# Numeric IRF Inversion with Accelerated Integrals

X. Kong, 2025

There are two equivalent viewpoints for the explanation of the $1/f$ tail cancellation in $\Im R(f)$

## Two perspectives - full Fourier Transform and Fourier Sine Transform

Write $R(f) = A(f) + iB(f)$ with $A = \Re R$ (even) and $B = \Im R$ (odd).
For a causal $K(t)$ with a jump $K_0 = K(0^+)$, the high-frequency tail is

$$B(f) = -\frac{K_0}{2\pi f} + \frac{\beta}{f^3} + O\Big(\frac{1}{f^5}\Big) \qquad (|f| \to \infty). \tag{1}$$

We modify $K$ by subtracting an exponential with the same jump:

$$\hat{K}(t) = K(t) - K_0 e^{-\alpha t} u(t).$$

The exponential part $K_0 e^{-\alpha t}$ has its Fourier transform that can be expressed as a sum of even and odd functions

$$\frac{K_0}{\alpha + i2\pi f} = \underbrace{\frac{K_0 \alpha}{\alpha^2 + (2\pi f)^2}}_{\text{real, even}} - i \underbrace{\frac{K_0\, 2\pi f}{\alpha^2 + (2\pi f)^2}}_{\text{imag, odd}}. \tag{2}$$

## Approach I: Full transform (subtract and invert)

Use $\omega = 2\pi f$ for simplicity, and define

$$\widetilde{R}(f) := R(f) - \frac{K_0}{\alpha + i2\pi f} = \Big(A - \frac{K_0 \alpha}{\alpha^2 + \omega^2}\Big) + i\Big(B + \frac{K_0 \omega}{\alpha^2 + \omega^2}\Big). \tag{3}$$

- Here the **real part** of the exponential combines with $A = \Re R$; it decays like $1/f^2$, so it is already fast and not the bottleneck.
- The **imaginary part** correction $+\dfrac{K_0\omega}{\alpha^2+\omega^2}$ is what matters: for large $|f|$,

$$\frac{K_0\omega}{\alpha^2+\omega^2} = \frac{K_0}{\omega}\left(1 - \frac{\alpha^2}{\omega^2} + O(\omega^{-4})\right),$$

so $B + \dfrac{K_0\omega}{\alpha^2+\omega^2}$ exactly cancels the $-K_0/\omega$ tail in $B$.

- Inverting $\widetilde{R}$ gives $\hat{K}$ directly; then add back $K_0 e^{-\alpha t}$.

So in this full view, the real part of $K_0/(\alpha + i2\pi f)$ is combined with $\Re R(f)$. But the slow convergent $1/f$ issue is in $B$, and that's where the key cancellation occurs.

---

## Approach II: Odd spectrum (sine-transform)

If we use a **sine transform** (odd extension forms a **purely imaginary** spectrum) by taking the double imaginary part:

$$\hat{R}_{\text{odd}}(f) = 2i\Im\left(R(f) - \frac{K_0}{\alpha + i2\pi f}\right) = 2iB(f) + i\frac{4\pi K_0 f}{\alpha^2 + (2\pi f)^2}. \tag{4}$$

Here the **real part is intentionally discarded**—by construction $\hat{R}_{\text{odd}}$ is purely imaginary (odd time signals have purely imaginary spectra). n invert $\hat{R}_{\text{odd}}$ with a sine transform to get $\hat{K}$.

### How the cancellation works, explicitly

From (1),

$$2iB(f) = i\left(-\frac{K_0}{\pi f} + \frac{2\beta}{f^3} + O(f^{-5})\right).$$

Expand the rational term:

$$i\frac{4\pi K_0 f}{\alpha^2 + (2\pi f)^2} = i\left(\frac{K_0}{\pi f} - \frac{K_0\alpha^2}{4\pi^3 f^3} + O(f^{-5})\right).$$

Add them:

$$\hat{R}_{\text{odd}}(f) = i\left[-\frac{K_0}{\pi f} + \frac{K_0}{\pi f}\right] + i\left(\frac{2\beta}{f^3} - \frac{K_0\alpha^2}{4\pi^3 f^3}\right) + O(f^{-5}),$$

so the $1/f$ terms **cancel exactly**, leaving $\hat{R}_{\text{odd}}(f) = O(i/f^3)$.

