

# YongSequenceTests

February 28, 2023

## 1 Working with Sequence Parameter Sets

### 1.0.1 Generating Sequence Parameter Sets

```
[4]: from mrftools import SequenceParameters, Perlin, Sequential,
      ↪ ScaledRectifiedSinusoid, types, InversionModule, FISPAcquisitionModule,
      ↪ SpoilerModule, DictionaryParameters, DeadtimeRecoveryModule,
      ↪ T2PreparationModule
      import numpy as np

      # Create a sequence definition programmatically
      numTimepoints=1776; numLobes=12; numSpirals=48
      TRs = np.ones(numTimepoints) * 0.0083
      TEs = np.ones(numTimepoints) * 0.0022

      flipangleLobeAmplitudeList = np.array(Perlin.Generate(numLobes, min=6, max=12,
      ↪ wavelength=1, firstValue=9, seed=5678))
      FAs = ScaledRectifiedSinusoid.Generate(numTimepoints,
      ↪ flipangleLobeAmplitudeList, minimum=4, patternInitialPhase=0)
      PHs = np.zeros(numTimepoints)
      IDs = Sequential.Generate(numTimepoints, numSpirals)

      numGroups = 12;
      timepointsPerModule = int(numTimepoints / numGroups)
      currentTimepoint = 0

      #https://www.sciencedirect.com/science/article/pii/S1053811918321190?via%3Dihub
      sequence = SequenceParameters("YongMyelinWaterFraction", [])

      # Group 1
      sequence.modules.append(InversionModule(totalDuration=0.020))
      acquisitionModule = FISPAcquisitionModule(dephasingRange=360)
      timepointRange = np.
      ↪ arange(currentTimepoint, currentTimepoint+timepointsPerModule-1);
      ↪ currentTimepoint = currentTimepoint+timepointsPerModule
      acquisitionModule.Initialize(TRs[timepointRange], TEs[timepointRange],
      ↪ FAs[timepointRange], PHs[timepointRange], IDs[timepointRange])
```

```

sequence.modules.append(acquisitionModule)

# Group 2
sequence.modules.append(DeadtimeRecoveryModule(0.300))
acquisitionModule = FISPAcquisitionModule(dephasingRange=360)
timepointRange = np.
    ↳arange(currentTimepoint,currentTimepoint+timepointsPerModule-1);↳
    ↳currentTimepoint = currentTimepoint+timepointsPerModule
acquisitionModule.Initialize(TRs[timepointRange], TEs[timepointRange],↳
    ↳FAs[timepointRange],PHs[timepointRange], IDs[timepointRange])
sequence.modules.append(acquisitionModule)

# Group 3
sequence.modules.append(DeadtimeRecoveryModule(0.300))
sequence.modules.append(T2PreparationModule(echoTime=0.040))
acquisitionModule = FISPAcquisitionModule(dephasingRange=360)
timepointRange = np.
    ↳arange(currentTimepoint,currentTimepoint+timepointsPerModule-1);↳
    ↳currentTimepoint = currentTimepoint+timepointsPerModule
acquisitionModule.Initialize(TRs[timepointRange], TEs[timepointRange],↳
    ↳FAs[timepointRange],PHs[timepointRange], IDs[timepointRange])
sequence.modules.append(acquisitionModule)

# Group 4
sequence.modules.append(DeadtimeRecoveryModule(0.300))
sequence.modules.append(T2PreparationModule(echoTime=0.080))
acquisitionModule = FISPAcquisitionModule(dephasingRange=360)
timepointRange = np.
    ↳arange(currentTimepoint,currentTimepoint+timepointsPerModule-1);↳
    ↳currentTimepoint = currentTimepoint+timepointsPerModule
acquisitionModule.Initialize(TRs[timepointRange], TEs[timepointRange],↳
    ↳FAs[timepointRange],PHs[timepointRange], IDs[timepointRange])
sequence.modules.append(acquisitionModule)

# Group 5
sequence.modules.append(DeadtimeRecoveryModule(0.300))
sequence.modules.append(InversionModule(totalDuration=0.100))
acquisitionModule = FISPAcquisitionModule(dephasingRange=360)
timepointRange = np.
    ↳arange(currentTimepoint,currentTimepoint+timepointsPerModule-1);↳
    ↳currentTimepoint = currentTimepoint+timepointsPerModule
acquisitionModule.Initialize(TRs[timepointRange], TEs[timepointRange],↳
    ↳FAs[timepointRange],PHs[timepointRange], IDs[timepointRange])
sequence.modules.append(acquisitionModule)

# Group 6

