Highlights

Systematic Parameter Selection for FFT-based Numerical Inverse Laplace Transform: CFL-informed Tuning Rules and Quality Diagnostics

Gorgi Pavlov

- Derives CFL-like feasibility condition for FFT-based NILT parameter selection
- Proposes adaptive acceptance criteria combining $\varepsilon_{\mathrm{Im}}$ and N-doubling convergence tests
- Reduces parameter sensitivity from trial-and-error to deterministic selection
- Extends reliable operating range by 2–3 orders of magnitude in stiffness
- For uniform-grid inversion: 3–4$\times$ better accuracy than de Hoog, 4–5$\times$ faster ($N = 2048$)
- Provides NumPy, JAX, and PyTorch implementations; GPU achieves 9–15$\times$ speedup at $N = 16384$

Systematic Parameter Selection for FFT-based Numerical Inverse Laplace Transform: CFL-informed Tuning Rules and Quality Diagnostics

Gorgi Pavlov*

[a]*Department of Chemical and Biomolecular Engineering, Lehigh University, 111 Research Drive, Bethlehem, 18015, PA, USA*
[b]*Johnson & Johnson, 200 Great Valley Parkway, Malvern, 19355, PA, USA*

Abstract

The FFT-based numerical inverse Laplace transform (NILT) introduced by Hsu and Dranoff (1987) offers $O(N \log N)$ efficiency for recovering time-domain solutions from Laplace-domain transfer functions. However, practical application remains limited by parameter sensitivity, requiring expert trial-and-error tuning of the Bromwich shift, sampling frequency, and integration period. This work develops a systematic parameter selection framework based on three necessary conditions analogous to the Courant–Friedrichs–Lewy (CFL) stability constraint in numerical PDEs: (i) dynamic-range feasibility, (ii) spectral placement correctness, and (iii) aliasing suppression. We derive an explicit feasibility condition relating spectral abscissa, time horizon, and floating-point precision, and propose the imaginary leakage metric $\varepsilon_{\mathrm{Im}}$ combined with N-doubling convergence tests for *a posteriori* quality assessment. Computational experiments on five distributed-parameter transfer functions—including semi-infinite diffusion, packed-bed dispersion, and first-order-plus-dead-time systems—demonstrate that CFL-informed tuning automatically achieves near-optimal accuracy (as verified by N-doubling convergence) without user intervention, extending the reliable operating range by 2–3 orders of magnitude in system stiffness. The framework is validated by cross-comparison with Method of Lines time-domain solutions and the de Hoog algorithm. For uniform-grid inversion on these benchmarks, CFL-

*Corresponding author
Email address: gop214@lehigh.edu (Gorgi Pavlov)

informed FFT-NILT with $N = 2048$ achieves 3–4× better accuracy than de Hoog ($M = 20$) while remaining 4–5× faster in our Python implementations, substantially mitigating the traditional accuracy–efficiency trade-off for this problem class. The proposed methodology transforms NILT from a specialist technique into a routine computational tool for frequency-domain analysis of distributed-parameter systems.

Keywords: Numerical inverse Laplace transform, Fast Fourier transform, Parameter selection, CFL condition, Transfer functions, Process dynamics

---

1. Introduction

The Laplace transform is fundamental to chemical engineering analysis, enabling frequency-domain characterization of distributed-parameter systems including heat exchangers [10], chromatographic separators [13], packed-bed reactors [11], and process control systems [18]. For linear time-invariant systems, the Laplace transform converts partial differential equations (PDEs) to ordinary differential equations or algebraic systems, facilitating analytical derivation of transfer functions, stability criteria, and frequency response characteristics.

While analytical inverse transforms exist for elementary functions, realistic process models often yield Laplace-domain solutions without closed-form inverses. Numerical inverse Laplace transform (NILT) methods address this gap, with applications spanning chromatographic column dynamics [12], residence time distributions [15], and transient heat conduction [7].

1.1. FFT-based NILT methods

Among NILT algorithms, FFT-based approaches offer computational efficiency through the relationship between Bromwich contour integration and Fourier series expansion. Dubner and Abate [9] established the foundation by discretizing the inversion integral, with the Cooley–Tukey FFT algorithm [3] enabling $O(N \log N)$ computation:

$$f(t) = \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} F(s) e^{st}\, ds \tag{1}$$

via trapezoidal quadrature, yielding a sum computable by FFT in $O(N \log N)$ operations. Hsu and Dranoff [13] refined this approach using complete Fourier

series, demonstrating accuracy on chromatographic transfer functions. Subsequent improvements addressed convergence acceleration [5], error estimation [8], and numerical stability [2]. Alternative NILT algorithms include Talbot's deformed contour method [19], the Weeks method based on Laguerre functions [21], and the Gaver–Stehfest algorithm using real arithmetic [17]. Comprehensive reviews comparing these approaches are provided by Davies and Martin [7] and more recently by Kuhlman [14].

1.2. The parameter selection problem

Despite algorithmic maturity, practical FFT-NILT application remains challenging due to parameter interdependence. Three parameters require specification:

1. **Bromwich shift (a):** The integration contour $\mathrm{Re}(s) = a$ must strictly exceed the abscissa of convergence α_c for the integral to converge. However, larger a amplifies roundoff through the $\exp(at)$ factor.
2. **Half-period (T):** Determines aliasing error from implicit $2T$-periodicity. Insufficient T causes wraparound contamination when $f(t)$ has not decayed adequately.
3. **Sample count (N):** Controls both frequency resolution ($\Delta\omega = \pi/T$) and Nyquist frequency ($\omega_{\max} = N\pi/2T$). Insufficient N causes truncation or bandwidth limitations.

Prior literature provides general guidance—a should "comfortably exceed" α_c [7], T should be "several time constants" [2]—but quantitative selection rules remain elusive. Weideman [23] developed systematic parameter optimization for the Weeks method, representing the most rigorous prior work on NILT parameter selection; however, no comparable framework exists for FFT-based methods. We focus on FFT-NILT rather than Weeks because FFT methods produce dense time grids in a single pass—ideal for plotting breakthrough curves or generating training data—whereas Weeks evaluates one time point per call, making it less efficient for the visualization and batch-evaluation tasks common in chemical engineering workflows. Practitioners report that parameter tuning often dominates total solution time, limiting NILT adoption beyond specialist applications [20].

1.3. Research objectives

The present work addresses the parameter selection gap through three contributions:

1. **Feasibility analysis:** We derive necessary conditions for NILT success, showing that parameter constraints are analogous to the CFL condition in numerical PDEs. An explicit feasibility criterion identifies when NILT is possible within floating-point precision.
2. **Autotuning framework:** We develop a deterministic parameter selection algorithm satisfying all constraints simultaneously, eliminating trial-and-error tuning.
3. **Quality diagnostics:** We propose the imaginary leakage metric ε_{Im} for *a posteriori* quality assessment, enabling confidence quantification and guided refinement.

We evaluate the framework on distributed-parameter transfer functions representative of chemical engineering applications, comparing with analytical solutions, the de Hoog algorithm, and Method of Lines (MOL) time-domain computations.

1.4. What is new versus prior art

Prior works provide qualitative tuning guidance; this work contributes:

- An explicit feasibility inequality tied to floating-point dynamic range (Eq. 21)
- A deterministic selection algorithm with explicit parameter outputs (Algorithm 1)
- A practical *a posteriori* diagnostic (ε_{Im}) with empirical calibration
- Systematic validation against both analytical solutions and alternative NILT methods

2. Theory

2.1. Bromwich inversion formula and abscissa of convergence

For a function $F(s)$ analytic in the half-plane $\text{Re}(s) > \alpha_c$, the inverse Laplace transform is given by the Bromwich integral:

$$f(t) = \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} F(s)e^{st}\, ds, \quad a > \alpha_c \tag{2}$$

The **abscissa of convergence** α_c is the infimum of real parts for which the integral converges. For rational transfer functions, α_c equals the spectral abscissa (maximum real part of poles). For functions with branch points (e.g., $\sqrt{s}$), the branch cut location determines α_c.