This is why, in the time domain, the **sine coefficients decay like** $1/n^3$ for the modified function $\hat{K}$ (it vanishes at both endpoints), and the inversion converges rapidly.

---

## Conclusion

- The phrase "the rational term cancels the $1/f$ tail in $\Im R(f)$" refers to the cancellation in the **imaginary part**; it happens whether the full transform (3) or the odd-spectrum form (4) are used.
- If the **full** transformation (3) is u, the **real** part $\dfrac{K_0\alpha}{\alpha^2 + (2\pi f)^2}$ simply subtracts from $\Re R(f)$ (it decays faster anyway, like $1/f^2$).
- If the **odd-spectrum / se is to use, real parts are neglected at all; by definition only $2i$ times the imaginary part is kept, which is why (4) contains only the imaginary pieces and is purely imaginary.

Either way, the slow $1/f$ tail is neutralized by the $\dfrac{K_0\omega}{\alpha^2 + \omega^2}$ term coming from the subtracted exponential, and the remaining spectrum decays like $1/f^3$.

## Numerical Verification

Below is a complete **numerical application** of the method on the analytic example

$$R(f) = \tfrac{1}{2}\mathbf{1}_{|f|\le\frac{1}{2}} - \frac{i}{2\pi}\log\left|\frac{f+\frac{1}{2}}{f-\frac{1}{2}}\right| \quad \Longleftrightarrow \quad K(t) = \frac{\sin(\pi t)}{\pi t}u(t).$$

Utilizing the acceleration techniques and **odd-spectrum / sine-transform**, these procedures are realized:

1. sample $A(f)$, $B(f)$ from the analytic $R(f)$;
2. estimate the jump $K_0$ from the high-frequency tail $-2\pi f B(f)$;

3. build the corrected odd spectrum

$$S(f) = 2B(f) + \frac{4\pi K_0 f}{\alpha^2 + (2\pi f)^2};$$

4. invert with a **discrete sine transform** on $[0, T]$;
5. restore $K(t) = \hat{K}(t) + K_0 e^{-\alpha t}$;
6. compare to the analytic $K(t)$.

Python code has been used to plot the results (all inline). Key points below:

- Estimated $K_0 = 1.00014307$ (true $K_0 = 1$).
- Sup-norm error on $[0, T]$: $6.53 \times 10^{-3}$; RMS error: $3.29 \times 10^{-3}$.
- The corrected spectrum $|S(f)|$ decays numerically like $f^{-3}$.

**Four figures show**:

1. **Analytical vs reconstructed** $K(t)$ on $[0, T]$ — the curves are visually indistinguishable at this scale.
2. **Reconstruction error** $K_{\text{rec}} - K_{\text{analytic}}$.
3. **Jump estimate**: plot of $-2\pi f B(f)$ tending to $K_0$.
4. **Log–log** plot showing $|S(f)| \sim f^{-3}$.

## Notes on the discretization

- We chose $T = 40.3$ so that $e^{-\alpha T} \approx e^{-8}$ is tiny and the frequency grid **avoids** sampling exactly at the logarithmic singularities $f = \pm\frac{1}{2}$ (which is also a kind of irregular frequencies).

- For the sine transform on $[0, T]$ with $N$ subintervals, the **natural frequency samples** are

$$f_n = \frac{n}{2T}, \qquad n = 1, \ldots, N-1, \quad \Delta f = \frac{1}{2T},$$

  and the interior time nodes are $t_j = j\Delta t$ with $\Delta t = T/N$. The Riemann-sum version of the continuous identity
  $\hat{K}(t) = -2\int_0^\infty S(f)\sin(2\pi f t)\mathrm{d}f$ becomes

$$\hat{K}(t_j) \approx -2\Delta f \sum_{n=1}^{N-1} S(f_n)\sin\left(\frac{\pi j n}{N}\right),$$

  which is exactly a discrete-sine-transform synthesis formula.

- Finally, we add back $K_0 e^{-\alpha t}$ to obtain the causal $K(t)$.

