ISBN: 978-3-95450-227-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2022-THPOPT068

LINEAR CANONICAL TRANSFORM LIBRARY FOR FAST COHERENT X-RAY WAVEFRONT PROPAGATION*

Boaz Nash, Dan Tyler Abell, Paul Moeller, Ilya V. Pogorelov,
RadiaSoft LLC, Boulder, Colorado
Nicholas Burke Goldring, STATE33 Inc., Portland, Oregon, USA

Abstract

X-ray beamlines are essential components of all synchrotron light sources, transporting radiation from the stored electron beam passing from the source to the sample. The linear optics of the beamline can be captured via an ABCD matrix computed using a ray tracing code. Once the transport matrix is available, one may then include diffraction effects and arbitrary wavefront structure by using that same information in a Linear Canonical Transform (LCT) applied to the initial wavefront. We describe our implementation of a Python-based LCT library for 2D synchrotron radiation wavefronts. We have thus far implemented the separable case and are in the process of implementing algorithms for the non-separable case. Rectangular apertures are also included. We have tested our work against corresponding wavefront computations using the Synchrotron Radiation Workshop (SRW) code. LCT vs. SRW timing and benchmark comparisons are given for undulator and bending magnet beamlines. This algorithm is being included in the Sirepo implementation of the Shadow ray tracing code. Finally, we describe our plans for application to partially coherent radiation.

INTRODUCTION

Linear Canonical Transforms (LCTs) [1, 2] are a class of integral transforms with applications in optics, quantum mechanics and signal processing. In the optical context, given an *ABCD* matrix representing a linear transformation in ray optics, the corresponding LCT transforms a coherent wavefront through that same optical system.

X-ray beamlines in synchrotron light sources transport the radiation from source to sample. Modeling with full wave optics and ray tracing is common during the design phase of a beamline, but less common during everyday operation. In order to develop a fast, simplified beamline model that one may use in combination with diagnostic measurements, we present the theory and implementation of the linear canonical transform. Although, true synchrotron radiation is only partially coherent, a coherent mode decomposition [3] may be performed and each mode propagated independently. We also work within a simplified framework we refer to as a *matrix-aperture beamline* [4]. Finally, we note that although many references describe fast LCT implementations, there exist, to our knowledge, no publicly available software libraries. Thus, our interest in developing such a library and herein documenting this work.

* This work is supported by the US Department of Energy, Office of Basic Energy Sciences under Award Number DE-SC0020593 and the Office of High Energy Physics under award number DE-SC0020931.

We start by presenting the theory and implementation of the 1D LCT, followed by the 2D separable case, constructed out of the 1D case. The non-separable 2D case remains for further development.

LINEAR CANONICAL TRANSFORMS IN ONE DEGREE OF FREEDOM

The *Linear Canonical Transform*, or LCT, of a function $f(u)$ is defined by the rule [1]

$$\mathcal{L}_M[f](v) = \mathrm{e}^{-\mathrm{i}\pi/4}\sqrt{\beta}\int_{-\infty}^{\infty} f(u)\,\mathrm{e}^{-\mathrm{i}\pi(\alpha v^2 - 2\beta uv + \gamma u^2)}\,\mathrm{d}u. \tag{1}$$

The properties of a particular LCT are determined by the associated 2×2 symplectic matrix

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} \gamma/\beta & 1/\beta \\ \alpha\gamma/\beta - \beta & \alpha/\beta \end{pmatrix}, \tag{2}$$

which describes the ray optics of an arbitrary beamline.

One may represent many of the well-known integral transforms as special cases of the general LCT. These include the Fourier transform, the fractional Fourier transform, the Fresnel transform, chirp multiplication, and scaling (or magnification). Moreover, LCTs obey the very important *group property*

$$\mathcal{L}_{M_2} \circ \mathcal{L}_{M_1} = \mathcal{L}_{M_2 \cdot M_1}. \tag{3}$$

As a consequence, one may compute a given LCT as a composition of simpler LCTs: First decompose the matrix M defining a given LCT into a product of simpler symplectic matrices, each of which defines one of a few specific special cases: scaling, Fourier transform, or chirp multiplication. Then compose those transformations.

The *scaling* operation corresponds to the matrix

$$M_m = \begin{pmatrix} m & 0 \\ 0 & 1/m \end{pmatrix}, \tag{4}$$

with the corresponding LCT given by

$$\mathcal{M}_m[f](u) = \frac{1}{\sqrt{m}} f\Big(\frac{u}{m}\Big). \tag{5a}$$

For numerical work, we use the equivalent form

$$\mathcal{M}_m[f](m\cdot u) = \frac{1}{\sqrt{m}} f(u). \tag{5b}$$

If $m < 0$, then one must introduce a factor of i.

