NewRelaxationTissueModel_Dummy

February 22, 2024

1 Dummy Code for New RelaxationTissueModel Class

Two cases of how it could be used: 1. Single composite for compartment model with unique relaxation values 2. Multiple composites that form a unique compartment model with their own unique relaxation values

1.1 1. Single Composite with Relaxation

1. Add a model from a tissue toolbox of choice (Dmipy as an example)
2. Translate into microtool usable tissue
3. Translate into RelaxationTissueModel: this can be either only T2 or T2+T1

RelaxationTissueModel is for composites, not compartments Thus, it allows to introduce different relaxation values for different models.

[ ]:
```
from microtool.dmipy import DmipyTissueModel
from microtool.tissue_model import RelaxationTissueModel
from dmipy.signal_models.gaussian_models import G1Ball

#Tissuemodel from dmipy
ball =  G1Ball(lambda_iso=1.7e-9)

#Translate into microtool
ball_microtool = DmipyTissueModel(ball)

#Translate into model that takes into account relaxations␣
 ↪(RelaxationTissueModel)
model_1 = RelaxationTissueModel(ball_microtool, T2 = 20)
```

1.2 2. Create a MultiTissueModel from several composites with relaxations

1. Add a new composite
2. Translate again into microtool usable tissue
3. Translate into RelaxationTissueModel
4. Add the two RelaxationTissueModels into a unique MultiTissueModel that serves as a wrapper of the RelaxationTissueModel

The final MultiTissueModel will contain within each composite the unique characteristics among which the specific relaxations are contained.

```
[ ]: from dmipy.signal_models import sphere_models

#Create second model from dmipy
sphere = sphere_models.S4SphereGaussianPhaseApproximation(diffusion_constant=2.
 ↪0e-9, diameter=10e-6)

#Translate into microtool
sphere_microtool = DmipyTissueModel(sphere)
sphere._dmipy_fix_parameters('S4SphereGaussianPhaseApproximation_1_diameter', 2.
 ↪0e-9)

#Translate into model that takes into account relaxations␣
 ↪(RelaxationTissueModel)
model_2 = RelaxationTissueModel(sphere_microtool, T2 = 10) #Allows for addition␣
 ↪of a different T2 value specific to the composite

from microtool.tissue_model import MultiTissueModel
#Combine models into MultiCompartmentModel
multitissuemodel_with_relaxations = MultiTissueModel(models = [model_1,␣
 ↪model_2], volume_fractions = [0.3, 0.7])
```

2 Proceed with acquisition protocol and optimization

Regardless of whether a single composite (unique RelaxationTissueModel) or several composites are considered (MultiTissueModel), the optimization procedure continues equally.

final_model will be either model_1 or multitissuemodel_with_relaxations

```
[ ]: #Initial scheme protocol
from microtool.acquisition_scheme import DiffusionAcquisitionScheme,␣
 ↪InversionRecoveryAcquisitionScheme
from microtool.gradient_sampling import sample_uniform_half_sphere

M = 10
N = 30

b_vals = np.concatenate([np.repeat(0, M), np.repeat(1000, M), np.repeat(3000,␣
 ↪M)])
pulse_widths = np.concatenate([np.repeat(0.019, M), np.repeat(0.016, M), np.
 ↪repeat(0.007, M)])
pulse_intervals = np.concatenate([np.repeat(0.030, M), np.repeat(0.027, M), np.
 ↪repeat(0.020, M)])

directions = sample_uniform_half_sphere(N)
initial_scheme = DiffusionAcquisitionScheme.from_bvals(b_values=b_vals,␣
 ↪b_vectors=directions, pulse_widths=pulse_widths,
```

```
                                                                    ␣
 ↪pulse_intervals=pulse_intervals)


#Optimization
from microtool.optimize import optimize_scheme
optimal_scheme, _ = optimize_scheme(initial_scheme, final_model,␣
 ↪noise_variance=0.02, method="trust-constr", solver_options={"verbose":2,␣
 ↪"maxiter": 10})

signal = final_model(optimal_scheme)

#Fitting
fitted_model = final_model.fit(optimal_scheme, signal,␣
 ↪use_parallel_processing=False)
fitted_model.fitted_parameters
```