---

## How the Python code os

1. Samples the analytic RAO $R(f) = \frac{1}{2}\mathbf{1}_{|f|\le 1/2} - \frac{i}{2\pi}\log\left|\frac{f+\frac{1}{2}}{f-\frac{1}{2}}\right|$ on the sine grid $f_n = n/(2T)$.
2. Estimates $K_0$ from the high-frequency tail of $\Im R$: $K_0 \approx -2\pi f B(f)$ for large $f$.
3. Forms the corrected odd spectrum $S(f) = 2B(f) + \dfrac{4\pi K_0 f}{\alpha^2 + (2\pi f)^2}$.
4. Inverts by a **discrete sine series** (DST-I grid): $\hat{K}(t_m) \approx -2\Delta f \sum_n S(f_n)\sin\left(\pi nm/N\right)$.
5. Restores $K(t) = \hat{K}(t) + K_0 e^{-\alpha t}$ and compares to the analytic $K(t) = \sin(\pi t)/(\pi t)$.
6. Plots: (i) $K_{\text{true}}$ vs $K_{\text{rec}}$; (ii) error; (iii) tail check for $K_0$; (iv) $|S(f)| \sim f^{-3}$.

From the run here:

- Estimated $K_0 = 1.00014307$ (true $= 1$).
- $|K_{\text{rec}} - K_{\text{true}}|_\infty \approx 6.53 \times 10^{-3}$, RMS error $\approx 3.29 \times 10^{-3}$.

---

# Python source code - DST

```
#!/usr/bin/env python3
# Invert analytic RAO R(f) to impulse response K(t) via accelerated sine inversion
# Example: causal half-sinc. Produces plots and CSVs.
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as scipint
# -------- Parameters --------
T = 40.3           # time window (non-integer so f=0.5 is not sampled exactly)
N = 2048           # grid size; interior time nodes are 1..N-1
alpha = 8.0 / T    # choose alpha so alpha*T ~ 8  -> e^{-alpha T} ≈ 3e-4
# ----------------------------

dt = T / N
t_full = np.linspace(0.0, T, N+1)
j = np.arange(1, N)               # interior time indices
df = 1.0 / (2.0 * T)
f = j * df
omega = 2.0 * np.pi * f

def A_of_f_pos(ff):
    return 0.5 * (ff <= 0.5).astype(float)
```

```
def B_of_f_pos(ff):
    return -(1.0/(2.0*np.pi)) * np.log(np.abs((ff + 0.5) / (ff - 0.5)))

A = A_of_f_pos(f)
B = B_of_f_pos(f)

# Step 2: jump estimate from tail of Im R
m = max(10, int(0.1 * f.size))
tail_idx = np.arange(f.size - m, f.size)
K0_est = np.median(-2.0 * np.pi * f[tail_idx] * B[tail_idx])

# Step 3: corrected odd spectrum S(f)
S = 2.0 * B + (4.0 * np.pi * K0_est * f) / (alpha**2 + omega**2)

# Step 4: discrete sine inversion on [0,T]
n = np.arange(1, N)
sin_matrix = np.sin(np.pi * np.outer(j, n) / N)
hatK_interior = -2.0 * df * (sin_matrix @ S)
hatK_full = np.zeros_like(t_full)
hatK_full[1:-1] = hatK_interior

# Step 5: restore K(t)
K_rec = hatK_full + K0_est * np.exp(-alpha * t_full)
K_true = np.sinc(t_full)  # sin(pi t)/(pi t)

# Diagnostics
err = K_rec - K_true
print(f"Estimated K0: {K0_est:.8f} (true K0 = 1.0)")
print(f"L_inf error: {np.max(np.abs(err)):.3e}")
print(f"RMS error: {np.sqrt(scipint.trapezoid(err**2, x=t_full) / T):.3e}")

# Plots (each chart in its own figure)
plt.figure()
plt.plot(t_full, K_true, label="Analytical K(t)")
plt.plot(t_full, K_rec, linestyle="--", label="Reconstructed K(t)")
plt.xlabel("t"); plt.ylabel("K(t)")
plt.title("Impulse response: analytical vs reconstructed")
plt.legend(); plt.grid(True); plt.show()

plt.figure()
plt.plot(t_full, err, label="Error")
plt.xlabel("t"); plt.ylabel("Error")
plt.title("Reconstruction error over [0, T]")
plt.legend(); plt.grid(True); plt.show()

plt.figure()
prod = -2.0 * np.pi * f * B
plt.plot(f, prod, label=r"-2π f B(f)")
```

```
plt.axhline(1.0, linestyle="--", label="True K0 = 1")
plt.axhline(K0_est, linestyle=":", label=f"Estimated K0 = {K0_est:.6f}")
plt.xlim(0, min(5.0, f[-1]))
plt.xlabel("f"); plt.ylabel(r"-2π f B(f)")
plt.title("Jump estimate from the tail of Im R(f)")
plt.legend(); plt.grid(True); plt.show()

plt.figure()
mask = f > 1.0
plt.loglog(f[mask], np.abs(S[mask]), label=r"|S(f)| after correction")
plt.loglog(f[mask], 1.0/(f[mask]**3), linestyle="--", label=r"f^{-3} reference")
plt.xlabel("f"); plt.ylabel(r"|S(f)|")
plt.title("Corrected odd spectrum decays ~ f^{-3}")
plt.legend(); plt.grid(True, which="both"); plt.show()

# Save CSVs
rao_path = "rao_samples.csv"
time_path = "impulse_response_comparison.csv"
spec_path = "odd_spectrum_corrected.csv"
np.savetxt(rao_path, np.column_stack([f, A, B]), delimiter=",", header="f, A(f), B(f)"
np.savetxt(time_path, np.column_stack([t_full, K_true, K_rec, hatK_full]), delimiter="
np.savetxt(spec_path, np.column_stack([f, S]), delimiter=",", header="f, S(f) (odd spe
print("Saved:", rao_path, time_path, spec_path)
```

