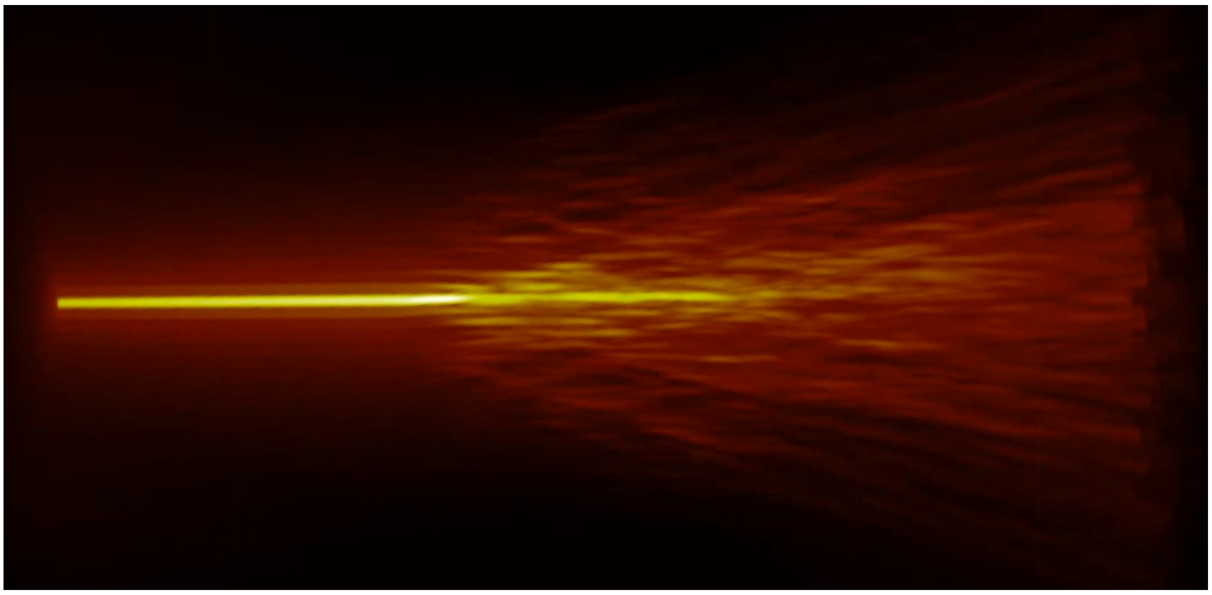
biobeam Documentation

Release 0.1.0

Martin Weigert

July 01, 2016

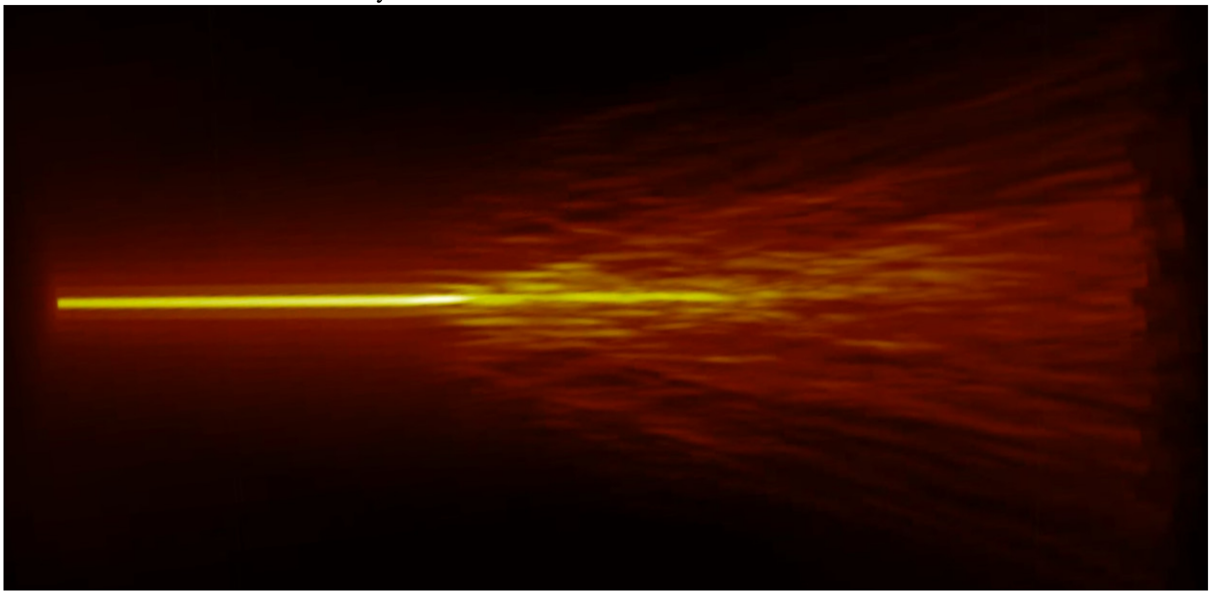
Contents



The full documentation with examples can be found here.

1 Introduction

Biobeams is awesome. And so are you!



This is something I want to say that is not in the docstring.

2 Installation

To nicely render the 3d output it is advisible to install *Spimagine*, an OpenCL accelerated renderer Spimagine

```
pip install spimagine
```

After that you should be able to simple do

```
pip install biobeams
```

3 Basic Usage

Contents

3.1 Beam propagation

class `biobeam.`**`Bpm3d`**(*size=None*, *shape=None*, *units=None*, *dn=None*, *lam=0.5*, *n0=1.0*, *simul_xy=None*, *simul_z=1*, *n_volumes=1*, *enforce_subsampled=False*, *fftplan_kwargs={}*)

the main class for gpu accelerated bpm propagation

`__init__`(*size=None*, *shape=None*, *units=None*, *dn=None*, *lam=0.5*, *n0=1.0*, *simul_xy=None*, *simul_z=1*, *n_volumes=1*, *enforce_subsampled=False*, *fftplan_kwargs={}*)

Parameters

- **`size`** (*`(Sx, Sy, Sz)`*) – the size of the geometry in microns (Sx,Sy,Sz)
