Micromagnetic simulations for magnonics

Ádám Papp

Pázmány Péter Catholic University, Budapest, Hungary

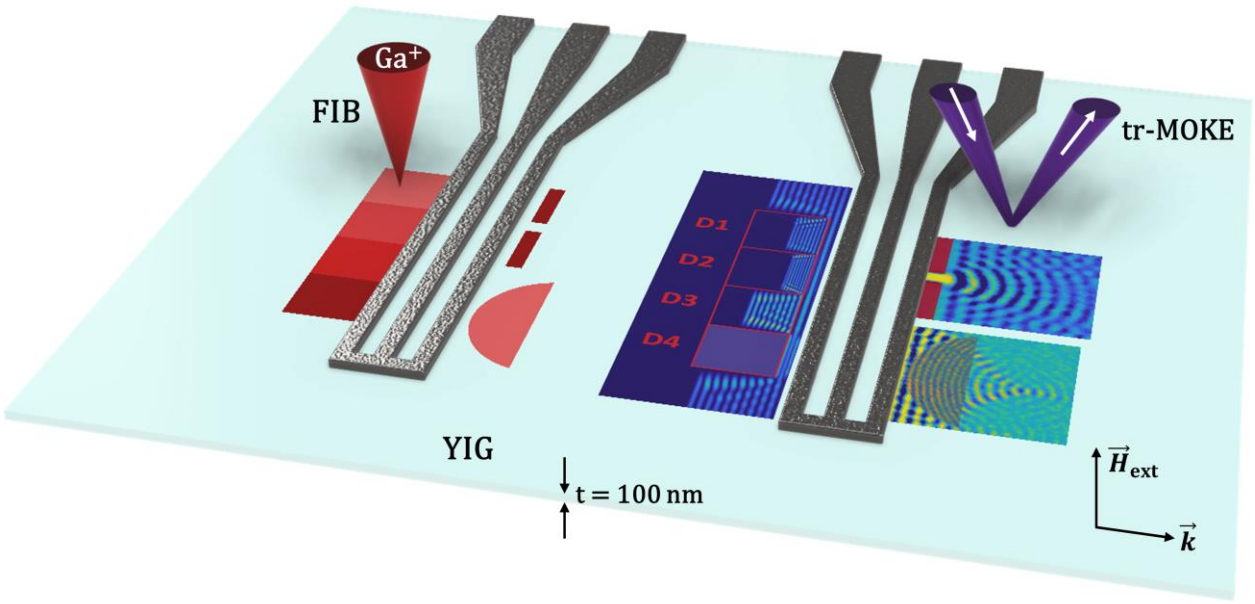
Contact: papp.adam@itk.ppke.hu

PÁZMÁNY PÉTER
CATHOLIC UNIVERSITY

UNIVERSITY OF
NOTRE DAME

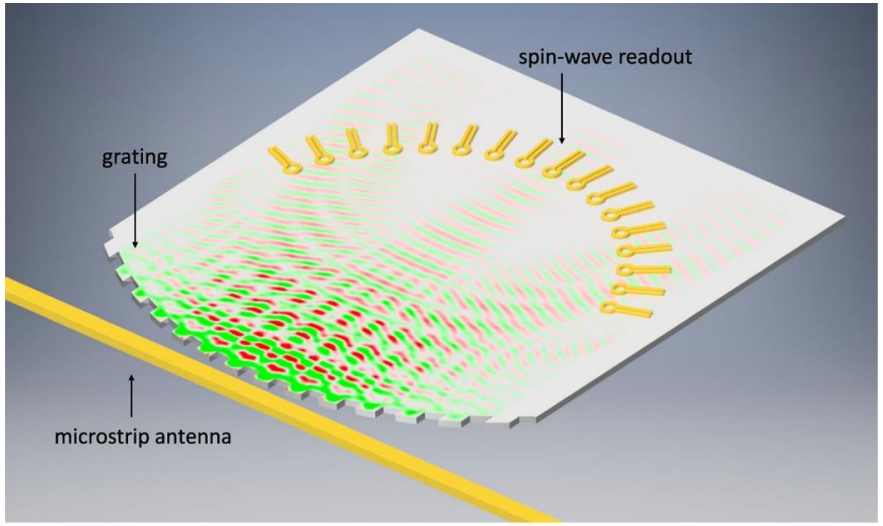
Motivation #1: Spin-wave optics

Focused-ion-beam irradiation of YIG



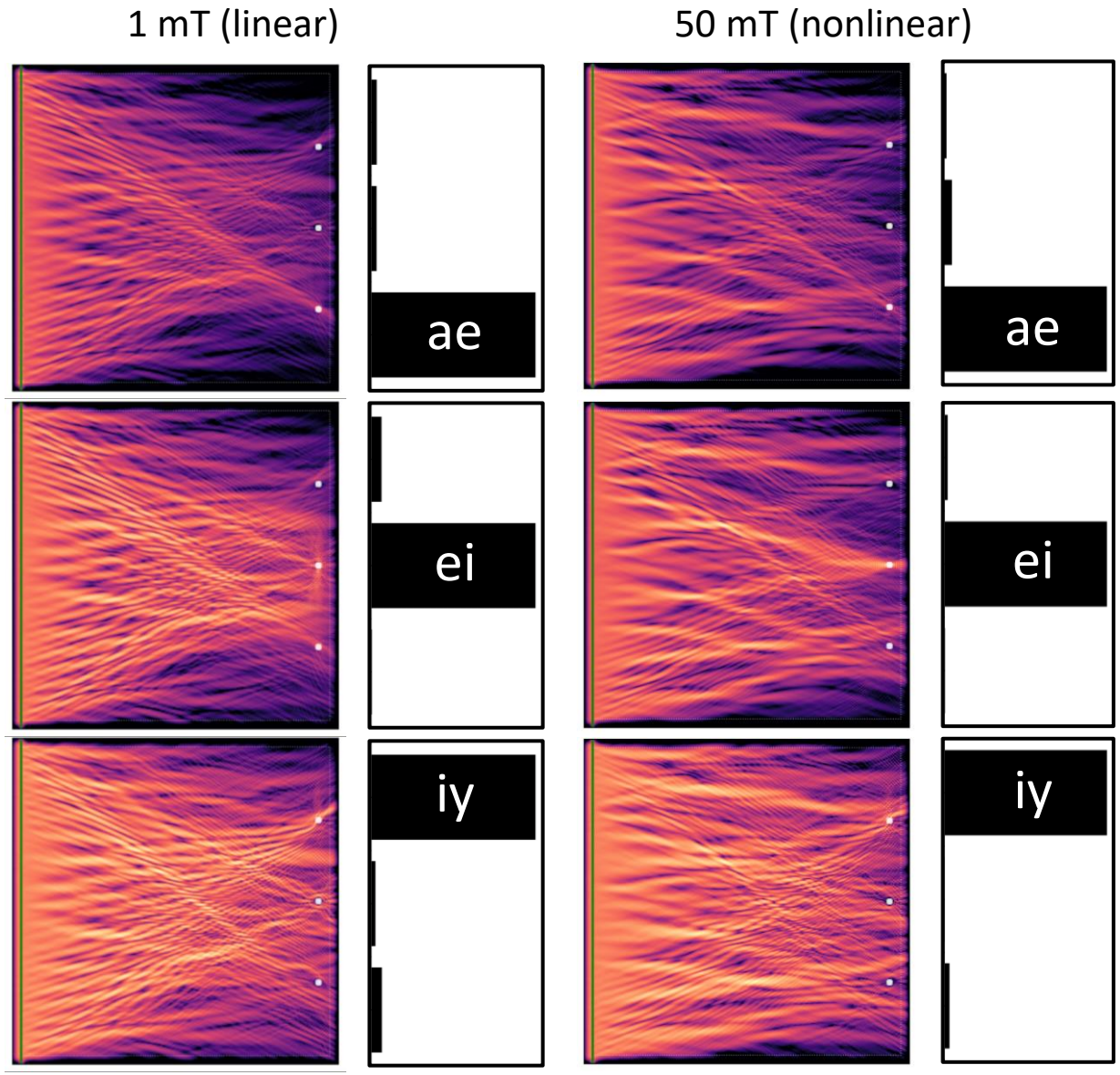
Spin-Wave Optics in YIG Realized by Ion-Beam Irradiation. *Small* 2023, 19, 2207293.

