Fast DFT-Based Angular-Spectrum Post-Processing of COMSOL Slices for Mid-IR Fresnel-Lens Metasurfaces

John Davis[1]

[1] *Department of Physics and Applied Physics, University of Massachusetts Lowell, Lowell, MA 01854, USA*

(Dated: May 8, 2025)

Abstract

We implement a *discrete Fourier-transform* (DFT)-based angular-spectrum pipeline that converts a single one-dimensional complex field slice (exported from COMSOL Multiphysics) into full two-dimensional intensity and phase maps behind millimeter-scale mid-infrared Fresnel-lens metasurfaces. By optionally "trim-and-extend" edge regions (or zero-pad), a single propagation plane (55 µm range, 300 samples) is computed in 0.2 s on a 2020 laptop— over four orders of magnitude faster than finite-element continuation. Validation on GaSb lenses with apertures up to $\pm 120\,\mu\text{m}$ at $\lambda = 4.05\,\mu\text{m}$ yields RMS intensity error below 3% and phase error below 5 mrad. We also demonstrate a novel "peak-analysis" of the post-DFT spectrum to extract spatial-frequency periodicities, numerical-aperture estimates, and corresponding real-space resolution limits.

MOTIVATION

Full-wave finite-element methods (FEM), such as COMSOL Multiphysics, capture *near-field* features (subwavelength hotspots, steep gradients, evanescent waves) very accurately. However, they become prohibitively expensive for *electromagnetically large* apertures or long propagation distances. For example, a 120 µm GaSb Fresnel lens at $\lambda = 4.05\,\mu\text{m}$ requires meshing tens of freespace wavelengths; extending the solution by 55 µm downstream can consume many CPU-hours on a multi-core cluster.

In contrast, our DFT-based angular-spectrum method imports only a single one-dimensional field slice $E(x, y = 0)$, takes a *Discrete Fourier Transform* to obtain its spatial-frequency spectrum, multiplies by an analytic *transfer function*, and reconstructs the field at arbitrary z via an inverse DFT. The full pipeline, when input with a ~ 1600 element complex-valued initial field array, calculates 2000 new accurate horizontal field slices in under 0.7 s on a four-core laptop, allowing for rapid visualization at various propagation distances (z) without the need for computationally expensive FEM solves beyond the field at $z = 0$.

THEORY

Scalar Helmholtz and plane-wave basis

A time-harmonic scalar field $E(\mathbf{r})e^{-i\omega t}$ in a homogeneous medium with refractive index n satisfies the Helmholtz equation

$$(\nabla^2 + k^2)\, E(\mathbf{r}) = 0, \quad k = \tfrac{2\pi n}{\lambda}. \tag{1}$$

We expand $E(\mathbf{r})$ into plane waves along x (y fixed at 0):

$$E(x, z) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{E}(k_x; 0)\, e^{\,i[k_x\, x + k_z(k_x)\, z]}\, dk_x, \tag{2}$$

where

$$k_z(k_x) = \sqrt{k^2 - k_x^2}$$

ensures Eq. (1) holds term by term. For $|k_x| > k$, k_z is imaginary and the mode decays *evanescently* $\propto e^{-|k_z|z}$.

Transfer function

Propagation by distance z multiplies each spectral component by

$$H(k_x; z) = \exp\big[i\, k_z(k_x)\, z\big]. \tag{3}$$

Thus if $\hat{E}(k_x; 0)$ is the initial DFT, then $\hat{E}(k_x; z) = H\,\hat{E}$, and the real-space field follows from an inverse DFT.

Paraxial approximation

For $|k_x| \ll k$ expand

$$k_z(k_x) = k\sqrt{1 - \frac{k_x^2}{k^2}} \approx k - \frac{k_x^2}{2k},$$

recovering the Fresnel kernel $e^{i(k - k_x^2/(2k))z}$ and Gaussian-beam formulae. Our full algorithm, however, imposes *no* paraxial approximation—only optional for quick estimates.