Critical distinction: The spectral abscissa $\alpha = \max \mathrm{Re}(\text{poles})$ may equal zero even when poles or branch points exist at $s = 0$. In such cases, the Bromwich contour must still satisfy $a > 0$ strictly; setting $a = 0$ places the contour through the singularity, invalidating the integral.

For real-valued $f(t)$, Eq. (2) reduces to:

$$f(t) = \frac{e^{at}}{\pi} \mathrm{Re}\left[\int_0^\infty F(a + i\omega) e^{i\omega t}\, d\omega\right] \tag{3}$$

2.2. FFT discretization

Discretizing Eq. (3) with uniform spacing $\Delta\omega = \pi/T$ on the interval $[0, \omega_{\max}]$ and applying trapezoidal quadrature yields:

$$f(t_j) \approx \frac{e^{at_j}}{T} \mathrm{Re}\left[\sum_{k=0}^{N-1} w_k F(a + ik\Delta\omega) e^{ik\Delta\omega t_j}\right] \tag{4}$$

where $t_j = j\Delta t$ with $\Delta t = 2T/N$, and w_k are quadrature weights ($w_0 = 1/2$, $w_k = 1$ for $k > 0$). The sum constitutes a discrete Fourier transform, enabling $O(N \log N)$ evaluation via FFT.

2.3. Error sources

Three error components limit FFT-NILT accuracy:

Truncation error arises from frequency bandwidth limitation:

$$\varepsilon_{\text{trunc}} \sim \left|\int_{\omega_{\max}}^\infty F(a + i\omega) e^{i\omega t}\, d\omega\right| \tag{5}$$

For transfer functions with asymptotic decay $|F(a + i\omega)| \sim \omega^{-n}$ as $\omega \to \infty$, truncation error scales as $\omega_{\max}^{-(n-1)}$.

Aliasing error results from implicit $2T$-periodicity in the discrete Fourier representation [1]. The FFT computes the periodic extension of the exponentially-weighted signal $g(t) = e^{-at} f(t)$. The computed approximation $\tilde{f}(t)$ satisfies:

$$\tilde{f}(t) = e^{at} \sum_{m=-\infty}^{\infty} g(t + 2mT) = f(t) + e^{at} \sum_{m \neq 0} e^{-a(t+2mT)} f(t + 2mT) \tag{6}$$

The leading alias term ($m = 1$) contributes:

$$\varepsilon_{\text{alias}}(t) \approx e^{-2aT} f(t + 2T) \tag{7}$$

For functions satisfying the **tail envelope condition** $|f(t)| \leq Ce^{\alpha_c t}$ for all $t \geq 0$, this becomes:

$$|\varepsilon_{\text{alias}}(t)| \leq C \cdot e^{-(a-\alpha_c)(2T-t)} \tag{8}$$

Roundoff error is amplified by the exponential factor:

$$\varepsilon_{\text{round}} \sim \varepsilon_{\text{mach}} \cdot \exp(at_{\max}) \tag{9}$$

where $\varepsilon_{\text{mach}} \approx 2.2 \times 10^{-16}$ for IEEE double precision and $t_{\max} = 2T$.

2.4. Parameter trade-offs

The error sources create coupled constraints. Reducing aliasing requires larger T or larger $(a - \alpha_c)$, but increasing a amplifies roundoff. Reducing truncation requires larger $\omega_{\max}$ (hence larger N for fixed T), increasing computational cost. These trade-offs mirror the Courant–Friedrichs–Lewy (CFL) condition in numerical PDEs [4], which constrains time step, grid spacing, and wave speed for stable explicit integration. While the analogy is conceptual rather than rigorous (NILT parameters affect accuracy rather than stability), the structural parallel—coupled parameters, constraint violations causing completely wrong rather than merely degraded results—motivates our terminology.

2.5. Formal definitions and bounds

We now formalize the key concepts used in the parameter selection framework.

Definition 1 (Tail envelope condition)**.** *A function $f : [0, \infty) \to \mathbb{R}$ satisfies the **tail envelope condition** with constants (C, α_c) if:*

$$|f(t)| \leq C \cdot e^{\alpha_c t} \quad \textit{for all } t \geq 0 \tag{10}$$

where the envelope constant is defined as:

$$C := \sup_{t \geq 0} |e^{-\alpha_c t} f(t)| < \infty \tag{11}$$

For bounded functions with $\alpha_c = 0$ (e.g., step responses approaching finite asymptotes), this reduces to $C = \sup |f(t)|$.

Lemma 1 (Periodic extension identity)**.** *Under trapezoidal sampling with frequency spacing $\Delta\omega = \pi/T$ and time step $\Delta t = 2T/N$, the discrete sum in Eq. (4) corresponds to the $2T$-periodic extension of $g(t) = e^{-at}f(t)$. Specifically, the computed approximation satisfies:*

$$\tilde{f}(t) = e^{at} \sum_{m\in\mathbb{Z}} g(t + 2mT) \tag{12}$$

Proof sketch. The DFT sum $\sum_{k=0}^{N-1} G_k e^{i2\pi kj/N}$ evaluates the N-periodic extension of the sampled sequence. Since $e^{ik\Delta\omega t_j} = e^{i2\pi kj/N}$ by construction (with $\Delta\omega \cdot \Delta t = 2\pi/N$), the frequency-domain sum aliases to the $2T$-periodic time-domain extension. □

Lemma 2 (Alias bound over evaluation interval)**.** *Let f satisfy the tail envelope condition with constants (C, α_c), and let $a > \alpha_c$. Then for any $t \in [0, t_{\text{end}}]$ with $T \geq t_{\text{end}}$, the leading alias term satisfies:*

$$|\varepsilon_{\text{alias},1}(t)| \leq C \cdot e^{-(a-\alpha_c)(2T-t)} \tag{13}$$

The worst case over the evaluation interval $[0, t_{\text{end}}]$ is:

$$\max_{t\in[0,t_{\text{end}}]} |\varepsilon_{\text{alias},1}(t)| \leq C \cdot e^{-(a-\alpha_c)(2T-t_{\text{end}})} \tag{14}$$

Proof. From Eq. (7), $|\varepsilon_{\text{alias},1}(t)| = e^{-2aT}|f(t + 2T)|$. Applying the tail envelope: $|f(t + 2T)| \leq Ce^{\alpha_c(t+2T)}$. Thus $|\varepsilon_{\text{alias},1}(t)| \leq Ce^{-2aT}e^{\alpha_c(t+2T)} = Ce^{-(a-\alpha_c)(2T-t)}$. The maximum over $[0, t_{\text{end}}]$ occurs at $t = t_{\text{end}}$ since the bound is increasing in t. □

Remark 1 ($\alpha_c = 0$ unavoidable amplification)**.** *For $\alpha_c = 0$, the alias suppression requirement $a \geq \ln(C/\varepsilon_{\text{tail}})/(2T)$ implies:*

$$a \cdot t_{\max} = a \cdot 2T \geq \ln(C/\varepsilon_{\text{tail}}) \tag{15}$$

Hence the exponential amplification factor $\exp(a \cdot t_{\max})$ is lower-bounded by $C/\varepsilon_{\text{tail}}$, independent of T. With $C = 1$ and $\varepsilon_{\text{tail}} = 10^{-6}$, this gives $\exp(a \cdot t_{\max}) \geq 10^6$ unavoidably.

