Rayleigh-Sommerfeld Formula

Date: 2023-09-22
Tags: RC
Created by: Reinhard Caspary

Preliminary Notes

1. The equations in the following text are scale independent. All length values are given in multiples of the wavelength and thus are unit-less. The wave number therefore is $k = 2\pi$ and unit-less as well.
2. Image quantities $\hat{U}$, $\hat{x}$, $\hat{y}$, $\hat{z}$ are marked by a hat accent while the source quantities (DOE) have no hat.

Rayleigh-Sommerfeld Integral

The goal of this work was a Python implementation of the Rayleigh-Sommerfeld diffraction formula (see e.g. J. W. Goodman: *Introduction to Fourier Optics*). The image field $\hat{U}$ is given as convolution of the source field U and the impuls response h:

$$\hat{U}(\hat{x},\hat{y},\hat{z}) = \iint U(x,y,0)\, h(\hat{x}-x,\hat{y}-y,\hat{z})\, dx\, dy$$

with

$$h(x,y,z) = \hat{z}\left(\frac{1}{2\pi r} - i\right)\frac{\exp(i\,2\pi r)}{r^2}$$

and the distance

$$r = \sqrt{x^2+y^2+z^2}\,.$$

Rayleigh-Sommerfeld Sum

Any point grid may be taken for the discretization of the RS-integral. However, I choose an equidistant discretization of the coordinates with the pixel pitches p_x, p_y and $\hat{p}_x$, $\hat{p}_y$ for the source and image coordinates, respectively. The coordinates are choosen symmetric to the optical axis:

$$x_k = \left(k - \frac{N_x-1}{2}\right) p_x\,, \quad y_l = \left(l - \frac{N_y-1}{2}\right) p_y\,, \quad \hat{x}_m = \left(m - \frac{\hat{N}_x-1}{2}\right) \hat{p}_x\,, \quad \hat{y}_n = \left(n - \frac{\hat{N}_y-1}{2}\right) \hat{p}_y$$

with

$$k = 0\ldots N_x-1\,, \quad l = 0\ldots N_y-1\,, \quad m = 0\ldots \hat{N}_x-1\,, \quad n = 0\ldots \hat{N}_y-1\,.$$

Using the abbreviations

$U_{kl} = U(x_k, y_l, 0)\,, \quad \hat{U}_{mn} = \hat{U}(\hat{x}_m, \hat{y}_n, \hat{z})\,, \quad g_{klmn} = h(\hat{x}_m - x_k, \hat{y}_n - y_l, \hat{z})/\hat{z}\,, \quad \phi_{klmn} = 2\pi r = 2\pi\sqrt{(\hat{x}_m - x_k)^2 + (\hat{y}_n - y_l)^2 + \hat{z}^2}\,,$

the RS-integral in discrete form is given by the sum

$$\hat{U}_{mn} = 4\pi^2 \hat{z}\, p_x p_y \sum_{k,l} U_{kl}\, g_{klmn}$$

with

$$g_{klmn} = \left(\frac{1}{\phi_{klmn}} - i\right) \frac{\exp(i\,\phi_{klmn})}{\phi_{klmn}^2}\;.$$

Implementation

The implementation of the RS-sum in Numpy is simple and straight-forward. However, the 4D-matrix g_{klmn} quickly becomes huge. A double precision complex floating point number occupies 16 bytes of memory. The size of a complex 4D-matrix with 128 pixel resolution on all coordinates is thus 4 GB. For 256 pixel the matrix requires already 64 GB. Therefore, this not a reasonable approach.

In fact, the elements of the matrix g_{klmn} can easily be calculated on-the-fly based on just the image distance, the pixel pitches and the pixel numbers:

$$\hat{z}\,,\quad p_x\,,\quad p_y\,,\quad \hat{p}_x\,,\quad \hat{p}_y\,,\quad N_x\,,\quad N_y\,,\quad \hat{N}_x\,,\quad \hat{N}_y\,.$$

However, this calculation requires four nested loops and would be incredibly slow in native Python. I decided to implement a Numpy extension in C, therefore. The function

```
rs(z, px_src, py_src, px_img, py_img, U_src, U_img)
```

takes the pixel number of the four coordinates from the dimensions of the source and image field matrices. The algorithm is a perfect match for multithreading. Therefore, the outmost for-clause is prepended by

```
#pragma omp parallel for
```

and the code is linked to the OpenMP library.

Results

For small pixel numbers, the runtime of the Numpy/C implementation is identical to the native Numpy/Python version. Runtimes on my Dell Latitude 7330 notebook with Intel i7-1265U Gen12, 1.80 GHz with 12 virtual CPUs and 16 GB RAM with quadratic source and image fields with the same pixel number:

Pixels	Runtime w/o OpenMP	Runtime with OpenMP
64	1 second	
128	12 seconds	2 seconds
256	3,5 minutes	40 seconds

512		24 minutes
1024		5 hours

Choice of Sampling Parameters

The Rayleigh-Sommerfeld method has no restrictions despite the fact that it describes the propagation of waves as scalar instead of vector fields. It is well known that the predictions of this approach are quite accurate even for large diffraction angles. At the cost of a high computational complexity of $O(N^4)$ the method allows to choose all sampling points on the source and image field completely free. The RS sum formula delivers the precise superposition of the elementary spherical waves originating from all chosen sampling points of the source field for each desired image point.

However, in practice discrete diffraction formulas are usually used to approximate the propagation of a certain continuous source field. The discretization decisions are thus determining whether the calculated discrete result is a useful approximation to the real continuous field or not.

