simple_mesh_simulation

March 7, 2021

Working copy of SIGMA Detector simulation

[1]:
```
println("Number of threads: $(Base.Threads.nthreads())")
```

```
Number of threads: 14
```

[2]:
```
using Plots; pyplot();
```

[3]:
```
# Load required Julia packages:

using ArraysOfArrays, StaticArrays, Tables, TypedTables
using Statistics, Random, Distributions, StatsBase
using Unitful
import HDF5
# import Pkg; Pkg.add("SolidStateDetectors")
# push!(LOAD_PATH, "/home/fraser/Videos")
# import Pkg; Pkg.add("PackageSpec")
# import PackageSpec
# import Pkg; Pkg.add("Formatting")
# import Pkg; Pkg.add("Interpolations")
# import Pkg; Pkg.add("JSON")
# import Pkg; Pkg.add("IntervalSets")
# import Pkg; Pkg.add("Requires")
# import Pkg; Pkg.add("Rotations")
# import Pkg; Pkg.add("YAML")
# import Pkg; Pkg.add("Clustering")
# import Pkg; Pkg.add("ProgressMeter")
# include("/home/fraser/Videos/SolidStateDetectors.jl/src/SolidStateDetectors.
 ↪jl")

using MeshIO
using GeometryTypes
using FileIO
using NPZ
using LinearAlgebra
using Distances
using SolidStateDetectors
```

```
using SolidStateDetectors: CylindricalGrid, PointTypes
using RadiationSpectra
using RadiationDetectorSignals
using RadiationDetectorSignals: group_by_evtno, ungroup_by_evtno,␣
 ↪group_by_evtno_and_detno
using LegendDataTypes
using LegendHDF5IO: readdata, writedata
using LegendTextIO
# import Pkg; Pkg.add("NPZ")
# import Pkg
# Pkg.add("FileIO")
# Pkg.add("MeshIO")
# using FileIO
# using MeshIO
# using NPZ
```

```
WARNING: could not import HDF5.HDF5Group into _hdf5_implementation
WARNING: could not import HDF5.HDF5Dataset into _hdf5_implementation
  Warning: Error requiring `HDF5` from `Plots`
    exception =
(LoadError("/home/fraser/.julia/packages/Plots/uCh2y/src/backends/hdf5.jl", 162,
UndefVarError(:HDF5Group)), Union{Ptr{Nothing}, Base.InterpreterIP}[Ptr{Nothing}
@0x00007f651bee273f, Ptr{Nothing} @0x00007f651bf7308c, Ptr{Nothing}
@0x00007f651bf735d5, Ptr{Nothing} @0x00007f651bf7323f, Ptr{Nothing}
@0x00007f651bf73c43, Ptr{Nothing} @0x00007f651bf74ea7, Base.InterpreterIP in
top-level CodeInfo for Plots._hdf5_implementation at statement 4, Ptr{Nothing}
@0x00007f651bf92129, Ptr{Nothing} @0x00007f651bf928a2, Ptr{Nothing}
@0x00007f651bf91c8a, Ptr{Nothing} @0x00007f651bf91c29, Ptr{Nothing}
@0x00007f651bf64905, Ptr{Nothing} @0x00007f651bf936d1, Ptr{Nothing}
@0x00007f650d9b24ed, Ptr{Nothing} @0x00007f64c01232ce, Ptr{Nothing}
@0x00007f651bf5a2f2, Ptr{Nothing} @0x00007f651bf735f5, Ptr{Nothing}
@0x00007f651bf7323f, Ptr{Nothing} @0x00007f651bf73c43, Ptr{Nothing}
@0x00007f651bf74ea7, Base.InterpreterIP in top-level CodeInfo for Plots at
statement 10, Ptr{Nothing} @0x00007f651bf92129, Ptr{Nothing}