## Python source code - Both FFT and DST

```
#!/usr/bin/env python3
"""
Invert analytic RAO R(f) to impulse response K(t) with either:
  - DST (sine-series) inversion on [0,T], or
  - FFT (full complex subtraction) on period 2T

Usage:
  python invert_rao_to_impulse_both.py --method dst
  python invert_rao_to_impulse_both.py --method fft
  python invert_rao_to_impulse_both.py --method both     # writes *both* sets (with _d

Options:
  --T <float>           time window T (default: 40.3)
  --N <int>             grid size used for DST (default: 2048)
  --alphaT <float>      choose alpha via alpha = alphaT / T (default: 8.0 → e^{-alpha
  --plot                (optional) show overlay plots; otherwise TSVs only

Outputs (for single-method runs, TAB-separated):
  rao_samples.csv                 (f, A(f), B(f))  on sine grid f_n = n/(2T)
```

```
  odd_spectrum_corrected.csv       (f, S(f))        diagnostic ~ f^{-3}
  impulse_response_comparison.csv  (t, K_true, K_rec, K_hat)

Outputs for --method both (TAB-separated):
  rao_samples_dst.csv, odd_spectrum_corrected_dst.csv, impulse_response_comparison_dst
  rao_samples_fft.csv, odd_spectrum_corrected_fft.csv, impulse_response_comparison_fft
# Discrete Sine Transform (writes the standard filenames)
python invert_rao_to_impulse_both.py --method dst

# FFT with full complex subtraction (default)
python invert_rao_to_impulse_both.py --method fft --T 40.3 --N 2048 --alphaT 8 --plot

# DST inversion
python invert_rao_to_impulse_both.py --method dst --T 40.3 --N 2048 --alphaT 8 --plot

# Write both sets of CSVs (with _dst/_fft suffixes) and compare plots
python invert_rao_to_impulse_both.py --method both --T 40.3 --N 2048 --alphaT 8 --plot

"""
import argparse
import numpy as np

# Import matplotlib safely *once* at module scope; don't reassign 'plt' inside functio
try:
    import matplotlib.pyplot as plt
except Exception:
    plt = None

# ---------- Analytic RAO ----------
def A_of_f(ff):
    return 0.5 * (np.abs(ff) <= 0.5).astype(float)

def B_of_f(ff):
    ratio = np.abs((ff + 0.5) / (ff - 0.5 + 0j))
    ratio = np.maximum(ratio, 1e-15)
    return -(1.0/(2.0*np.pi)) * np.log(ratio)

def estimate_K0_from_tail(fpos, Bpos):
    m = max(10, int(0.1 * fpos.size))
    tail_idx = np.arange(fpos.size - m, fpos.size)
    return float(np.median(-2.0 * np.pi * fpos[tail_idx] * Bpos[tail_idx]))

# ---------- DST inversion on [0,T] ----------
def invert_dst(T, N, alpha, K0_est):
    # positive sine grid
    n = np.arange(1, N)
    f_pos = n / (2.0 * T)
    w_pos = 2.0 * np.pi * f_pos
```

```
    A_pos = A_of_f(f_pos)
    B_pos = B_of_f(f_pos)
    # odd-spectrum diagnostic
    S_pos = 2.0 * B_pos + (4.0 * np.pi * K0_est * f_pos) / (alpha**2 + w_pos**2)
    # discrete sine inversion (matrix formulation)
    df = 1.0/(2.0*T)
    j = np.arange(1, N)
    sin_matrix = np.sin(np.pi * np.outer(j, n) / N)
    K_hat_interior = -2.0 * df * (sin_matrix @ S_pos)
    # assemble \hat K on [0,T]
    t_full = np.linspace(0.0, T, N+1)
    K_hat = np.zeros_like(t_full)
    K_hat[1:-1] = K_hat_interior
    # restore causal
    K_rec = K_hat + K0_est * np.exp(-alpha * t_full)
    K_true = np.sinc(t_full)
    return (f_pos, A_pos, B_pos, S_pos), (t_full, K_true, K_rec, K_hat)

# ---------- FFT inversion on period 2T ----------
def invert_fft(T, N, alpha, K0_est):
    # FFT grid over 2T with M=2N so dt matches DST
    M = 2 * N
    L = 2.0 * T
    df_fft = 1.0 / L
    dt_fft = L / M
    k = np.arange(-M//2, M//2)
    f_fft = k * df_fft
    w_fft = 2.0 * np.pi * f_fft
    # analytic RAO on FFT grid and full subtraction
    R_fft = A_of_f(f_fft) + 1j * B_of_f(f_fft)
    R_tilde = R_fft - K0_est / (alpha + 1j * w_fft)
    # inverse FFT with integral scaling Δf
    F_shifted = np.fft.ifftshift(R_tilde)
    K_hat_periodic = np.fft.ifft(F_shifted) * M * df_fft
    K_hat_periodic = K_hat_periodic.real
    # restrict to [0,T]
    t_full = np.arange(M) * dt_fft
    idx_half = np.arange(0, N+1)
