nrefocus Documentation

Release 0.4.2

Paul Müller

Apr 23, 2021

CONTENTS:

Nrefocus is a Python 3 library that allows to numerically refocus (including autofocusing) complex wave fields. This is the documentaion of nrefocus version 0.4.2.

CHAPTER

ONE

INTRODUCTION

This package provides methods for numerical propagation of a complex wave in free space. The available propagators are the angular spectrum method (*helmholtz*) and the Fresnel approximation (*fresnel*). Both implementations are convolution-based. The angular spectrum method is suited for near-field propagation (numerical focusing) and yields better results than the Fresnel approximation. The single Fourer transform-based Fresnel propagation method which is suitable for far-field propagation is not implemented in this package.

1.1 Obtaining nrefocus

You can install nrefocus via:

```
pip install nrefocus
```

If you would like to take advantage of fast Fourer transforms with PyFFTW, please also install the *pyfftw* package or use the extras key *FFTW*:

```
pip install nrefocus[FFTW]
```

The source code of nrefocus is available at https://github.com/RI-imaging/nrefocus.

1.2 Citing nrefocus

Please cite this package if you are using it in a scientific publication.

This package should be cited like this[1].

You can find out what version you are using by typing (in a Python console):

```
>>> import nrefocus
>>> nrefocus.__version__
'0.1.2'
```

[1] Paul Müller (2013) *nrefocus: Python algorithms for numerical focusing* (Version x.x.x) [Software]. Available at https://pypi.python.org/pypi/nrefocus/.

1.3 Acknowledgments

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 282060.

1.4 References

CHAPTER

TWO

THEORY

The derivations given here are treated in more detail in the relevant literature, e.g. [[ST91]] and [[Goo05]].

2.1 Optical transfer function

Let us consider a wave field $u(\mathbf{r_0})$ whose values we know at an initial plane $\mathbf{r_0} = (x_0, y_0, z_0)$ (z_0 fixed). The field has a certain vacuum wavelength λ and is traveling through a homogeneous medium with refractive index n_m. From the knowledge of the wave field at the plane $\mathbf{r_0}$ and its wavelength λ/n_m, we can infer the direction of propagation of the wave field for every point in $\mathbf{r_0}$. We rewrite the field at $\mathbf{r_0}$ as an angular spectrum, a sum over all possible directions $\mathbf{s} = (p, q, M)$, assuming that the field is only traveling from left to right

$$\begin{aligned}
u(\mathbf{r_0}) &= \iint dpdq\, A(p,q) e^{i k_\mathrm{m}(p x_0 + q y_0 + M z_0)} \\
|\mathbf{s}| &= p^2 + q^2 + M^2 = 1 \\
M &= \sqrt{1 - p^2 - q^2}.
\end{aligned}$$

The equation above describes the Huygens-Fresnel principle: the value of the field u at a certain position $\mathbf{r_0}$ at the initial plane (point source) is defined as an integral over all possible plane waves with wavenumber $k_\mathrm{m} = \frac{2\pi n_\mathrm{m}}{\lambda}$, weighted with the amplitude $A(p, q)$.

Let us now consider the 2D Fourier transform of $u(\mathbf{r_0})$.

$$\begin{aligned}
\widehat{U}_0(k_\mathrm{x}, k_\mathrm{y}) &= \frac{1}{2\pi} \iint dx_0 dy_0 \iint dpdq\, A(p,q) e^{i k_\mathrm{m}(p x_0 + q y_0 + M z_0)} e^{-i(k_\mathrm{x} x_0 + k_\mathrm{y} y_0)} \\
&= \frac{1}{2\pi} \iint dx_0 dy_0 \iint dpdq\, A(p,q) e^{i k_\mathrm{m} M z_0} e^{i x_0 (k_\mathrm{m} p - k_\mathrm{x})} e^{i y_0 (k_\mathrm{m} q - k_\mathrm{y})} \\
&= \frac{2\pi}{k_\mathrm{m}^2} A(k_\mathrm{x}, k_\mathrm{y}) e^{i k_\mathrm{m} M z_0}
\end{aligned}$$

