Analytical Computation Software Module in Python for Automating the Representation of Equations for Further Numerical Modeling of the Chain of Nanomagnets Associated with the Josephson Junction

Majed Nashaat AbdelGhani[1,2], Kirill V. Kulikov[1], Andrey V. Nechaevskiy[3,4], Adiba R. Rahmonova[3,4], Oksana I. Streltsova[3,4], Maxim I. Zuev[4]

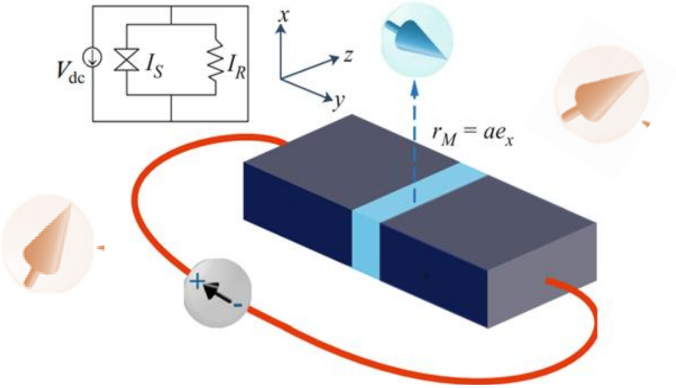
[1] *Bogolyubov Laboratory of Theoretical Physics, JINR, Dubna, Russia*
[2] *Department of Physics, Faculty of Science, Cairo University, Egypt*
[3] *Dubna State University, Dubna, Russia*
[4] *Meshcheryakov Laboratory of Information Technologies JINR, Dubna, Russia*

hYBRI LIT/JINR

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

Research of systems based on Josephson junctions

Symbolic computation block



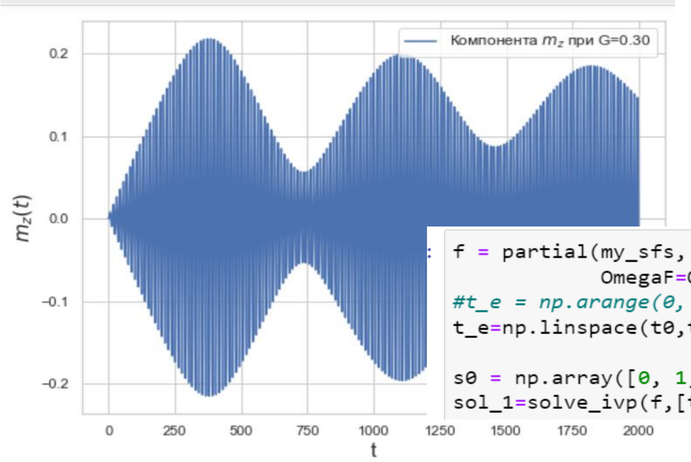
$$\gamma_{m_i} \quad = \quad = -\frac{\mu_0}{2\Phi_0}\int d\mathbf{r}_i \frac{\mathbf{M}_i \times \mathbf{r}_i}{r^3}$$

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

SymPy is a Python library for symbolic mathematics.

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

Block of numerical 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.

Acceleration of multiparameter calculations

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.

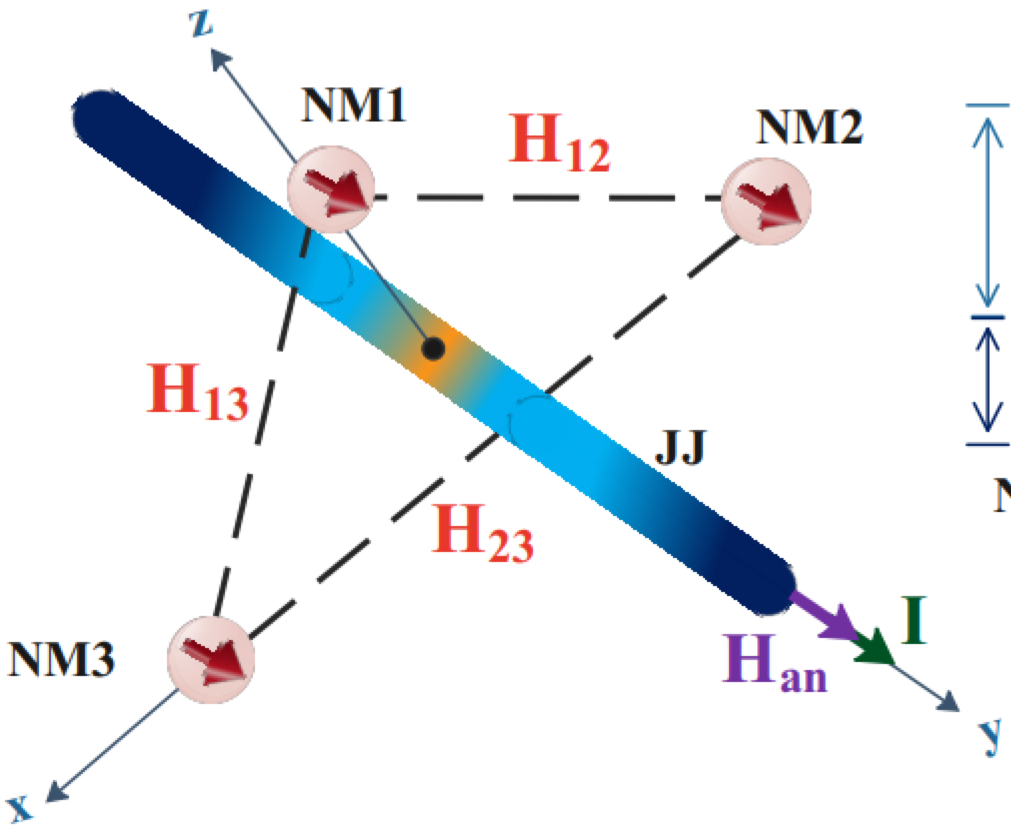
System of nanomagnets and Josephson junction