    t_half = t_full[idx_half]
    K_hat_half = K_hat_periodic[idx_half]
    # restore causal
    K_rec = K_hat_half + K0_est * np.exp(-alpha * t_half)
    K_true = np.sinc(t_half)
    # For RAO/S(f) exports we still use the positive sine grid
    n = np.arange(1, N)
    f_pos = n / (2.0 * T)
    w_pos = 2.0 * np.pi * f_pos
    A_pos = A_of_f(f_pos)
```

```
    B_pos = B_of_f(f_pos)
    S_pos = 2.0 * B_pos + (4.0 * np.pi * K0_est * f_pos) / (alpha**2 + w_pos**2)
    return (f_pos, A_pos, B_pos, S_pos), (t_half, K_true, K_rec, K_hat_half)

# ---------- Save helpers ----------
def save_standard(rao, time, prefix=""):
    f, A, B, S = rao
    t, K_true, K_rec, K_hat = time
    dlm = "\t"
    if prefix:
        np.savetxt(f"rao_samples_{prefix}.csv", np.column_stack([f, A, B]), delimiter=
                   header="f\tA(f)\tB(f)", comments="")
        np.savetxt(f"odd_spectrum_corrected_{prefix}.csv", np.column_stack([f, S]), de
                   header="f\tS(f) (odd spectrum multiplier)", comments="")
        np.savetxt(f"impulse_response_comparison_{prefix}.csv", np.column_stack([t, K_
                   header="t\tK_true\tK_rec\tK_hat", comments="")
    else:
        np.savetxt("rao_samples.csv", np.column_stack([f, A, B]), delimiter=dlm,
                   header="f\tA(f)\tB(f)", comments="")
        np.savetxt("odd_spectrum_corrected.csv", np.column_stack([f, S]), delimiter=dl
                   header="f\tS(f) (odd spectrum multiplier)", comments="")
        np.savetxt("impulse_response_comparison.csv", np.column_stack([t, K_true, K_re
                   header="t\tK_true\tK_rec\tK_hat", comments="")

def main():
    import argparse
    ap = argparse.ArgumentParser()
    ap.add_argument("--method", choices=["dst","fft","both"], default="fft")
    ap.add_argument("--T", type=float, default=40.3)
    ap.add_argument("--N", type=int, default=2048)
    ap.add_argument("--alphaT", type=float, default=8.0, help="alpha*T; alpha = alphaT
    ap.add_argument("--plot", action="store_true", help="optional quick plots")
    args = ap.parse_args()

    T, N = args.T, args.N
    alpha = args.alphaT / T

    # Common RAO on sine grid to estimate K0 (same for both methods)
    n = np.arange(1, N)
    f_pos = n / (2.0 * T)
    B_pos = B_of_f(f_pos)
    K0_est = estimate_K0_from_tail(f_pos, B_pos)

    if args.method in ("dst","both"):
        rao_d, time_d = invert_dst(T, N, alpha, K0_est)
        save_standard(rao_d, time_d, prefix="dst" if args.method=="both" else "")
    if args.method in ("fft","both"):
        rao_f, time_f = invert_fft(T, N, alpha, K0_est)
```

```
        save_standard(rao_f, time_f, prefix="fft" if args.method=="both" else "")

    # Optional quick plots (shared for both) — use module-scope 'plt' safely
    if args.plot and plt is not None:
        if args.method == "both":
            data_dst = np.loadtxt("impulse_response_comparison_dst.csv", delimiter="\t
            data_fft = np.loadtxt("impulse_response_comparison_fft.csv", delimiter="\t
            t_d, K_true_d, K_rec_d = data_dst[:,0], data_dst[:,1], data_dst[:,2]
            t_f, K_true_f, K_rec_f = data_fft[:,0], data_fft[:,1], data_fft[:,2]
            plt.figure()
            plt.plot(t_d, K_true_d, label="K_true")
            plt.plot(t_d, K_rec_d, linestyle="--", label="K_rec (DST)")
            plt.plot(t_f, K_rec_f, linestyle=":", label="K_rec (FFT)")
            plt.legend(); plt.grid(True); plt.xlabel("t"); plt.ylabel("K(t)"); plt.tit
            plt.show()
        else:
            data = np.loadtxt("impulse_response_comparison.csv", delimiter="\t", skipr
            t, K_true, K_rec = data[:,0], data[:,1], data[:,2]
            plt.figure()
            plt.plot(t, K_true, label="K_true")
            plt.plot(t, K_rec, linestyle="--", label="K_rec")
            plt.legend(); plt.grid(True); plt.xlabel("t"); plt.ylabel("K(t)"); plt.tit
            plt.show()

if __name__ == "__main__":
    main()
```