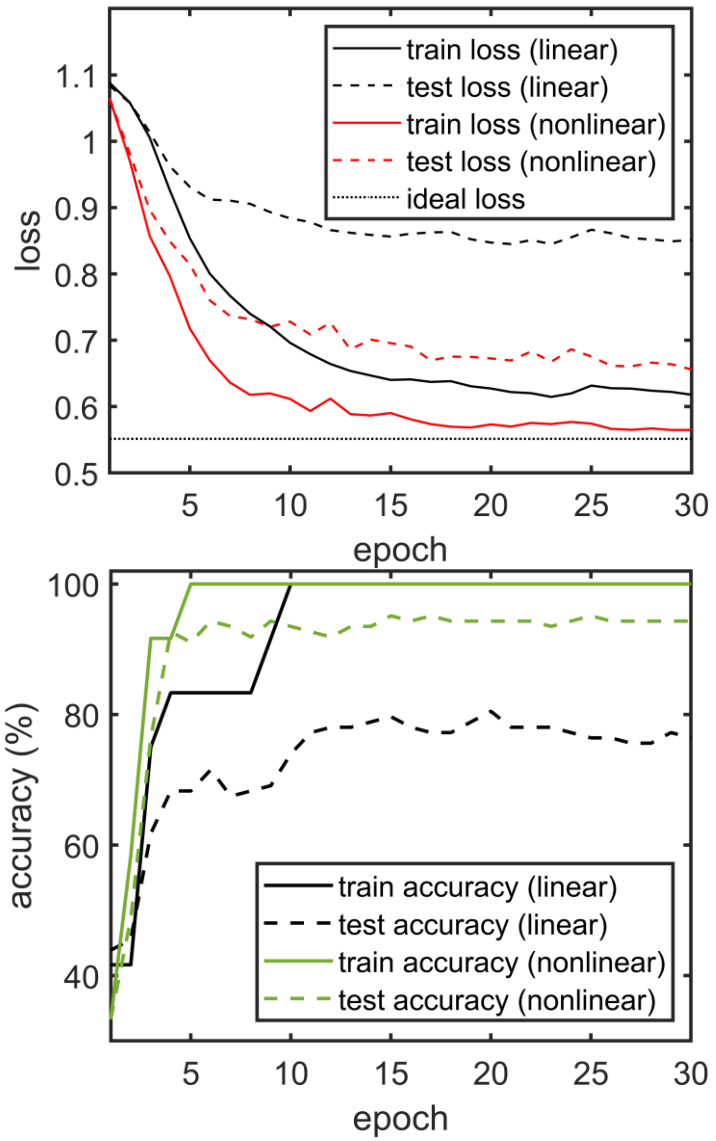
Spin-wave-based microwave spectrum analyzer



Nanoscale spectrum analyzer based on spin-wave interference. *Sci Rep* **7,** 9245 (2017).

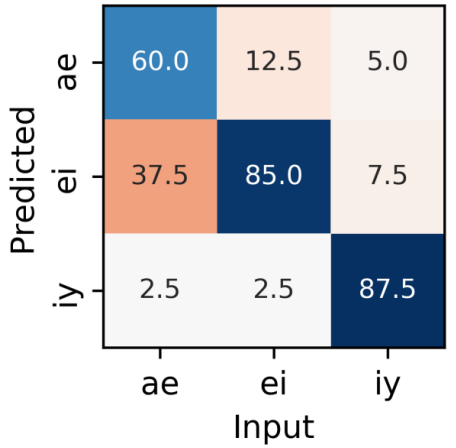
Motivation #2: Neural networks based on spin waves



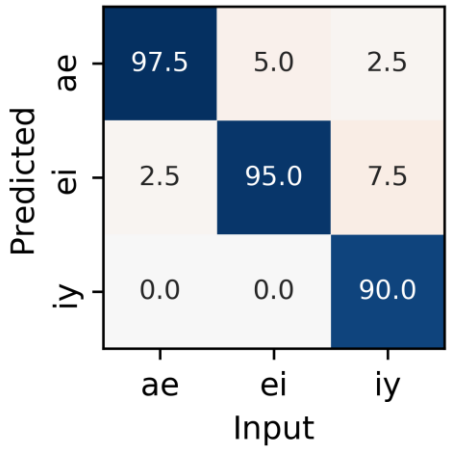


Confusion matrices:

linear



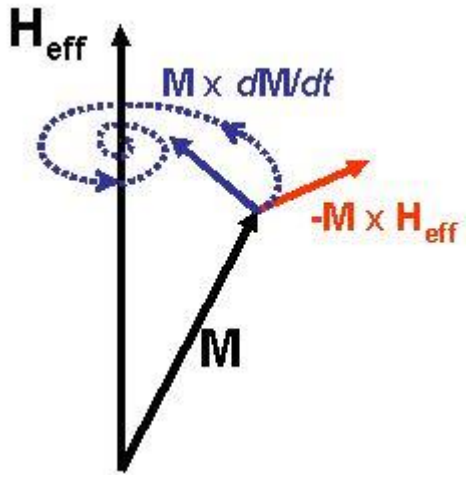
nonlinear



Papp, Á. et al., Nanoscale neural network using non-linear spin-wave interference. *Nat Commun* **12,** 6422 (2021).

LLG equation

$$\frac{\partial \vec{\mathbf{m}}}{\partial t} = \vec{\tau}$$



$$\vec{\tau}_{\mathrm{LL}} = \gamma_{\mathrm{LL}} \frac{1}{1+\alpha^2} \left(\vec{\mathbf{m}} \times \vec{\mathbf{B}}_{\mathrm{eff}} + \alpha \left(\vec{\mathbf{m}} \times \left(\vec{\mathbf{m}} \times \vec{\mathbf{B}}_{\mathrm{eff}} \right) \right) \right)$$

$$\vec{\mathbf{B}}_{\mathrm{ext}}$$

$$\vec{\mathbf{B}}_{\mathrm{demag}\ i} = \hat{\mathbf{K}}_{ij} * \vec{\mathbf{M}}_j$$

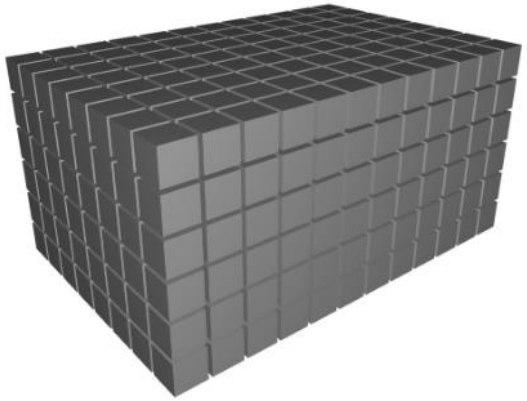
convolution by FFT
-> GPU!

$$\vec{\mathbf{B}}_{\mathrm{exch}} = 2 \frac{A_{\mathrm{ex}}}{M_{\mathrm{sat}}} \Delta \vec{\mathbf{m}}$$

small-angle approx.

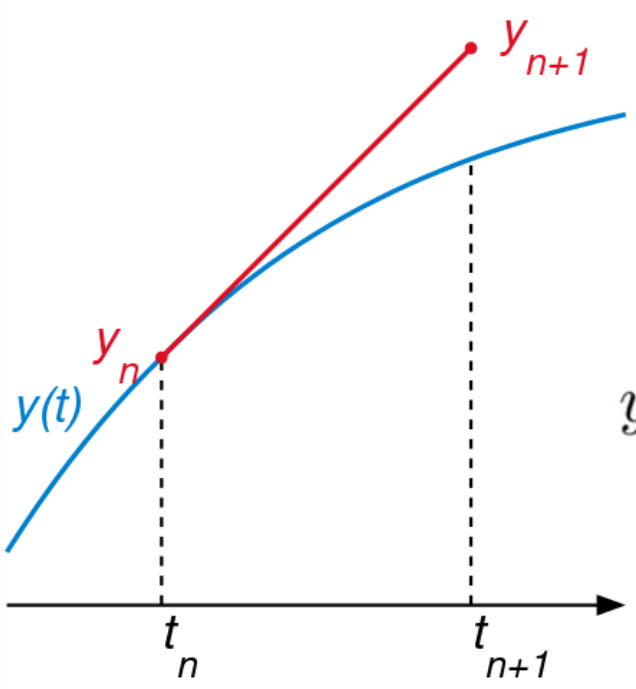
$$\vec{\mathbf{B}}_{\mathrm{exch}} = 2 \frac{A_{\mathrm{ex}}}{M_{\mathrm{sat}}} \sum_i \frac{(\vec{\mathbf{m}}_i - \vec{\mathbf{m}})}{\Delta_i^2}$$

FDTD method



Solution is calculated on a discrete grid, starting with an initial condition and marching in time.