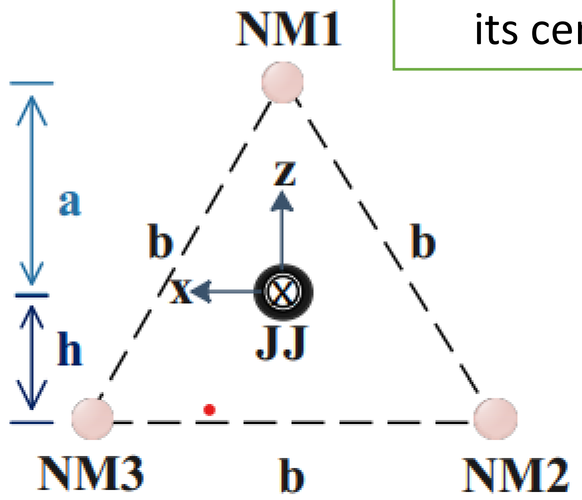
SymPy is a Python library for syml mathematics.

Statement of the problem

Consider:

- a system n nanomagnets;
- Josephson Junction (JJ) is located along the y-axis and its center is at the origin.





For definiteness, we will demonstrate the operation of the software module for $n = 3$

System of nanomagnets and Josephson junction

The dynamics of a system of nanomagnets with JJ is described by the Landau - Lifshitz - Hilbert equations

$$\frac{d\mathbf{m}_1}{dt} = -\frac{\Omega_{F,1}}{1+\alpha^2}\left(\mathbf{m}_1 \times \mathbf{h}_{eff,1} + \alpha\mathbf{m}_1 \times (\mathbf{m}_1 \times \mathbf{h}_{eff,1})\right)$$

$$\frac{d\mathbf{m}_2}{dt} = -\frac{\Omega_{F,2}}{1+\alpha^2}\left(\mathbf{m}_2 \times \mathbf{h}_{eff,2} + \alpha\mathbf{m}_2 \times (\mathbf{m}_2 \times \mathbf{h}_{eff,2})\right)$$

$$\frac{d\mathbf{m}_3}{dt} = -\frac{\Omega_{F,3}}{1+\alpha^2}\left(\mathbf{m}_3 \times \mathbf{h}_{eff,3} + \alpha\mathbf{m}_3 \times (\mathbf{m}_3 \times \mathbf{h}_{eff,3})\right)$$

$\boldsymbol{h_{eff,i}}$, $\boldsymbol{i = 1, \dots, n}$ - effective field

$$h_{eff,i} = h_{ij} + h_{an,i} + h_{J,i} + h_{ext}$$

$\boldsymbol{n = 1, 2, \dots 100, \dots.}$

Calculation of integrals

$$\gamma_{m_i} = -\frac{\mu_i V_{F,i}}{2\Phi_0}\int d\mathbf{r}_i \frac{\mathbf{M}_i \times \mathbf{r}_i}{r^3}$$

System of nanomagnets and Josephson junction

1. Geometry

SymPy is a Python library for symbolic mathematics.