@0x00007f651bf933a0, Ptr{Nothing} @0x00007f64c01bfbd9, Ptr{Nothing}
@0x00007f64c01bfc0d, Ptr{Nothing} @0x00007f651bf59e27, Ptr{Nothing}
@0x00007f64ca3dee24, Ptr{Nothing} @0x00007f64c01bfb1d, Ptr{Nothing}
@0x00007f64c01bfb4d, Ptr{Nothing} @0x00007f651bf59e27, Ptr{Nothing}
@0x00007f64ca3de6eb, Ptr{Nothing} @0x00007f64c01bfa84, Ptr{Nothing}
@0x00007f64c01bfabd, Ptr{Nothing} @0x00007f651bf59e27, Ptr{Nothing}
@0x00007f651bf6a0d3, Ptr{Nothing} @0x00007f651bf6a9b1, Ptr{Nothing}
@0x00007f64ca3a4e8f, Ptr{Nothing} @0x00007f64ca3a5011, Ptr{Nothing}
@0x00007f651bf5a2f2, Ptr{Nothing} @0x00007f651bf6a0d3, Ptr{Nothing}
@0x00007f651bf6a9b1, Ptr{Nothing} @0x00007f64c018d5da, Ptr{Nothing}
@0x00007f64c018f822, Ptr{Nothing} @0x00007f651bf5a2f2, Ptr{Nothing}
@0x00007f651bf90a8a, Ptr{Nothing} @0x00007f651bf91fcd, Ptr{Nothing}
```

```
@0x00007f651bf64905, Ptr{Nothing} @0x00007f650d1536f3, Ptr{Nothing}
@0x00007f64ca366399, Ptr{Nothing} @0x00007f651bf5a2f2, Ptr{Nothing}
@0x00007f64c013eec3, Ptr{Nothing} @0x00007f651bf59e27, Ptr{Nothing}
@0x00007f651bf6a0d3, Ptr{Nothing} @0x00007f651bf6a9b1, Ptr{Nothing}
@0x00007f64ca3532df, Ptr{Nothing} @0x00007f64ca35351e, Ptr{Nothing}
@0x00007f64ca35353c, Ptr{Nothing} @0x00007f651bf59e27, Ptr{Nothing}
@0x00007f651bf7896e, Ptr{Nothing} @0x0000000000000000])
 @ Requires /home/fraser/.julia/packages/Requires/035xH/src/require.jl:44
 Info: Precompiling SolidStateDetectors [71e43887-2bd9-5f77-aebd-47f656f0a3f0]
 @ Base loading.jl:1278
 Warning: Package SolidStateDetectors does not have MeshIO in its dependencies:
 - If you have SolidStateDetectors checked out for development and have
   added MeshIO as a dependency but haven't updated your primary
   environment's manifest file, try `Pkg.resolve()`.
 - Otherwise you may need to report an issue with SolidStateDetectors
 Loading MeshIO into SolidStateDetectors from project dependency, future
warnings for SolidStateDetectors are suppressed.
WARNING: using GeometryBasics.Sphere in module SolidStateDetectors conflicts
with an existing identifier.
WARNING: using GeometryBasics.AbstractGeometry in module SolidStateDetectors
conflicts with an existing identifier.
WARNING: using Meshes.sample in module SolidStateDetectors conflicts with an
existing identifier.
WARNING: using Meshes.Geometry in module SolidStateDetectors conflicts with an
existing identifier.
WARNING: using Meshes.Sphere in module SolidStateDetectors conflicts with an
existing identifier.
WARNING: using Meshes.Box in module SolidStateDetectors conflicts with an
existing identifier.
WARNING: using Meshes.vertices in module SolidStateDetectors conflicts with an
existing identifier.
```