Here we made use of the identity of the delta distribution

$$\begin{aligned}
\frac{1}{2\pi} \int dx_0\, e^{i x_0 (k_\mathrm{m} p - k_\mathrm{x})} &= \delta(k_\mathrm{m} p - k_\mathrm{x}) = \frac{1}{k_\mathrm{m}} \delta(p - k_\mathrm{x}/k_\mathrm{m}) \\
\frac{1}{2\pi} \int dy_0\, e^{i y_0 (k_\mathrm{m} q - k_\mathrm{y})} &= \delta(k_\mathrm{m} q - k_\mathrm{y}) = \frac{1}{k_\mathrm{m}} \delta(q - k_\mathrm{y}/k_\mathrm{m})
\end{aligned}$$

If we now perform the same procedure for a different position $\mathbf{r}_\mathrm{d} = (x_0, y_0, z_\mathrm{d})$, we will see that the Fourier transform of the field becomes

$$\widehat{U}_\mathrm{d}(k_\mathrm{x}, k_\mathrm{y}) = \frac{2\pi}{k_\mathrm{m}^2} A(k_\mathrm{x}, k_\mathrm{y}) e^{i k_\mathrm{m} M z_\mathrm{d}}.$$

Thus, the propagation of the field $u(\mathbf{r_0})$ by a distance $d = z_\mathrm{d} - z_0$ is described by a multiplication with the transfer function

$$\mathcal{H}^\mathrm{Helmholtz} = e^{ik_\mathrm{m}Md}$$

in Fourier space. This is the basis of the convolution-based numerical propagation algorithms implemented in nrefocus. The process of numerical propagation with the angular spectrum method can be written as

$$u(\mathbf{r_d}) = \mathcal{F}^{-1}\left\{\mathcal{F}\{u(\mathbf{r_0})\} \cdot e^{ik_\mathrm{m}Md}\right\}$$

with the Fourier transform $\mathcal{F}$ and its inverse $\mathcal{F}^{-1}$. With the convolution operator $*$, we may rewrite this equation to

$$u(\mathbf{r_d}) = u(\mathbf{r_0}) * \mathcal{F}^{-1}\left\{e^{ik_\mathrm{m}Md}\right\}.$$

2.2 Fresnel approximation

The Fresnel approximation (or paraxial approximation) uses a Taylor expansion to simplify the exponent of the transfer function $e^{ik_\mathrm{m}Md}$. The exponent can be rewritten as

$$ik_\mathrm{m}Md = ik_\mathrm{m}d\left(1 - p^2 - q^2\right)^{1/2}.$$

If the angles of propagation θ_x and θ_y for each plane wave of the angular spectrum is small, then we can make the paraxial approximation:

$$\theta_\mathrm{x} \approx p$$
$$\theta_\mathrm{y} \approx q$$
$$\theta^2 = \theta_\mathrm{x}^2 + \theta_\mathrm{y}^2 \approx p^2 + q^2$$

We now Taylor-expand the exponent around small values of θ

$$ik_\mathrm{m}d\left(1 - \theta^2\right)^{1/2} \approx ik_\mathrm{m}d\left(1 - \frac{\theta^2}{2} + \frac{\theta^4}{8} - \dots\right).$$

The Fresnel approximation discards the third term ($\sim \theta^4$) and the transfer function then reads:

$$e^{ik_\mathrm{m}Md} \approx e^{ik_\mathrm{m}d} \cdot e^{-\frac{ik_\mathrm{m}d(p^2+q^2)}{2}}$$
$$e^{i\sqrt{k_\mathrm{m}^2 - k_\mathrm{x}^2 - k_\mathrm{y}^2}d} \approx e^{ik_\mathrm{m}d} \cdot e^{-\frac{id(k_\mathrm{x}^2+k_\mathrm{y}^2)}{2k_\mathrm{m}}}$$
$$\mathcal{H}^\mathrm{Fresnel} = e^{ik_\mathrm{m}d} \cdot e^{-\frac{id(k_\mathrm{x}^2+k_\mathrm{y}^2)}{2k_\mathrm{m}}}$$