- **`shape`** (*`(Nx, Ny, Nz)`*) – the shape of the geometry in pixels (Nx,Ny,Nz)
- **`units`** (*`(dx, dy, dz)`*) – the voxelsizes in microns (dx,dy,dz)
- **`dn`** (*`ndarray (float32|complex64)`*) – refractive index distribution, dn.shape != (Nz,Ny,Nx)
- **`lam`** (*`float`*) – the wavelength in microns
- **`n0`** (*`float`*) – the refractive index of the surrounding media
- **`simul_xy`** (*`(Nx, Ny, Nz), optional`*) – the shape of the 2d computational geometry in pixels (Nx,Ny) (e.g. subsampling in xy)
- **`simul_z`** (*`int, optional`*) – the subsampling factor along z
- **`n_volumes`** (*`int`*) – splits the domain into chunks if GPU memory is not large enough (will be set automatically)

Example

```
>>> m = Bpm3d(size = (10,10,10),shape = (256,256,256),units = (0.1,0.1,0.1),lam = 0.488,r
```

`aberr_at`(*NA=0.4*, *center=(0, 0, 0)*, *n_zern=20*, *n_integration_steps=200*)

c = (cx,cy,cz) in realtive pixel coordinates wrt the center

returns phi, zern

`aberr_field_grid`(*NA*, *cxs*, *cys*, *cz*, *n_zern=20*, *n_integration_steps=200*)

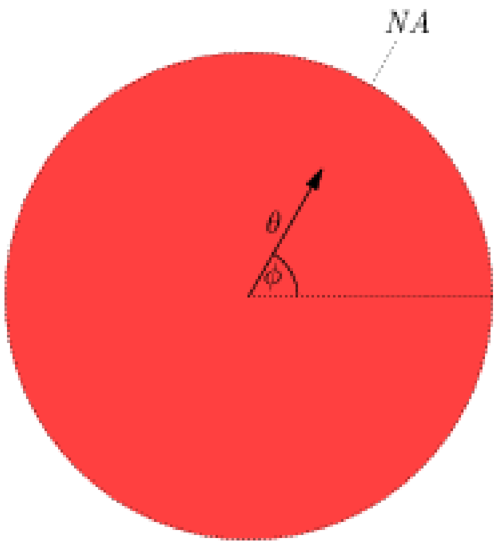
cxs, cys are equally spaced 1d arrays defining the grid

