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()
```



Example 1. 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

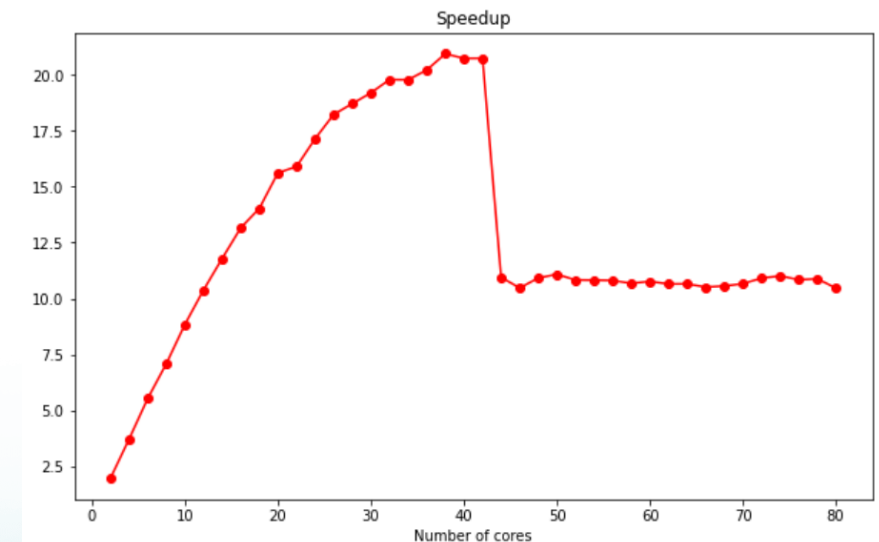
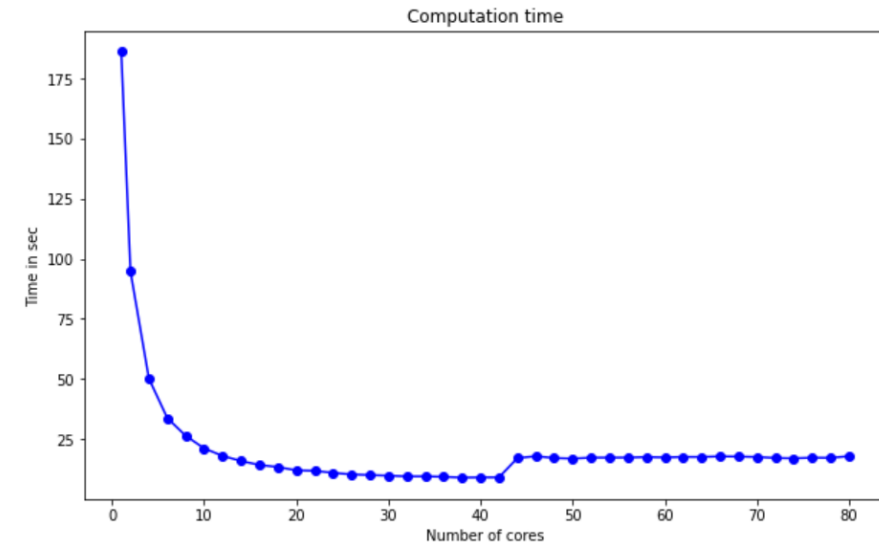
```
t0 = time.time()
rez= Parallel(n_jobs=1)\
    (delayed(funk_parall)(k) for k in range(N*N) )
t1 = time.time()
print(f'Execution time {t1 - t0} s')
```

Execution time 159.9254457950592 s

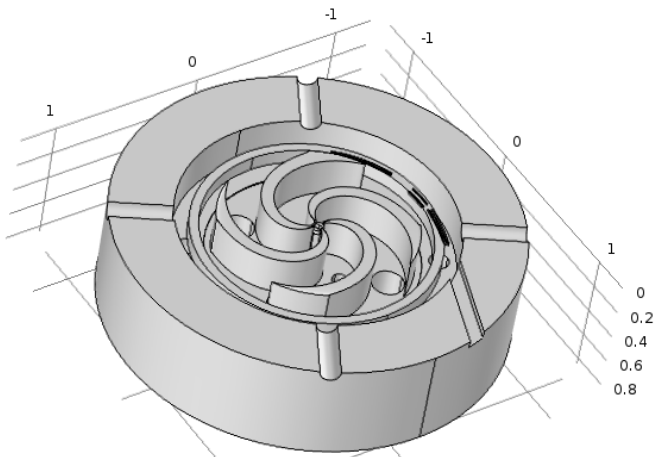
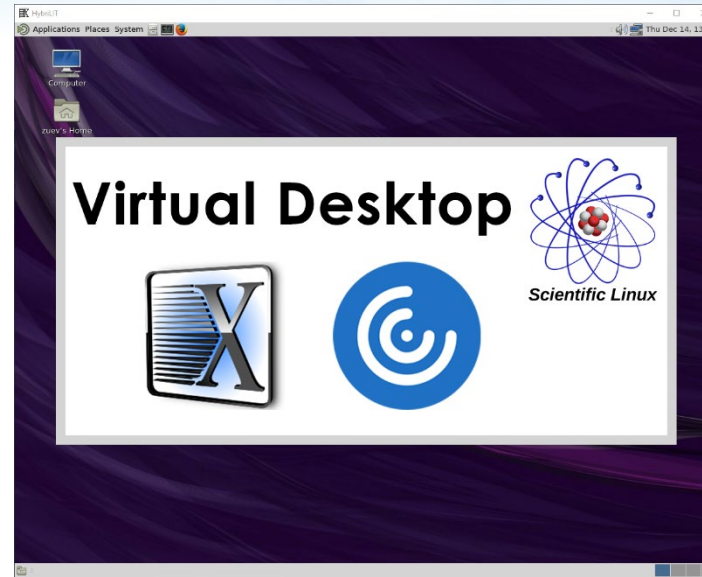
Computing in Parallel Mode

```
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')
```

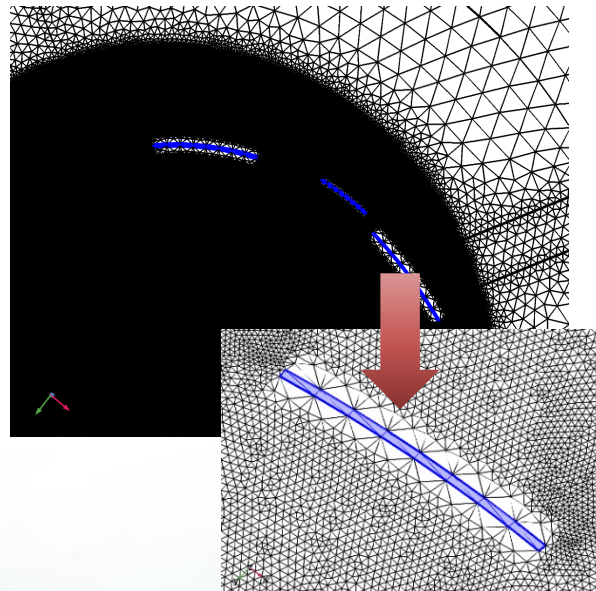