Thus, the propagation by a distance distance $d = z_\mathrm{d} - d$ in the Fresnel approximation can be written in the form of the convolution

$$u(\mathbf{r_d}) = e^{ik_\mathrm{m}d} \cdot u(\mathbf{r_0}) * \mathcal{F}^{-1}\left\{e^{-\frac{id(k_\mathrm{x}^2+k_\mathrm{y}^2)}{2k_\mathrm{m}}}\right\}.$$

Note that the Fresnel approximation results in paraboloidal waves ($p^2 + q^2$) whereas spherical waves are used with the Helmholtz equation.

2.3 Transfer functions in nrefocus

The numerical focusing algorithms in this package require the input data u_{in} to be normalized by the incident plane wave $u_0(\mathbf{r_0})$ according to

$$u_{\text{in}}(\mathbf{r_0}) = \frac{u(\mathbf{r_0})}{u_0(\mathbf{r_0})}$$

As a result, the transfer functions change to

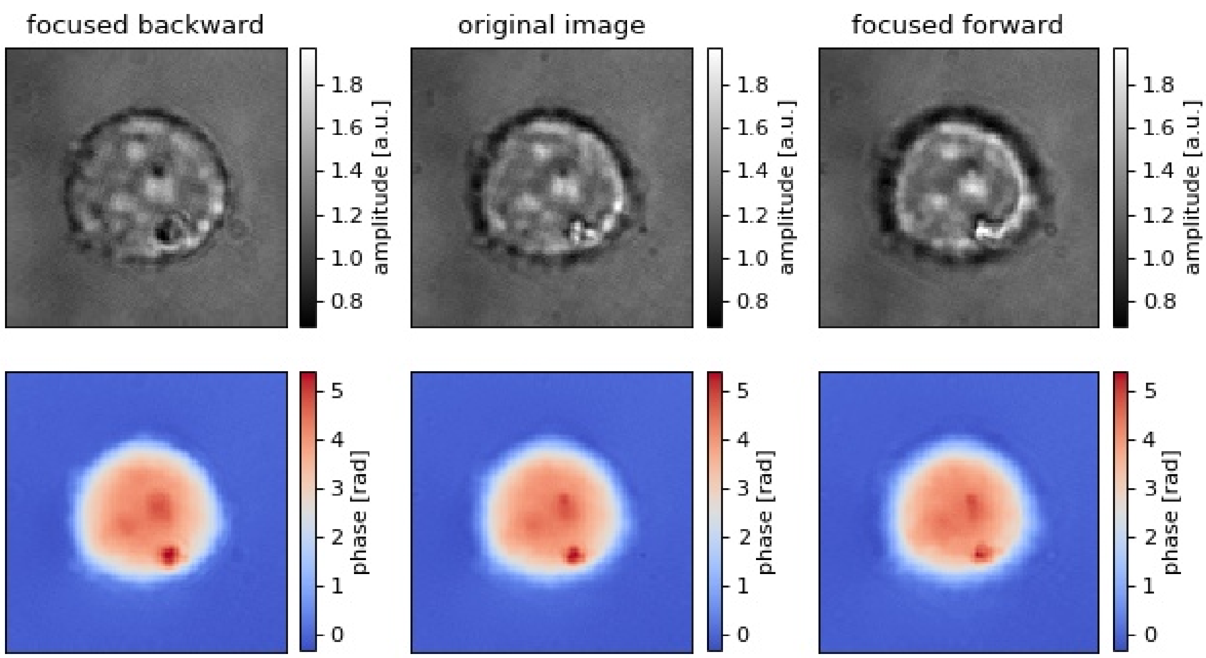
$$\begin{aligned}\mathcal{H}_{\text{norm}}^{\text{Helmholtz}} &= e^{ik_\mathrm{m}(M-1)d} = e^{id\left(\sqrt{k_\mathrm{m}^2 - k_\mathrm{x}^2 - k_\mathrm{y}^2} - k_\mathrm{m}\right)}\\ \mathcal{H}_{\text{norm}}^{\text{Fresnel}} &= e^{-\frac{id(k_\mathrm{x}^2 + k_\mathrm{y}^2)}{2k_\mathrm{m}}}.\end{aligned}$$