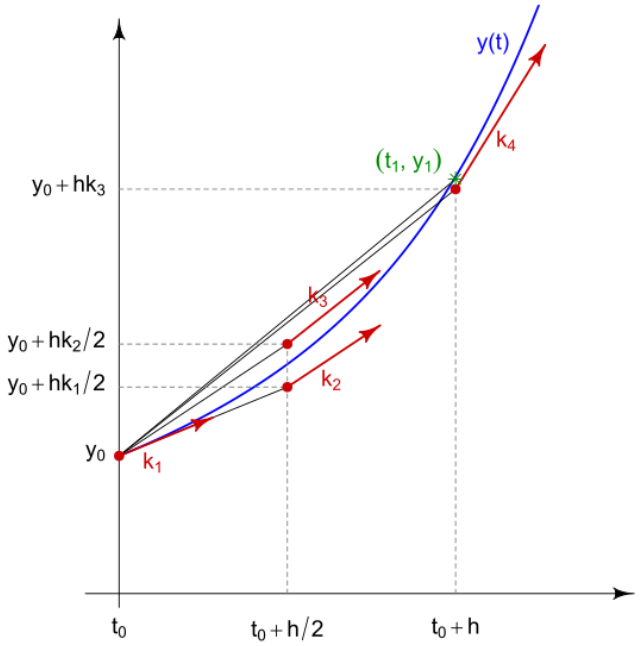
Euler method



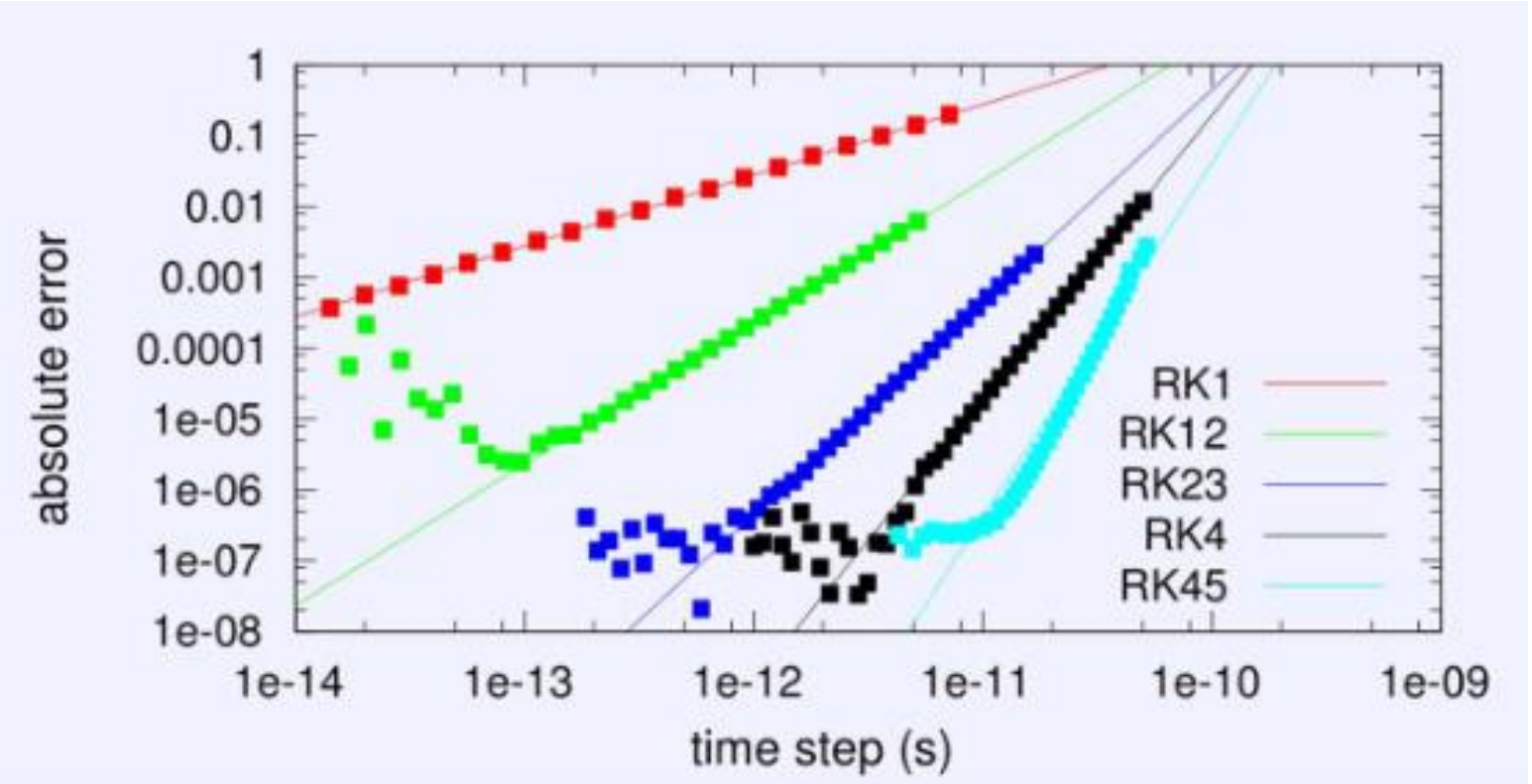
$$\frac{dy}{dt} = f(t, y)$$

$$y_{n+1} = y_n + hf(t, y_n)$$

Runge-Kutta (RK4) method



Convergence of different methods



Higher order methods require multiple calculations of derivatives but converge faster.

By default, mumax3 uses the adaptive Dormand-Prince method (RK45) → Best for most cases.
The relax() function uses Bogacki-Shampine (3rd order) method (besides it excludes dynamic term).

Exchange length and cell size/timestep size

Neighboring cells are held together by exchange interaction.

→cellsize below the exchange length is usually safe

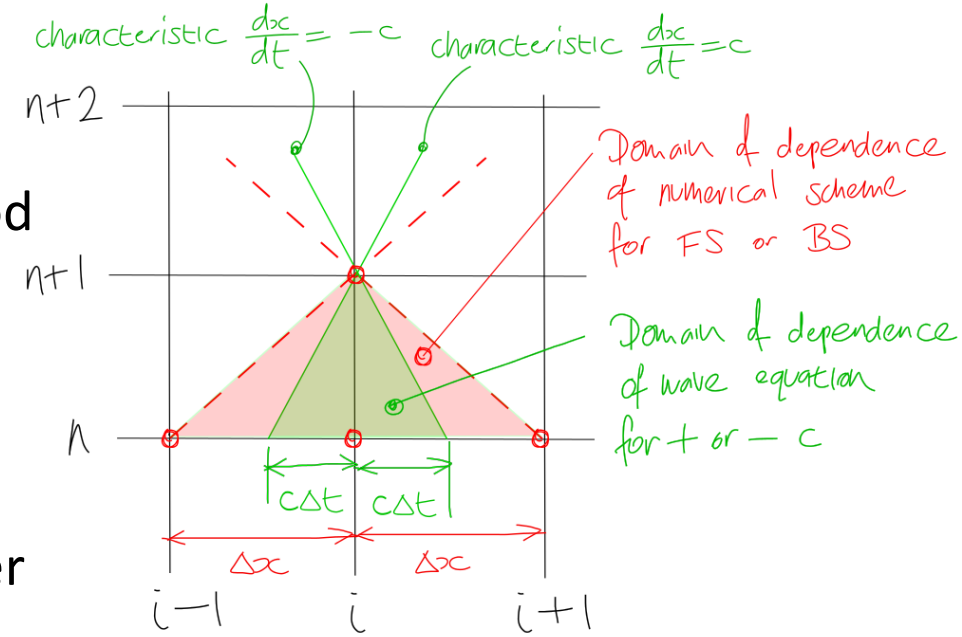
$$l_{ex} = \sqrt{\frac{2A_{ex}}{\mu_0 M_s^2}} \ (\approx 17 \text{ nm for YIG})$$

It is usually recommended to use at least 10 timesteps/period (for the highest frequency).

CFL condition: $$v_g < \frac{\Delta x}{\Delta t}$$

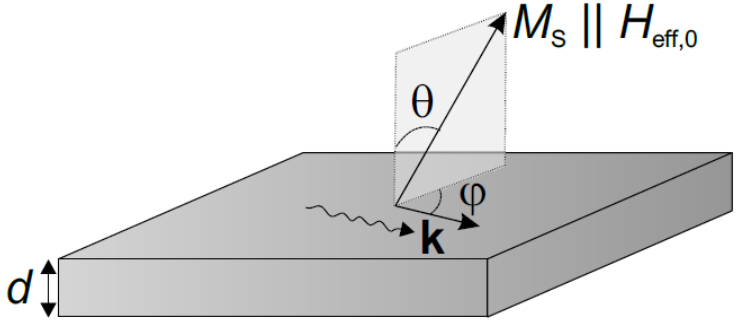
For spin waves, v_g is in the order of 100 nm/ns, much smaller than speed of light.