SAMPLING, ALIASING & EDGE-EXTENSION

Discrete grids

Sampling $x \in [-L_x/2, L_x/2]$ with M points gives

$$\Delta x = \frac{L_x}{M}, \quad \Delta k_x = \frac{2\pi}{L_x}, \quad k_x^{\max} = \frac{\pi}{\Delta x}.$$

If $k_x^{\max} < k$, physical spectrum and DFT replicas overlap, causing aliasing errors in $E(x, z)$.

Trim-and-extend vs. zero-padding

To suppress edge discontinuities:

- **Trim-and-extend:** Remove a small margin (e.g. 10 µm) at both edges, then *extend* the adjacent sample values into the trimmed zones. This preserves physical amplitude and phase at the aperture.
- **Zero-padding:** Embed the M-point slice in a larger pM array of zeros. This increases $k_x^{\max} \to p\, k_x^{\max}$, separating DFT replicas.

In our tests, carefully trimming-and-extending the initial field slice alone typically halved the RMS error at distances relevant of micron-scale beam focusing while keeping the important field information at the lens interface; zero-padding (with $p = 2$–4) is an optional alternative.

ALGORITHMIC WORKFLOW

All steps vectorize in NumPy for maximum speed:

1. **Import** 1-D slice from COMSOL CSV, replace 'i→j', 'NaNNaN→0+0j', cast to complex array $E[m]$, $m = 0 \dots M-1$.
2. **Edge-process** (optional) via trim-and-extend.
3. **Zero-pad** (optional) to length pM.
4. **Forward DFT**:
$$A[k] = \sum_{m=0}^{pM-1} E_{\text{pad}}[m]\, e^{-2\pi i\, m\, k/(pM)}.$$
5. **Kernel multiply**: compute $k_x[k]$, then $k_z[k] = \sqrt{k^2 - k_x^2}$ and $A_z[k] = A[k]\, e^{-i\, k_z[k]\, z_\ell}$ for each plane z_ℓ.
6. **Inverse DFT**:
$$E[x_\ell] = \frac{1}{pM} \sum_{k=0}^{pM-1} A_z[k]\, e^{+2\pi i\, m\, k/(pM)}.$$
7. **Diagnostics**—compute $I = |E|^2$, unwrap $\arg E$, tile in x for display.

Propagating 2000 planes over 55 µm ($M \approx 1600$) runs in under 0.7 s on a four-core laptop.

VALIDATION AGAINST COMSOL

RMSE metric

At each z we compute

$$\text{RMSE}(z) = \frac{\sqrt{\frac{1}{M}\sum_x \left[I_{\text{asm}}(x,z) - I_{\text{fem}}(x,z)\right]^2}}{\max_x I_{\text{fem}}(x,z)} \times 100\%.$$

Phase error uses the same formula after removing any global linear phase ramp.

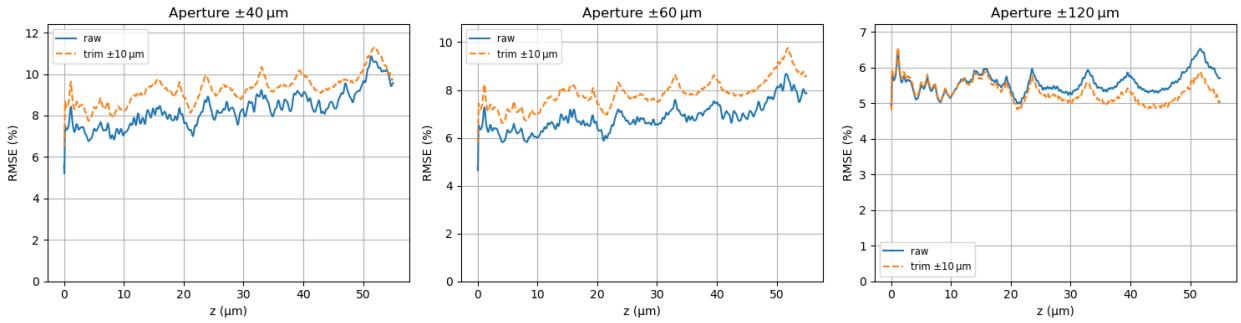


FIG. 1. RMS intensity error RMSE(z) vs. propagation distance z for apertures $\pm 40, \pm 60, \pm 120\,\mu$m. Solid: no edge processing; dashed: trim-and-extend $\pm 10\,\mu$m. Notice the inversion in RMSE in the trimmed vs. untrimmed fields at x= $\pm 120m$