## Python source code - FFT, DST .vs. WAMIT

```
#!/usr/bin/env python3
"""
Unified inversion of RAO R(f) -> impulse response K(t)

Methods:
  dst        : Sine-series (DST-I quadrature) after odd-spectrum correction
  fft        : Jump-removal + IFFT on period 2T (periodic trapezoid)
  filon_si   : Filon on [0,F] (direct) + truncation tail  (K_tail = (K0/π)[π/2 - Si(2π
  filon_exp  : Filon on [0,F] of preconditioned spectrum (subtract K0/(α+i2πf)) + expo
  all        : write all four sets with method-specific suffixes

I/O is TAB-separated to match plotting pipeline.

# JR-FFT (exponential tail)
python invert_rao_to_impulse_both_v2.py --method fft --T 40.3 --N 2048 --alphaT 8 --pl
```

```
# Filon + truncation tail (Si)
python invert_rao_to_impulse_both_v2.py --method filon --T 40.3 --N 2048 --alphaT 8 \
  --fsing 0.5 --eps 2e-3 --plot

# Filon on preconditioned spectrum + exponential tail
python invert_rao_to_impulse_both_v2.py --method filon_exp --T 40.3 --N 2048 --alphaT
  --fsing 0.5 --eps 2e-3 --plot

# All four, side-by-side (writes *_dst/_fft/_filon/_filon_exp CSVs)
python invert_rao_to_impulse_both_v2.py --method all --T 40.3 --N 2048 --alphaT 8 --fs
"""
import argparse
import numpy as np
import scipy.integrate as scipint

try:
    import matplotlib.pyplot as plt
except Exception:
    plt = None

# ---------- Analytic RAO (half-sinc demo) ----------
def A_of_f(ff):
    return 0.5 * (np.abs(ff) <= 0.5).astype(float)

def B_of_f(ff):
    ratio = np.abs((ff + 0.5) / (ff - 0.5 + 0j))
    ratio = np.maximum(ratio, 1e-15)
    return -(1.0/(2.0*np.pi)) * np.log(ratio)

# ---------- Utilities ----------
def estimate_K0_from_tail(fpos, Bpos):
    m = max(10, int(0.1 * fpos.size))
    tail_idx = np.arange(fpos.size - m, fpos.size)
    return float(np.median(-2.0 * np.pi * fpos[tail_idx] * Bpos[tail_idx]))

def save_standard(rao, time, prefix=""):
    f, A, B, S = rao
    t, K_true, K_rec, K_hat = time
    dlm = "\t"
    if prefix:
        np.savetxt(f"rao_samples_{prefix}.csv", np.column_stack([f, A, B]), delimiter=
                   header="f\tA(f)\tB(f)", comments="")
        np.savetxt(f"odd_spectrum_corrected_{prefix}.csv", np.column_stack([f, S]), de
                   header="f\tS(f) (odd spectrum multiplier)", comments="")
        np.savetxt(f"impulse_response_comparison_{prefix}.csv", np.column_stack([t, K_
                   header="t\tK_true\tK_rec\tK_hat", comments="")
    else:
        np.savetxt("rao_samples.csv", np.column_stack([f, A, B]), delimiter=dlm,
```

```
                header="f\tA(f)\tB(f)", comments="")
        np.savetxt("odd_spectrum_corrected.csv", np.column_stack([f, S]), delimiter=dl
                header="f\tS(f) (odd spectrum multiplier)", comments="")