Adaptive methods use an error limit to set dt automatically.



https://www.thevisualroom.com/_images/domain_of_dependence.png

Dispersion curves



The dispersion curves for an infinitely large film:

$$\omega^2 = \omega_H(\omega_H + \omega_M F(\varphi, \theta))$$

$$\omega_H = \gamma\mu_0\left(H_{eff,0} + \frac{2A}{\mu_0 M_S}k^2\right)$$

$$\omega_M = \gamma\mu_0 M_S$$

$$F(\varphi, \theta) = P + \sin^2\theta\left[1 - P(1 + \cos^2\varphi) + \omega_M\frac{P(1-P)\sin^2\varphi}{\omega_H}\right]$$

$$P = 1 - \frac{1 - e^{-kd}}{kd}$$

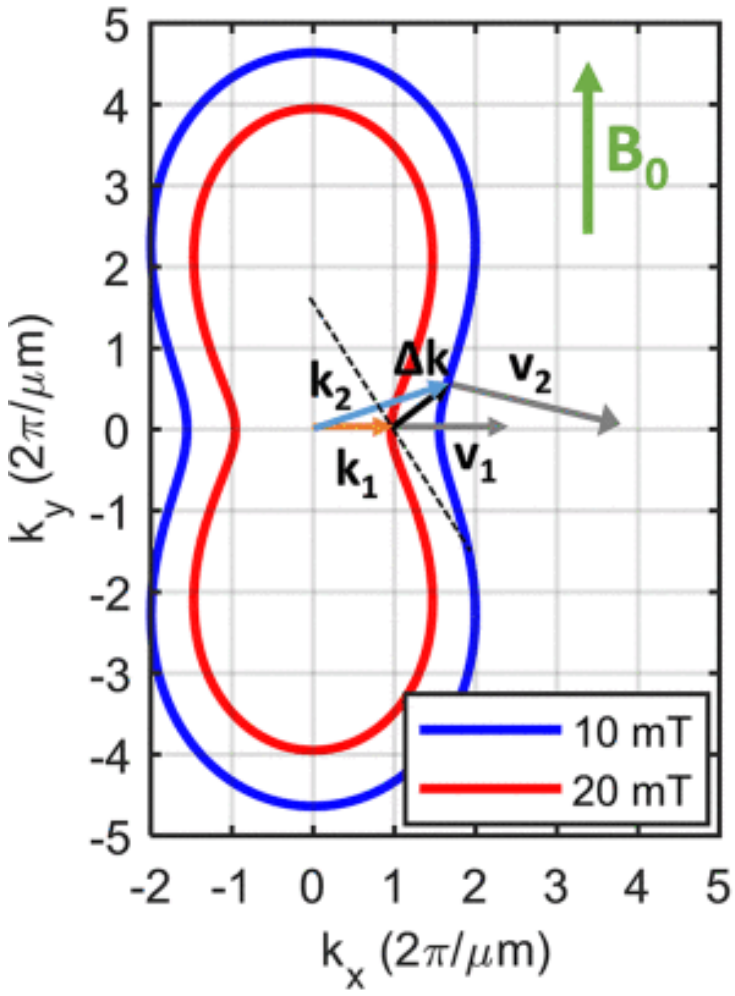
	θ	φ	$H_{eff,0}$
FV	0°	-	$H_0 - M_s$
DE	90°	90°	H_0
BV	90°	0°	H_0

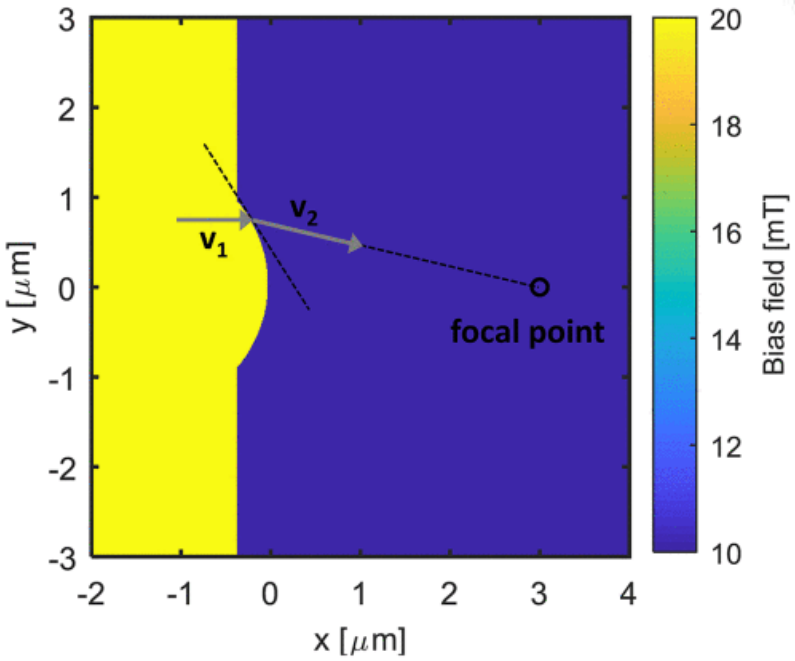
Check: if k=0 we get back the Kittel formula for FMR.

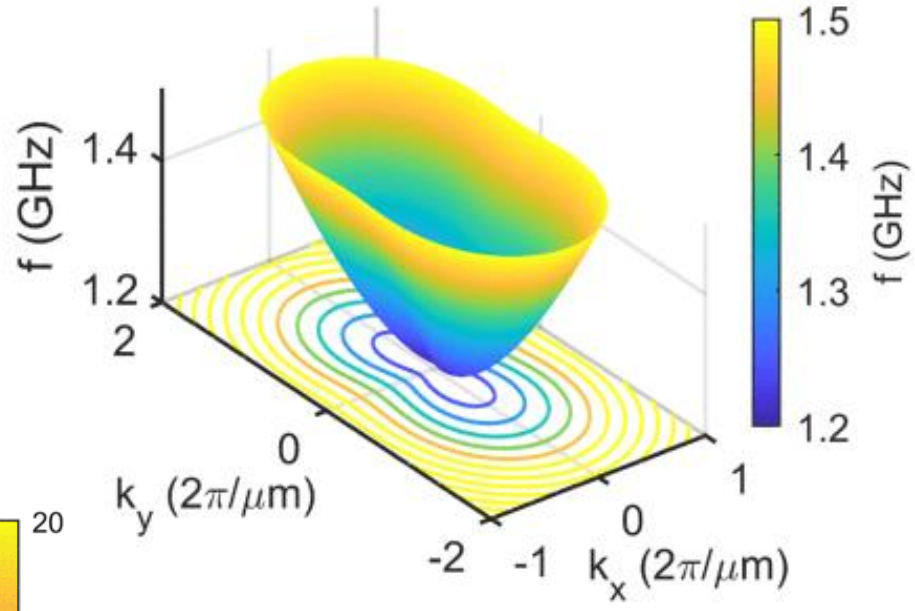
Magnonic Crystals: From Simple Models toward Applications
Jaroslaw W. Klos and Maciej Krawczyk (pg. 288)

Code: dispersion_code.m

Isofrequency curves





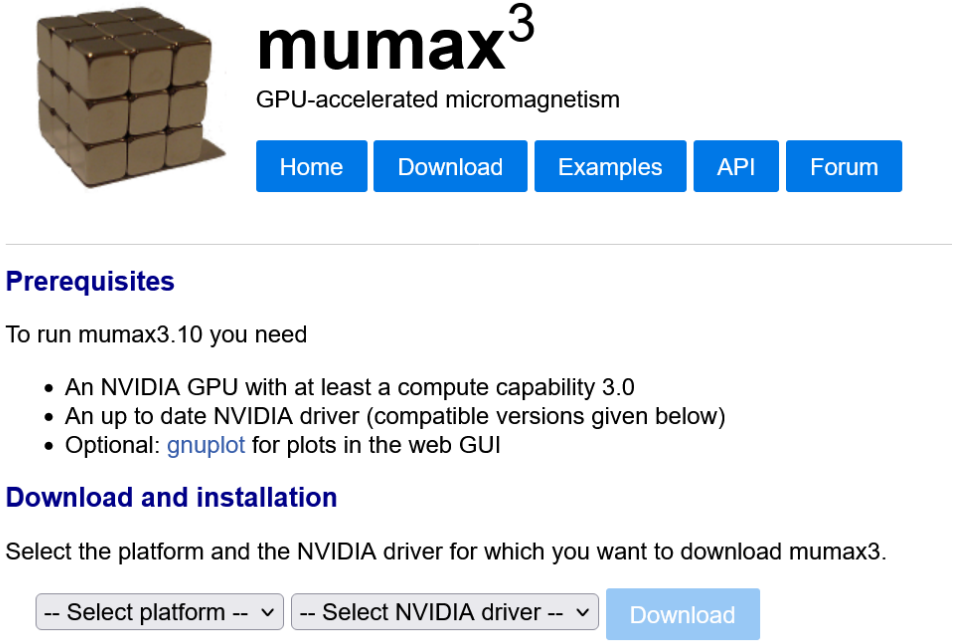


Code: dispersion_code_2D.m

Á. Papp and G. Csaba, "Lens Design for Computing With Anisotropic Spin Waves," in *IEEE Magnetics Letters*, vol. 9, pp. 1-5, 2018, Art no. 3706405, doi: 10.1109/LMAG.2018.2872127.

mumax^3 installation and usage

Download binary from http://mumax.github.io/download.html



Very good tutorial lectures and examples from the authors: https://mumax.ugent.be/mumax3-workshop/

Google Colab

If you don't have a GPU, you can try running mumax using Google Colab (for free – limited resources).