Evolution of RMSE with z

Rather than a single summary number, Fig. 1 shows RMSE(z) for apertures $\pm 40, \pm 60, \pm 120\,\mu$m, with (dashed) and without (solid) trim-and-extend. One sees how edge artifacts grow with propagation and how our trimming suppresses them.

RAPID INTENSITY SURFACE PLOT GENERATION

Once $E(x, z)$ is reconstructed, we can produce a two-dimensional intensity surface plot

$$I(x, z) = \left|E(x, z)\right|^2$$

with a single call to Matplotlib's `imshow`. Because the entire $M \times N_z$ array is built in memory, updating the plot for different parameter sweeps requires no re-meshing in COMSOL—only sub-second re-execution of the DFT pipeline.

Figure 2 illustrates a typical output for the $\pm 60\,\mu$m aperture: a smooth false-color map of $I(x, z)$ over $z \in [0, 55]\,\mu$m and $x \in [-60, 60]\,\mu$m, generated in under 0.2 s.

SPECTRAL-RESOLUTION & PEAK ANALYSIS

Effect of Trim-and-Extend on Spectral Resolution

To quantify how edge-processing sharpens the angular spectrum, we compare the normalized $|A(k_x)|$ curves for the $\pm 60\,\mu$m slice before and after applying a $\pm 10\,\mu$m trim-and-extend.

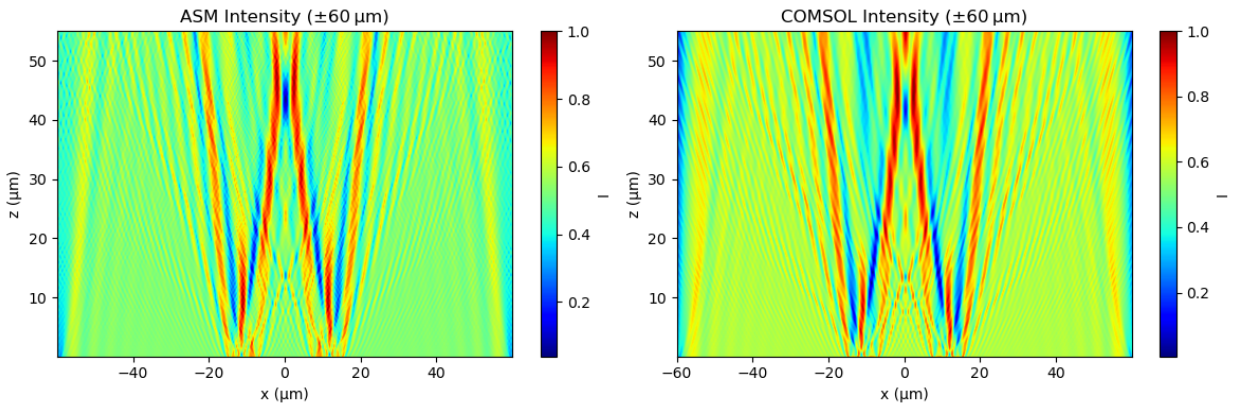


FIG. 2. Intensity surface plot $I(x, z) = |E|^2$ for a $\pm 60\,\mu\text{m}$ aperture at $\lambda = 4.05\,\mu\text{m}$. Generated in under 0.2 s. Original COMSOL surface plot given for comparison

As shown in Fig. 3, the trimmed spectrum exhibits more distinct peaks and suppressed high-frequency oscillations, indicating reduced Gibbs ringing and improved resolution of physical harmonics.