The *Fourier transform* corresponds to the matrix

$$F_{\mathrm{LC}} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \tag{6}$$

ISBN: 978-3-95450-227-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2022-THPOPT068

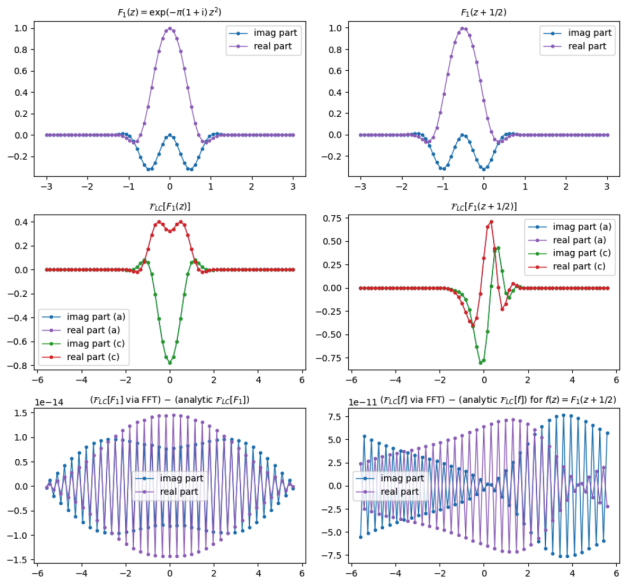

Figure 1: Test of the LC Fourier transform acting on the function $F_1(z) = \exp[-\pi(1+\mathrm{i})z^2]$ (left-hand column), and the same function shifted one-half unit to the left.

with the corresponding LCT given by

$$\mathcal{F}_{\mathrm{LC}}[f](v) = \mathrm{e}^{-\mathrm{i}\pi/4} \int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}2\pi uv} f(u)\, \mathrm{d}u. \tag{7}$$

This differs from the standard Fourier transform by the leading phase factor $\mathrm{e}^{-\mathrm{i}\pi/4}$. For numerical work, we approximate the integral so as to use the Fast Fourier Transform:

$$\begin{aligned} \int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}2\pi uv} f(u)\, \mathrm{d}u &\approx \int_{-P/2}^{P/2} \mathrm{e}^{-\mathrm{i}2\pi uv} f(u)\, \mathrm{d}u \\ &\approx \frac{P}{N} \sum_{j}^{N} \mathrm{e}^{-\mathrm{i}2\pi jk/N} f\Big(j\frac{P}{N}\Big). \end{aligned} \tag{8}$$

Of course for this to work well, the signal f must vanish—or very nearly so—outside the domain $[-P/2, P/2]$.

Finally, the operation of *chirp multiplication* corresponds to the matrix

$$Q_q = \begin{pmatrix} 1 & 0 \\ -q & 1 \end{pmatrix}, \tag{9}$$

with the corresponding LCT given by

$$\mathcal{Q}_q[f](u) = \mathrm{e}^{-\mathrm{i}\pi q u^2} f(u) \tag{10}$$

In addition to the above trio of basic operations, one also requires the operation of *resampling*. This need derives from the fact that applying chirp multiplication to a signal increases the resolution required for accurate representation of the resulting signal. See, for example [2, 5, 6].

We have implemented in Python[1] the full set of functions required for the computation of 1D LCTs. The appendix contains a very brief overview. And in Figs. 1 and 2 we show some tests of those functions.

[1] https://github.com/radiasoft/rslight

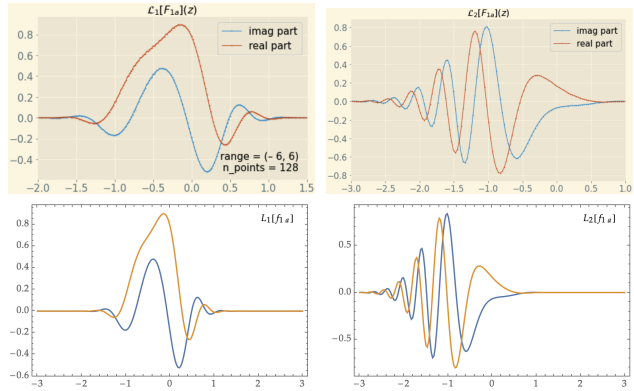

Figure 2: Tests of two different 1D LCTs acting on the shifted version of the function used in Fig. 1: $\mathcal{L}_1$, defined by $(\alpha, \beta, \gamma) = (-3, -2, -1)$ (left-hand column); and $\mathcal{L}_2$, defined by $(\alpha, \beta, \gamma) = (-4/5, 1, 2)$ (right-hand column). The upper row displays results computed using our Python library implementation `lctLIB`. For comparison, the lower row displays results computed using brute-force numerical integration.