```

```

sequence.modules.append(DeadtimeRecoveryModule(0.300))
acquisitionModule = FISPACquisitionModule(dephasingRange=360)
timepointRange = np.
    ↳arange(currentTimepoint,currentTimepoint+timepointsPerModule-1);↳
    ↳currentTimepoint = currentTimepoint+timepointsPerModule
acquisitionModule.Initialize(TRs[timepointRange], TEs[timepointRange],↳
    ↳FAs[timepointRange],PHs[timepointRange], IDs[timepointRange])
sequence.modules.append(acquisitionModule)

# Group 7
sequence.modules.append(DeadtimeRecoveryModule(0.300))
sequence.modules.append(T2PreparationModule(echoTime=0.040))
acquisitionModule = FISPACquisitionModule(dephasingRange=360)
timepointRange = np.
    ↳arange(currentTimepoint,currentTimepoint+timepointsPerModule-1);↳
    ↳currentTimepoint = currentTimepoint+timepointsPerModule
acquisitionModule.Initialize(TRs[timepointRange], TEs[timepointRange],↳
    ↳FAs[timepointRange],PHs[timepointRange], IDs[timepointRange])
sequence.modules.append(acquisitionModule)

# Group 8
sequence.modules.append(DeadtimeRecoveryModule(0.300))
sequence.modules.append(T2PreparationModule(echoTime=0.080))
acquisitionModule = FISPACquisitionModule(dephasingRange=360)
timepointRange = np.
    ↳arange(currentTimepoint,currentTimepoint+timepointsPerModule-1);↳
    ↳currentTimepoint = currentTimepoint+timepointsPerModule
acquisitionModule.Initialize(TRs[timepointRange], TEs[timepointRange],↳
    ↳FAs[timepointRange],PHs[timepointRange], IDs[timepointRange])
sequence.modules.append(acquisitionModule)

# Group 9
sequence.modules.append(DeadtimeRecoveryModule(0.300))
sequence.modules.append(InversionModule(totalDuration=0.250))
acquisitionModule = FISPACquisitionModule(dephasingRange=360)
timepointRange = np.
    ↳arange(currentTimepoint,currentTimepoint+timepointsPerModule-1);↳
    ↳currentTimepoint = currentTimepoint+timepointsPerModule
acquisitionModule.Initialize(TRs[timepointRange], TEs[timepointRange],↳
    ↳FAs[timepointRange],PHs[timepointRange], IDs[timepointRange])
sequence.modules.append(acquisitionModule)

# Group 10
sequence.modules.append(DeadtimeRecoveryModule(0.300))
acquisitionModule = FISPACquisitionModule(dephasingRange=360)

```

```

timepointRange = np.
    ↳arange(currentTimepoint,currentTimepoint+timepointsPerModule-1);␣
    ↳currentTimepoint = currentTimepoint+timepointsPerModule
acquisitionModule.Initialize(TRs[timepointRange], TEs[timepointRange],␣
    ↳FAs[timepointRange],PHs[timepointRange], IDs[timepointRange])
sequence.modules.append(acquisitionModule)

# Group 11
sequence.modules.append(DeadtimeRecoveryModule(0.300))
sequence.modules.append(T2PreparationModule(echoTime=0.040))
acquisitionModule = FISPACquisitionModule(dephasingRange=360)
timepointRange = np.
    ↳arange(currentTimepoint,currentTimepoint+timepointsPerModule-1);␣
    ↳currentTimepoint = currentTimepoint+timepointsPerModule
acquisitionModule.Initialize(TRs[timepointRange], TEs[timepointRange],␣
    ↳FAs[timepointRange],PHs[timepointRange], IDs[timepointRange])
sequence.modules.append(acquisitionModule)

# Group 12
sequence.modules.append(DeadtimeRecoveryModule(0.300))
sequence.modules.append(T2PreparationModule(echoTime=0.080))
acquisitionModule = FISPACquisitionModule(dephasingRange=360)
timepointRange = np.
    ↳arange(currentTimepoint,currentTimepoint+timepointsPerModule-1);␣
    ↳currentTimepoint = currentTimepoint+timepointsPerModule
acquisitionModule.Initialize(TRs[timepointRange], TEs[timepointRange],␣
    ↳FAs[timepointRange],PHs[timepointRange], IDs[timepointRange])
sequence.modules.append(acquisitionModule)

#partitionSpoilingModule = SpoilerModule(dephasingRange=4*np.pi,␣
    ↳totalDuration=20/1000)
sequence.ExportToJson(castToIntegers=True)
print(sequence.modules[1].timepoints[0])