Execution time 34.51503801345825 s



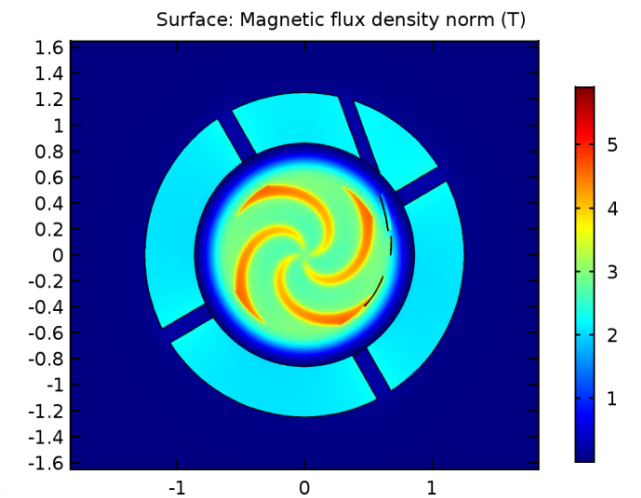
HLIT-VDI – Virtual desktops system



Superconducting magnet SC200 designed for medical application



Computational mesh



Magnetic field distribution at median plane of the magnet

Example 2. MATLAB Integration for Jupyter *



<https://jhub2.jinr.ru>

The screenshot displays the JupyterLab interface with the following components:

- Launcher:** Contains icons for Notebook, Console, and Other. The Notebook icon is highlighted with a red arrow pointing to the MATLAB integration code in the Live Editor.