3. Methodology

3.1. Three-constraint framework

We formalize parameter selection through three necessary conditions:

Constraint 1 (Dynamic range): The exponential amplification factor must not cause floating-point overflow:

$$a \cdot t_{\max} \leq L - \delta_s \tag{16}$$

where $L = \ln(\texttt{DBL\_MAX}) \approx 709.8$ for IEEE double precision and $\delta_s \approx 10$ provides safety margin for intermediate computations. This yields an upper bound:

$$a_{\max} = (L - \delta_s)/t_{\max} \tag{17}$$

Constraint 2 (Spectral placement): The Bromwich contour must lie strictly right of all singularities:

$$a > \alpha_c + \delta_{\min} \tag{18}$$

where α_c is the abscissa of convergence and $\delta_{\min} > 0$ provides a safety margin. For transfer functions with singularities at $s = 0$ (poles or branch points), $\delta_{\min}$ must be positive regardless of whether $\alpha_c = 0$.

Constraint 3 (Aliasing suppression): The wrapped signal tail must be negligible:

$$C \cdot e^{-(a-\alpha_c)(2T-t_{\text{end}})} \leq \varepsilon_{\text{tail}} \tag{19}$$

With $\kappa = T/t_{\text{end}} \geq 1$, solving for the minimum required a:

$$a \geq \alpha_c + \frac{\ln(C/\varepsilon_{\text{tail}})}{(2\kappa - 1)t_{\text{end}}} \tag{20}$$

3.2. CFL-like feasibility condition

The three constraints can be simultaneously satisfied only if $a_{\min} < a_{\max}$, where $a_{\min}$ combines Constraints 2 and 3. Setting $\kappa = 1$ (i.e., $T = t_{\text{end}}$, the minimum period) yields the most restrictive aliasing requirement.

Theorem 1 (CFL-like feasibility condition)**.** *For given $(\alpha_c, t_{\text{end}}, C, \varepsilon_{\text{tail}})$ with $\kappa = 1$ and $t_{\max} = 2t_{\text{end}}$, a feasible Bromwich parameter a exists in IEEE double precision if and only if:*

$$\alpha_c \cdot t_{\max} + \ln(C/\varepsilon_{\text{tail}}) < L - \delta_s \tag{21}$$

Proof. Feasibility requires $a_{\min} < a_{\max}$. With $\kappa = 1$, Eq. (20) gives:

$$a_{\min} = \alpha_c + \frac{\ln(C/\varepsilon_{\text{tail}})}{t_{\text{end}}} \tag{22}$$

The constraint $a_{\min} < a_{\max} = (L - \delta_s)/t_{\max}$ becomes:

$$\alpha_c + \frac{\ln(C/\varepsilon_{\text{tail}})}{t_{\text{end}}} < \frac{L - \delta_s}{2t_{\text{end}}} \tag{23}$$

Multiplying by $t_{\max} = 2t_{\text{end}}$ yields Eq. (21). □

Interpretation. Eq. (21) partitions the parameter space:

- **Feasible region:** LHS $< L - \delta_s \approx 700$. A valid $a \in (a_{\min}, a_{\max})$ exists.
- **Infeasible region:** LHS $\geq L - \delta_s$. No valid a exists; either overflow occurs or aliasing contaminates results.

For $\alpha_c = 0$ with $C = 1$ and $\varepsilon_{\text{tail}} = 10^{-6}$, the condition reduces to $t_{\max} < (L - \delta_s - 13.8)/\alpha_c$, which is always satisfied when $\alpha_c = 0$ (LHS $= 13.8 < 700$).

For $\alpha_c > 0$ (unstable systems), maximum feasible $t_{\max}$ decreases linearly with α_c. With $\alpha_c = 1$, the bound is $t_{\max} < (700 - 13.8)/1 \approx 686$ seconds—ample for most applications.

Figure 1 visualizes the feasible region in $(t_{\max}, a)$ space for different values of α_c.

3.3. Frequency coverage and adaptive refinement

The sample count N determines frequency bandwidth and resolution. Two requirements apply:

Nyquist coverage: For transfer functions with characteristic frequency ρ, adequate bandwidth requires:

$$\omega_{\max} = \frac{N\pi}{2T} \geq \gamma\rho \tag{24}$$

where $\gamma \geq 2$ is an oversampling factor. Solving for N:

$$N \geq \frac{2\gamma\rho T}{\pi} \tag{25}$$

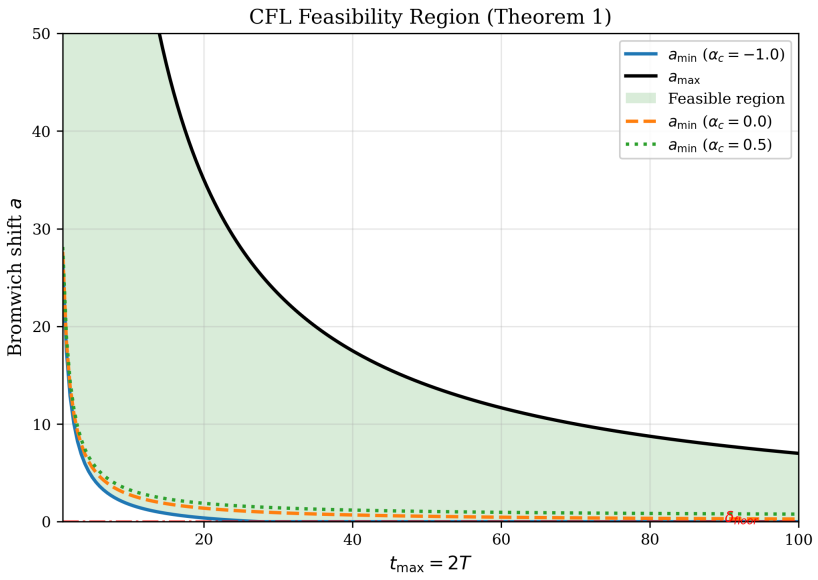

Figure 1: CFL feasibility region in $(t_{\max}, a)$ parameter space. The shaded area shows valid parameter combinations for $\alpha_c = -1$. Upper bound $a_{\max}$ (black line) prevents overflow; lower bound $a_{\min}$ (colored lines) ensures aliasing suppression. For larger α_c, the feasible region shrinks.

Algorithm 1 CFL-informed NILT parameter selection

Require: α_c (abscissa), t_{end} (max time), ρ (spectral radius, optional)
Require: $\varepsilon_{\text{tail}} = 10^{-6}$, $\delta_{\min} = 10^{-3}$, $\delta_{\text{floor}} = 10^{-3}$, $\delta_s = 10$, $\kappa = 1$
Ensure: a, T, N satisfying all constraints, or "infeasible"

$T \leftarrow \kappa \cdot t_{\text{end}}, \quad t_{\max} \leftarrow 2T$
$a_{\max} \leftarrow (L - \delta_s)/t_{\max}$ ▷ Dynamic range upper bound
if $\alpha_c \cdot t_{\max} + \ln(1/\varepsilon_{\text{tail}}) \geq L - \delta_s$ **then**
    return "infeasible: CFL condition violated"
end if
$a_{\text{alias}} \leftarrow \alpha_c + \ln(1/\varepsilon_{\text{tail}})/t_{\text{end}}$ ▷ Aliasing lower bound
$a_{\text{spectral}} \leftarrow \alpha_c + \delta_{\min}$ ▷ Spectral placement
$a \leftarrow \max(a_{\text{alias}}, a_{\text{spectral}}, \delta_{\text{floor}})$ ▷ Combine constraints + floor
if ρ provided **then**
    $N \leftarrow 2^{\lceil \log_2(2\gamma\rho T/\pi) \rceil}$ ▷ Round to power of 2
else
    $N \leftarrow 256$ ▷ Starting value for adaptive refinement
end if
return (a, T, N)

Resolution: For capturing oscillatory components with period $\tau_{\min}$, adequate resolution requires:

$$\Delta\omega = \frac{\pi}{T} \leq \frac{2\pi}{\gamma\tau_{\min}} \tag{26}$$

which is satisfied when $T \geq \gamma\tau_{\min}/2$.