Data structure for radius vectors of nanomagnets:

$$\underbrace{Geom\_NM}_{3xn} = \begin{pmatrix} r_{0,x} & r_{1,}x & \dots & r_{n-1,x} \\ r_{0,y} & r_{1,}y & \dots & r_{n-1,y} \\ r_{0,z} & r_{1,}z & \dots & r_{n-1,z} \end{pmatrix}$$

```
import sympy as sp
import numpy as np
from sympy import *
```

```
a, b, C = sp.symbols('a b C', positive = True, real = True)
```

```
from sympy.vector import CoordSys3D, ParametricRegion, ImplicitRegion, vector_integrate
from sympy.abc import r, x, y, z, theta, phi, t, V, t
#https://docs.sympy.org/latest/modules/vector/vector_integration.html
L, K, S = sp.symbols('L K S', positive = True, real = True)
```

```
n = 3
phi0 = 2*pi/n
phi0
```

$$\frac{2\pi}{3}$$

```
Geom_NM = zeros(3, n)
for i in range(n):
    Geom_NM[0,i] = -a*cos(pi/2-phi0*i)
    Geom_NM[1,i] =  0
    Geom_NM[2,i] = a*sin(pi/2-phi0*i)
```

```
Geom_NM
```

$$\begin{bmatrix} 0 & -\frac{\sqrt{3}a}{2} & \frac{\sqrt{3}a}{2} \\ 0 & 0 & 0 \\ a & -\frac{a}{2} & -\frac{a}{2} \end{bmatrix}$$

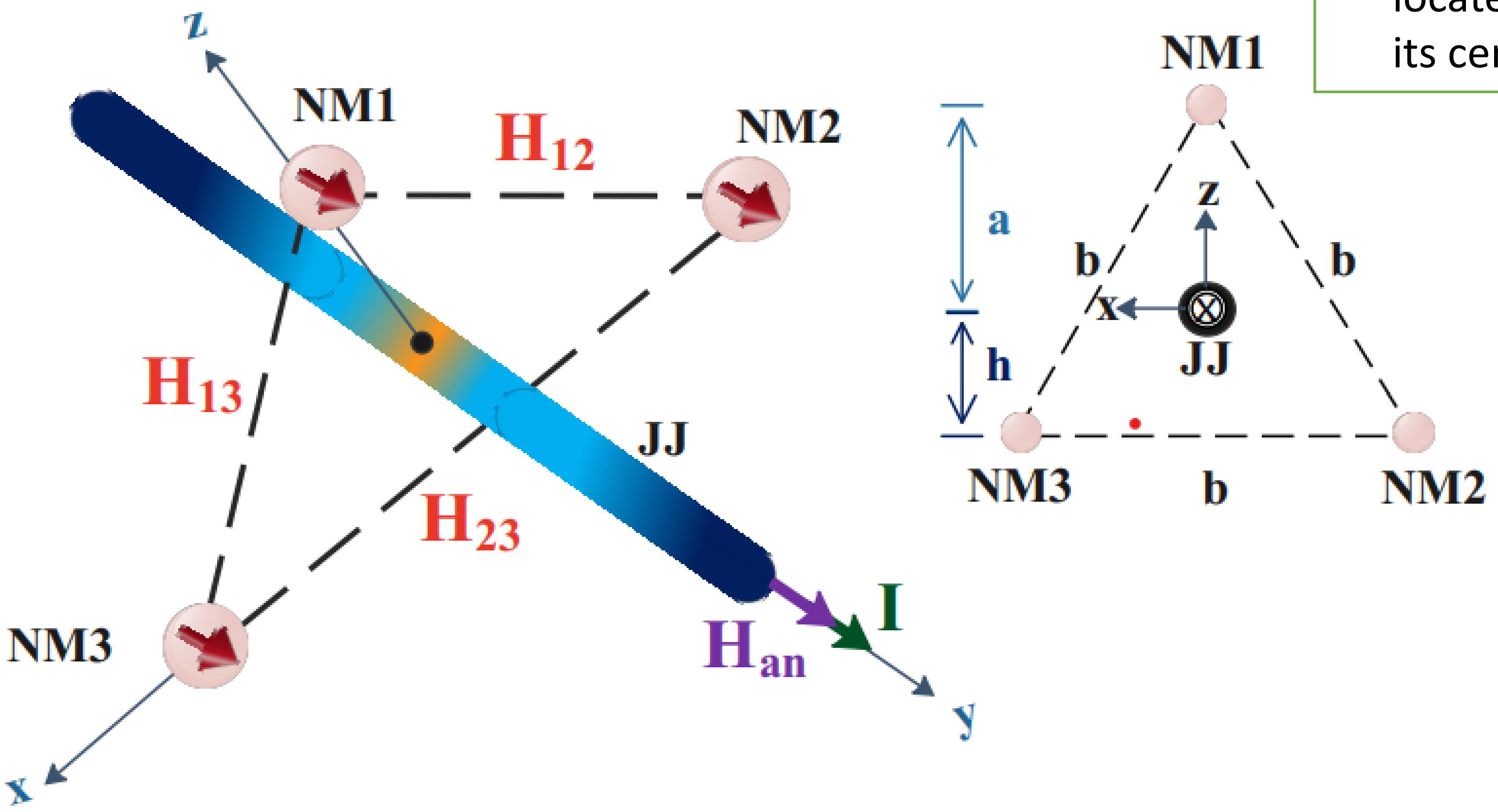
System of nanomagnets and Josephson junction

SymPy is a Python library for syml mathematics.