0.1 Calculation of detector potentials and fields

0.1.1 Detector definition

First, load a detector definition - here, a simple mesh:

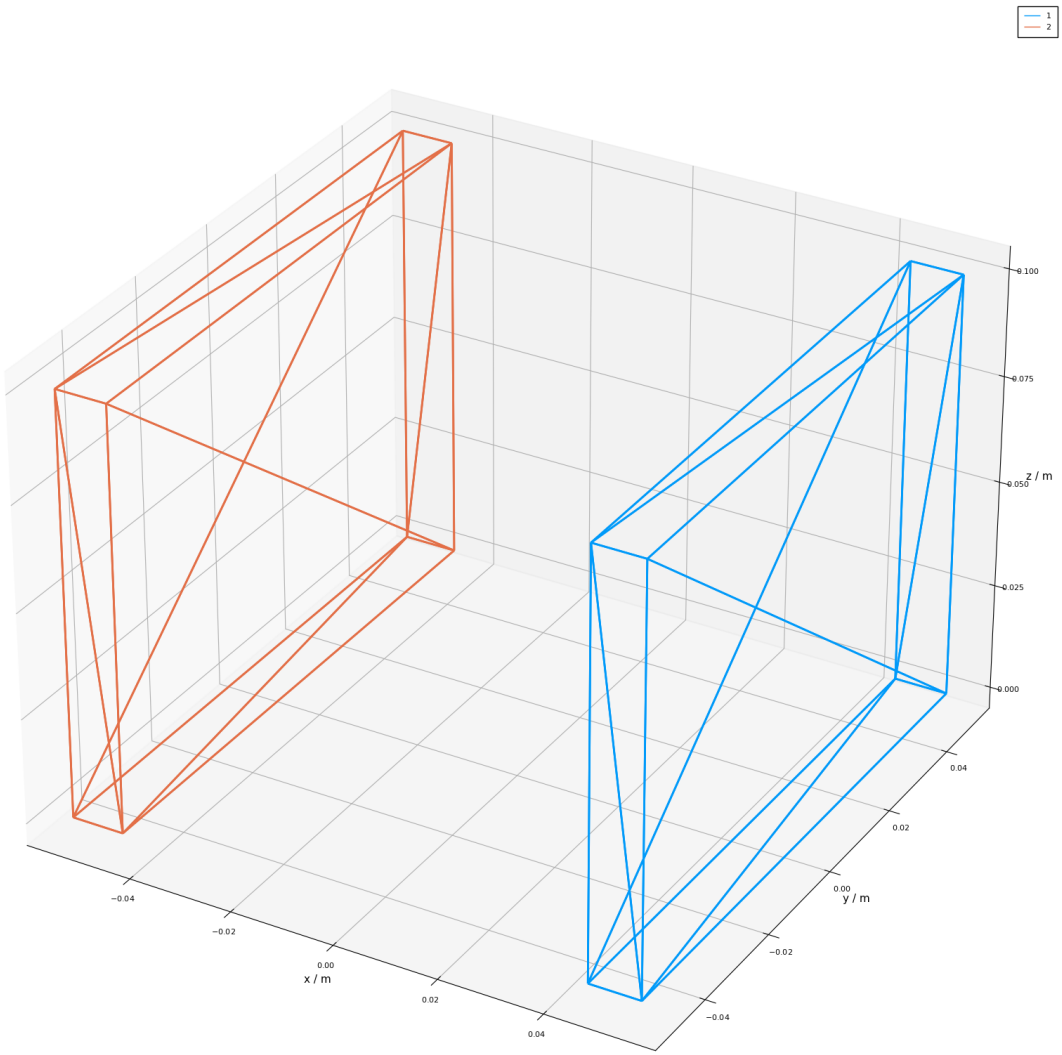
[4]:
```
#detector_config_filename = SSD_examples[:Coax]
T = Float32 # Optional; Default is Float32, but works with Float64 as well
#detector = SolidStateDetector{T}(detector_config_filename)
#simulation = Simulation{T}(SSD_examples[:InvertedCoax])
#simulation = Simulation{T}("/home/fraser/Julia/SolidStateDetectors.jl/examples/
 ↪example_detector_config_files/AGATA_crystal_cart_config.json")
simulation = Simulation{T}("mesh_cube.json")
# simulation = Simulation{T}("AGATA_test.json")
# simulation = Simulation{T}("/home/fraser/Julia/legend-julia-tutorial/
 ↪AGATA-crystal-simplified.json")
```

```
#simulation = Simulation{T}("SIGMA_config.json")
#simulation = Simulation{T}("public_Coax_config.json")
plot(simulation.detector, size= (1600, 1600))
```

```
Reading mesh: /home/fraser/Documents/solidstatedetectors/legend-julia-
tutorial/MESH/cube.stl
STL mesh read
Reading mesh: /home/fraser/Documents/solidstatedetectors/legend-julia-
tutorial/MESH/contact_1.stl
STL mesh read
Reading mesh: /home/fraser/Documents/solidstatedetectors/legend-julia-
tutorial/MESH/contact_2.stl
STL mesh read
```

```
WARNING: both GeometryBasics and IntervalSets export "width"; uses of it in
module SolidStateDetectors must be qualified
```