CHAPTER THREE

EXAMPLES

3.1 2D Refocusing of an HL60 cell

The data show a live HL60 cell imaged with quadriwave lateral shearing interferometry (SID4Bio, Phasics S.A., France). The diameter of the cell is about 20µm.



refocus_cell.py

```
import matplotlib.pylab as plt
import numpy as np
import unwrap

import nrefocus

from example_helper import load_cell

# load initial cell
cell1 = load_cell("HL60_field.zip")
```

(continues on next page)

(continued from previous page)

```python

# refocus to two different positions
cell2 = nrefocus.refocus(cell1, 15, 1, 1)  # forward
cell3 = nrefocus.refocus(cell1, -15, 1, 1)  # backward

# amplitude range
vmina = np.min(np.abs(cell1))
vmaxa = np.max(np.abs(cell1))
ampkw = {"cmap": plt.get_cmap("gray"),
         "vmin": vmina,
         "vmax": vmaxa}

# phase range
cell1p = unwrap.unwrap(np.angle(cell1))
cell2p = unwrap.unwrap(np.angle(cell2))
cell3p = unwrap.unwrap(np.angle(cell3))
vminp = np.min(cell1p)
vmaxp = np.max(cell1p)
phakw = {"cmap": plt.get_cmap("coolwarm"),
         "vmin": vminp,
         "vmax": vmaxp}

# plots
fig, axes = plt.subplots(2, 3, figsize=(8, 4.5))
axes = axes.flatten()
for ax in axes:
    ax.xaxis.set_major_locator(plt.NullLocator())
    ax.yaxis.set_major_locator(plt.NullLocator())

# titles
axes[0].set_title("focused backward")
axes[1].set_title("original image")
axes[2].set_title("focused forward")

# data
mapamp = axes[0].imshow(np.abs(cell3), **ampkw)
axes[1].imshow(np.abs(cell1), **ampkw)
axes[2].imshow(np.abs(cell2), **ampkw)
mappha = axes[3].imshow(cell3p, **phakw)
axes[4].imshow(cell1p, **phakw)
axes[5].imshow(cell2p, **phakw)

# colobars
cbkwargs = {"fraction": 0.045}
plt.colorbar(mapamp, ax=axes[0], label="amplitude [a.u.]", **cbkwargs)
plt.colorbar(mapamp, ax=axes[1], label="amplitude [a.u.]", **cbkwargs)
plt.colorbar(mapamp, ax=axes[2], label="amplitude [a.u.]", **cbkwargs)
plt.colorbar(mappha, ax=axes[3], label="phase [rad]", **cbkwargs)
plt.colorbar(mappha, ax=axes[4], label="phase [rad]", **cbkwargs)
plt.colorbar(mappha, ax=axes[5], label="phase [rad]", **cbkwargs)

plt.tight_layout()
plt.show()
```

CHAPTER

FOUR

CODE REFERENCE

4.1 Refocus interface

Refocus is a user-convenient interface for numerical refocusing. Each class implements refocusing for a specific dimensionality (1D or 2D fields) using a specific method for refocusing (e.g. numpy FFT or FFTW).

`nrefocus.`**`get_best_interface`**`()`

Return the fastest refocusing interface available

If *pyfftw* is installed, `nrefocus.RefocusPyFFTW` is returned. The fallback is `nrefocus.RefocusNumpy`.

class `nrefocus.`**`RefocusPyFFTW`**(*field*, *wavelength*, *pixel_size*, *medium_index=1.3333*, *distance=0*, *kernel='helmholtz'*, *padding=True*)

Refocusing with FFTW

New in version 0.4.0.