1. Geometry

Consider:

- a system n nanomagnets;
- Josephson Junction (*JJ*) is located along the y-axis and its center is at the origin.



For definiteness, we will demonstrate the operation of the software module for $n = 3$

System of nanomagnets and Josephson junction

SymPy is a Python library for symbolic mathematics.

1. Geometry

Matrix of distances between nanomagnets *Rij_NM*

$$\underbrace{Rij\_NM}_{3xn^2}\Big|_{n=3} = \left(\begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ r_{00,x}=0 & r_{01,x} & r_{02,x} & r_{10,x} & r_{11,x}=0 & r_{12,x} & r_{20,x} & r_{21,x} & r_{22,x}=0 \\ r_{00,y}=0 & r_{01,y} & r_{02,y} & r_{10,y} & r_{11,y}=0 & r_{12,y} & r_{20,y} & r_{21,y} & r_{22,y}=0 \\ r_{00,z}=0 & r_{01,z} & r_{02,z} & r_{10,z} & r_{11,z}=0 & r_{12,z} & r_{20,z} & r_{21,z} & r_{22,z}=0 \end{array}\right)$$

$$\begin{bmatrix} 0 & -\frac{\sqrt{3}a}{2} & \frac{\sqrt{3}a}{2} & \frac{\sqrt{3}a}{2} & 0 & \sqrt{3}a & -\frac{\sqrt{3}a}{2} & -\sqrt{3}a & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{3a}{2} & -\frac{3a}{2} & \frac{3a}{2} & 0 & 0 & \frac{3a}{2} & 0 & 0 \end{bmatrix}$$

System of nanomagnets and Josephson junction

SymPy is a Python library for symbolic mathematics.

1. Geometry

Matrix of distances between nanomagnets *Rij_NM*

$$\underbrace{Rij\_NM}_{3xn^2}\Big|_{n=3} = \begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} r_{00,x}=0 & r_{01,x} & r_{02,x} & r_{10,x} & r_{11,x}=0 & r_{12,x} & r_{20,x} & r_{21,x} & r_{22,x}=0 \\ r_{00,y}=0 & r_{01,y} & r_{02,y} & r_{10,y} & r_{11,y}=0 & r_{12,y} & r_{20,y} & r_{21,y} & r_{22,y}=0 \\ r_{00,z}=0 & r_{01,z} & r_{02,z} & r_{10,z} & r_{11,z}=0 & r_{12,z} & r_{20,z} & r_{21,z} & r_{22,z}=0 \end{pmatrix}$$

```python
for i in range(n):
    for j in range(n):
        Rij_NM[:,j+n*i] = Geom_NM[:,j] -Geom_NM[:,i]
        print(i,j, j+n*i)
```

$$\begin{bmatrix} 0 & -\frac{\sqrt{3}a}{2} & \frac{\sqrt{3}a}{2} & \frac{\sqrt{3}a}{2} & 0 & \sqrt{3}a & -\frac{\sqrt{3}a}{2} & -\sqrt{3}a & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{3a}{2} & -\frac{3a}{2} & \frac{3a}{2} & 0 & 0 & \frac{3a}{2} & 0 & 0 \end{bmatrix}$$

System of nanomagnets and Josephson junction

2. Model parameters

Constants:

- α is the Gilbert damping constant
- $\omega_c = \frac{2\pi}{\Phi_0} I_c R$ - timescale t is unitless and is reduced
- ϵ_J

Let's introduce a **row matrix**, dimensions $(1 \times n)$:

- $\gamma = [\gamma_0, \gamma_1, \ldots, \gamma_{n-1}]$ is the gyromagnetic ratio for all NM,
- $M_0 = [M_{0,0}, M_{0,1}, \ldots, M_{0,n-1}]$ is the saturation magnetization of a simulated ferromagnet
- $\mu = [\mu_0, \mu_1, \ldots, \mu_{n-1}]$ is permeability (H/m)
- $V_F = [V_{F,0}, V_{F,1}, \ldots, V_{F,n-1}]$ is volumes of NM
- $K_{an} = [K_{an,0}, K_{an,1}, \ldots, K_{an,n-1}]$ is anisotropy

Designation row matrix:

- $\Omega_F = [\Omega_{F,0}, \Omega_{F,1}, \ldots, \Omega_{F,n-1}]$
- $C = [C_0, C_1, \ldots, C_{n-1}]$
- $\epsilon = [\epsilon_0, \epsilon_1, \ldots, \epsilon_{n-1}]$

$$\Omega_{F,i} = \frac{\mu_i M_{0,i} \gamma_i}{\omega_c}, i = 0, 1, \ldots, n-1.$$

$$C_i = \frac{V_{F,i}}{4\pi}, i = 0, 1, \ldots, n-1.$$

$$\epsilon_i = \frac{\epsilon_J}{\mu_i V_{F,i} M_{0,i}^2}, i = 0, 1, \ldots, n-1.$$

```
epsilon = zeros(1, n)
for i in range(n):
    epsilon[i] = epsilon_J/(mu[i]*M0_NM[i]*M0_NM[i]*V_F[i])
epsilon
```

$$\left[\begin{matrix}\frac{\epsilon_J}{M_{00}^2 V_{F0} \mu_0} & \frac{\epsilon_J}{M_{01}^2 V_{F1} \mu_1} & \frac{\epsilon_J}{M_{02}^2 V_{F2} \mu_2}\end{matrix}\right]$$

SymPy is a Python library for symbolic mathematics.

3. Required vectors

Magnetization vector (normalized) NM: *M_NM*

```
M_NM=zeros(3, n)
for i in range(n):
    M_NM[0,i] = Function(symbols("m"+ str(0)+str(i)))(t)
    M_NM[1,i] = Function(symbols("m"+ str(1)+str(i)))(t)
    M_NM[2,i] = Function(symbols("m"+ str(2)+str(i)))(t)
M_NM
```

Functions

$$\begin{bmatrix} \mathrm{m}_{00}(t) & \mathrm{m}_{01}(t) & \mathrm{m}_{02}(t) \\ \mathrm{m}_{10}(t) & \mathrm{m}_{11}(t) & \mathrm{m}_{12}(t) \\ \mathrm{m}_{20}(t) & \mathrm{m}_{21}(t) & \mathrm{m}_{22}(t) \end{bmatrix}$$

System of nanomagnets and Josephson junction

4. Effective field

SymPy is a Python library for symbolic mathematics.

The effective field is consist of 4 terms:

$$\boxed{h_{eff,i} = h_{ij} + h_{an,i} + h_{J,i} + h_{ext}}$$

where

- h_{ij} - is an effective field duo to the dipole interaction,
- $h_{an,i}$ - is an effective field duo to the magnetic anisotropy,
- $h_{J,i}$ - is an effective field duo to the current through JJ and
- h_{ext} - is an external magnetic field, which is in our case zero ($h_{ext} = 0$).

System of nanomagnets and Josephson junction

4. Effective field

Finding magnetizations

h_{ij} is an effective field duo to the dipole interaction

The magnetic field of a magnetic dipole in vector notation is:

$$B_{ij}(r_{ij}, m_i) = \frac{V_{F,i}\mu_i}{4\pi}(\frac{3(m_i \cdot \mathbf{r})\mathbf{r}}{b^5} - \frac{m_i}{b^3})$$

where H is the field, r is the vector from the position of the dipole to the position where the field is being measured, b is the distance between nanomagnets, m_i is the magnetization vector, μ_i is the permeability.

Data structure

```
Bij_NM=zeros(3, n*n)
Bij_NM
```

```
for i in range(n):
    for j in range(n):
        if(i != j ):
            b = sqrt( Rij_NM[:,j+n*i].dot(Rij_NM[:,j+n*i]))
            Bij_NM[:,j+n*i] = C[i]*mu[i] *( 3* (M_NM[:,i].dot(Rij_NM[:,j+n*i]) )/b**5*Rij_NM[:,j+n*i] -M_NM[:,i]/b**3  )
Bij_NM = simplify(Bij_NM)
Bij_NM
```

System of nanomagnets and Josephson junction

4. Effective field

```
for i in range(n):
    for j in range(n):
        if(i != j ):
            b = sqrt( Rij_NM[:,j+n*i].dot(Rij_NM[:,j+n*i]))
            Bij_NM[:,j+n*i] = C[i]*mu[i] *( 3* (M_NM[:,i].dot(Rij_NM[:,j+n*i]) )/b**5*Rij_NM[:,j+n*i] -M_NM[:,i]/b**3  )
Bij_NM = simplify(Bij_NM)
Bij_NM
```