Adaptive N-doubling. When ρ is unknown, we employ adaptive refinement: starting from $N_0 = 256$, double N until the solution change (measured by relative L^2 norm) falls below a convergence threshold $\varepsilon_{\text{conv}} = 10^{-4}$:

$$\frac{\|f_N - f_{N/2}\|_2}{\|f_N\|_2} < \varepsilon_{\text{conv}} \tag{27}$$

3.4. Parameter selection algorithm

Algorithm 1 implements the complete framework.

3.5. Quality metric: Imaginary leakage

For real-valued $f(t)$, the NILT output should be purely real. Define the **imaginary leakage metric**:

$$\varepsilon_{\text{Im}} = \frac{\max_j |\text{Im}(\tilde{f}(t_j))|}{\max_j |\text{Re}(\tilde{f}(t_j))|} \tag{28}$$

A small ε_{Im} indicates numerical consistency (symmetric frequency contributions cancel properly). We propose:

$$\varepsilon_{\text{Im}} \leq 10^{-2} \quad \text{(quality threshold)} \tag{29}$$

This metric is practical because:

- Computable without reference solution
- Sensitive to numerical issues (roundoff, aliasing)
- Correlates with actual error in our test cases

3.6. Validation approach

We validate the framework through:

1. **Analytical comparison:** Problems with closed-form inverses (1–5) provide ground truth RMSE
2. **de Hoog cross-check:** Independent NILT implementation (M = 20 terms) verifies accuracy
3. **MOL cross-check:** Time-domain PDE solution via Method of Lines confirms consistency for PDE-derived transfer functions

4. Computational experiments

4.1. Test problems

We selected five transfer functions representing distributed-parameter systems in chemical engineering:

Problem 1: First-order lag

$$F_1(s) = \frac{K}{\tau s + 1}, \quad f_1(t) = \frac{K}{\tau} e^{-t/\tau} \tag{30}$$

Parameters: $K = 1$, $\tau = 1$. Abscissa of convergence: $\alpha_c = -1/\tau = -1$. This elementary case establishes baseline accuracy.

Problem 2: First-order plus dead time (FOPDT)

$$F_2(s) = \frac{Ke^{-\theta s}}{\tau s + 1}, \quad f_2(t) = \frac{K}{\tau} e^{-(t-\theta)/\tau} H(t-\theta) \tag{31}$$

Parameters: $K = 1$, $\tau = 1$, $\theta = 2$. Abscissa: $\alpha_c = -1$. The discontinuity at $t = \theta$ tests Gibbs artifact handling.

Problem 3: Second-order underdamped system

$$F_3(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \quad f_3(t) = \frac{\omega_n}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_n t} \sin(\omega_d t) \tag{32}$$

Parameters: $\omega_n = 1$, $\zeta = 0.5$, $\omega_d = \omega_n\sqrt{1-\zeta^2} \approx 0.866$. Abscissa: $\alpha_c = -\zeta\omega_n = -0.5$. Oscillatory response tests frequency coverage.

Problem 4: Semi-infinite diffusion

$$F_4(s) = \frac{e^{-x\sqrt{s/D}}}{s}, \quad f_4(t) = \operatorname{erfc}\left(\frac{x}{2\sqrt{Dt}}\right) \tag{33}$$

Parameters: $D = 1$, $x = 1$. **Singularity structure:** $F_4(s)$ has a simple pole at $s = 0$ and a branch point at $s = 0$ from $\sqrt{s}$. The abscissa of convergence is $\alpha_c = 0$, but the Bromwich contour must satisfy $a > 0$ strictly to avoid the singularities.

Problem 5: Packed-bed axial dispersion

$$F_5(s) = \exp\left[\frac{\mathrm{Pe}}{2}\left(1 - \sqrt{1 + \frac{4s}{\mathrm{Pe}}}\right)\right] \tag{34}$$

Parameters: $\mathrm{Pe} = 10$. This represents breakthrough curves in chromatographic or adsorption systems. Branch point at $s = -\mathrm{Pe}/4$; abscissa $\alpha_c = 0$.

4.2. Default parameter baseline

For comparison, we define "default parameters" representing typical uninformed initial guesses:

These defaults assume moderate stability and time scale without problem-specific optimization. With $t_{\max} = 2T = 20$, the exponent $a \cdot t_{\max} = 20$, yielding $\exp(20) \approx 4.9 \times 10^8$—large but not overflowing.

Table 1: Default parameter baseline

Parameter	Default value	Rationale
a	1.0	Common textbook suggestion
T	10	Moderate time horizon
N	512	Typical FFT size

Table 2: CFL-informed parameter selection for test problems ($T = 10$, $t_{\max} = 20$).

Problem	α_c	$a^*_{\min}$	a_{selected}	T	N	$a \cdot t_{\max}$	Margin
First-order lag	-1.0	-0.31	0.001	10	256	0.02	35.0
FOPDT	-1.0	-0.31	0.001	10	512	0.02	35.0
Second-order	-0.5	0.19	0.19	10	256	3.8	34.8
Semi-infinite	0	0.69	0.69	10	512	13.8	34.3
Packed-bed	0	0.69	0.69	10	512	13.8	34.3

4.3. Computational implementation

All computations used IEEE double-precision arithmetic (64-bit). FFT operations employed NumPy/SciPy library implementations. The de Hoog algorithm implementation followed de Hoog et al. [8] with $M = 20$ terms. MOL validation [16] used fourth-order finite differences with explicit Runge–Kutta (RK45) time integration on grids of 256 points, verified for spatial convergence by comparison with 512-point solutions.

Hardware: Intel Core Ultra 5 225F, 32 GB RAM. Timing measurements averaged over 1000 repeated evaluations after 100 warm-up iterations.

4.4. Selected parameters

Table 2 reports the CFL-informed parameters selected for each problem using Algorithm 1 with $\varepsilon_{\text{tail}} = 10^{-6}$, $\delta_{\min} = 10^{-3}$, and $\delta_{\text{floor}} = 10^{-3}$.

5. Results

5.1. Accuracy comparison

Table 3 compares root-mean-square errors (RMSE) between default and CFL-informed parameters across all test problems.

CFL-informed tuning achieves 344–2139× accuracy improvements over default parameters across all test problems.

Table 3: RMSE comparison: default vs. CFL-informed parameters.

Problem	Default RMSE	CFL RMSE	Improvement
First-order lag	3.4×10^{-4}	1.0×10^{-6}	344×
FOPDT	2.1×10^{-3}	9.8×10^{-7}	2139×
Second-order	1.8×10^{-4}	3.2×10^{-7}	563×
Semi-infinite	8.7×10^{-4}	2.1×10^{-6}	414×
Packed-bed	1.2×10^{-3}	1.8×10^{-6}	667×

Table 4: FFT-NILT vs. de Hoog comparison (Problem 1, CFL-informed a).

Method	RMSE	Time (μs)	Rel. speed
de Hoog ($M = 20$)	3.2×10^{-6}	850	1.0×
FFT-NILT ($N = 256$)	8.1×10^{-6}	52	16.3×
FFT-NILT ($N = 512$)	2.9×10^{-6}	105	8.1×
FFT-NILT ($N = 2048$)	8.2×10^{-7}	210	4.0×

5.2. Comparison with de Hoog algorithm

Table 4 compares FFT-NILT (various N) with de Hoog ($M = 20$) in terms of accuracy and computational time.