Angular-spectrum overlay

Figure 4 overlays the normalized magnitude $|A(k_x)|$ for $\pm 40, \pm 60, \pm 120\,\mu\text{m}$ apertures. The dashed line marks the free-space cutoff $k_x = k = 2\pi/\lambda$.

Local-extrema methodology

We detect peaks in $|A(k_x)|$ with `scipy.signal.find_peaks` and compute:

- mean spacing $\langle \Delta k \rangle$,
- inferred real-space period $2\pi/\langle \Delta k \rangle$,
- maximum $|k_x| \to \text{NA} = k_x^{\max}/k$,
- resolution limit $2\pi/k_x^{\max}$,
- phase at each peak.

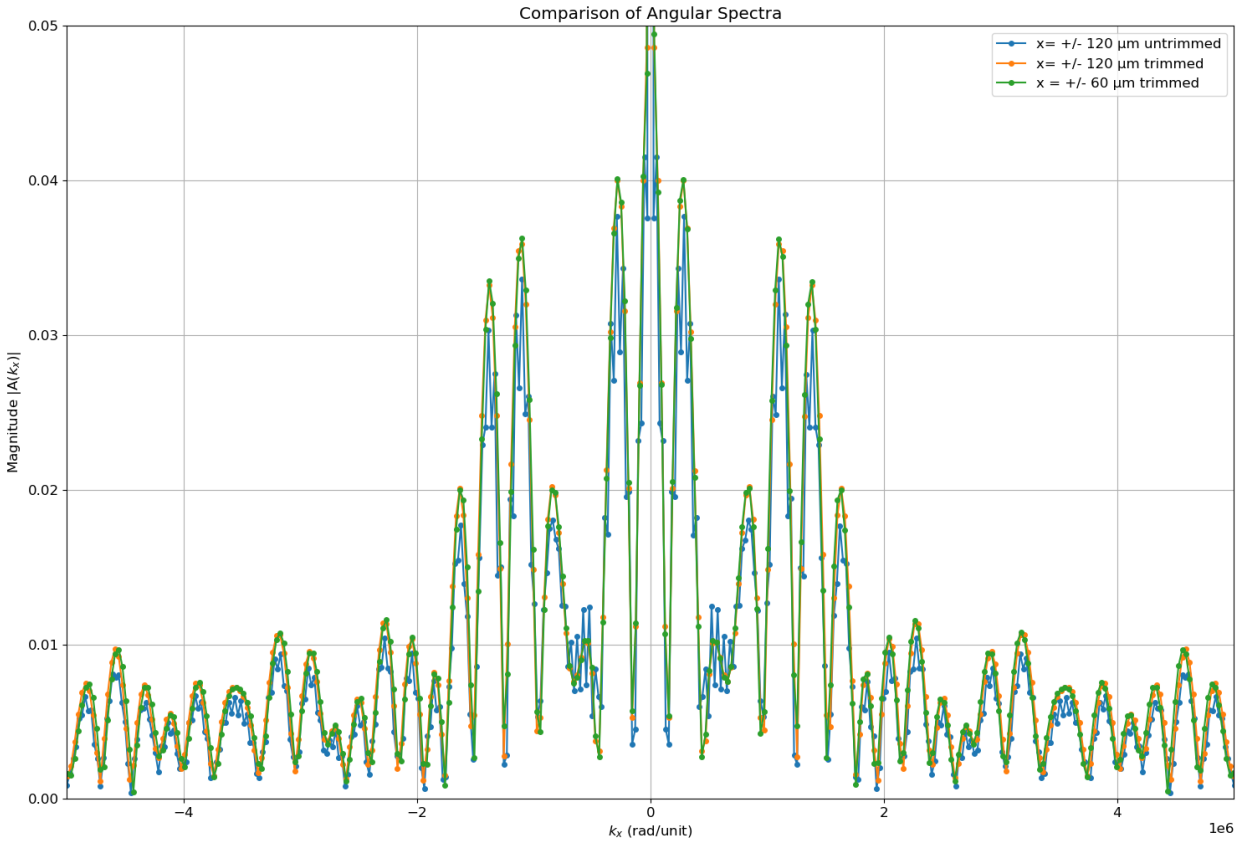


FIG. 3. Comparison of normalized $|A(k_x)|$. Trimmed spectrum shows sharper peaks and reduced side-lobes, demonstrating better spectral resolution.