$$\left[\begin{matrix}0 & \frac{V_{F0}\mu_{0}\left(-\sqrt{3}m_{00}(t)+9m_{20}(t)\right)}{144\pi a^{3}} & -\frac{V_{F0}\mu_{0}\left(\sqrt{3}m_{00}(t)+9m_{20}(t)\right)}{144\pi a^{3}} & \frac{V_{F1}\mu_{1}\left(-\sqrt{3}m_{01}(t)+9m_{21}(t)\right)}{144\pi a^{3}} & 0 & \frac{\sqrt{3}V_{F1}\mu_{1}m_{01}(t)}{18\pi a^{3}} & -\frac{V_{F2}\mu_{2}\left(\sqrt{3}m_{02}(t)+9m_{22}(t)\right)}{144\pi a^{3}} & \frac{\sqrt{3}V_{F2}\mu_{2}m_{02}(t)}{18\pi a^{3}} & 0\\0 & -\frac{\sqrt{3}V_{F0}\mu_{0}m_{10}(t)}{36\pi a^{3}} & -\frac{\sqrt{3}V_{F0}\mu_{0}m_{10}(t)}{36\pi a^{3}} & -\frac{\sqrt{3}V_{F1}\mu_{1}m_{11}(t)}{36\pi a^{3}} & 0 & -\frac{\sqrt{3}V_{F1}\mu_{1}m_{11}(t)}{36\pi a^{3}} & -\frac{\sqrt{3}V_{F2}\mu_{2}m_{12}(t)}{36\pi a^{3}} & -\frac{\sqrt{3}V_{F2}\mu_{2}m_{12}(t)}{36\pi a^{3}} & 0\\0 & \frac{V_{F0}\mu_{0}\left(9m_{00}(t)+5\sqrt{3}m_{20}(t)\right)}{144\pi a^{3}} & \frac{V_{F0}\mu_{0}\left(-9m_{00}(t)+5\sqrt{3}m_{20}(t)\right)}{144\pi a^{3}} & \frac{V_{F1}\mu_{1}\left(9m_{01}(t)+5\sqrt{3}m_{21}(t)\right)}{144\pi a^{3}} & 0 & -\frac{\sqrt{3}V_{F1}\mu_{1}m_{21}(t)}{36\pi a^{3}} & \frac{V_{F2}\mu_{2}\left(-9m_{02}(t)+5\sqrt{3}m_{22}(t)\right)}{144\pi a^{3}} & -\frac{\sqrt{3}V_{F2}\mu_{2}m_{22}(t)}{36\pi a^{3}} & 0\end{matrix}\right]$$

System of nanomagnets and Josephson junction

4. Effective field

```
for i in range(n):
    for j in range(n):
        if(i != j ):
            b = sqrt( Rij_NM[:,j+n*i].dot(Rij_NM[:,j+n*i]))
            Bij_NM[:,j+n*i] = C[i]*mu[i] *( 3* (M_NM[:,i].dot(Rij_NM[:,j+n*i]) )/b**5*Rij_NM[:,j+n*i] -M_NM[:,i]/b**3  )
Bij_NM = simplify(Bij_NM)
Bij_NM
```

$$\left[\begin{matrix}0 & \frac{V_{F0}\mu_0\left(-\sqrt{3}m_{00}(t)+9m_{20}(t)\right)}{144\pi a^3} & -\frac{V_{F0}\mu_0\left(\sqrt{3}m_{00}(t)+9m_{20}(t)\right)}{144\pi a^3} & \frac{V_{F1}\mu_1\left(-\sqrt{3}m_{01}(t)+9m_{21}(t)\right)}{144\pi a^3} & 0 & \frac{\sqrt{3}V_{F1}\mu_1 m_{01}(t)}{18\pi a^3} & -\frac{V_{F2}\mu_2\left(\sqrt{3}m_{02}(t)+9m_{22}(t)\right)}{144\pi a^3} & \frac{\sqrt{3}V_{F2}\mu_2 m_{02}(t)}{18\pi a^3} & 0\\0 & -\frac{\sqrt{3}V_{F0}\mu_0 m_{10}(t)}{36\pi a^3} & -\frac{\sqrt{3}V_{F0}\mu_0 m_{10}(t)}{36\pi a^3} & -\frac{\sqrt{3}V_{F1}\mu_1 m_{11}(t)}{36\pi a^3} & 0 & -\frac{\sqrt{3}V_{F1}\mu_1 m_{11}(t)}{36\pi a^3} & -\frac{\sqrt{3}V_{F2}\mu_2 m_{12}(t)}{36\pi a^3} & -\frac{\sqrt{3}V_{F2}\mu_2 m_{12}(t)}{36\pi a^3} & 0\\0 & \frac{V_{F0}\mu_0\left(9m_{00}(t)+5\sqrt{3}m_{20}(t)\right)}{144\pi a^3} & \frac{V_{F0}\mu_0\left(-9m_{00}(t)+5\sqrt{3}m_{20}(t)\right)}{144\pi a^3} & \frac{V_{F1}\mu_1\left(9m_{01}(t)+5\sqrt{3}m_{21}(t)\right)}{144\pi a^3} & 0 & -\frac{\sqrt{3}V_{F1}\mu_1 m_{21}(t)}{36\pi a^3} & \frac{V_{F2}\mu_2\left(-9m_{02}(t)+5\sqrt{3}m_{22}(t)\right)}{144\pi a^3} & -\frac{\sqrt{3}V_{F2}\mu_2 m_{22}(t)}{36\pi a^3} & 0\end{matrix}\right]$$