```

Unique IDs: 48 || Total Length: 1776  
(0.0083, 0.0022, 4., 0., 0)

```

[2]: import json
from mrftools import SequenceParameters, SequenceUnits, Units
with open("YongMyelinWaterFraction_dev.sequence") as inputFile:
    inputJson = json.load(inputFile)
    importedSequence = SequenceParameters.FromJson(inputJson)

sequence.ConvertUnits(SequenceUnits(Units.SECONDS, Units.DEGREES))

```

Input file mrftools Version: 0.2.1  
Time: Units.MICROSECONDS Angle: Units.CENTIDEGREES (8300., 2200., 400., 0., 0)

Time: Units.SECONDS Angle: Units.DEGREES (0.0083, 0.0022, 4., 0., 0)

## 1.1 Simulation

```
[3]: import torch
import numpy as np
from matplotlib import pyplot as plt
from mrftools import DictionaryParameters, Simulation, WHITE_MATTER_3T,
    ↳GREY_MATTER_3T, CSF_3T, DictionaryEntry
from mpl_toolkits.axes_grid1 import make_axes_locatable

# Create dictionary definition programmatically
TenPctDict = DictionaryParameters.GenerateFixedPercent("TenPctDict",
    ↳percentStepSize=10)

# Run the simulation
newSimulation = Simulation(sequence, TenPctDict, "newSimulation", numSpins=300)
newSimulation.Execute(numBatches=10)

# From https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6263822/
LYMPH_NODE_CORTEX_3T = np.array([(1.435, 0.102, 1.)],dtype=DictionaryEntry)
LYMPH_NODE_HILUM_3T = np.array([(0.714, 0.119, 1.)],dtype=DictionaryEntry)
# From https://mri-q.com/uploads/3/4/5/7/34572113/normal_relaxation_times_at_3t.
    ↳pdf
FAT_3T = np.array([(0.400, 0.025, 1.)],dtype=DictionaryEntry)

# Find Dictionary Entry Numbers with tissues of interest
WMIndex, WMEntry = TenPctDict.GetNearestEntry(WHITE_MATTER_3T['T1'],
    ↳WHITE_MATTER_3T['T2'])
GMIndex, GMEntry = TenPctDict.GetNearestEntry(GREY_MATTER_3T['T1'],
    ↳GREY_MATTER_3T['T2'])
CSFIndex, CSFEntry = TenPctDict.GetNearestEntry(CSF_3T['T1'], CSF_3T['T2'])
LNCIndex, LNCEnt = TenPctDict.GetNearestEntry(LYMPH_NODE_CORTEX_3T['T1'],
    ↳LYMPH_NODE_CORTEX_3T['T2'])
LNHIndex, LNHEnt = TenPctDict.GetNearestEntry(LYMPH_NODE_HILUM_3T['T1'],
    ↳LYMPH_NODE_HILUM_3T['T2'])
FATIndex, FATEnt = TenPctDict.GetNearestEntry(FAT_3T['T1'], FAT_3T['T2'])
tissuesIndicesToPlot = [WMIndex, GMIndex, LNCIndex, LNHIndex, FATIndex,
    ↳CSFIndex];
tissueLabelsToPlot = ["White Matter", "Grey Matter", "Lymph Node (Cortex)",
    ↳"Lymph Node (Hilum)", "Fat", "CSF"]

# Plot the timecourses in time domain and readout domain
plt.figure(figsize=(20,10))
plt.subplot(211);plt.plot(newSimulation.times, np.abs(newSimulation.
    ↳timeDomainResults[:,tissuesIndicesToPlot]))
plt.legend(tissueLabelsToPlot); plt.title('Magnetization vs Time (s)')
```

```
plt.subplot(212);plt.plot(np.abs(newSimulation.results[:,tissuesIndicesToPlot]))
plt.legend(tissueLabelsToPlot); plt.title('Magnetization vs Readout')

# Plot the inner product differences of the tissues of interest
plt.figure()
innerProductMatrix = Simulation.GetInnerProducts(newSimulation.results[:,
↪,tissuesIndicesToPlot], newSimulation.results[:,tissuesIndicesToPlot])
fig, ax = plt.subplots()
divider = make_axes_locatable(ax)
cax = divider.append_axes('right', size='5%', pad=0.05)
im = ax.imshow(np.abs(innerProductMatrix)**10, 'gray');
ax.set_xticklabels([""]+tissueLabelsToPlot,rotation='vertical');
ax.xaxis.tick_top()
ax.set_yticklabels([""]+tissueLabelsToPlot);
fig.colorbar(im, cax=cax, orientation='vertical')
plt.show()
```