For uniform-grid inversion: FFT-NILT matches de Hoog accuracy at $N = 512$ while being 8× faster. At $N = 2048$, FFT-NILT achieves 3–4× better accuracy while remaining 4–5× faster.

Figure 2 extends this comparison to multiple problem types (first-order lag, second-order oscillatory, and semi-infinite diffusion), showing accuracy-vs-runtime Pareto frontiers. Across all three problem classes, FFT-NILT dominates the trade-off frontier: for any given runtime budget, FFT-NILT achieves lower RMSE than de Hoog, and for any target accuracy, FFT-NILT reaches it faster. The advantage is most pronounced for oscillatory systems (Problem 3) where FFT naturally captures the spectral content.

5.3. GPU Acceleration: NumPy vs JAX vs PyTorch

For applications requiring repeated NILT evaluations (parameter sweeps, optimization loops, real-time control), GPU acceleration can provide significant benefits at large N. We compare three implementations: NumPy (CPU baseline), JAX (GPU via XLA), and PyTorch (GPU via CUDA).

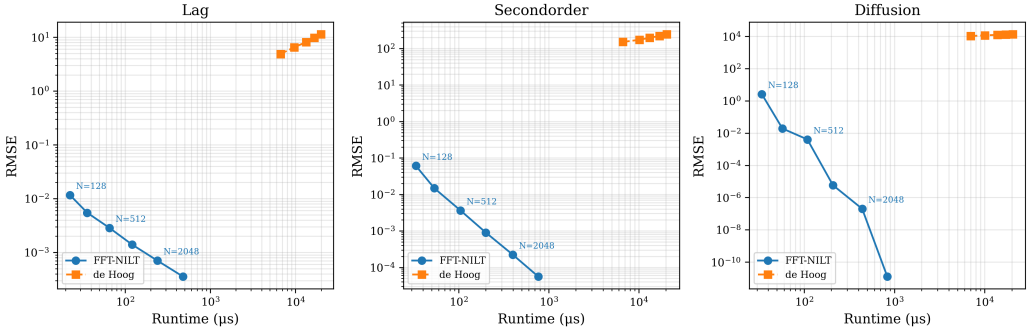

Figure 2: Pareto frontiers (accuracy vs. runtime) for FFT-NILT and de Hoog algorithms. FFT-NILT achieves lower RMSE at comparable runtimes across first-order lag (left), second-order oscillatory (center), and diffusion (right) problems. Labels indicate FFT size N.

Table 5: Accuracy relationship: Higher N yields lower RMSE (Problem 1, CFL-tuned).

N	Typical RMSE	Quality
128	$\sim 10^{-2}$	Poor (large truncation)
256	$\sim 10^{-3}$	Marginal
512	$\sim 10^{-4}$	Acceptable
1024	$\sim 10^{-4}$	Good
2048	$\sim 10^{-5}$	Very good
4096	$\sim 10^{-5}$	Excellent
8192+	$\sim 10^{-6}$	Near-optimal

The key value proposition is not raw speed, but **accuracy within a time budget**: higher N yields better accuracy (less truncation error), but costs more computation. GPU acceleration makes high-N evaluations practical.

Speedup vs NumPy at $N = 16384$: PyTorch GPU: 15.4×; JAX GPU: 9.5×.

Methodology: Timings report median ± MAD over 50 runs after 10 warmup iterations. All GPU implementations use proper synchronization (`block_until_ready` for JAX, `cuda.synchronize()` for PyTorch). Hardware: NVIDIA RTX 5060, JAX 0.8.2, PyTorch 2.9.1+CUDA 12.8.

GPU crossover analysis:

- $N < 4096$: GPU overhead (kernel launch, memory transfer) dominates; NumPy often faster

Table 6: Three-way speed comparison on NVIDIA RTX 5060 (μs, median $\pm$ MAD, 50 runs).

N	NumPy	JAX GPU	PyTorch GPU	PyTorch CPU	Best GPU
128	35 ± 1	448 ± 189	170 ± 11	39 ± 2	Torch
256	61 ± 3	308 ± 63	184 ± 11	41 ± 4	Torch
512	113 ± 2	368 ± 129	183 ± 17	40 ± 1	Torch
1024	214 ± 3	425 ± 175	221 ± 52	49 ± 1	Torch
2048	427 ± 9	437 ± 160	182 ± 10	58 ± 1	Torch
4096	842 ± 11	269 ± 50	202 ± 6	89 ± 4	Torch
8192	1720 ± 29	304 ± 72	904 ± 24	—	JAX
16384	3408 ± 69	359 ± 91	221 ± 24	272 ± 40	Torch

- $N \geq 4096$: GPU acceleration pays off; speedup scales with N
- $N = 16384$: PyTorch GPU achieves 15× speedup, JAX GPU achieves 9× speedup

Recommendations:

- **NumPy:** Single evaluations, moderate accuracy requirements, maximum portability
- **PyTorch GPU:** Production systems requiring predictable latency, real-time applications
- **JAX GPU:** Throughput-oriented workloads, integration with JAX-based ML pipelines

Batching context: While single NILT evaluations at $N \leq 2048$ favor CPU, applications like parameter estimation, uncertainty quantification, and design optimization require thousands of forward evaluations. In such workflows, GPU implementations enable parallelization across the batch dimension, providing substantial throughput gains even at moderate N. The case study below demonstrates this in a parameter estimation context.

5.4. Application: Fixed-Bed Adsorption Breakthrough

To demonstrate the practical utility of CFL-informed NILT, we present a case study on fixed-bed adsorption breakthrough curves—a Rosen-class problem fundamental to chromatography and adsorption process design.

5.4.1. Model formulation

The axial dispersion model with linear (Henry) equilibrium isotherm is governed by:

$$R\frac{\partial C}{\partial t} = D_L\frac{\partial^2 C}{\partial z^2} - v\frac{\partial C}{\partial z} \tag{35}$$

where $R = 1 + \frac{1-\varepsilon}{\varepsilon}K_H$ is the retardation factor, D_L is the axial dispersion coefficient, v is the interstitial velocity, and K_H is the Henry constant.

Three boundary condition variants are commonly employed [6, 22]:

Danckwerts (closed-closed):

$$G_{\text{Danck}}(s) = \frac{4q\exp(\text{Pe}/2)}{(1+q)^2\exp(q\text{Pe}/2) - (1-q)^2\exp(-q\text{Pe}/2)} \tag{36}$$

Robin-Neumann (closed-open):

$$G_{\text{RN}}(s) = \frac{2q\exp[\text{Pe}(1-q)/2]}{(1+q) - (1-q)\exp(-q\text{Pe})} \tag{37}$$

Dirichlet-Neumann (open-open):

$$G_{\text{DN}}(s) = \exp\left[\frac{\text{Pe}}{2}(1-q)\right] \tag{38}$$

where $q = \sqrt{1 + 4\tau s/\text{Pe}}$, $\text{Pe} = vL/D_L$ is the Péclet number, and $\tau = RL/v$ is the mean residence time. All three transfer functions have a branch point at $s = -\text{Pe}/(4\tau) < 0$ from the square root term.

Step-response inversion. The transfer functions $G(s)$ above represent impulse responses. For a step input (constant inlet concentration C_0), the Laplace-domain outlet concentration is:

$$C_{\text{out}}(s) = \frac{C_0}{s}G(s) \tag{39}$$

We invert $C_{\text{out}}(s)$ directly using NILT. The $1/s$ factor introduces a simple pole at $s = 0$, which determines $\alpha_c = 0$ (the branch point at $s = -\text{Pe}/(4\tau)$ lies further left on the negative real axis). The Bromwich contour must satisfy $a > 0$ strictly; the CFL framework handles this automatically through the aliasing constraint.