$$h_{ij}(r_{ij}, m_i) = C_i\left(\frac{3(m_i\cdot\mathbf{r})\mathbf{r}}{b^2} - m_i\right)$$

where $C_i = \dfrac{V_{F,i}\mu_i M_{0,i}}{4\pi\mu_1 M_{0,1} b^3}$ - is a dimensionless coefficient.

```
for i in range(n):
    for j in range(n):
        if(i != j ):
            hij_NM[:,i] =  hij_NM[:,i] +  Bij_NM[:,i+n*j]


hij_NM= simplify(hij_NM)
hij_NM
```

4. Effective field

$h_{an,i}$ - is an effective field duo to the magnetic anisotropy

The magnetic anisotropy field is given by:

$$h_{an,i} = \tilde{K}_{an,i} m_{y,i},$$

where

$$\tilde{K}_{an,i} = \frac{K_{an,i}}{\mu_i M_{0,i}^2}, i = 0, 1, \ldots, n-1.$$

```
for i in range(n):
    h_an[1,i] = K_an[i] * M_NM[1,i]/(mu[i] *  M0_NM[i]*  M0_NM[i])

h_an
```

$$\begin{bmatrix} 0 & 0 & 0 \\ \frac{K_{an0} m_{10}(t)}{M_{00}^2 \mu_0} & \frac{K_{an1} m_{11}(t)}{M_{01}^2 \mu_1} & \frac{K_{an2} m_{12}(t)}{M_{02}^2 \mu_2} \\ 0 & 0 & 0 \end{bmatrix}$$

System of nanomagnets and Josephson junction

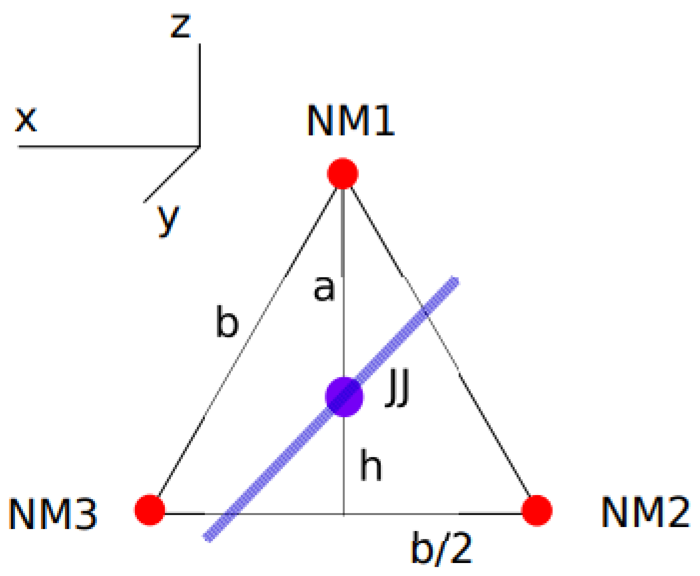
4. Effective field

$\boldsymbol{h_{J,i}}$ - is an effective field duo to the current through *JJ*

The superconductive phase shift γ_{m_1,m_2} caused by the nanomagnets is given by

$$\gamma_{m_i} = -\frac{\mu_i V_{F,i}}{2\Phi_0}\int d\mathbf{r}_i \frac{\mathbf{M}_i \times \mathbf{r}_i}{r^3}$$

I. Polygon




```python
from sympy.vector import CoordSys3D, ParametricRegion, ImplicitRegion, vector_integrate
from sympy.abc import r, x, y, z, theta, phi, t, V, t
#https://docs.sympy.org/latest/modules/vector/vector_integration.html
L, K, S = sp.symbols('L K S', positive = True, real = True)
```

```python
tp = sp.symbols('tp', positive = True, real = True)
```

```python
from sympy.geometry import Point, Polygon
curve = ParametricRegion( (0, tp, 0),  (tp, -L/2, L/2))
C = CoordSys3D('C')
```

System of nanomagnets and Josephson junction

4. Effective field

$\boldsymbol{h_{J,i}}$ - is an effective field duo to the current through *JJ*

The superconductive phase shift γ_{m_1,m_2} caused by the nanomagnets is given by

$$\gamma_{m_i} = -\frac{\mu_i V_{F,i}}{2\Phi_0}\int d\mathbf{r}_i \frac{\mathbf{M}_i \times \mathbf{r}_i}{r^3}$$