Parameters

- **field** (*2d complex-valued ndarray*) – Input field to be refocused
- **wavelength** (*float*) – Wavelength of the used light [m]
- **pixel_size** (*float*) – Pixel size of the input image [m]
- **medium_index** (*float*) – Refractive index of the medium, defaults to water (1.3333 at 21.5°C)
- **distance** (*float*) – Initial focusing distance [m]
- **kernel** (*str*) – Propagation kernel, one of
  - ”helmholtz”: the optical transfer function $\exp\left(id\left(\sqrt{k_\mathrm{m}^2 - k_\mathrm{x}^2 - k_\mathrm{y}^2} - k_\mathrm{m}\right)\right)$
  - ”fresnel”: paraxial approximation $\exp(-id(k_\mathrm{x}^2 + k_\mathrm{y}^2)/2k_\mathrm{m})$
- **padding** (*bool*) – Whether or not to perform zero-padding

`autofocus`(*metric='average gradient'*, *minimizer='legacy'*, *minimizer_kwargs=None*, *interval=(None, None)*, *roi=None*)

Autofocus the initial field

Parameters

- **metric** (*str*) –
  - “average gradient” : average gradient metric of amplitude
  - ”rms contrast” : RMS contrast of phase data

- "spectrum" : sum of filtered Fourier coefficients

- **minimizer** (*str*) –
  - "legacy": custom nrefocus minimizer
- **interval** (*tuple of floats*) – Approximate interval to search for optimal focus [m]
- **roi** (*rectangular region of interest (x1, y1, x2, y2)*) – Region of interest of *field* for which the metric will be minimized. If not given, the entire *field* will be used.
- **minimizer_kwargs** (*dict*) – Any additional keyword arguments for the minimizer

Returns

- **af_field** (*2d ndarray*) – Autofocused field
- **af_distance** (*float*) – Autofocusing distance
- *[other]* – Any other objects returned by *minimizer* defined via *minimizer_kwargs*

get_kernel(*distance*)

Return the current kernel

Ther kernel type *self.kernel* is used (see `Refocus.__init__()`)

propagate(*distance*)

Propagate the initial field to a certain distance

Parameters **distance** (*float*) – Absolute focusing distance [m]

Returns **refocused_field** – Initial field refocused at *distance*

Return type 2d ndarray

Notes

Any subclass should perform padding with `nrefocus.pad.pad_rem()` during initialization.

property shape

Shape of the padded input field or Fourier transform

class nrefocus.**RefocusNumpy**(*field, wavelength, pixel_size, medium_index=1.3333, distance=0, kernel='helmholtz', padding=True*)

Refocusing with numpy-based Fourier transform

New in version 0.3.0.

Parameters

- **field** (*2d complex-valued ndarray*) – Input field to be refocused
- **wavelength** (*float*) – Wavelength of the used light [m]
- **pixel_size** (*float*) – Pixel size of the input image [m]
- **medium_index** (*float*) – Refractive index of the medium, defaults to water (1.3333 at 21.5°C)
- **distance** (*float*) – Initial focusing distance [m]
- **kernel** (*str*) – Propagation kernel, one of
  - "helmholtz": the optical transfer function $\exp\left(id\left(\sqrt{k_\mathrm{m}^2 - k_\mathrm{x}^2 - k_\mathrm{y}^2} - k_\mathrm{m}\right)\right)$

  - "fresnel": paraxial approximation $\exp(-id(k_\mathrm{x}^2 + k_\mathrm{y}^2)/2k_\mathrm{m})$
- **padding** (*bool*) – Whether or not to perform zero-padding

autofocus (*metric='average gradient'*, *minimizer='legacy'*, *minimizer_kwargs=None*, *interval=(None, None)*, *roi=None*)

Autofocus the initial field

Parameters

- **metric** (*str*) –
  - "average gradient" : average gradient metric of amplitude
  - "rms contrast" : RMS contrast of phase data
  - "spectrum" : sum of filtered Fourier coefficients
- **minimizer** (*str*) –
  - "legacy": custom nrefocus minimizer
- **interval** (*tuple of floats*) – Approximate interval to search for optimal focus [m]
- **roi** (*rectangular region of interest (x1, y1, x2, y2)*) – Region of interest of *field* for which the metric will be minimized. If not given, the entire *field* will be used.
- **minimizer_kwargs** (*dict*) – Any additional keyword arguments for the minimizer