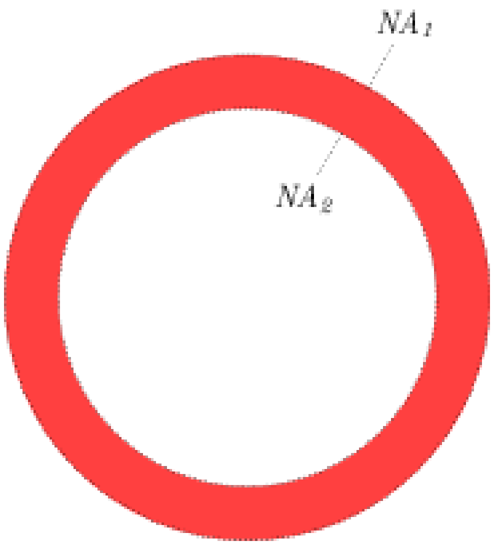
4 Input Beam patterns

Contents

4.1 Gaussian/Bessel beams

Gaussian/Bessel beams





`biobeam.`**`focus_field_beam`**(*shape*, *units*, *lam=0.5*, *NA=0.6*, *n0=1.0*, *return_all_fields=False*, *n_integration_steps=200*)

calculates the focus field for a perfect, aberration free optical system for x polzarized illumination via the vectorial debye diffraction integral (see [1]). The pupil function is given by numerical aperture(s) NA (that can be a list to model bessel beams, see further below)

Parameters

- **`shape`** (*`Nx, Ny, Nz`*) – the shape of the geometry
- **`units`** (*`dx, dy, dz`*) – the pixel sizes in microns

[1] Matthew R. Foreman, Peter Toeroek, *Computational methods in vectorial imaging*, Journal of Modern Optics, 2011, 58, 5-6, 339

- **lam** (*float*) – the wavelength of light used in microns
- **NA** (*float/list*) – the numerical aperture(s) of the illumination objective that is either a single number (for gaussian beams) or an even length list of NAs (for bessel beams), e.g. *NA = [0.5,0.55]* lets light through the annulus 0.5<0.55 (making a bessel beam) or *NA = [0.1,0.2,0.5,0.6]* lets light through the annulus 0.1<0.2 and 0.5<0.6 making a beating double bessel beam...
- **n0** (*float*) – the refractive index of the medium
- **n_integration_steps** (*int*) – number of integration steps to perform
- **return_all_fields** (*boolean*) – if True returns also the complex vectorial field components

Returns

- **u** (*ndarray*) – the intensity of the focus field
- **(u,ex,ey,ez)** (*list(ndarray)*) – the intensity of the focus field and the complex field components (if return_all_fields is True)

Example

```
>>> u = focus_field_beam((128,128,128), (0.1,0.1,.1), lam=.5, NA = .4)
```

References

biobeam.**focus_field_beam_plane**(*shape=(128, 128)*, *units=(0.1, 0.1)*, *z=0.0*, *lam=0.5*, *NA=0.6*, *n0=1.0*, *ex_g=None*, *n_integration_steps=200*)

calculates the complex 2d input field at position -z of a perfect, aberration free optical system

Parameters

- **shape** (*Nx, Ny*) – the 2d shape of the geometry
- **units** (*dx, dy*) – the pixel sizes in microns
- **z** (*float*) – defocus position in microns, such that the beam would focus at z e.g. an input field with z = 10. would hav its focus spot after 10 microns
- **lam** (*float*) – the wavelength of light used in microns
- **NA** (*float/list*) – the numerical aperture(s) of the illumination objective that is either a single number (for gaussian beams) or an even length list of NAs (for bessel beams), e.g. NA = [0.5,0.55] lets light through the annulus 0.5<0.55 (making a bessel beam) or NA = [0.1,0.2,0.5,0.6] lets light through the annulus 0.1<0.2 and 0.5<0.6 making a beating double bessel beam...
- **n0** (*float*) – the refractive index of the medium
- **n_integration_steps** (*int*) – number of integration steps to perform