II. Vector integrate

```
Am_vec1 = -K*nm.cross(r1m)/sqrt(r1m.dot(r1m))**3
Am_vec1
```

$$\left(-\frac{K\left(-\mathbf{y_C} m_{20}(t) - a m_{10}(t)\right)}{\left(\mathbf{y_C}^2 + a^2\right)^{\frac{3}{2}}}\right)\hat{\mathbf{i}}_{\mathbf{C}} + \left(-\frac{K a m_{00}(t)}{\left(\mathbf{y_C}^2 + a^2\right)^{\frac{3}{2}}}\right)\hat{\mathbf{j}}_{\mathbf{C}} + \left(-\frac{\mathbf{y_C} K m_{00}(t)}{\left(\mathbf{y_C}^2 + a^2\right)^{\frac{3}{2}}}\right)\hat{\mathbf{k}}_{\mathbf{C}}$$

Data structure:

$\gamma = [\gamma_{0,}\gamma_{1,}\gamma_{2,}]$

```
for i in range(n):
    r1m=  (0-Geom_NM[0,i])*C.i+ (C.y -Geom_NM[1,i] )*C.j + (0-Geom_NM[2,i]) *C.k
    nm = M_NM[0,i]*C.i+ M_NM[1,i]*C.j+M_NM[2,i]*C.k
    Am_vec1 = -mu[i]* V_F[i]/(2*Phi_0) *nm.cross(r1m)/sqrt(r1m.dot(r1m))**3
    gamma_m[i] = vector_integrate(Am_vec1, curve)
gamma_m
```

$$\left[-\frac{L V_{F0}\mu_0\, m_{00}(t)}{2\Phi_0 a\sqrt{L^2+a^2}} \quad \frac{L V_{F1}\mu_1\left(m_{01}(t)-\sqrt{3}\, m_{21}(t)\right)}{4\Phi_0 a\sqrt{L^2+a^2}} \quad \frac{L V_{F2}\mu_2\left(m_{02}(t)+\sqrt{3}\, m_{22}(t)\right)}{4\Phi_0 a\sqrt{L^2+a^2}}\right]$$

System of nanomagnets and Josephson junction

5. The current flowing through the JJ

The current flowing through the JJ is given by

$$I \quad = \quad \sin[Vt + \gamma_{m_1} + \gamma_{m_2} + \gamma_{m_3}] + V + \dot{\gamma}_{m_1} + \dot{\gamma}_{m_2} + \dot{\gamma}_{m_3}.$$

$\boldsymbol{h_{J,i}}$ - is an effective field duo to the current through *JJ*

Data structure:
$\gamma = [\gamma_{0,}\gamma_{1,}\gamma_{2,}]$

```
for i in range(n):
    r1m=  (0-Geom_NM[0,i])*C.i+ (C.y -Geom_NM[1,i] )*C.j + (0-Geom_NM[2,i]) *C.k
    nm = M_NM[0,i]*C.i+ M_NM[1,i]*C.j+M_NM[2,i]*C.k
    Am_vec1 = -mu[i]* V_F[i]/(2*Phi_0) *nm.cross(r1m)/sqrt(r1m.dot(r1m))**3
    gamma_m[i] = vector_integrate(Am_vec1, curve)
gamma_m
```

$$\left[-\frac{LV_{F0}\mu_0 \mathrm{m}_{00}(t)}{2\Phi_0 a\sqrt{L^2+a^2}} \quad \frac{LV_{F1}\mu_1\left(\mathrm{m}_{01}(t) - \sqrt{3}\,\mathrm{m}_{21}(t)\right)}{4\Phi_0 a\sqrt{L^2+a^2}} \quad \frac{LV_{F2}\mu_2\left(\mathrm{m}_{02}(t) + \sqrt{3}\,\mathrm{m}_{22}(t)\right)}{4\Phi_0 a\sqrt{L^2+a^2}} \right]$$

System of nanomagnets and Josephson junction

6. LLH system of equations

To further solve the Cauchy problem (**IVP**) numerically, it is necessary to reduce the system of ordinary differential equations to the form:

$$\frac{d\vec{y}}{dt} = F(t, \vec{y})$$

$$\frac{d\mathbf{m}_i}{dt} = -\frac{\Omega_{F,i}}{1+\alpha^2}\Big(\mathbf{m}_i \times \mathbf{h}_{eff,i} + \alpha \mathbf{m}_i \times (\mathbf{m}_i \times \mathbf{h}_{eff,i})\Big), i = 0, 1, \dots, n-1.$$

During the numerical calculation, a system of linear algebraic equations is solved at each time step

Note:

$\frac{d\vec{m_i}}{dt}$ in $h_{eff,j}$

Conclusion

To study the chain of nanomagnets associated with the Josephson junction, a **software module** has been developed to output equations in symbolic form for further numerical modeling.

In this case, the equations are reduced to the form of a system of ordinary differential equations resolved with respect to the derivative at each integration step.

The implementation is carried out using the **SymPy** library for symbolic operations.