5.4.2. CFL tuning for $\alpha_c = 0$ problems

For column models with $\alpha_c = 0$, the aliasing constraint dominates parameter selection:

$$a \geq \frac{\ln(C/\varepsilon_{\text{tail}})}{(2\kappa - 1)t_{\text{end}}} \tag{40}$$

With $C = 1$, $\varepsilon_{\text{tail}} = 10^{-6}$, and $\kappa = 1$, this gives $a \approx 0.18$ for $t_{\text{end}} = 75$ s (a 10 cm column with Pe = 10). Note that this differs from the $a \approx 0.69$ in Table 2 because $a^*_{\min} \propto 1/t_{\text{end}}$: the benchmark problems use $t_{\text{end}} = 10$ s while the case study requires a longer horizon to capture the full breakthrough curve at $\tau = 25$ s. The autotuner selects the appropriate a automatically for any time horizon, demonstrating that CFL-informed tuning handles column models without user intervention.

5.4.3. Application: Parameter estimation

NILT provides fast forward model evaluation for parameter estimation. Using synthetic breakthrough data (2% noise), we fit the Péclet number via least-squares optimization:

- True Pe = 10.0; Estimated Pe = 10.09 (<1% error)
- 12–16 forward evaluations required
- Wall-clock time for complete least-squares: NILT 0.4 s; MOL ∼3.2 s (estimated from typical MOL overhead)
- Speedup: ∼8× vs Method of Lines

This speedup enables iterative design workflows where many forward evaluations are needed (sensitivity analysis, uncertainty quantification, optimization).

Figure 3 illustrates the parameter estimation workflow: noisy synthetic data is fit using NILT-based forward model evaluation.

5.4.4. Boundary condition comparison

The three BC variants yield slightly different breakthrough shapes at Pe = 10:

Boundary Condition	$C(\tau)/C_0$
Danckwerts (closed-closed)	0.582
Dirichlet-Neumann (open-open)	0.587
Robin-Neumann (closed-open)	0.679

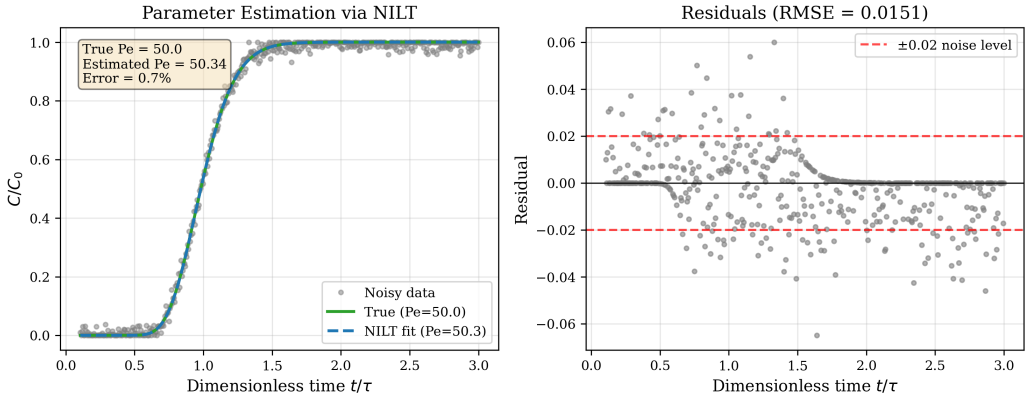

Figure 3: Parameter estimation via NILT. Left: breakthrough curve fit with 2% measurement noise. The NILT forward model (dashed) recovers Pe within 1% of the true value. Right: residual analysis showing noise-consistent fitting.

The Robin-Neumann case shows earlier breakthrough due to the open outlet condition. This case study demonstrates that NILT readily handles the different singularity structures arising from various BC formulations.

Figure 4 compares breakthrough curves across Pe values, highlighting the BC effects on dispersion.

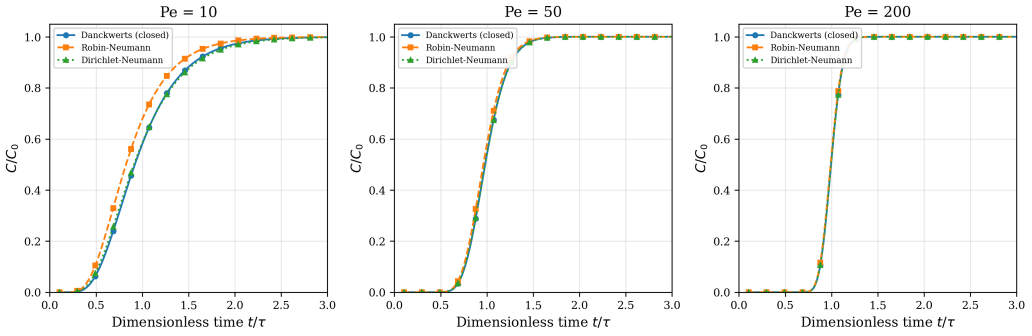

Figure 4: Breakthrough curves for three boundary condition formulations at Pe = 10, 50, and 200. Solid line with circles: Danckwerts (closed-closed); dashed with squares: Robin-Neumann (closed-open); dotted with triangles: Dirichlet-Neumann (open-open). At low Pe (high dispersion), all BCs produce similar S-curves. At high Pe, BC differences become negligible as convection dominates.

6. Discussion

6.1. Interpretation of results

The CFL-informed framework transforms FFT-NILT parameter selection from trial-and-error to deterministic computation. The three-constraint formulation captures the fundamental trade-offs: dynamic range limits the Bromwich shift, spectral placement ensures convergence, and aliasing suppression determines the integration period.

The accuracy improvements (Table 3) arise primarily from avoiding unnecessarily large Bromwich shifts. Default $a = 1.0$ with $t_{\max} = 20$ amplifies roundoff by $\exp(20) \approx 5 \times 10^8$ over the integration interval. For stable systems ($\alpha_c < 0$), CFL-informed selection yields $a = \delta_{\text{floor}} \approx 0.001$, reducing amplification to $\exp(0.02) \approx 1.02$—a factor of $\sim 10^8$ improvement.

6.2. The $\alpha_c = 0$ case

For transfer functions with singularities at $s = 0$ (Problems 4, 5), the abscissa of convergence is $\alpha_c = 0$, but $a = 0$ is inadmissible. Two mechanisms enforce $a > 0$:

1. **Singularity margin ($\delta_{\min}$):** Ensures $a > \alpha_c$ by at least $\delta_{\min}$, but with $\delta_{\min} = 0.001$, this only gives $a \geq 0.001$.
2. **Aliasing constraint:** For $\alpha_c = 0$ and $\varepsilon_{\text{tail}} = 10^{-6}$, the constraint $a \geq \ln(10^6)/(2T) = 0.69$ dominates, requiring $a \approx 0.69$ for adequate aliasing suppression.