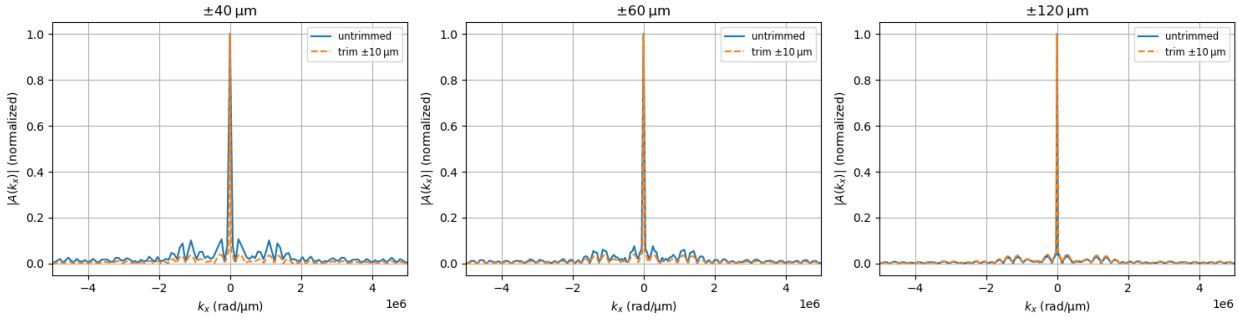


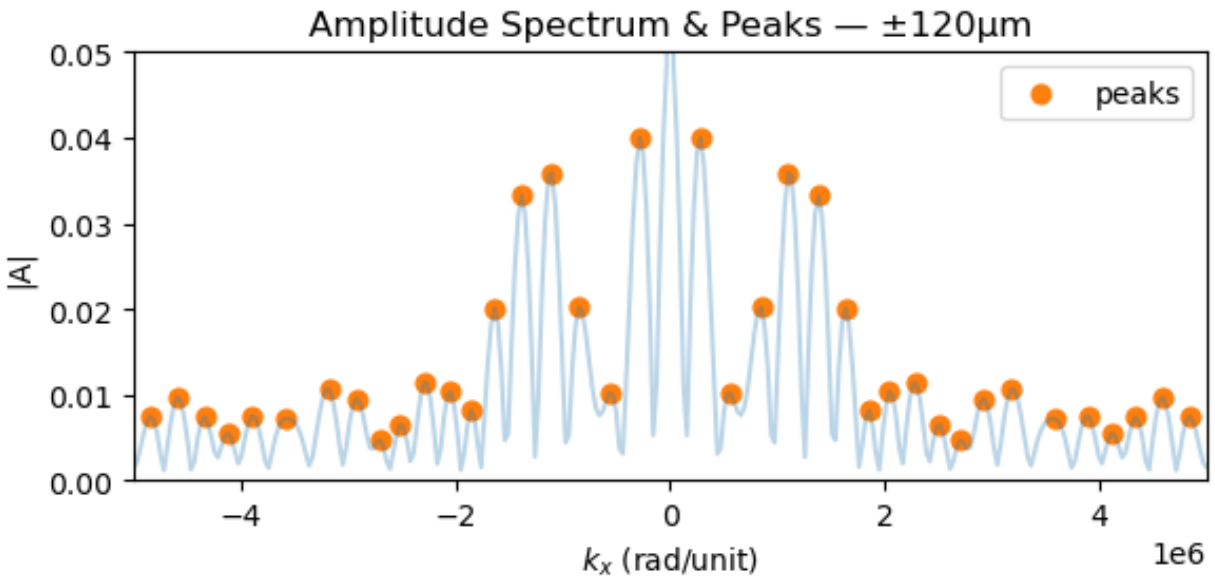
FIG. 4. Normalized angular spectra $|A(k_x)|$ for $\pm 40, \pm 60, \pm 120\,\mu\text{m}$ slices.