- Notebook:** Shows a list of files and folders in the left sidebar, including AnyLogic_Models, ayss2016, BUILD, ctxvdi, daos, PROBLEMS-TESTS, SCRIPTS, SOURCE, venv-3.6.8-daos-test, Work, bash_history, etc.tar.xz, export-Python-3.6.8.sh, export-textlive, test-cir.q.py, Untitled.ipynb, and untitled.m.
- Console:** Shows a Python 3 (ipykernel) console.
- Live Editor:** Displays the MATLAB integration code in the Live Editor. The code is as follows:

```
1 x = linspace(-2,2,20);
2 y = x';
3 z = x .* exp(-x.^2 - y.^2);
4 surf(x,y,z)
5 view([-119.307 28.266])
```
- Figure:** Shows a 3D surface plot of the function $z = x \cdot \exp(-x^2 - y^2)$ over the domain $x \in [-2, 2]$ and $y \in [-2, 2]$.
- Workspace:** Shows the variables x , y , and z with their respective values and sizes.
- Command Window:** Shows the MATLAB command prompt `>>`.

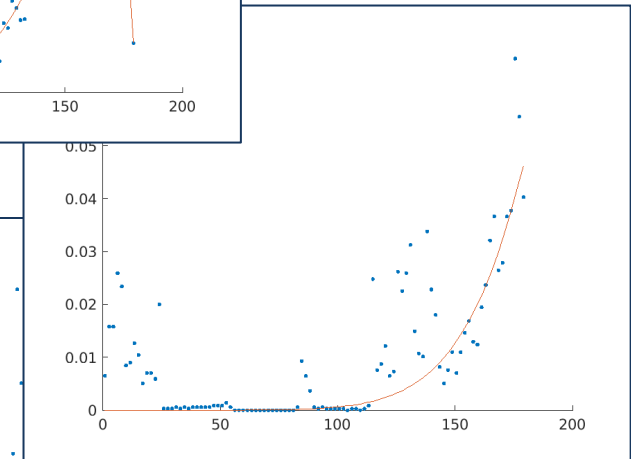
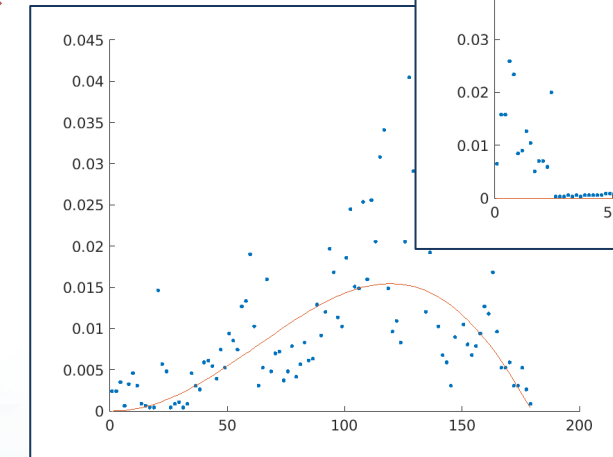
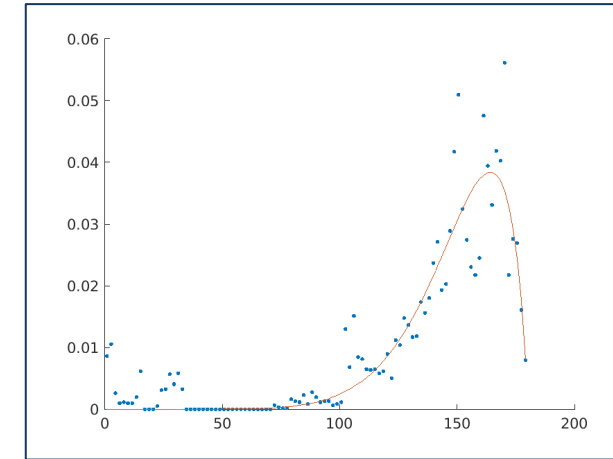
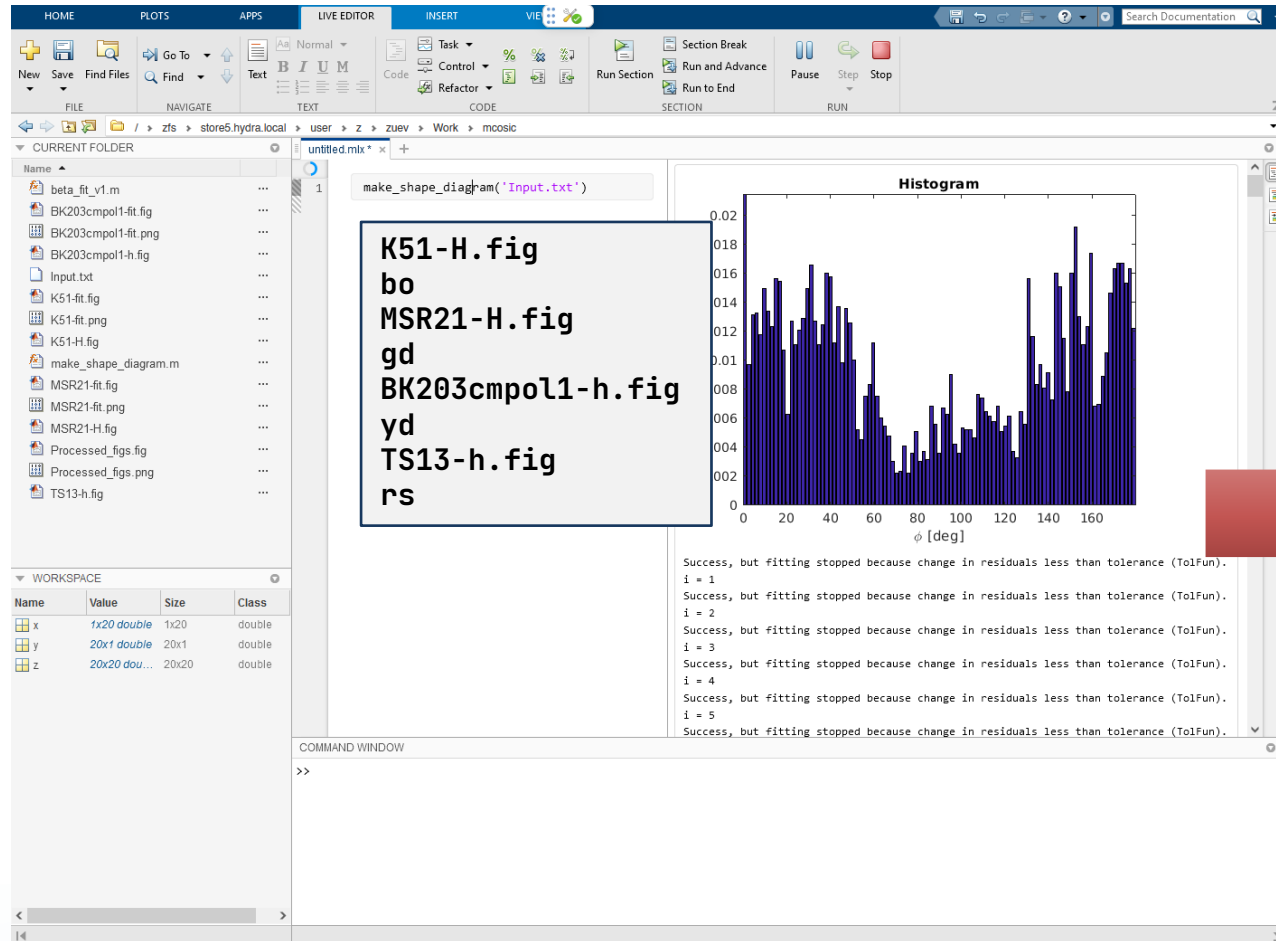
* <https://www.mathworks.com/products/reference-architectures/jupyter.html>

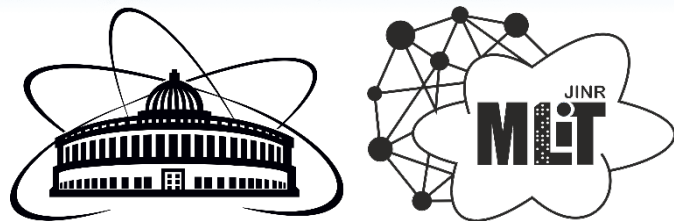
Example 2. MATLAB Integration for Jupyter



Collaboration with Marko Čosić (Laboratory of Physics, Vinča Institute of Nuclear Sciences – National Institute of the Republic of Serbia)

<https://jhub2.jinr.ru>





Thanks for your attention!

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

**The 6th International Workshop on Deep Learning in Computational Physics (DLCP-2022)
Dubna, JINR, 6-8 July 2022**