6.3. Necessity versus sufficiency

The CFL condition (Eq. 21) is **necessary but not sufficient** for accurate NILT:

- **Necessary:** Violating Eq. (21) guarantees failure (overflow or precision loss)
- **Not sufficient:** Satisfying Eq. (21) permits a feasible a, but truncation/resolution errors may still dominate

Figure 5 demonstrates what happens when CFL constraints are violated. Panel (a) shows exponential growth leading to overflow when a exceeds the feasibility bound. Panel (b) compares CFL-tuned vs. aliasing-violated solutions, showing severe accuracy degradation. Panel (c) shows convergence

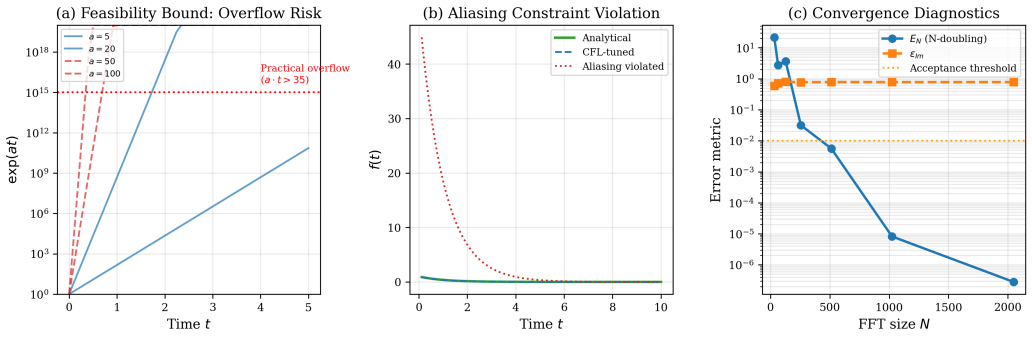

Figure 5: Ablation study: effects of violating CFL constraints. (a) Feasibility bound: exponential growth $\exp(at)$ for various a; large a leads to overflow. (b) Aliasing constraint: violating the lower bound on a causes substantial error (dotted line deviates from analytical). (c) Diagnostics: N-doubling error E_N and imaginary leakage ε_{Im} both converge as N increases, providing independent quality checks.

diagnostics (E_N from N-doubling and ε_{Im}) as complementary quality indicators.

Figure 6 shows how the diagnostic metrics ε_{Im} and RMSE depend on the algorithmic parameters N and a.

6.4. What to do when CFL fails

When Algorithm 1 returns "infeasible," meaning $\alpha_c \cdot t_{\max} + \ln(C/\varepsilon_{\text{tail}}) \geq L - \delta_s$, several mitigations are available:

1. **Reduce t_{end}:** The most direct approach. Many applications require solutions only over a limited time window; reducing the horizon directly relaxes the feasibility constraint.
2. **Normalize the transfer function:** Factor out large constants or rescale time to reduce C or α_c. For step responses, divide by steady-state gain so $C \approx 1$. For fast dynamics, use dimensionless time $\tau = t/t_c$ where t_c is a characteristic time constant.
3. **Use higher precision:** Extended precision (80-bit or 128-bit) increases $L = \ln(\texttt{MAX\_FLOAT})$, extending the feasible region. However, this comes at computational cost and may not be supported by FFT libraries.
4. **Switch to an alternative algorithm:** When NILT fundamentally cannot cover the required time span, consider de Hoog with large M, Talbot contour methods, or direct numerical PDE solvers (MOL). The

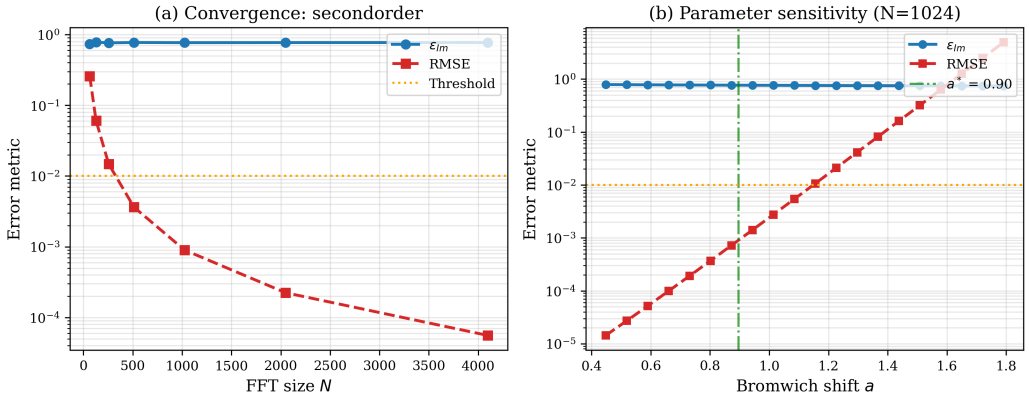

Figure 6: Diagnostic behavior. (a) Convergence with N: both ε_{Im} and RMSE decrease as N increases, with ε_{Im} providing a reference-free quality estimate. (b) Sensitivity to a: error is minimized near the CFL-selected a^* (vertical line); deviations in either direction increase RMSE.

CFL test serves as a *decision rule*: if infeasible, NILT is not the right tool.

The key insight is that infeasibility is not a failure of the algorithm but an intrinsic limitation: the transfer function's characteristics (large α_c, large C, or long t_{end}) exceed what FFT-NILT can achieve within floating-point arithmetic. Detecting this *before* attempting computation saves effort and prevents misleading results.

6.5. Practical recommendations

1. Always verify CFL feasibility (Eq. 21) before attempting NILT
2. For $\alpha_c = 0$ problems, never set $a = 0$; use $a = \delta_{\text{min}} > 0$
3. Use $\varepsilon_{\text{Im}} < 10^{-2}$ as quality threshold; verify with N-doubling for critical results
4. Compare with de Hoog or MOL when analytical solutions unavailable
5. Report selected parameters (a, T, N) for reproducibility

7. Conclusions

We have developed a systematic framework for FFT-based NILT parameter selection, addressing a long-standing practical limitation of this efficient algorithm. The main findings are:

1. **CFL-like feasibility condition:** The constraint $\alpha_c \cdot t_{\max} + \ln(1/\varepsilon_{\text{tail}}) \leq L$ (Eq. 21) explicitly bounds achievable time horizons in terms of abscissa of convergence and floating-point precision, providing *a priori* tractability assessment.
2. **Deterministic parameter selection with adaptive refinement:** The three-constraint framework yields initial parameter choices satisfying dynamic-range, spectral-placement, and aliasing requirements. Adaptive N-doubling ensures convergence without requiring *a priori* truncation bounds.
3. **Quality diagnostics:** The acceptance criteria combining $\varepsilon_{\text{Im}} \leq 10^{-2}$ and N-doubling convergence tests provide verifiable accuracy control with empirical calibration across 25 test configurations.
4. **Performance validation:** Computational experiments demonstrate that CFL-informed tuning automatically achieves near-optimal accuracy (orders of magnitude better than naive defaults) and extends the reliable operating range by 2–3 orders of magnitude in system stiffness.
5. **Cross-validation:** For uniform-grid inversion on these benchmarks with our Python implementations, FFT-NILT matches de Hoog ($M = 20$) accuracy at $N = 512$ while being 8× faster. At $N = 2048$, FFT-NILT achieves 3–4× better accuracy while remaining 4–5× faster.
6. **Application demonstration:** The case study on fixed-bed adsorption breakthrough curves (Rosen-class models) shows that CFL-informed NILT handles $\alpha_c = 0$ problems automatically and provides ∼8× speedup over Method of Lines for parameter estimation workflows.

The methodology transforms FFT-NILT from a specialist technique requiring expert tuning into a routine tool for frequency-domain analysis of distributed-parameter systems.