Dictionary Parameter set 'TenPctDict' initialized with 2352 entries

Simulating 10 batch(s) of ~235 dictionary entries

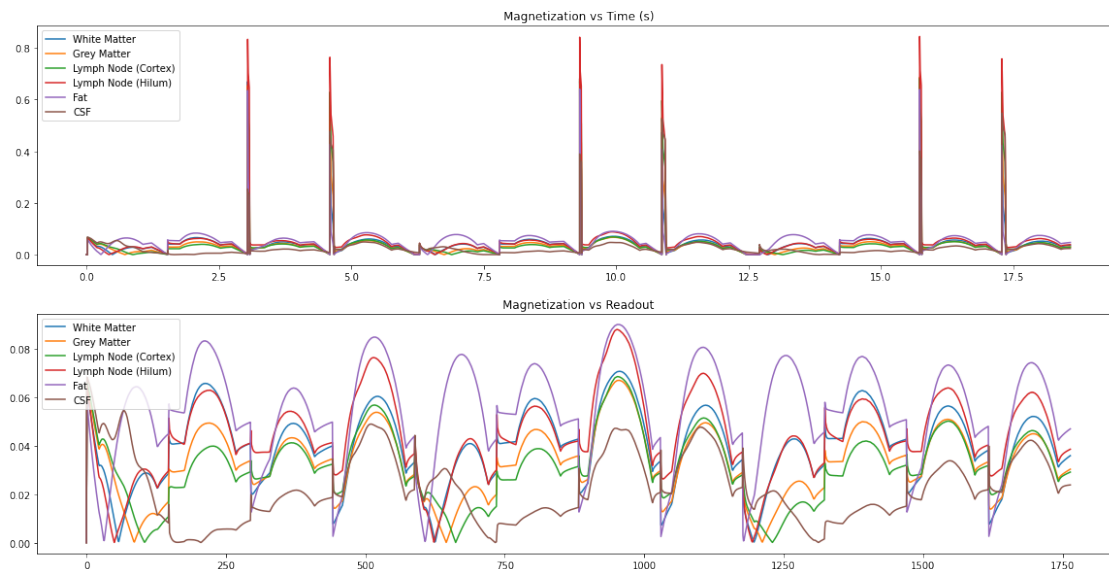
100%| | 10/10 [01:07<00:00, 6.79s/it]

/tmp/ipykernel\_1312530/536812272.py:44: UserWarning: FixedFormatter should only be used together with FixedLocator

```
ax.set_xticklabels([""]+tissueLabelsToPlot,rotation='vertical');
```

/tmp/ipykernel\_1312530/536812272.py:46: UserWarning: FixedFormatter should only be used together with FixedLocator

```
ax.set_yticklabels([""]+tissueLabelsToPlot);
```



<Figure size 432x288 with 0 Axes>