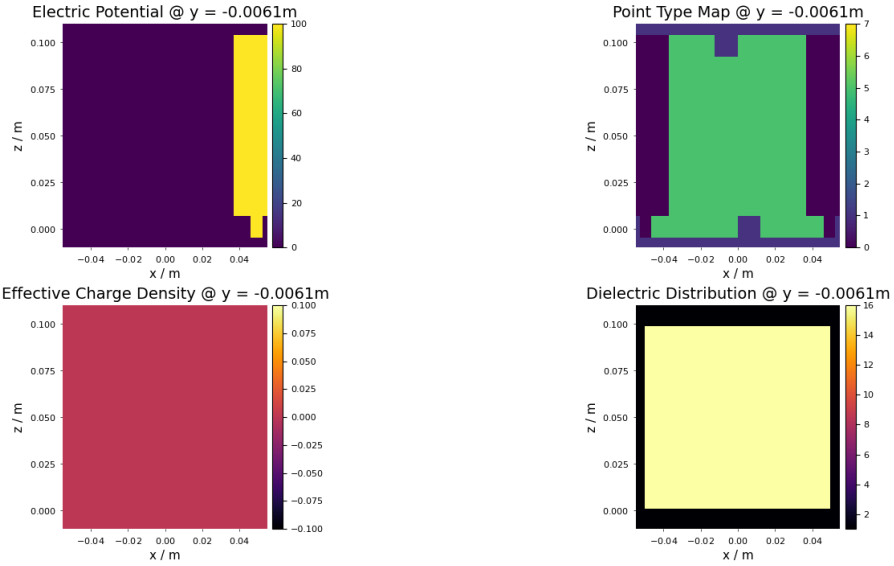
[4]:



[5]:
```
apply_initial_state!(simulation, ElectricPotential) # optional
```

[6]:
```
plot(
    plot(simulation.electric_potential, y = 0.00), # initial electric potential␣
↪(boundary conditions)
    plot(simulation.point_types, y=0), # map of different point types: fixed␣
↪point / inside or outside detector volume / depleted/undepleted
    plot(simulation. , y=0), # charge density distribution
    plot(simulation. , y=0), # dielectric distribution
    layout = (2, 2), size = (1400, 700)
)
```

[6]:



Next, calculate the electric potential:

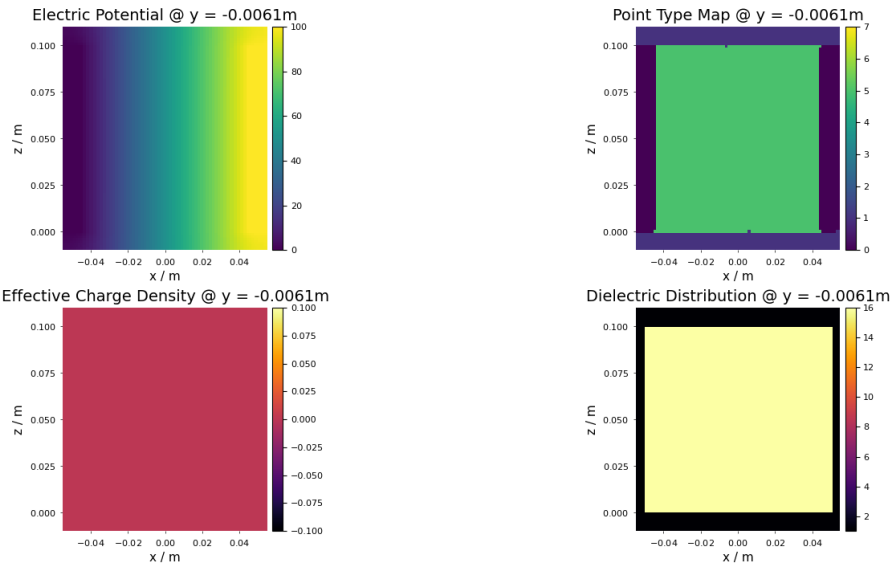
[7]:
```
calculate_electric_potential!(simulation, max_refinements=3)
```

```
Electric Potential Calculation
Bias voltage: 100.0 V
Precision: Float32
Convergence limit: 1.0e-7 => 1.0e-5 V
Threads: 14
Coordinate system: cartesian
Initial grid dimension: (12, 10, 11)
Refine? -> true
Refinement parameters:
        maximum number of refinements: 3
```

```
        minimum grid spacing:
                r: 1.0e-5 m
                 : 1.0e-5 rad
                z: 1.0e-5 m
        Refinement limits:
                r: 1.0e-5 -> 0.001 V
                 : 1.0e-5 -> 0.001 V
                z: 1.0e-5 -> 0.001 V


Convergence:  (thresh = 1e-05, value = 3.05176e-05)  Info: Maximum
number of iterations reached. (`n_iterations = 53025`)
  @ SolidStateDetectors /home/fraser/.julia/packages/SolidStateDetectors/48ipd/s
rc/PotentialSimulation/ConvergenceAndRefinement.jl:57
Convergence:  Time: 0:00:01 (8 iterations)
  Info: New Grid Size = (22, 18, 21)
  @ SolidStateDetectors /home/fraser/.julia/packages/SolidStateDetectors/48ipd/s
rc/Simulation/Simulation.jl:460
Convergence:  Time: 0:00:00 (11 iterations)5)m
  Info: New Grid Size = (42, 34, 41)
  @ SolidStateDetectors /home/fraser/.julia/packages/SolidStateDetectors/48ipd/s
rc/Simulation/Simulation.jl:460
Convergence:  Time: 0:00:00 (14 iterations)05)
  Info: New Grid Size = (82, 54, 78)
  @ SolidStateDetectors /home/fraser/.julia/packages/SolidStateDetectors/48ipd/s
rc/Simulation/Simulation.jl:460
Convergence:  Time: 0:00:00 (45 iterations)97)
```