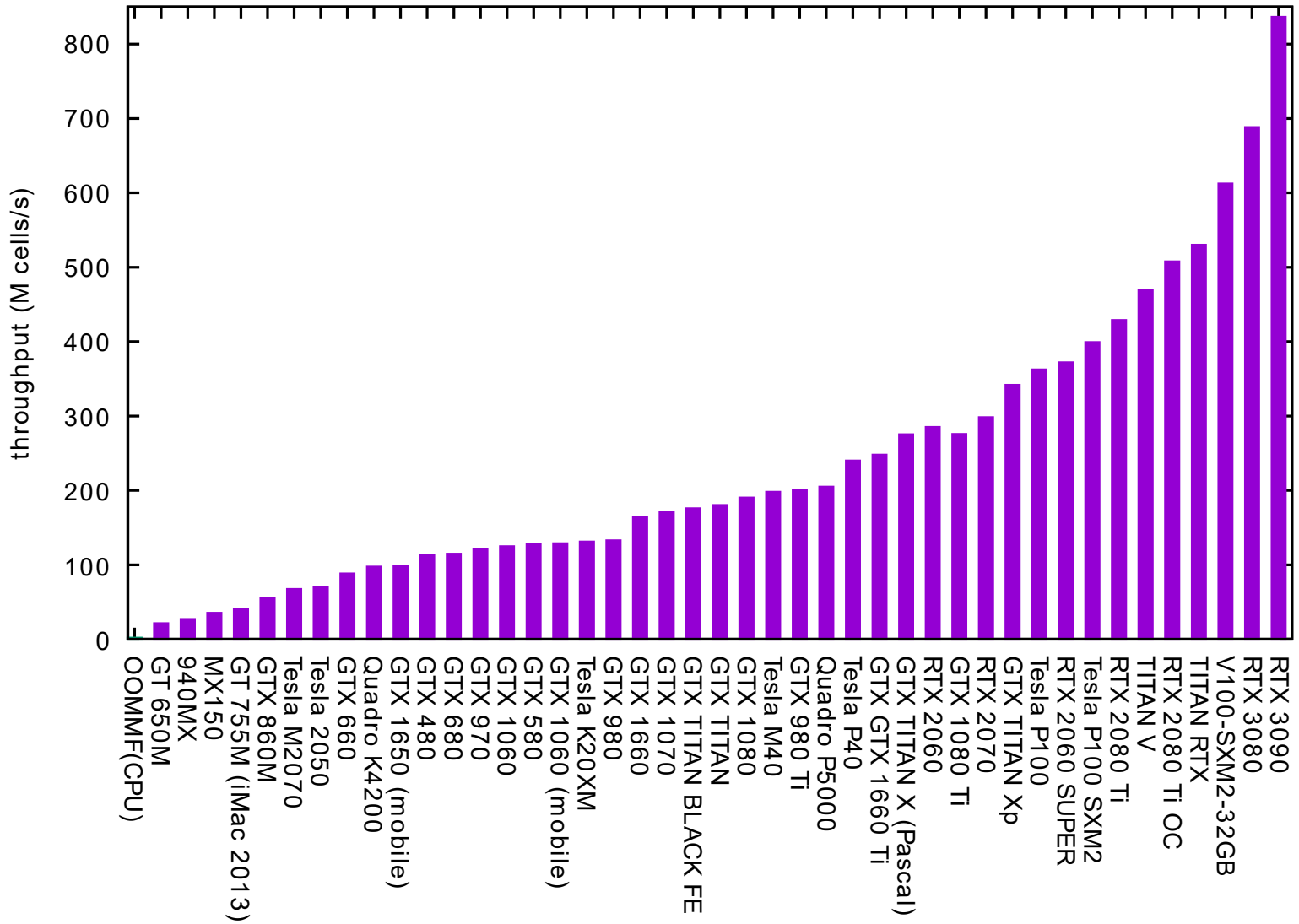
Download example from here:

mumax in Colab example

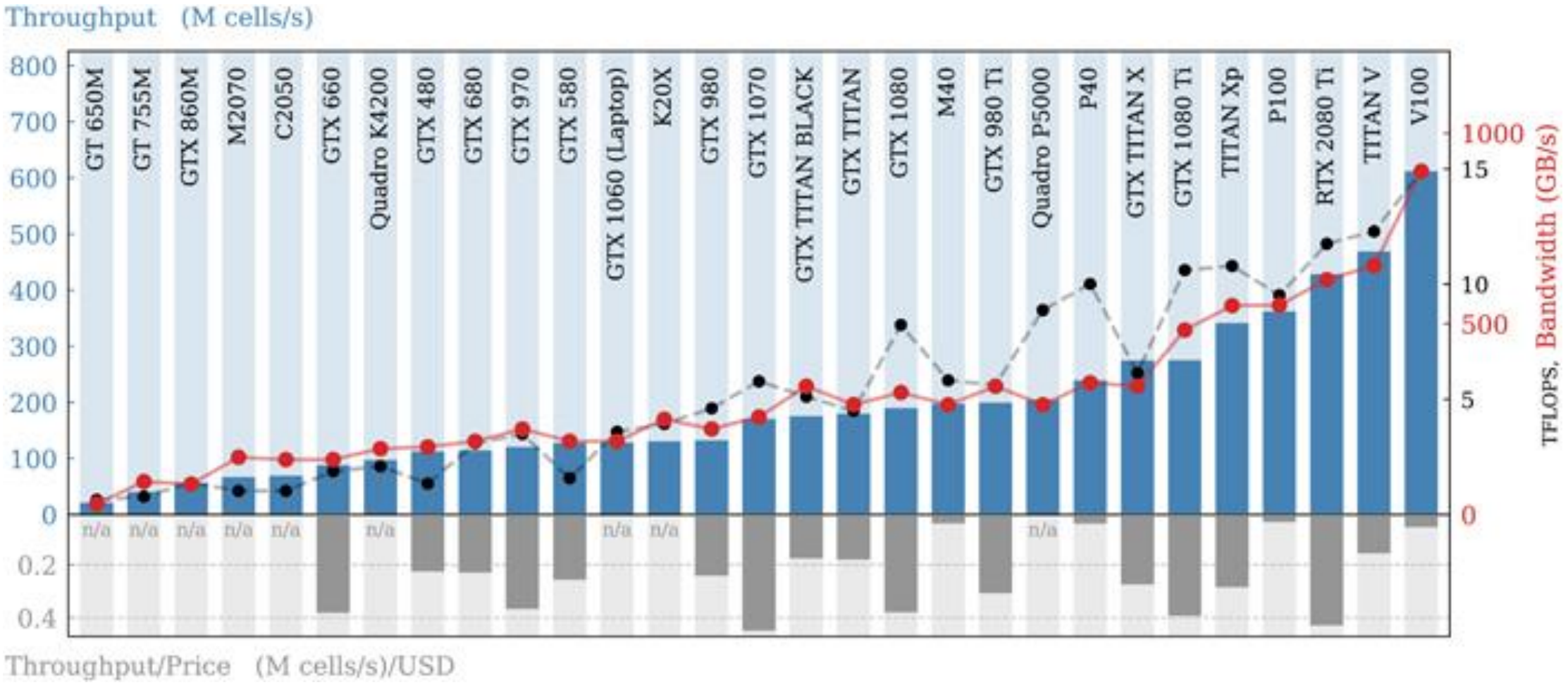
Also, postprocessing can be done inside the jupyter notebook:

Colab link for postprocessing

GPU requirements



How to choose a GPU – it's mostly about bandwidth



Professional-grade GPUs have double-precision cores, but this is not used by mumax3.

Journal of Applied Physics 125, 180901 (2019) https://doi.org/10.1063/1.5093730

Example #1: Spin-wave generation

- How to excite spin waves
- Add a lens
- Absorbing boundaries
- Periodic boundary condition (PBC)
- Nonlinear amplitude
- Finite temperature

Thermal noise is added by a random effective field:

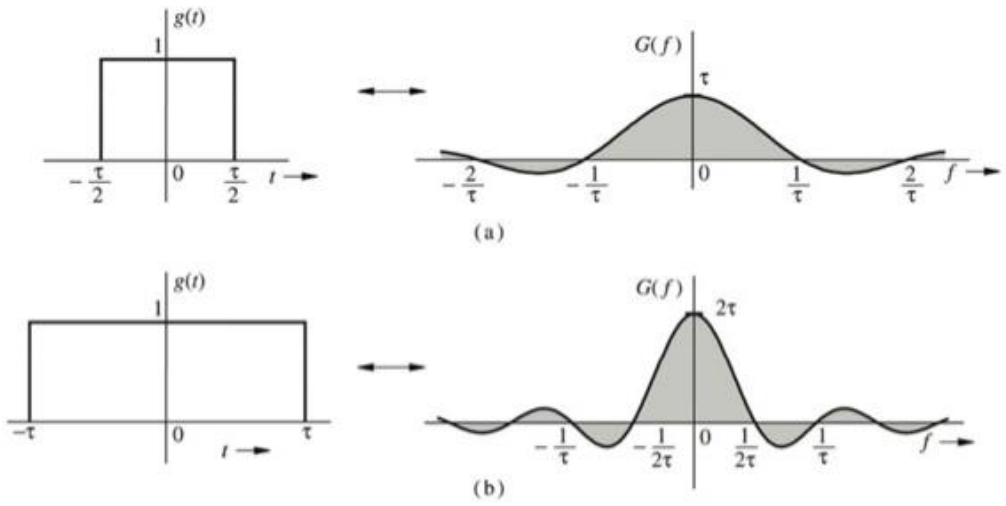
$$\vec{\mathbf{B}}_{\text{therm}} = \vec{\eta}\left(\text{step}\right)\sqrt{\frac{2\mu_0\alpha k_{\text{B}}T}{B_{\text{sat}}\gamma_{\text{LL}}\Delta V\Delta t}}$$

Code: spinwave.mx3

Vansteenkiste, Arne, et al. "The design and verification of MuMax3." *AIP advances* 4.10 (2014): 107133.