Returns **ex** – the complex field

Return type ndarray

Example

```
>>> # the input pattern of a bessel beam that will focus after 4 microns
>>> ex = focus_field_beam_plane((256,256), (0.1,0.1), z = 4, lam=.5, NA = (.4,.5))
```

See also:

`biobeam.focus_field_beam()` the 3d function

4.2 Cylindrical Lens

`biobeam.`**`focus_field_cylindrical`**(*shape*, *units*, *lam=0.5*, *NA=0.3*, *n0=1.0*, *return_all_field=False*, *n_integration_steps=100*)

calculates the focus field for a perfect, aberration free cylindrical lens after x polarized illumination via the vectorial debye diffraction integral (see [2]). The pupil function is given by the numerical aperture NA

Parameters

- **`shape`** (*`Nx, Ny, Nz`*) – the shape of the geometry
- **`units`** (*`dx, dy, dz`*) – the pixel sizes in microns
- **`lam`** (*`float`*) – the wavelength of light used in microns
- **`NA`** (*`float`*) – the numerical aperture of the lens
- **`n0`** (*`float`*) – the refractive index of the medium
- **`return_all_fields`** (*`boolean`*) – if True, returns u,ex,ey,ez where ex/ey/ez are the complex field components
- **`n_integration_steps`** (*`int`*) – number of integration steps to perform
- **`return_all_fields`** – if True returns also the complex vectorial field components

Returns

- **u** (*ndarray*) – the intensity of the focus field
- **(u,ex,ey,ez)** (*list(ndarray)*) – the intensity of the focus field and the complex field components (if return_all_fields is True)

Example

```
>>> u, ex, ey, ez = focus_field_cylindrical((128,128,128), (0.1,0.1,.1), lam=.5, NA = .4, ret
```

References

`biobeam.`**`focus_field_cylindrical_plane`**(*shape=(128, 128)*, *units=(0.1, 0.1)*, *z=0.0*, *lam=0.5*, *NA=0.6*, *n0=1.0*, *ex_g=None*, *n_integration_steps=200*)

calculates the complex 2d input field at position -z of a perfect, aberration free optical system

Parameters

- **`shape`** (*`Nx, Ny`*) – the 2d shape of the geometry
- **`units`** (*`dx, dy`*) – the pixel sizes in microns
- **`z`** (*`float`*) – defocus position in microns, such that the beam would focus at z e.g. an input field with z = 10. would hav its focus spot after 10 microns
- **`lam`** (*`float`*) – the wavelength of light used in microns

[2] Colin J. R. Sheppard: Cylindrical lenses—focusing and imaging: a review, Appl. Opt. 52, 538-545 (2013)

- **NA** (*float/list*) – the numerical aperture(s) of the illumination objective that is either a single number (for gaussian beams) or an even length list of NAs (for bessel beams), e.g. NA = [0.5,0.55] lets light through the annulus 0.5<0.55 (making a bessel beam) or NA = [0.1,0.2,0.5,0.6] lets light through the annulus 0.1<0.2 and 0.5<0.6 making a beating double bessel beam...
- **n0** (*float*) – the refractive index of the medium
- **n_integration_steps** (*int*) – number of integration steps to perform

Returns **ex** – the complex field

Return type ndarray

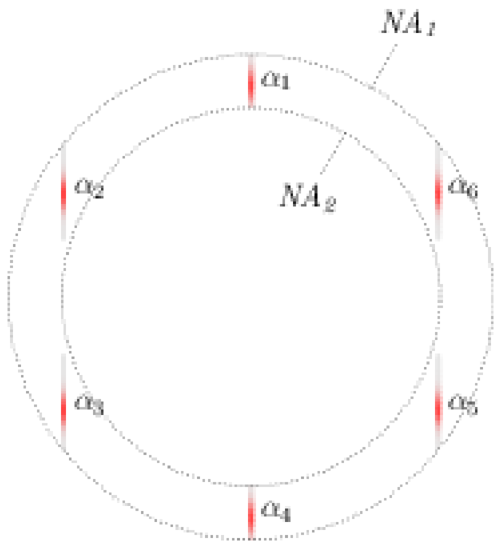
Example

```
>>> # the input pattern of a bessel beam that will focus after 4 microns
>>> ex = focus_field_cylindrical_plane((256,256), (0.1,0.1), z = 4., lam=.5, NA = (.4,.5))
```