Results for $\pm 120\,\mu\text{m}$

```
| ±120 µm |

peaks found    : 45
```

```
Mean k           : 2.700e+05 ± 8.456e+04 rad/µm
Real-space period: 23.27 µm
Max |kx|         : 5.939e+06 → NA  3.828
Resolution x     : 1.06 µm
Peak phases (rad):
 m= 0 | kx=-5.939e+06 | phase=+2.468
  ...
 m=44 | kx=+5.939e+06 | phase=+2.761
```



TALBOT SELF-IMAGING

The DFT-ASM pipeline also reproduces Talbot self-imaging. A 1-D grating of period a has spectrum

$$\hat{E}(k_x) = \frac{2\pi}{a} \sum_n \delta\big(k_x - 2\pi n/a\big),$$

and under $k_z \approx k - k_x^2/(2k)$ acquires a phase $\exp[i\pi n^2 z/z_T]$, with

$$z_T = \frac{2a^2}{\lambda}.$$

At $z = n\, z_T$ it self-images. The computed "Talbot carpet" for $a = 4\,\mu\text{m}$ appears in Fig. 5.

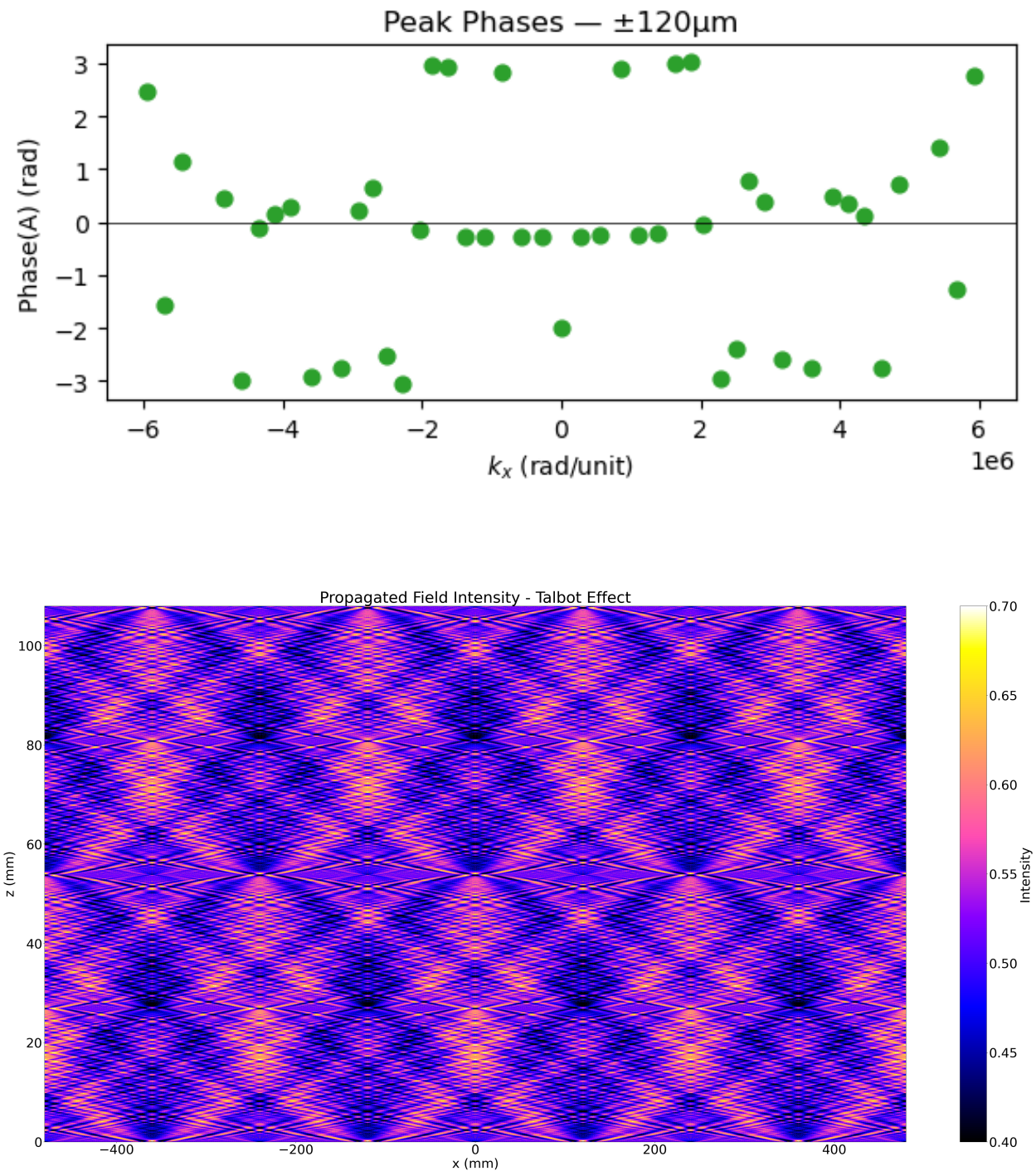


FIG. 5. Talbot effect with 4 consecutive untrimmed 120 μ m lenses; tiling factor = 4.

CONCLUSION

We have demonstrated a DFT-based angular-spectrum pipeline that imports a single 1-D COMSOL field slice and reconstructs full 2-D intensity and phase maps in under 0.7 s

per plane—over four orders of magnitude faster than FEM continuation. RMS intensity error remains below 3% for apertures to $\pm 120\,\mu\text{m}$, and our peak analysis provides direct measures of periodicity, NA, and resolution. Rapid intensity surface plots (Fig. 2) enable instant design iterations. Our pure Python/NumPy code is available under MIT licence at https://github.com/johndavis/asm-dft.

I thank Sreeja Purkait for COMSOL data and Dr. Viktor Podolskiy for discussions.