Returns

- **af_field** (*2d ndarray*) – Autofocused field
- **af_distance** (*float*) – Autofocusing distance
- *[other]* – Any other objects returned by *minimizer* defined via *minimizer_kwargs*

get_kernel (*distance*)

Return the current kernel

Ther kernel type *self.kernel* is used (see `Refocus.__init__()`)

propagate (*distance*)

Propagate the initial field to a certain distance

Parameters **distance** (*float*) – Absolute focusing distance [m]

Returns **refocused_field** – Initial field refocused at *distance*

Return type 2d ndarray

Notes

Any subclass should perform padding with `nrefocus.pad.pad_rem()` during initialization.

property shape

Shape of the padded input field or Fourier transform

class nrefocus.**RefocusNumpy1D** (*field*, *wavelength*, *pixel_size*, *medium_index=1.3333*, *distance=0*, *kernel='helmholtz'*, *padding=True*)

Refocus a 1D field with numpy

New in version 0.3.0.

Parameters

- **field** (*1d complex-valued ndarray*) – Input 1D field to be refocused
- **wavelength** (*float*) – Wavelength of the used light [m]
- **pixel_size** (*float*) – Pixel size of the input image [m]
- **medium_index** (*float*) – Refractive index of the medium, defaults to water (1.3333 at 21.5°C)
- **distance** (*float*) – Initial focusing distance [m]
- **kernel** (*str*) – Propagation kernel, one of
  - ”helmholtz”: the optical transfer function $\exp\left(id\left(\sqrt{k_\mathrm{m}^2 - k_\mathrm{x}^2} - k_\mathrm{m}\right)\right)$
  - ”fresnel”: paraxial approximation $\exp(-idk_\mathrm{x}^2/2k_\mathrm{m})$
- **padding** (*bool*) – Whether or not to perform zero-padding

autofocus (*metric='average gradient'*, *minimizer='legacy'*, *minimizer_kwargs=None*, *interval=(None, None)*, *roi=None*)

Autofocus the initial field

Parameters

- **metric** (*str*) –
  - “average gradient” : average gradient metric of amplitude
  - ”rms contrast” : RMS contrast of phase data
  - ”spectrum” : sum of filtered Fourier coefficients
- **minimizer** (*str*) –
  - “legacy”: custom nrefocus minimizer
- **interval** (*tuple of floats*) – Approximate interval to search for optimal focus [m]
- **roi** (*rectangular region of interest (x1, y1, x2, y2)*) – Region of interest of *field* for which the metric will be minimized. If not given, the entire *field* will be used.
- **minimizer_kwargs** (*dict*) – Any additional keyword arguments for the minimizer

Returns

- **af_field** (*2d ndarray*) – Autofocused field
- **af_distance** (*float*) – Autofocusing distance
- *[other]* – Any other objects returned by *minimizer* defined via *minimizer_kwargs*

get_kernel (*distance*)

Return the kernel for a 1D propagation

propagate (*distance*)

Propagate the initial field to a certain distance

Parameters **distance** (*float*) – Absolute focusing distance [m]

Returns **refocused_field** – Initial 1D field refocused at *distance*

Return type 1d ndarray

property shape

Shape of the padded input field or Fourier transform

4.2 Metrics

`nrefocus.metrics.`**`metric_average_gradient`**(*rfi*, *distance*, *roi=None*, **kwargs*)
Compute mean average gradient norm of the amplitude

Notes

The absolute value of the gradient is returned.

`nrefocus.metrics.`**`metric_rms_contrast`**(*rfi*, *distance*, *roi=None*, **kwargs*)
Compute RMS contrast of the phase

Notes

The negative angle of the field is used for contrast estimation.

`nrefocus.metrics.`**`metric_spectrum`**(*rfi*, *distance*, *roi=None*, ***kwargs*)
Compute spectral contrast

Performs bandpass filtering in Fourier space according to optical limit of detection system, approximated by twice the wavelength.