LINEAR CANONICAL TRANSFORMS IN TWO DEGREES OF FREEDOM

The Linear Canonical Transform, or LCT, in two degrees of freedom is governed by a 4×4 real symplectic matrix

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}.$$

in this context, the symplecticity of M means that the 2×2 submatrices A, B, C, and D must obey the relations [2]

$$A^{\mathrm{tr}}C = C^{\mathrm{tr}}A, \quad B^{\mathrm{tr}}D = D^{\mathrm{tr}}B, \quad A^{\mathrm{tr}}D - C^{\mathrm{tr}}B = I \tag{11a}$$

$$AB^{\mathrm{tr}} = BA^{\mathrm{tr}}, \quad CD^{\mathrm{tr}} = DC^{\mathrm{tr}}, \quad AD^{\mathrm{tr}} - BC^{\mathrm{tr}} = I, \tag{11b}$$

where the superscript 'tr' denotes matrix transposition, and I denotes the 2×2 identity matrix. The fact that the submatrices obey these particular relations tells us that the matrix M acts on phase-space variables in the order (q_1, q_2, p_1, p_2) (see, for example, [7, §3.2]). This means that if one extracts the ray optical matrix M with the phase-space variables given in some other order, *e.g.* (q_1, p_1, q_2, p_2), then one must—for the purposes of the computation described here—make sure to appropriately permute the entries of M.

For a function $f(\vec{u})$, the LCT governed by matrix M is defined by the rule [1, 2]

$$\mathcal{L}_M[f](\vec{u}) = \frac{1}{\sqrt{\det \mathrm{i}B}} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\vec{v})\, \exp[\mathrm{i}\pi p(\vec{u}, \vec{v})], \tag{12a}$$

where

$$p(\vec{u}, \vec{v}) = \vec{u}^{\,\mathrm{tr}} D B^{-1} \vec{u} - 2\vec{v}^{\,\mathrm{tr}} B^{-1} \vec{u} + \vec{v}^{\,\mathrm{tr}} B^{-1} A \vec{v}. \tag{12b}$$

As in the 1D case, the phase of the leading factor in (12a) "requires some care" [1, Ch. 1]. In addition, the 2D LCT also

ISBN: 978-3-95450-227-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2022-THPOPT068

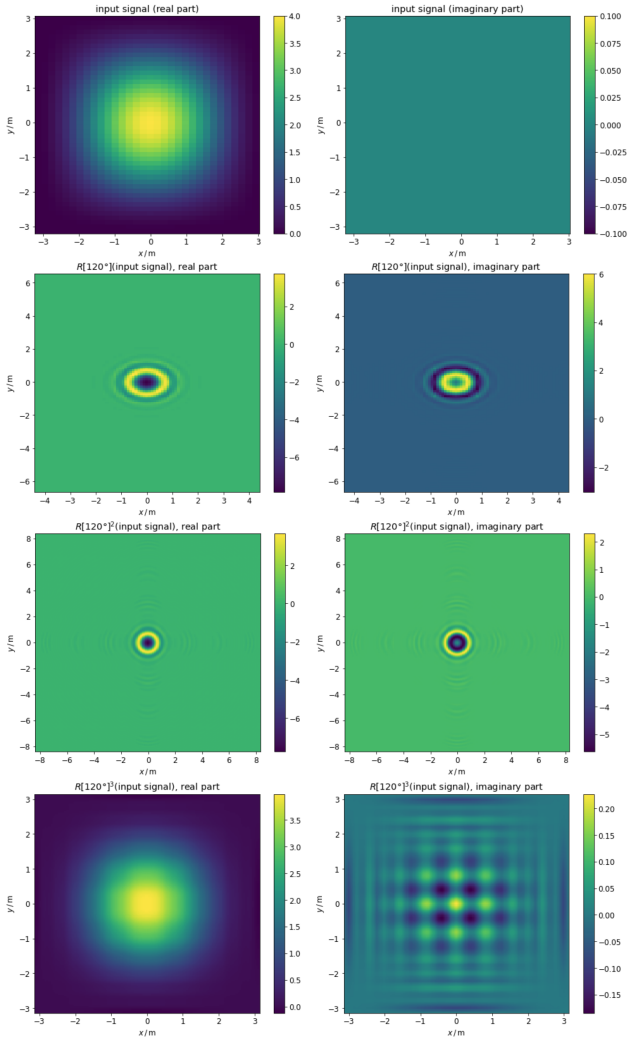

Figure 3: Test of separable 2D LCT. Top row: real and imaginary parts of initial 2D signal. Subsequent rows: respectively one, two, and three applications of the LCT corresponding to $M_x = M_y = R(120°)$, a 120° degree rotation. The last row should therefore appear identical to the first, and differences reflect the accuracy in this example.

obeys the group property (3), with the same consequences for ease of computation.

For the case of an optical system with no coupling between the planes, the corresponding LCT is called *separable*, and can be computed as a concatenation 1D LCTs applied in the separate planes. Figure 3 shows the results of a simple test.