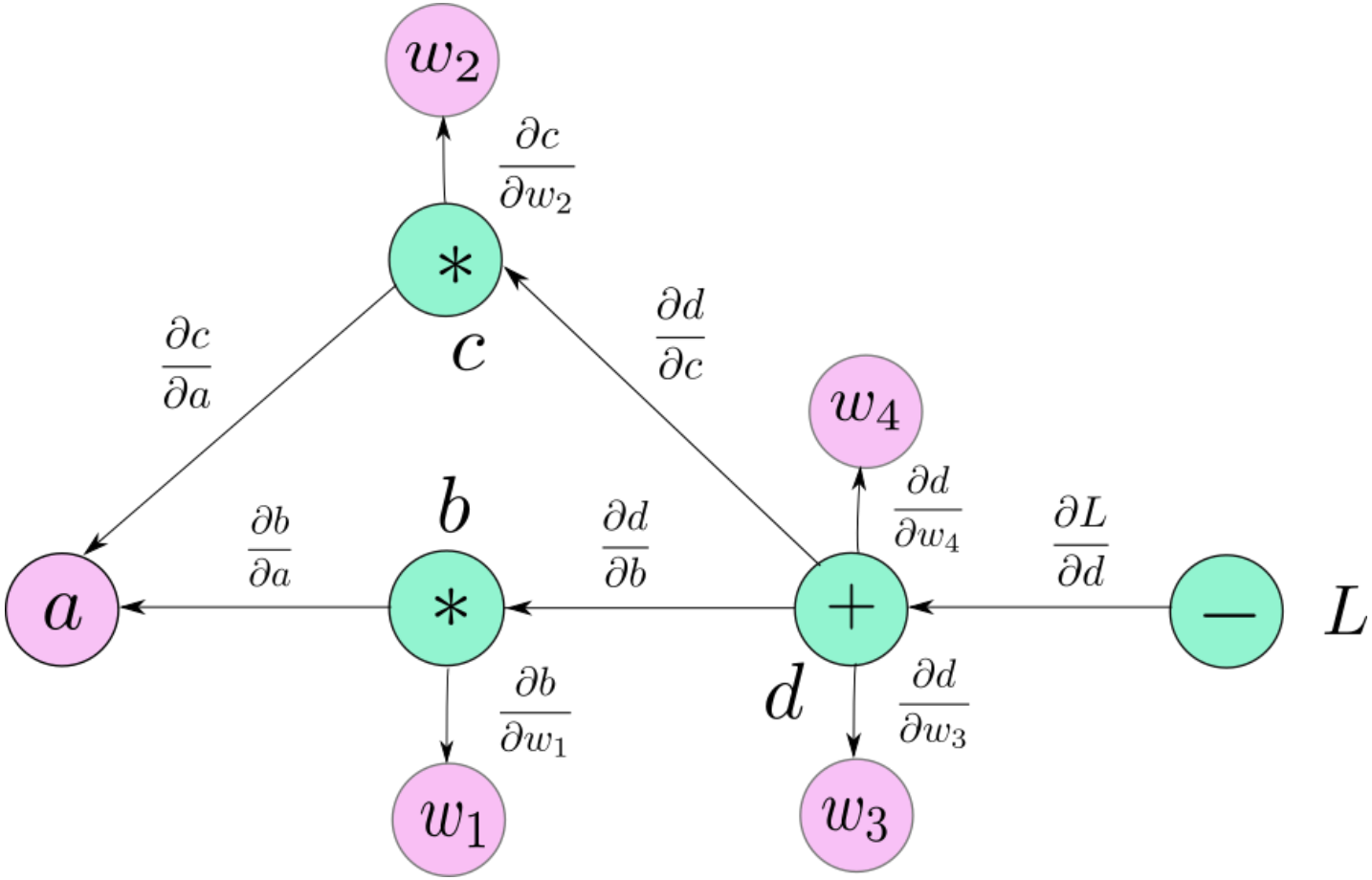
Example #2: Spin-wave dispersion

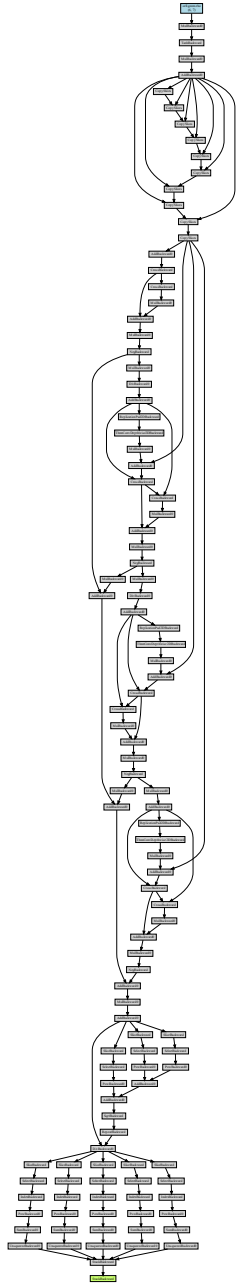
- Broadband excitation in space and time
  - Note: excitation efficiency of a transducer depends on the wavelength of the spin wave → can be calculated using the Fourier transform of the field profile.
  - We have to obey the Nyquist criterion to avoid aliasing.
- Calculate the dispersion curve by FFT



Code: spinwave_dispersion.mx3
loadmag_compress.m
dispersion_FFT.m

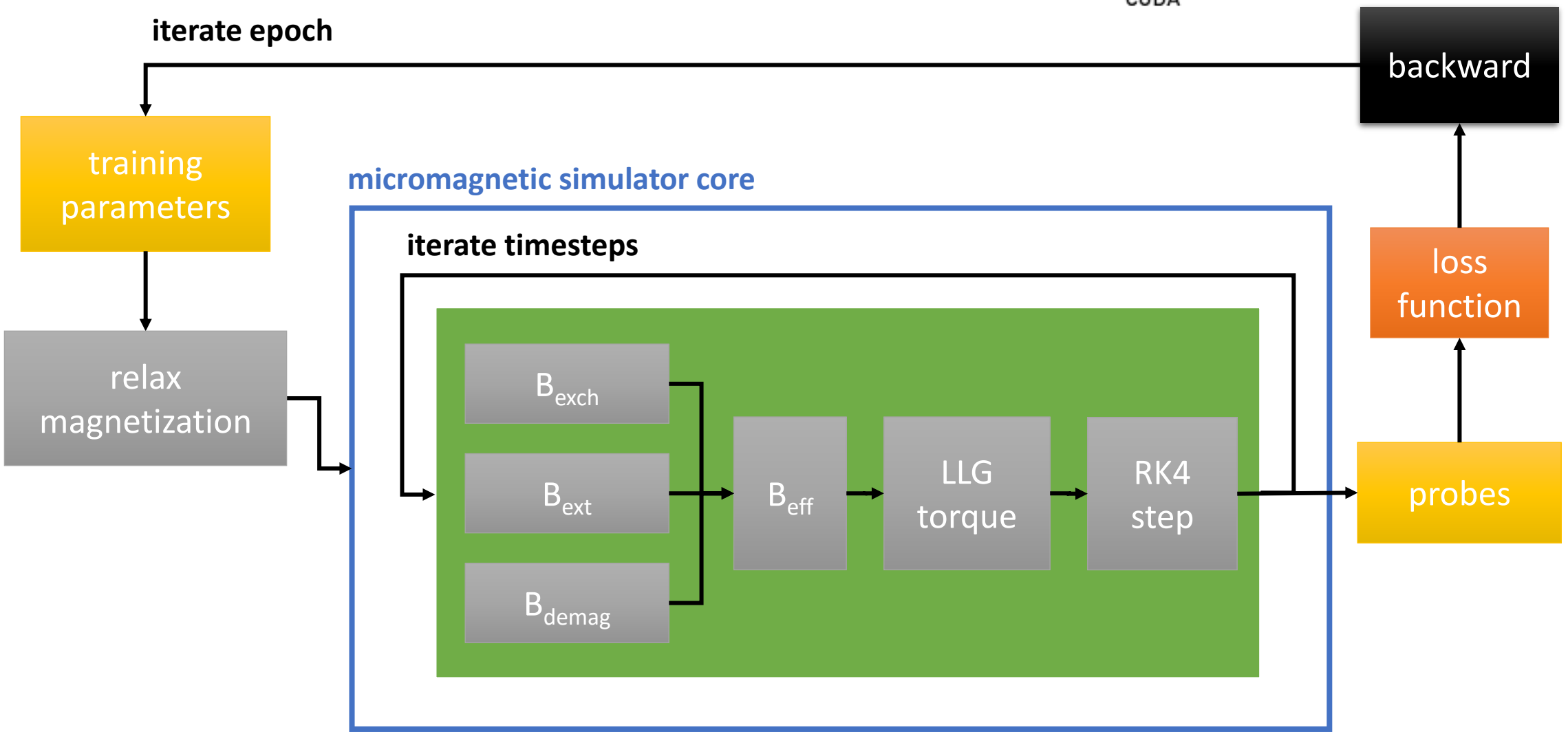
Inverse design with backpropagation





https://blog.paperspace.com/pytorch-101-understanding-graphs-and-automatic-differentiation/