[8]:
```
plot(
    plot(simulation.electric_potential, y = 0.00), # initial electric potential␣
↪(boundary conditions)
    plot(simulation.point_types, y=0), # map of different point types: fixed␣
↪point / inside or outside detector volume / depleted/undepleted
    plot(simulation. , y=0), # charge density distribution
    plot(simulation. , y=0), # dielectric distribution
    layout = (2, 2), size = (1400, 700)
)
```

[8]:



[9]:
```
npzwrite("cube/epot.npy", simulation.electric_potential)
npzwrite("cube/eps.npy", simulation.point_types)
npzwrite("cube/ephi.npy", simulation. )
npzwrite("cube/ee.npy", simulation. )
```

SolidStateDetectors.jl supports active (i.e. depleted) volume calculation:

[10]:
```
println(simulation.electric_potential.grid)

npzwrite("cube/grid_spacing_x.npy", simulation.electric_potential.grid[1])
npzwrite("cube/grid_spacing_y.npy", simulation.electric_potential.grid[2])
npzwrite("cube/grid_spacing_z.npy", simulation.electric_potential.grid[3])
```

```
Grid{Float32, 3, cartesian}
 Axis 1: -0.055..0.055 - length = 82
 Axis 2: -0.055..0.055 - length = 54
 Axis 3: -0.01..0.11 - length = 78
```

[ ]:
```
```

[11]:
```
for contact in simulation.detector.contacts
    calculate_weighting_potential!(simulation, contact.id, max_refinements = 2,␣
 ↪n_points_in_  = 2, verbose = true)
end
```

