Dynamic OCT simulator: Module design

2024-09-11 (Wed)

# Class: scattererPositions

## Members

* Array of scatterers
* i-th scatterer
  + [xi, yi, zi]: Physical position
  + ri: Complex reflectivity
  + Motion (velocity)
* Physical field size (x, y, z)

## Methods

### scattererPositions::constructor (initializeScattererPositions)

* Role  
  This function seeds scatterers in the ScattererPosition object.
* Input
* ScattererPositions object (this)
* Scatterer density
* Reflectivity r (single constant)
* Outputs
  + None (the ScattererPositions object of the input will be updated.

### scattererPositions::positionsUpdate()

* Inputs
  + ScattererPositions object (this)
  + Time interval t [s] (real double)
* Output
  + None (the ScattererPositions object of the input will be updated.

## scateererPositions::velocitiesSet

* Role  
  This function updates the velocity of each scatterer based on the given motion type and its parameter.
* Inputs
* ScattererPositions object (this)
* Motion type {“flow”, “random ballistic”, “diffusion”}
* Motion parameters (flow speed for flow and random ballistic, diffusion coefficient for diffusion)

# Class: complexPsfField

## Role

To the 3D complex array of a complex PSF.

* **Member variables**
  + psf : 3D complex array (i.e., complex field)
  + psfSpectrum : 3D complex array (i.e., complex field) in frequency space
    - flag\_psfSpectrum = True if spectrum has been computed. False if not.
    - buff\_zeroMerginSize [pix]
  + numerical3dFieldparameters
  + octSystemParameters
  + complexPsfParameters (Class)

## Methods

### complexPsfField::constructor ()

* Inputs
  + numerical3dFieldparameters
  + Resolutions (wx, wy, wz, resolution type)
  + Peak intensity = 1.0
* Process
  + flag\_psfSpectrum == False

### complexPsfField::psfSpectrumGet()

* Input
  + zeroMarginSize
* Output
  + psfSpectrum
* Process  
  If flag\_psfSpectrum == False OR self.zeroMerginSize != zeroMerginSize  
   Compute FFT of psf  
   set the result to psfSpectrum  
   flag\_psfSpectrum = True  
   self.zeroMerginSize = zeroMerginSize  
   return (psfSpectrum)  
  else:

return(psfSpectrum)

### complexPsfField::optimalZeroMerginSizeForFftGet()

# Class: complexPsfParameters

## Members

* Resolutions (wx, wy, wz, resolution type = {‘FWHM of intensity’, ‘1/e2 of intensity’, ‘FWHM of amplitude’...})
* PSF type “Gaussian”
* (Defocus [m])
* (Aberration: Zernike coefficients)

# Class: scattererField

## Role

To holds the complex field of scatterers

* **Member variables**
  + field (3D complex array)
  + numerical3dFieldparameters
  + octSystemParameters

## Methods

### scattererField::generate()

* Inputs
  + Obj: scattererPositions

### scattererField::constructor

Do nothing

# Class (structure): numerical3dFieldParameters

## Members

* Physical field size (x, y, z)
* Analytical field size (x,y,z)
* Pixel numbers (x, y, z)
* Pixel\_separation (x, y, z)
* Probe wavelength

# Class: complexOctField

## Role

To holds the complex OCT signal (3D volume)

* **Member variables**
  + field (3D complex array)
  + numerical3dFieldparameters
  + psfSpectrumBuffer : 3D complex array (i.e., complex field) in frequency space
    - flag\_psfSpectrum = True if spectrum has been computed. False if not.

## Methods

### complexOctField::generate()

* Inputs
  + scattererField class instance
  + complexPsfField class instance
* Outputs
  + Note (overwrite self.field)
* Actions
  + Convolve the scattererField with complexPsfField.
  + Note: Before FFT the fields, we need to add zeros at the meringues (or embedded the fields into a larger zero-fields)

# Class: complexNoiseField

## Members

* field (3D complex array)
* numerical3dFieldParameters
* complexPsfField (Class)

## Methods

### complexNoiseField::generate()

* Inputs
  + Noise energies (shot, RIN, non-optical)

### ~~complexNoiseField::generateEachNoise()~~

* ~~Role  
  Generate noise field of each type with normalized energy~~
* ~~Inputs~~ 
  + ~~Mean energy~~
  + ~~noiseType = {‘shot’, ‘RIN’, ‘non-optical’}~~

# ~~Class: octSystemParameters~~

## ~~Members~~

* ~~(complexPsfParameters)~~
* ~~(Scan speed)~~
* ~~(PSF parameters?)~~

**scattererPositions**

**complexPsfField**

**complexPsfParameters**

**scattererField**

**numerical3dFieldParameters**

**complexOctField**

**complexNoiseField**