Simulation for 2024 paper using Julia

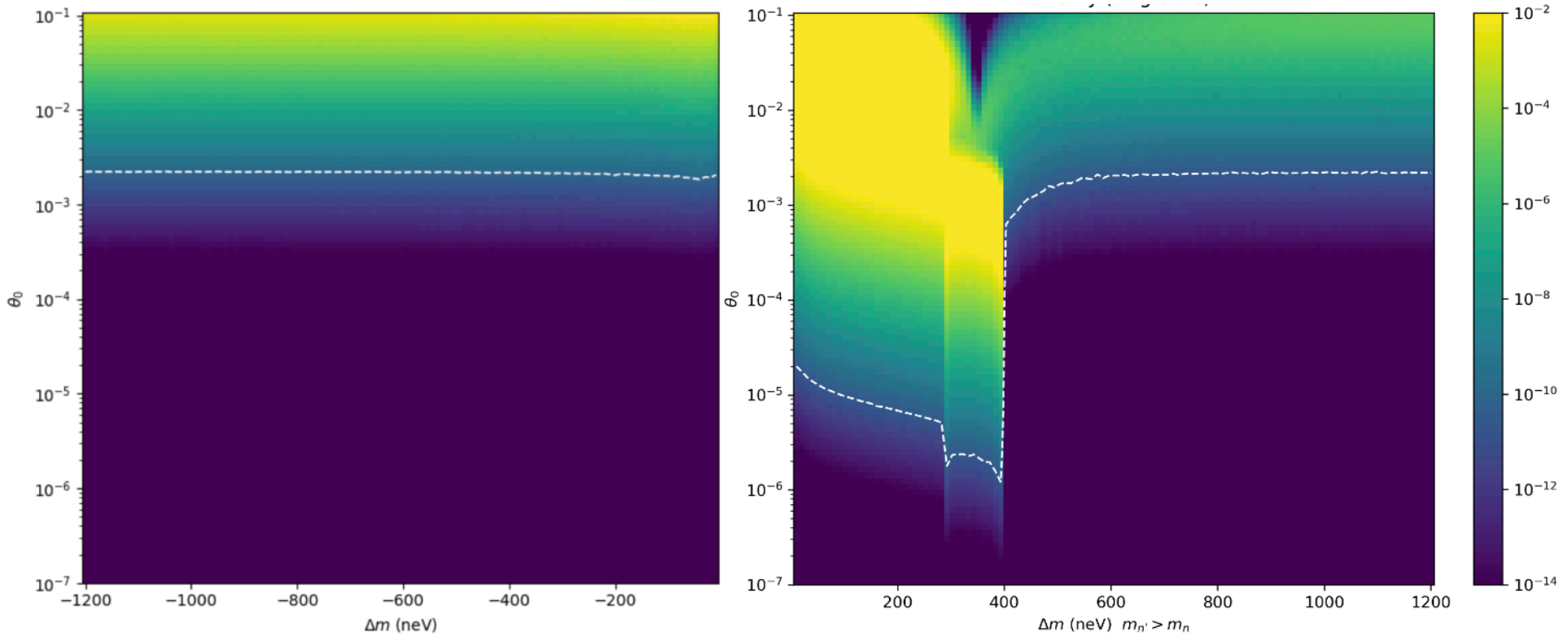
2026-02-17

linus.persson@fysik.lu.se

LUND UNIVERSITY

Changes to the code

- New velocity sampling implemented, new magnetic field reading, cubic Hermite field extrapolation.
- Corrected Cd thickness.



Remaining things to be clarified

- $U = V - \Delta m \pm \mu B$
- Should run with opposite polarization to see if distinctive features show up for negative Δm.
- Some differing conventions, see next slide. I think current one is incorrect, but need to investigate.