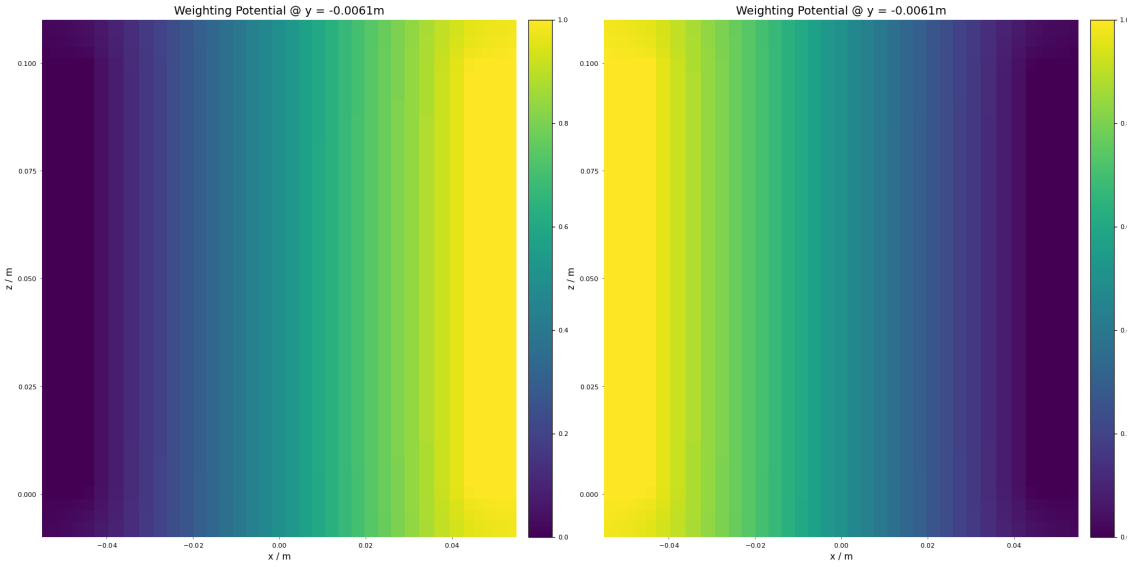
```
Weighting Potential Calculation
Precision: Float32
Convergence limit: 1.0e-7 => 1.0e-7 V
Threads: 14
```

```
Coordinate system: cartesian
Initial grid dimension: (12, 10, 11)
Refine? -> true
Refinement parameters:
        maximum number of refinements: 2
        minimum grid spacing:
                r: 1.0e-5 m
                 : 1.0e-5 rad
                z: 1.0e-5 m
        Refinement limits:
                r: 1.0e-5 -> 1.0e-5 V
                 : 1.0e-5 -> 1.0e-5 V
                z: 1.0e-5 -> 1.0e-5 V


Convergence:  (thresh = 1e-07, value = 2.98023e-07)  Info: Maximum
number of iterations reached. (`n_iterations = 53025`)
  @ SolidStateDetectors /home/fraser/.julia/packages/SolidStateDetectors/48ipd/s
rc/PotentialSimulation/ConvergenceAndRefinement.jl:57
Convergence:  Time: 0:00:01 (8 iterations)
  Info: New Grid Size = (22, 18, 21)
  @ SolidStateDetectors /home/fraser/.julia/packages/SolidStateDetectors/48ipd/s
rc/Simulation/Simulation.jl:464
Convergence:  Time: 0:00:00 (11 iterations)07)
  Info: New Grid Size = (42, 34, 41)
  @ SolidStateDetectors /home/fraser/.julia/packages/SolidStateDetectors/48ipd/s
rc/Simulation/Simulation.jl:464
Convergence:  Time: 0:00:00 (14 iterations)07)

Weighting Potential Calculation
Precision: Float32
Convergence limit: 1.0e-7 => 1.0e-7 V
Threads: 14
Coordinate system: cartesian
Initial grid dimension: (12, 10, 11)
Refine? -> true
Refinement parameters:
        maximum number of refinements: 2
        minimum grid spacing:
                r: 1.0e-5 m
                 : 1.0e-5 rad
                z: 1.0e-5 m
        Refinement limits:
                r: 1.0e-5 -> 1.0e-5 V
                 : 1.0e-5 -> 1.0e-5 V
                z: 1.0e-5 -> 1.0e-5 V


Convergence:  (thresh = 1e-07, value = 2.38419e-07)  Info: Maximum
```

```
number of iterations reached. (`n_iterations = 53025`)
  @ SolidStateDetectors /home/fraser/.julia/packages/SolidStateDetectors/48ipd/s
rc/PotentialSimulation/ConvergenceAndRefinement.jl:57
Convergence:  Time: 0:00:01 (8 iterations)
  Info: New Grid Size = (22, 18, 21)
  @ SolidStateDetectors /home/fraser/.julia/packages/SolidStateDetectors/48ipd/s
rc/Simulation/Simulation.jl:464
Convergence:  Time: 0:00:00 (11 iterations)07)
  Info: New Grid Size = (42, 34, 41)
  @ SolidStateDetectors /home/fraser/.julia/packages/SolidStateDetectors/48ipd/s
rc/Simulation/Simulation.jl:464
Convergence:  Time: 0:00:00 (14 iterations)07)
```

[12]:
```
plot(plot(simulation.weighting_potentials[1], y=0),
     plot(simulation.weighting_potentials[2], y=0),
     size = (1900, 1700)
)
```

[12]:



[13]:
```
npzwrite("cube/wpot-1.npy", simulation.weighting_potentials[1])
npzwrite("cube/wpot-2.npy", simulation.weighting_potentials[2])
npzwrite("cube/wpot-1x.npy", simulation.weighting_potentials[1].grid[1])
npzwrite("cube/wpot-2x.npy", simulation.weighting_potentials[2].grid[1])
npzwrite("cube/wpot-2y.npy", simulation.weighting_potentials[2].grid[2])
npzwrite("cube/wpot-1z.npy", simulation.weighting_potentials[1].grid[3])
npzwrite("cube/wpot-2z.npy", simulation.weighting_potentials[2].grid[3])
```

[ ]:
```

```