Although not exactly necessary, an equidistant sampling grid is usually chosen for convenience. Since the sample field is known, it is usually an easy task to chose a pixel pitch which makes sure that every significant variation of the source field is sampled. Together with the size of the source field, this determines the number or pixels required to describe the source field.

The determination of a valid set of pixel pitch $\hat{p}$ and pixel number $\hat{N}$ of the image field requires more consideration, since both are affected and somehow restricted by the choice of the source sampling parameters. Sampling of a continuous function $u(x, y)$ may be described as the multiplication of this function with a grid of delta functions $g(x, y)$:

$$u_s(x, y) = u(x, y) \cdot g(x, y)$$

The spectrum of a product function is given by the convolution of the spectra of both factor functions:

$$U_s(f_x, f_y) = U(f_x, f_y) * G(f_x, f_y)$$

Since the spectrum G of the sampling grid g is also a grid, the calculation will deliver an infinite number of identical patches in the frequency space. This leads to two important consequences. At first, there is no sense in calculating more than one (the central) patch. The pixel pitch p of the source determines the maximum useful size $\hat{s}$ of the image for each axis. The highest spatial frequency is equivalent to a phase period of $2p$ in the source field. In wavelength units the respective propagation angle is

$$\alpha = \arcsin\left(\frac{1}{2p}\right).$$

Over a propagation distance $\hat{z}$ along the optical axis, this angle restricts the size of the image to

$$\hat{s} = 2\hat{z}\,\tan\alpha = 2\hat{z}\,\tan\left[\arcsin\left(\frac{1}{2p}\right)\right].$$

In the paraxial case $\alpha \ll 1$, this simplifies to $\hat{s} = \hat{z}/p$.

The second consequence is that the magnitude of the image field must be very close to zero at the boundaries of this region, otherwise it would contain portions from the neighbouring spatial frequency patches. This provides a simple check for the validity of the chosen source pixel pitch: If the image magnitude is not almost zero near the boundary, you must reduce the source pitch p.

The size $s = Np$ of the source field determines the lowest spatial frequency and is thus a useful hint for the choice of the image pixel pitch $\hat{p}$. A reference pitch $\hat{p}_r$ is defined by the source size as being

$$\hat{p}_r = \hat{z}\,\tan\left[\arcsin\left(\frac{1}{Np}\right)\right] \approx \frac{\hat{z}}{Np}\ ,$$

where the approximation always may be used, since in any practical case the propagation angle will be very small. A certain amount of oversampling is required by chosing $\hat{p} < \hat{p}_r$ in order to resolve the diffraction fringes caused by a clipping source aperture. However, if the magnitude of the source field goes smoothly down to zero at the source field boundaries, $\hat{p} = \hat{p}_r$ is an appropriate choice.

In many cases an image field smaller than the maximum one defined above may be chosen. For this choice it is helpful to determine the power fraction covered by the chosen image dimensions. In order to compare the power of the source and image fields, they need to be normalized by the respective pixel densities. We thus define the field power to be

$$P = p_x p_y \sum_{k,l} |U_{kl}|^2\ , \quad \hat{P} = \hat{p}_x \hat{p}_y \sum_{i,j} |\hat{U}_{ij}|^2\ .$$

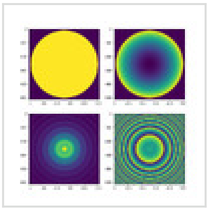
With this definition, we must always have $\hat{P} = P$, if we choose the image to be large enough. If used correctly, no power losses result from the Rayleigh-Sommerfeld formula.

For other discrete diffraction calculations, zero padding of the source field is the typical way to increase the resolution of the image. This approach is useless and expensive in terms of computation time when using the Rayleigh-Sommerfeld formula. You are always free to increase the sampling density of the image. Zero-padding of the source field does not change the image field at all, because no elementary waves originate from those points.

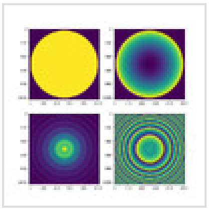
Attached files

test.py (Test script calculating the field in the focus plane of a thin lens.)
sha256: 8ff2d83a2e57d53c299834a66758572aa260cf131f87e51282435682c52769e6

lens_focus_512.png (Result of the test script with 512 pixel resolution.)
sha256: 1f03be490ae98957774fab210744bd6eb6ad1f0d6050f87f14490495d2067d2e



lens_focus_1024.png (Result of the test script with 1024 pixel resolution.)
sha256: 678ae3ab54f6234c836c08f0891f46e39918917502ec7b1df4e60ed1c6e05d05



beam.py
sha256: 0c3e62faea22109bf12312df9a4801331354e8269472ba602e434d396883032f

diffract.c
sha256: 5b96f3ad294f57e1f36dcd14cbf769b518a4716801fb28434fdb1fa5288ce815

diffract.py
sha256: a77ad018c4e073dfc11e6376285c68832d79cf8d69e318c9fdf0e532a7f8c0eb

params.py
sha256: 42a738b92dcb2446a3462753731ce44da6709e6067e4304f9db6495789dcfcca

wave.py
sha256: 61276ba37d91f8b913c4fa2828f7cc0cb5b82ab58a67e2159eaf944b09311582

init__.py
sha256: f6e01d3997ea300ce1fc1235254e6063802200449ad684ecde0cd9bf01522354

Unique eLabID: 20230922-d450808fa7cebea15c193ce37279af5e112a16bc
Link: https://elab.pxd.uni-hannover.de/experiments.php?mode=view&id=188