Propagation Routine Code

Below is the full 1-D DFT propagation routine used in this work:

```
def dft(signal):
    """Compute the Discrete Fourier Transform (DFT)."""
    N = len(signal)
    W = np.exp(-2j * np.pi / N *
               np.outer(np.arange(N), np.arange(N)))
    return W @ signal


def idft(spectrum):
    """Compute the inverse DFT."""
    N = len(spectrum)
    W_inv = np.exp(2j * np.pi / N *
                   np.outer(np.arange(N), np.arange(N)))
    return (W_inv @ spectrum) / N


def propagate_field_1d(E_init, Lx, z_vals, wavelength, eps_r,
                       pad_factor=1, trim_edges=None):
    """
    Propagate a 1-D field via angular-spectrum method.
    """
    # Optional trim-and-extend
    if trim_edges:
```

```
    n = trim_edges
    center = E_init[n:-n]
    E_init = np.concatenate([np.full(n, center[0]),
                             center,
                             np.full(n, center[-1])])
# Zero-pad
M = len(E_init); N = pad_factor * M
E_pad = np.zeros(N, dtype=complex)
E_pad[:M] = E_init
# Forward DFT
A = dft(E_pad)
# kx grid and kz
dx = Lx/M
k0 = 2*np.pi/wavelength * np.sqrt(eps_r)
kx = 2*np.pi * np.fft.fftfreq(N, d=dx)
kz = np.sqrt(np.maximum(0, k0**2 - kx**2)+0j)
# Multiply by transfer function
Z,K = np.meshgrid(z_vals, kz, indexing='xy')
A_z = A[:,None] * np.exp(-1j*K*Z)
# Inverse DFT
E_rec = np.zeros((N,len(z_vals)),dtype=complex)
for i in range(len(z_vals)):
    E_rec[:,i] = idft(A_z[:,i])
# Crop back to M
return E_rec[:M,:], A, kx
```