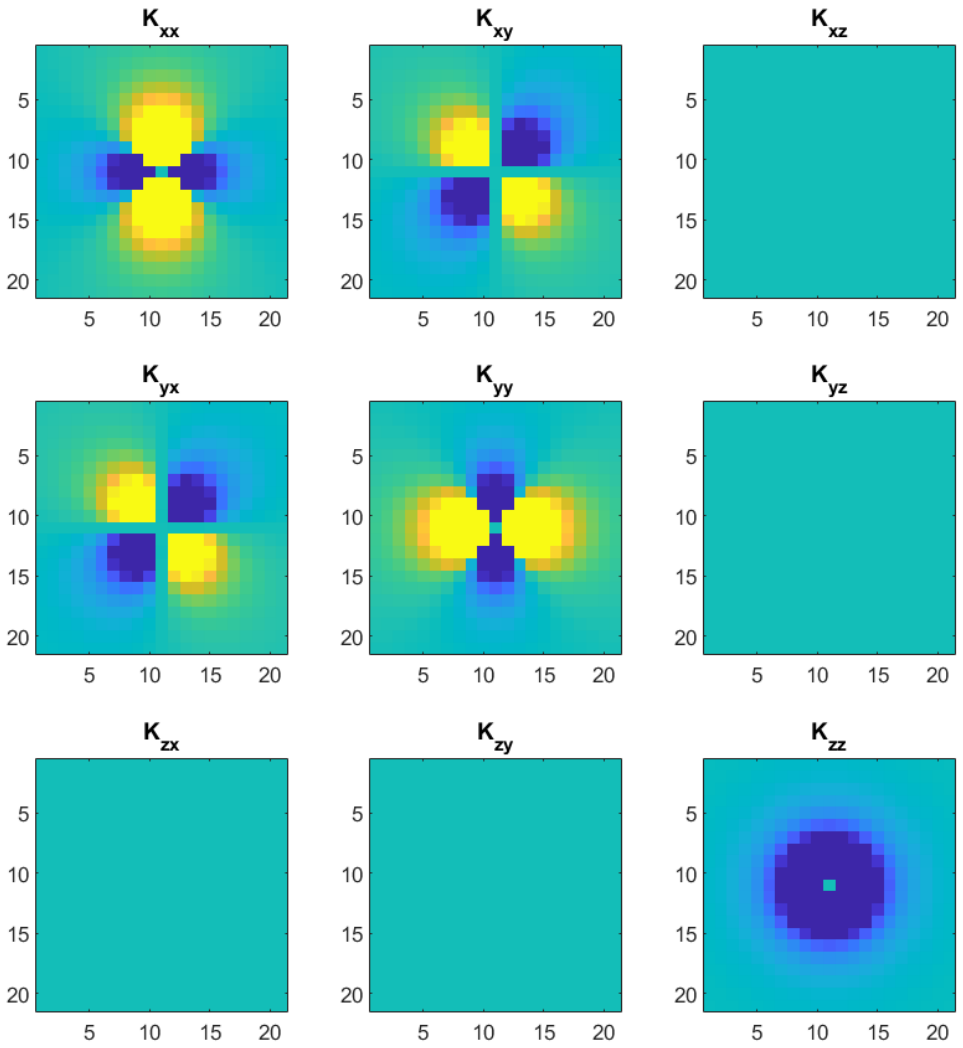
SpinTorch



Available online: https://github.com/a-papp/SpinTorch

Demag field calculation

Demag kernel (2D)

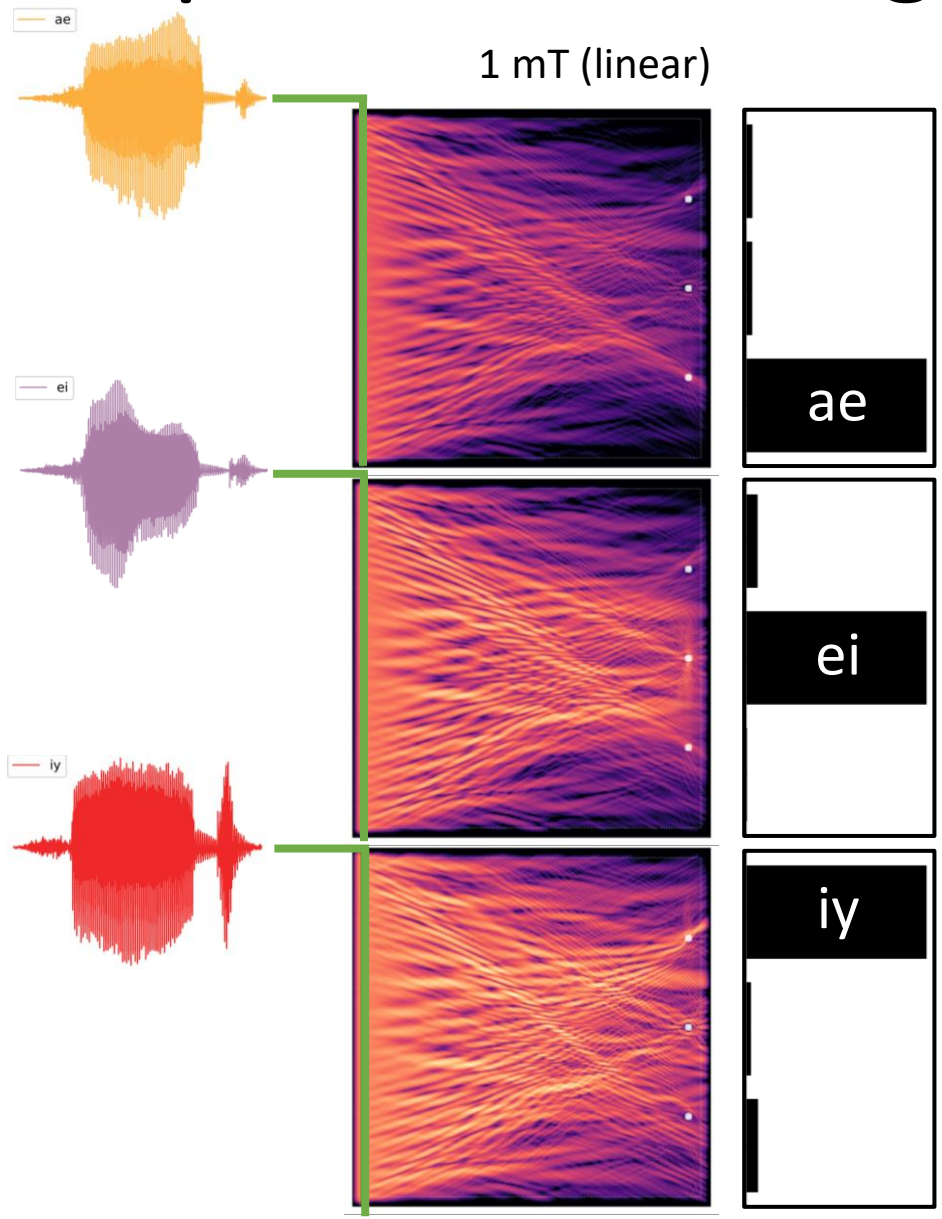


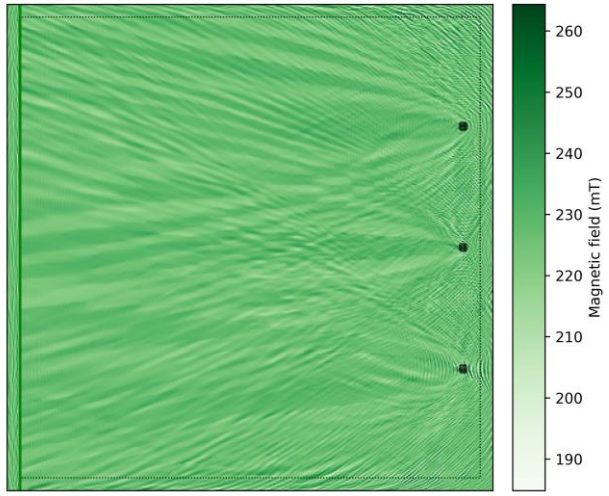
$$\vec{\mathbf{B}}_{\text{demag } i} = \hat{\mathbf{K}}_{ij} * \vec{\mathbf{M}}_j$$