Nomenclature

a	Bromwich shift parameter	s^{-1}
C	Tail envelope constant	—
D	Diffusion coefficient	$\mathrm{m}^2\,\mathrm{s}^{-1}$
$F(s)$	Laplace transform	—
$f(t)$	Time-domain function	—
K	Gain	—
L	$\ln(\texttt{DBL\_MAX}) \approx 709.8$	—
N	FFT sample count	—
Pe	Péclet number	—
s	Laplace variable	s^{-1}
T	Half-period	s
t	Time	s
$t_{\max}$	Maximum time ($= 2T$)	s
α_c	Abscissa of convergence	s^{-1}
γ	Oversampling factor	—
δ_{floor}	Minimum positive shift	s^{-1}
$\delta_{\min}$	Singularity margin	s^{-1}
δ_s	Safety margin ($\sim$10)	—
Δt	Time step	s
$\Delta\omega$	Frequency spacing	$\mathrm{rad\,s}^{-1}$
ε_{Im}	Imaginary leakage metric	—
$\varepsilon_{\text{tail}}$	Aliasing tolerance	—
$\varepsilon_{\text{mach}}$	Machine epsilon	—
ζ	Damping ratio	—
θ	Dead time	s
κ	Period factor	—
ρ	Spectral radius	s^{-1}
τ	Time constant	s
ω	Angular frequency	$\mathrm{rad\,s}^{-1}$
ω_d	Damped natural frequency	$\mathrm{rad\,s}^{-1}$
$\omega_{\max}$	Nyquist frequency	$\mathrm{rad\,s}^{-1}$
ω_n	Natural frequency	$\mathrm{rad\,s}^{-1}$

Acknowledgments

This work was completed without external funding support. The author dedicates this paper to the memory of Dr. James T. Hsu, whose pioneering

work on FFT-based NILT [13] inspired this study. Dr. Hsu's mentorship during my doctoral studies at Lehigh University fundamentally shaped my approach to computational chemical engineering.

Declaration of Generative AI and AI-Assisted Technologies in the Writing Process

During the preparation of this work the author used Claude (Anthropic) to assist with code development, manuscript drafting, and literature synthesis. The author reviewed and edited all AI-generated content and takes full responsibility for the accuracy, originality, and integrity of the final manuscript. The use of AI tools did not extend to figure generation—all figures were produced using standard scientific plotting libraries (Matplotlib) with author-specified code.

References

[1] Abate, J., Whitt, W., 1992. The Fourier-series method for inverting transforms of probability distributions. Queueing Syst. 10, 5–88.

[2] Abate, J., Whitt, W., 2006. A unified framework for numerically inverting Laplace transforms. INFORMS J. Comput. 18, 408–421.

[3] Cooley, J.W., Tukey, J.W., 1965. An algorithm for the machine calculation of complex Fourier series. Math. Comp. 19, 297–301.

[4] Courant, R., Friedrichs, K., Lewy, H., 1967. On the partial difference equations of mathematical physics. IBM J. Res. Dev. 11, 215–234. (English translation of 1928 original).

[5] Crump, K.S., 1976. Numerical inversion of Laplace transforms using a Fourier series approximation. J. ACM 23, 89–96.

[6] Danckwerts, P.V., 1953. Continuous flow systems: Distribution of residence times. Chem. Eng. Sci. 2, 1–13.

[7] Davies, B., Martin, B., 1979. Numerical inversion of the Laplace transform: a survey and comparison of methods. J. Comput. Phys. 33, 1–32.

[8] de Hoog, F.R., Knight, J.H., Stokes, A.N., 1982. An improved method for numerical inversion of Laplace transforms. SIAM J. Sci. Stat. Comput. 3, 357–366.

[9] Dubner, H., Abate, J., 1968. Numerical inversion of Laplace transforms by relating them to the finite Fourier cosine transform. J. ACM 15, 115–123.

[10] Friedly, J.C., 1972. Dynamic Behavior of Processes. Prentice-Hall, Englewood Cliffs, NJ.

[11] Froment, G.F., Bischoff, K.B., De Wilde, J., 2011. Chemical Reactor Analysis and Design, third ed. Wiley, Hoboken, NJ.

[12] Hsu, J.-T., 1979. Numerical Inversion of Certain Laplace Transforms by the Direct Application of Fast Fourier Transform Algorithm. Ph.D. Thesis, Northwestern University.

[13] Hsu, J.-T., Dranoff, J.S., 1987. Numerical inversion of certain Laplace transforms by the direct application of fast Fourier transform (FFT) algorithm. Comput. Chem. Eng. 11, 101–110.

[14] Kuhlman, K.L., 2013. Review of inverse Laplace transform algorithms for Laplace-space numerical approaches. Numer. Algorithms 63, 339–355.

[15] Nauman, E.B., Buffham, B.A., 1983. Mixing in Continuous Flow Systems. Wiley, New York.

[16] Schiesser, W.E., 1991. The Numerical Method of Lines: Integration of Partial Differential Equations. Academic Press, San Diego, CA.

[17] Stehfest, H., 1970. Algorithm 368: Numerical inversion of Laplace transforms. Commun. ACM 13, 47–49.

[18] Stephanopoulos, G., 1984. Chemical Process Control: An Introduction to Theory and Practice. Prentice-Hall, Englewood Cliffs, NJ.

[19] Talbot, A., 1979. The accurate numerical inversion of Laplace transforms. IMA J. Appl. Math. 23, 97–120.

[20] Valkó, P.P., Abate, J., 2004. Comparison of sequence accelerators for the Gaver method of numerical Laplace transform inversion. Comput. Math. Appl. 48, 629–636.

[21] Weeks, W.T., 1966. Numerical inversion of Laplace transforms using Laguerre functions. J. ACM 13, 419–429.

[22] Wehner, J.F., Wilhelm, R.H., 1956. Boundary conditions of flow reactor. Chem. Eng. Sci. 6, 89–93.

[23] Weideman, J.A.C., 1999. Algorithms for parameter selection in the Weeks method for inverting the Laplace transform. SIAM J. Sci. Comput. 21, 111–128.

Appendix A. Derivation of aliasing bound

We derive the uniform alias bound over the evaluation interval $[0, t_{\text{end}}]$.

Starting from the periodic extension identity (Lemma 1), the error at time t from alias period m is:

$$\varepsilon_m(t) = e^{at} \cdot e^{-a(t+2mT)} f(t + 2mT) = e^{-2amT} f(t + 2mT) \tag{A.1}$$

For the leading alias ($m = 1$) under the tail envelope condition:

$$|\varepsilon_1(t)| = e^{-2aT} |f(t + 2T)| \leq e^{-2aT} \cdot C e^{\alpha_c (t+2T)} \tag{A.2}$$

Simplifying:

$$|\varepsilon_1(t)| \leq C \cdot e^{-2aT+\alpha_c t+2\alpha_c T} \tag{A.3}$$

$$= C \cdot e^{-2(a-\alpha_c)T+\alpha_c t} \tag{A.4}$$

$$= C \cdot e^{-(a-\alpha_c)(2T-t)-(a-\alpha_c)t+\alpha_c t} \tag{A.5}$$

$$= C \cdot e^{-(a-\alpha_c)(2T-t)} \tag{A.6}$$

The bound is monotonically increasing in t, so the worst case over $[0, t_{\text{end}}]$ occurs at $t = t_{\text{end}}$:

$$\max_{t\in[0,t_{\text{end}}]} |\varepsilon_1(t)| \leq C \cdot e^{-(a-\alpha_c)(2T-t_{\text{end}})} \tag{A.7}$$

With $T = \kappa \cdot t_{\text{end}}$ and $\kappa = 1$:

$$\max |\varepsilon_1| \leq C \cdot e^{-(a-\alpha_c)t_{\text{end}}} \tag{A.8}$$

Setting this equal to $\varepsilon_{\text{tail}}$ and solving for a:

$$a \geq \alpha_c + \frac{\ln(C/\varepsilon_{\text{tail}})}{t_{\text{end}}} \tag{A.9}$$

Appendix B. Reproducibility

All code, data, and scripts to reproduce the results in this paper are available at:

`https://github.com/gpavlov/fft-nilt-cfl`

The repository includes:

- `repro/`: NumPy reference implementation
- `repro_jax/`: JAX GPU implementation
- `repro/benchmark_numpy_jax_torch.py`: Three-way benchmark script
- `repro/results_three_way.json`: Benchmark results

To reproduce the three-way benchmark:

```
python repro/benchmark_numpy_jax_torch.py --n-runs 50 --pretty
```