FUTURE WORK

In future work on `libLCT`, we shall (i) implement the non-separable 2D LCT; (ii) implement a unit test suite; and (iii) execute and publish a set of computational benchmarks.

APPENDIX: IMPLEMENTATION OF AN OPEN-SOURCE LCT LIBRARY

Herewith a précis of our Python library `libLCT`.

```
# NB: 'in_signal' must have the form
#     ( dX, signal_array )
# where signal_array denotes a 1d numpy array
# representing a signal with sample spacing dX.

# === Manifest ===
# - convert between parameter representations
convert_params_3to4(alpha, beta, gamma)
convert_params_4to3(M_lct)
# - component functions
lct_abscissae(nn, du, ishift = False)
resample_signal(k, in_signal)
scale_signal(m, in_signal)
lct_fourier(in_signal)
chirp_multiply(q, in_signal)
# - decompose LCT matrix
lct_decomposition(M_lct)
# - apply sequence of transformations
apply_lct_1d(M_lct, in_signal)

# decompose LCT matrix
def lct_decomposition(M_lct):
    alpha, beta, gamma = convert_params_4to3(M_lct)
    ag = abs(gamma)
    if ag <= 1:
        k = 1 + ag + abs(alpha) / beta ** 2 \
                   * (1 + ag) ** 2
        seq = [[ 'SCL',   beta              ],
               ['RSMP',     2.              ],
               [  'CM', - gamma / beta ** 2 ],
               ['LCFT',     0               ],
               ['RSMP',    k/2              ],
               [  'CM', - alpha             ]]
    else:
        k = 1 + 1/ag + abs(alpha - beta ** 2 / gamma) \
                     / beta ** 2 * (1 + ag) ** 2
        seq = [[ 'SCL', - gamma / beta              ],
               ['LCFT',     0                       ],
               ['RSMP',     2.                      ],
               [  'CM',   gamma / beta ** 2         ],
               ['LCFT',     0                       ],
               ['RSMP',    k/2                      ],
               [  'CM', - alpha + beta ** 2 / gamma ]]
    return seq

# apply sequence of transformations
def apply_lct(M_lct, in_signal):
    seq = lct_decomposition(M_lct)
    signal0 = in_signal
    for lct in seq:
        if   lct[0] == 'CM':
            signal1 = chirp_multiply(lct[1], signal0)
            signal0 = signal1
        elif lct[0] == 'LCFT':
            signal1 = lct_fourier(signal0)
            signal0 = signal1
        elif lct[0] == 'SCL':
            signal1 = scale_signal(lct[1], signal0)
            signal0 = signal1
        elif lct[0] == 'RSMP':
            signal1 = resample_signal(lct[1], signal0)
            signal0 = signal1
        else:
            assert False, 'LCT code ' + lct[0] + \
                          ' not recognized! Exiting now.'
            return -1
    return signal1
```

REFERENCES

[1] J. J. Healy, M. A. Kutay, H. M. Ozaktas, and J. T. Sheridan, Eds., *Linear Canonical Transforms: Theory and Applications*,

ISBN: 978-3-95450-227-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2022-THPOPT068

ser. Springer Series in Optical Sciences, New York: Springer, 2016, vol. 198.

[2] A. Koç, "Fast algorithms for digital computationof linear canonical transforms," Ph.D. dissertation, Stanford University, Stanford, CA, USA, Mar. 2011.

[3] R. Li and O. Chubar, "CPU and memory efficient coherent mode decomposition for the partially coherent x-ray simulations," *Advances in Computational Methods for X-Ray Optics V*, O. Chubar and K. Sawhney, Eds., vol. 11493, International Society for Optics and Photonics, SPIE, 2020, pp. 78–87. doi:10.1117/12.2569058

[4] B. Nash, N. Goldring, J. Edelen, C. Federer, P. Moeller, and S. Webb, "Reduced model representation of x-ray transport suitable for beamline control," *Advances in Computational Methods for X-Ray Optics V*, International Society for Optics and Photonics, Aug. 2020, vol. 11493, p. 114930C. doi:10.1117/12.2568187

[5] A. Stern, "Sampling of linear canonical transformed signals," *Signal Process.*, vol. 86, no. 7, pp. 1421–1425, Jul. 2006. doi:10.1016/j.sigpro.2005.07.031

[6] F. S. Oktem and H. M. Ozaktas, "Exact relation between continuous and discrete linear canonical transforms," *IEEE Signal Process. Lett.*, vol. 16, no. 8, pp. 727–730, May 2009. doi:10.1109/LSP.2009.2023940

[7] A. J. Dragt, *Lie Methods for Nonlinear Dynamics with Applications to Accelerator Physics*, Nov. 2020, latest version available at http://www.physics.umd.edu/dsat/.