See also:

biobeam.focus_field_cylindrical() the 3d function

4.3 Bessel Lattices



biobeam.**focus_field_lattice**(*shape*, *units*, *lam=0.5*, *NA1=0.4*, *NA2=0.5*, *sigma=0.1*, *kpoints=6*, *n0=1.0*, *n_integration_steps=100*)

calculates the focus field for a bessel lattice The pupil function consists out of discrete points (kpoints) superimposed on an annulus (NA1<NA2) which are smeared out by a 1d gaussian of given sigma creating an array of bessel beams in the focal plane (see [3]).

Parameters

- **shape** (*Nx, Ny, Nz*) – the shape of the geometry
- **units** (*dx, dy, dz*) – the pixel sizes in microns
- **lam** (*float*) – the wavelength of light used in microns
- **NA1** (*float/list*) – the numerical aperture of the inner ring
- **NA2** (*float/list*) – the numerical aperture of the outer ring

[3] Chen et al. Lattice light-sheet microscopy: imaging molecules to embryos at high spatiotemporal resolution. Science 346, (2014).

- **`sigma`** (*`float`*) – the standard deviation of the gaussian smear function applied to each point on the aperture (the bigger sigma, the tighter the sheet in y)
- **`kpoints`** (*`int/ (2,N) array`*) – defines the set of points on the aperture that create the lattice, can be - a (2,N) ndarray, such that kpoints[:,i] are the coordinates of the ith point - a single int, defining points on a regular polygon (e.g. 4 for a square lattice, 6 for a hex lattice) $k_i = \arcsin \frac{NA_1+NA_2}{2n_0} \begin{pmatrix} \cos\phi_i \\ \sin\phi_i \end{pmatrix}$ $, \phi_i = \frac{\pi}{2} + \frac{2i}{N}$
- **`n0`** (*`float`*) – the refractive index of the medium
- **`n_integration_steps`** (*`int`*) – number of integration steps to perform
- **`return_all_fields`** (*`boolean`*) – if True, returns u,ex,ey,ez where ex/ey/ez are the complex vector field components

Returns

- **u** (*ndarray*) – the intensity of the focus field
- **(u,ex,ey,ez)** (*list(ndarray)*) – the intensity of the focus field and the complex field components (if return_all_fields is True)

Example

```
>>> u = focus_field_lattice((128,128,128), (0.1,0.1,.1), lam=.5, NA1 = .44, NA2 = .55, kpoint
```

References

`biobeam.`**`focus_field_lattice_plane`**(*shape=(256, 256), units=(0.1, 0.1), z=0.0, lam=0.5, NA1=0.4, NA2=0.5, sigma=0.1, Npoly=6, n0=1.0, apodization_bound=10, ex_g=None, n_integration_steps=100*)

5 Examples

5.1 Plane wave scattered by sphere

```
# create the refractive index difference
x = 0.1 * np.arange(-128,128)
Z, Y, X = np.meshgrid(x,x,x,indexing = "ij")
R = np.sqrt(X**1+Y**2+Z**2)
dn = 0.05*(R<2.)

# create the computational geometry
m = Bpm3d(dn = dn, units = (0.1,0.1,0.1), lam = 0.5)

# propagate a plane wave and return the intensity
u = m._propagate()

# vizualize
import matplotlib.pyplot as plt
plt.subplot(1,2,1)
plt.imshow(u[...,128], cmap = "hot")
plt.title("zy slice")
plt.subplot(1,2,2)
plt.imshow(u[128,...], cmap = "hot")
plt.title("xy slice")
```

5.2 Light sheet through cell phantom

5.3 Computing the psf inside a cell phantom

5.4 Aberration from sphere

6 Some Examples

or not?

```
print "huhu"
```

```
huhu
```

Index