A strange comment?

```
function step_at_zStart!(neutrons, material, _dzs, _rands, pol::Float64)
    dzs = CUDA.Const(_dzs)
    rands = CUDA.Const(_rands)

    index = (blockIdx().x - 1) * blockDim().x + threadIdx().x
    stride = blockDim().x * gridDim().x
    for i = index:stride:length(neutrons)
        mat = material(neutrons[i].z)
        # Calculate parameters
        W::CuDF64   = CuDF64(mat.W0) + CuDF64(mat.Wv) * CuDF64(neutrons[i].vel)
        # Note: the neutrons[i].eps is the small Greek epsilon - the one in the
        # Material.jl file is the same.
        eps::CuDF64 = CuDF64(neutrons[i].eps)
        # Ditto, this u
        #U::CuDF64  = iszero(pol) ? CuDF64(mat.V) - CuDF64(neutrons[i].dm) :
        #    CuDF64(mat.V) + CuDF64(pol * kernel_uB(neutrons[i].z)) - CuDF64(neutrons[i].dm)
        U::CuDF64   = CuDF64(mat.V) + CuDF64(pol * kernel_uB(neutrons[i].z)) - CuDF64(0.5 * neutrons[i].dm)

        # This is to account for equation (19) in the SNS 2020 paper, the correct
        # formulation of the Hamiltonian, having an opposite sign of U than this code.
        # This is technically a bug and needs to be addressed (in the step(...) function),
        # but this gives the correct behavior and minimizes code change (in the actual
        # computing functions).
        U = -U

        @inbounds neutrons[i] = initialize_as_ktvb_22(neutrons[i], eps/hbar, U/hbar, W/hbar, dzs[i], rands[i])
        #@inbounds neutrons[i] = initialize_as_averaged(neutrons[i], eps/hbar, U/hbar, W/hbar, dzs[i])
    end
```

```
function step_array_with_prob!(neutrons, _dzs, _physArr, p11Vec, paVec, pbVec, p22Vec)::Nothing
  dzs = CUDA.Const(_dzs)
  pArr= CUDA.Const(_physArr)

  index = (blockIdx().x - 1) * blockDim().x + threadIdx().x
  stride = blockDim().x * gridDim().x
  for i = index:stride:length(neutrons)
    # Get the parameters
    j = neutrons[i].zIdx
    W::CuDF64 = CuDF64(pArr[j,4]) + CuDF64(pArr[j,5]) * CuDF64(neutrons[i].vel)
    eoh::CuDF64 = CuDF64(neutrons[i].eps / hbar)
    U::CuDF64 = CuDF64(pArr[j,3]) + CuDF64(pArr[j,2]) - CuDF64(neutrons[i].dm) # V + uB - dm

    # Move to the next step
    if iszero(U)
      if iszero(W)
        @inbounds neutrons[i] = stepUW0(neutrons[i], eoh, dzs[i])
      elseif isnan(W)
        @inbounds neutrons[i] = stepInf(neutrons[i], dzs[i])
      else
        @inbounds neutrons[i] = stepU0(neutrons[i], eoh, W/hbar, dzs[i])
      end
    elseif iszero(W)
      @inbounds neutrons[i] = stepW0(neutrons[i], eoh, U/hbar, dzs[i])
    else
      @inbounds neutrons[i] = step(neutrons[i], eoh, U/hbar, W/hbar, dzs[i])
    end
    @inbounds p11Vec[i] = HI(neutrons[i].p11)
    @inbounds paVec[i]  = 0.0 #HI(neutrons[i].pa)
    @inbounds pbVec[i]  = 0.0 #HI(neutrons[i].pb)
    @inbounds p22Vec[i] = 0.0 #HI(neutrons[i].p22)
  end

  return nothing
end
```

LUND UNIVERSITY

Current plan

- Have received access to local LU server with basically no job queue. Have not set up the code to run there yet but will hopefully soon.
- Today started a job with negative Δm and negative polarization.
- Nathan has some spikes not seen in Julia for the negative masses, but overall agreement is better than before.
- Next weeks are busy, but I will try to start runs occasionally.

nTMM experiment

- Have given my nTMM COMSOL model to German, which he will use to run simulation with his fitted magnetic scalar potential.
- Investigating the use of different optimization routines.
- He is looking into impact of removing or shifting the two strange "rows" of vectors.