→ **FFT accelerated convolution**

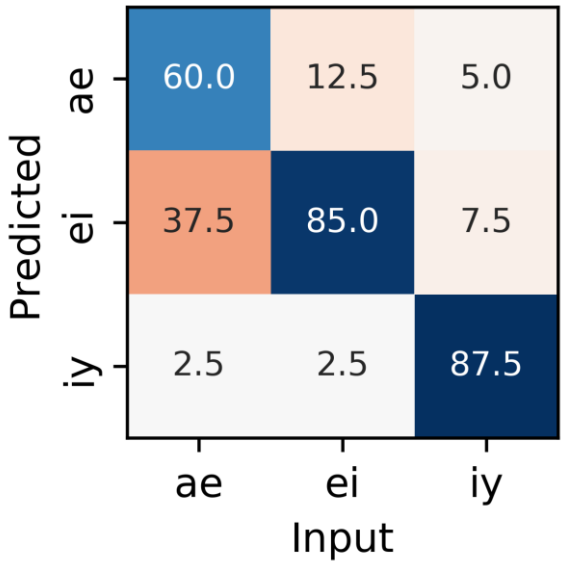
```python
@jit(nopython=True)
def demag_tensor_2D(nx, ny, dx):
    """
    Calculates the demagnetization tensor for 2D problems.
    Numba is used to accelerate the calculation.
    Inputs: nx, ny, nz number of cells in x/y/z, dx cellsize
    Outputs: demag tensor elements (numpy.array)
    """
    # Initialization of demagnetization tensor
    Kxx = zeros((nx*2, ny*2))
    Kyy = zeros((nx*2, ny*2))
    Kzz = zeros((nx*2, ny*2))
    Kxy = zeros((nx*2, ny*2))

    K = 0
    for J in range(-ny+1, ny):
        for I in range(-nx+1, nx):
            L, M = (I+nx-1), (J+ny-1)  # non-negative indices
            for i in (-0.5, 0.5):
                for j in (-0.5, 0.5):
                    for k in (-0.5, 0.5):
                        sgn = (-1)**(i+j+k+1.5)/(4*pi)
                        r = sqrt((I+i)**2 + (J+j)**2 + (K+k)**2)
                        Kxx[L, M] += sgn * arctan((K+k)*(J+j)/(I+i)/r)
                        Kyy[L, M] += sgn * arctan((I+i)*(K+k)/(J+j)/r)
                        Kzz[L, M] += sgn * arctan((J+j)*(I+i)/(K+k)/r)
                        Kxy[L, M] -= sgn * log((K+k+r))# * dx)

    return Kxx, Kyy, Kzz, Kxy
```

Example: Vowel recognition





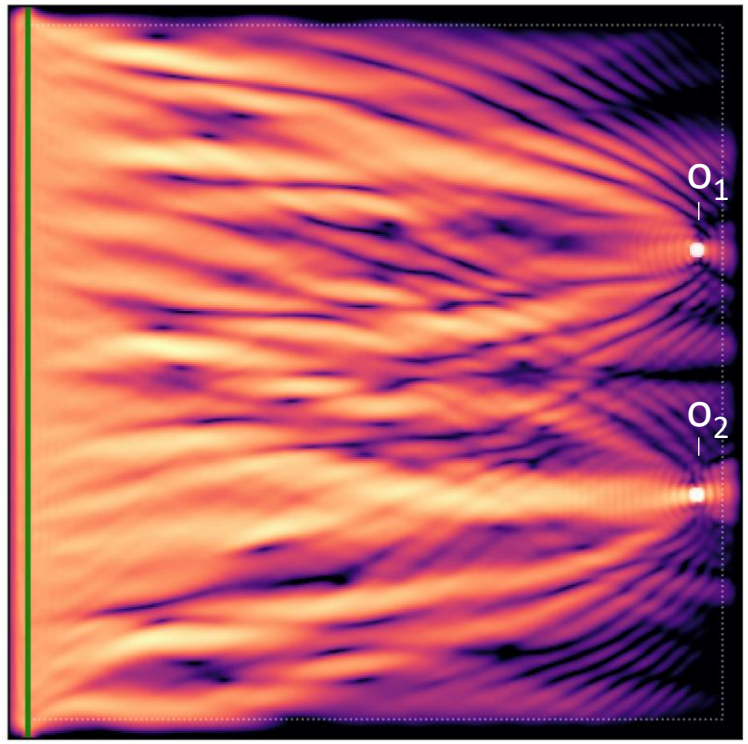
Confusion matrix:



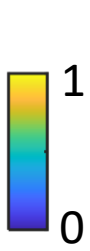
Train set: 12 samples
Test set: 123 samples

Nonlinear test function

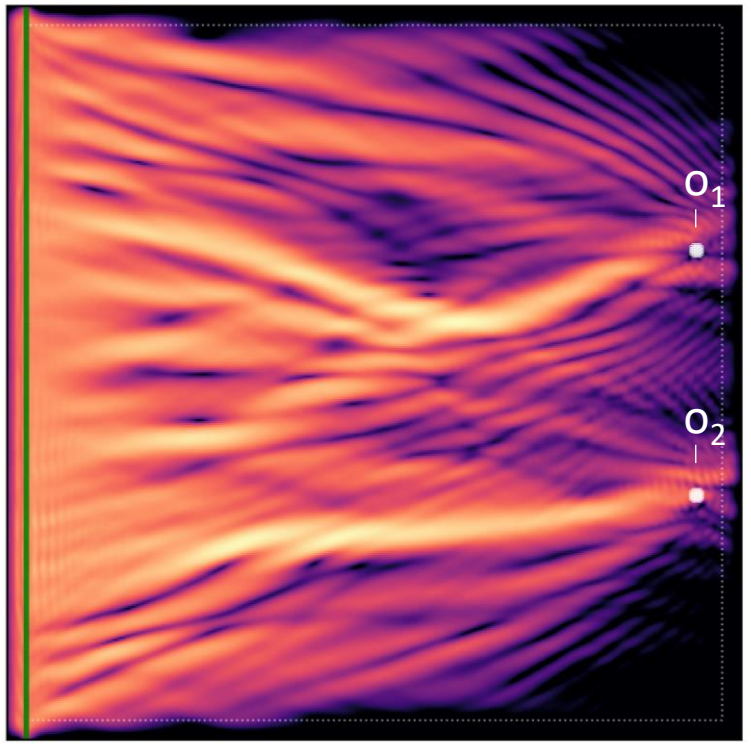
1 mT



	f_1	f_2	(f_1+f_2)
o_1	1	1	0
o_2	0	0	1

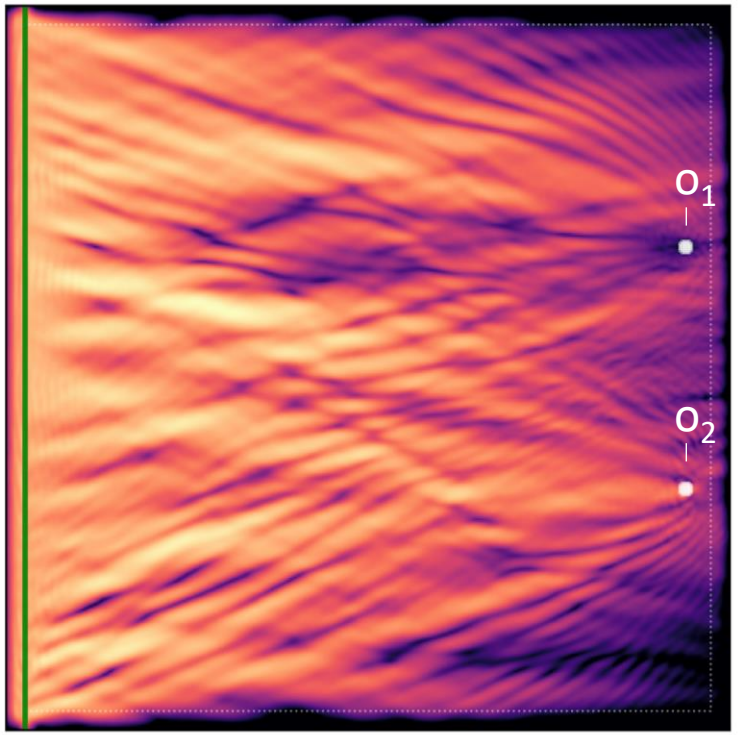


20 mT



	f_1	f_2	(f_1+f_2)
o_1	1	1	0
o_2	0	0	1

50 mT



	f_1	f_2	(f_1+f_2)
o_1	1	1	0
o_2	0	0	1

Papp, Á. et al., Nanoscale neural network using non-linear spin-wave interference. *Nat Commun* **12,** 6422 (2021).

SpinTorch demo

Download source code from: https://github.com/a-papp/SpinTorch