`nrefocus.metrics.`**`METRICS = {'average gradient': <function metric_average_gradient>, 'rms c`**
Available metrics

4.3 Legacy methods

These methods are legacy functions which are kept for backwards-compatibility.

4.3.1 Refocusing

`nrefocus.`**`refocus`**(*field*, *d*, *nm*, *res*, *method='helmholtz'*, *padding=True*)
Refocus a 1D or 2D field

Parameters

- **field** (*1d or 2d array*) – 1D or 2D background corrected electric field (Ex/BEx)
- **d** (*float*) – Distance to be propagated in pixels (negative for backwards)
- **nm** (*float*) – Refractive index of medium
- **res** (*float*) – Wavelenth in pixels
- **method** (*str*) – Defines the method of propagation; one of
  - ”helmholtz” : the optical transfer function *exp(idk(M-1))*
  - ”fresnel” : paraxial approximation $exp(idk^2/k)$
- **padding** (*bool*) – perform padding with linear ramp from edge to average to reduce ringing artifacts.

  New in version 0.1.4.

Returns

Return type Electric field at *d*.

Notes

This method uses `nrefocus.RefocusNumpy` for refocusing of 2D fields. This is because the `nrefocus.refocus_stack()` function uses *async* which appears to not work with e.g. `pyfftw`.

nrefocus.**refocus_stack**(*fieldstack, d, nm, res, method='helmholtz', num_cpus=2, copy=True, padding=True*)

Refocus a stack of 1D or 2D fields

Parameters

- **fieldstack** (*2d or 3d array*) – Stack of 1D or 2D background corrected electric fields (Ex/BEx). The first axis iterates through the individual fields.
- **d** (*float*) – Distance to be propagated in pixels (negative for backwards)
- **nm** (*float*) – Refractive index of medium
- **res** (*float*) – Wavelenth in pixels
- **method** (*str*) – Defines the method of propagation; one of
  - ”helmholtz” : the optical transfer function *exp(idk(M-1))*
  - ”fresnel” : paraxial approximation *exp(idk*^{2}*/k)*
- **num_cpus** (*int*) – Defines the number of CPUs to be used for refocusing.
- **copy** (*bool*) – If False, overwrites input stack.
- **padding** (*bool*) – Perform padding with linear ramp from edge to average to reduce ringing artifacts.

  New in version 0.1.4.

Returns

Return type Electric field stack at *d*.

4.3.2 Autofocusing

nrefocus.**autofocus**(*field, nm, res, ival, roi=None, metric='average gradient', padding=True, ret_d=False, ret_grad=False, num_cpus=1*)

Numerical autofocusing of a field using the Helmholtz equation.

Parameters

- **field** (*1d or 2d ndarray*) – Electric field is BG-Corrected, i.e. field = EX/BEx
- **nm** (*float*) – Refractive index of medium.
- **res** (*float*) – Size of wavelength in pixels.
- **ival** (*tuple of floats*) – Approximate interval to search for optimal focus in px.
- **roi** (*rectangular region of interest (x1, y1, x2, y2)*) – Region of interest of *field* for which the metric will be minimized. If not given, the entire *field* will be used.
- **metric** (*str*) –
  - “average gradient” : average gradient metric of amplitude

  - "rms contrast" : RMS contrast of phase data
  - "spectrum" : sum of filtered Fourier coefficients
- **padding** (*bool*) – Perform padding with linear ramp from edge to average to reduce ringing artifacts.

  Changed in version 0.1.4: improved padding value and padding location
- **ret_d** (*bool*) – Return the autofocusing distance in pixels. Defaults to False.
- **ret_grad** (*bool*) – Return the computed gradients as a list.
- **num_cpus** (*int*) – Not implemented.

Returns

- *field, [d, [grad]]*
- *The focused field and optionally, the optimal focusing distance and*
- *the computed gradients.*

Notes

This method uses `nrefocus.RefocusNumpy` for refocusing of 2D fields. This is because the `nrefocus.refocus_stack()` function uses *async* which appears to not work with e.g. `pyfftw`.

nrefocus.**autofocus_stack**(*fieldstack, nm, res, ival, roi=None, metric='average gradient', padding=True, same_dist=False, ret_ds=False, ret_grads=False, num_cpus=2, copy=True*)

Numerical autofocusing of a stack using the Helmholtz equation.

Parameters

- **fieldstack** (*2d or 3d ndarray*) – Electric field is BG-Corrected, i.e. Field = EX/BEx
- **nm** (*float*) – Refractive index of medium.
- **res** (*float*) – Size of wavelength in pixels.
- **ival** (*tuple of floats*) – Approximate interval to search for optimal focus in px.
- **roi** (*rectangular region of interest (x1, y1, x2, y2)*) – Region of interest of *field* for which the metric will be minimized. If not given, the entire *field* will be used.
- **metric** (*str*) – see *autofocus_field*.
- **padding** (*bool*) – Perform padding with linear ramp from edge to average to reduce ringing artifacts.

  Changed in version 0.1.4: improved padding value and padding location
- **same_dist** (*bool*) – Refocus entire sinogram with one distance.
- **ret_ds** (*bool*) – Return the autofocusing distances in pixels. Defaults to False. If sam_dist is True, still returns autofocusing distances of first pass. The used refocusing distance is the average.
- **ret_grads** (*bool*) – Return the computed gradients as a list.
- **num_cpus** (*int*) – Number of CPUs to use
- **copy** (*bool*) – If False, overwrites input array.

Returns

Return type The focused field (and the refocussing distance + data if d is None)

CHAPTER

FIVE

CHANGELOG

List of changes in-between nrefocus releases.

5.1 version 0.4.2

- docs: minor improvements

5.2 version 0.4.1

- fix: *autofocus* method of Refocus was not functional
- ref: use *Refocus.autofocus* for legacy autofocus method
- docs: fix rtd builds

5.3 version 0.4.0

- feat: implement nrefocus.RefocusPyFFTW for faster refocusing using pyfftw (#11)
- enh: speed-up propagation kernel computation using numexpr
- docs: cleanup

5.4 version 0.3.1

- dist: include submodules in wheel/dist

5.5 version 0.3.0

- feat: introduce nrefocus.RefocusNumpy and nrefocus.RefocusNumpy1D interface class for user-convenience and efficiency
- docs: cleanup
- ref: new submodule for metrics and metrics now accept a Refocus instance as an argument
- ref: new submodule for minimizers and minimizers now accept a Refocus instance
- ref: make legacy autofocusing code use the new Refocus class

5.6 version 0.2.1

- fix: fix several minor bugs (deprecations?) that caused the tests to faile
- ci: migrate to GitHub Actions
- setup: setup.py test is deprecated
- docs: refurbish documentation

5.7 version 0.2.0

- Drop support for Python 2 (#8)
- Code cleanup

5.8 version 0.1.8

- Include docs in sdist

5.9 version 0.1.7

- Update documentation and examples

5.10 version 0.1.6

- Move documentation from GitHub to readthedocs.io
- Add universal wheel on PyPI
- Update tests on travis with new versions of NumPy

5.11 version 0.1.5

- Code cleanup

5.12 version 0.1.4

- Padding is now available in all methods (#2)
- Added new convenient submodule *pad*
- Bugfix: autofocusing did not return the correct focusing distance. This resulted in a slight offset in the refocusing distance for the method *autofocus_stack* when *same_dist=True* was set.
- New test functions for *pad*

CHAPTER

SIX

BILBLIOGRAPHY

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[Goo05] Joseph W. Goodman. *Introduction to Fourier Optics 3rd ed*. Roberts & Company Publishers, 2005.

[ST91] Bahaa E. A. Saleh and Malvin Carl Teich. *Fundamentals of Photonics*. John Wiley & Sons, Inc., aug 1991. doi:10.1002/0471213748.

PYTHON MODULE INDEX

n

INDEX