        np.savetxt("impulse_response_comparison.csv", np.column_stack([t, K_true, K_re
                header="t\tK_true\tK_rec\tK_hat", comments="")

# ---------- Sine/DST inversion [0,T]----------
def invert_dst(T, N, alpha, K0_est):
    # positive sine grid
    n = np.arange(1, N)
    f_pos = n / (2.0 * T)
    w_pos = 2.0 * np.pi * f_pos
    A_pos = A_of_f(f_pos); B_pos = B_of_f(f_pos)
    # odd-spectrum diagnostic
    S_pos = 2.0 * B_pos + (4.0 * np.pi * K0_est * f_pos) / (alpha**2 + w_pos**2)
    # discrete sine inversion (matrix formulation)
    df = 1.0/(2.0*T)
    j = np.arange(1, N)
    sin_matrix = np.sin(np.pi * np.outer(j, n) / N)
    K_hat_interior = -2.0 * df * (sin_matrix @ S_pos)
    # assemble \hat K on [0,T]
    t_full = np.linspace(0.0, T, N+1)
    K_hat = np.zeros_like(t_full); K_hat[1:-1] = K_hat_interior
    # restore causal
    K_rec = K_hat + K0_est * np.exp(-alpha * t_full)
    K_true = np.sinc(t_full)
    return (f_pos, A_pos, B_pos, S_pos), (t_full, K_true, K_rec, K_hat)

# ---------- FFT inversion on period 2T ----------
def invert_fft(T, N, alpha, K0_est):
    # FFT grid over 2T with M=2N so dt matches DST
    M = 2 * N; L = 2.0 * T
    df_fft = 1.0 / L; dt_fft = L / M
    k = np.arange(-M//2, M//2); f_fft = k * df_fft; w_fft = 2.0 * np.pi * f_fft
    # analytic RAO on FFT grid and full subtraction
    R_fft = A_of_f(f_fft) + 1j * B_of_f(f_fft)
    R_tilde = R_fft - K0_est / (alpha + 1j * w_fft)
    # inverse FFT with integral scaling Δf
    F_shifted = np.fft.ifftshift(R_tilde)
    K_hat_periodic = np.fft.ifft(F_shifted) * M * df_fft
    K_hat_periodic = K_hat_periodic.real
    # restrict to [0,T]
    t_full = np.arange(M) * dt_fft
    idx_half = np.arange(0, N+1)
    t_half = t_full[idx_half]; K_hat_half = K_hat_periodic[idx_half]
    # restore causal
    K_rec = K_hat_half + K0_est * np.exp(-alpha * t_half)
    K_true = np.sinc(t_half)
```

```
    # For RAO/S(f) exports we still use the positive sine grid
    n = np.arange(1, N); f_pos = n / (2.0 * T); w_pos = 2.0 * np.pi * f_pos
    A_pos = A_of_f(f_pos); B_pos = B_of_f(f_pos)
    S_pos = 2.0 * B_pos + (4.0 * np.pi * K0_est * f_pos) / (alpha**2 + w_pos**2)
    return (f_pos, A_pos, B_pos, S_pos), (t_half, K_true, K_rec, K_hat_half)

# ---------- Filon building blocks ----------
def filon_block_uniform(f, g, t):
    # composite quadratic Filon on uniform f-grid for ∫ g(f) e^{i2π f t} df
    Np = len(f); Nint = Np - 1; assert Nint % 2 == 0
    h = f[1] - f[0]; M = Nint // 2
    y0 = g[0:-2:2]; y1 = g[1:-1:2]; y2 = g[2::2]
    a = y1; b = (y2 - y0)/(2*h); c = (y0 - 2*y1 + y2)/(2*h**2)
    fmid = f[1:-1:2]
    mu = 2*np.pi*t
    if abs(mu) < 1e-12:
        I0, I1, I2 = 2*h, 0j, 2*h**3/3
    else:
        th = mu*h
        I0 = 2*np.sin(th)/mu
        I1 = 2j*(-h*np.cos(th)/mu + np.sin(th)/mu**2)
        I2 = 2*(h**2*np.sin(th)/mu + 2*h*np.cos(th)/mu**2 - 2*np.sin(th)/mu**3)
    return np.sum(np.exp(1j*mu*fmid) * (a*I0 + b*I1 + c*I2))

def Si(x):
    # sine integral Si(x) with a simple numeric fallback
    try:
        import mpmath as mp
        return float(mp.si(x))
    except Exception:
        x = float(x)
        if x > 50: return np.pi/2
        u = np.linspace(0, x, 4001)
        u[0] = 1e-16
        return float(scipint.trapezoid(np.sin(u)/u, u))

# ---------- Filon (direct) + truncation tail ----------
def invert_filon_si(T, N, alpha, K0_est, fsing=None, eps=2e-3, F=None):
    # freq grid for Filon: use the same positive sine grid -> uniform Δf = 1/(2T)
    n = np.arange(1, N); f_pos = n/(2.0*T)
    if F is None: F = f_pos[-1]
    A_pos = A_of_f(f_pos); B_pos = B_of_f(f_pos)
    # Optional gap around a known singular frequency
    if fsing is not None:
        mask1 = f_pos <= (fsing - eps)
        mask2 = f_pos >= (fsing + eps)
        segs = [(f_pos[mask1], A_pos[mask1] + 1j*B_pos[mask1]),
                (f_pos[mask2], A_pos[mask2] + 1j*B_pos[mask2])]
```

```
    else:
        segs = [(f_pos, A_pos + 1j*B_pos)]
    # time grid (same as others)
    t_full = np.linspace(0.0, T, N+1); K_true = np.sinc(t_full)
    K_rec = np.zeros_like(t_full)
    K_hat_dummy = np.zeros_like(t_full)  # placeholder for CSV compatibility
    for j, tj in enumerate(t_full):
        if j == 0:
            # right limit at t=0
            I = 0j
            for fseg, Xseg in segs:
                if len(fseg) >= 3:
                    if (len(fseg)-1) % 2 == 1:  # even intervals required
                        fseg, Xseg = fseg[:-1], Xseg[:-1]
                    I += filon_block_uniform(fseg, Xseg, tj)
            # truncation tail from 1/f asymptotic: K_tail = (K0/π)(π/2 - Si(2π F t))
            K_tail = (K0_est/np.pi) * 0.5*np.pi  # at t=0: π/2
            K_rec[j] = 2*np.real(I) + K_tail
            continue
        I = 0j
        for fseg, Xseg in segs:
            if len(fseg) >= 3:
                if (len(fseg)-1) % 2 == 1:
                    fseg, Xseg = fseg[:-1], Xseg[:-1]
                I += filon_block_uniform(fseg, Xseg, tj)
        K_tail = (K0_est/np.pi) * (np.pi/2 - Si(2*np.pi*F*tj))
        K_rec[j] = 2*np.real(I) + K_tail
    # S(f) diag (same as others)
    w_pos = 2.0*np.pi*f_pos
    S_pos = 2.0*B_pos + (4.0*np.pi*K0_est*f_pos)/(alpha**2 + w_pos**2)
    return (f_pos, A_pos, B_pos, S_pos), (t_full, K_true, K_rec, K_hat_dummy)

# ---------- Filon on preconditioned spectrum + exponential tail ----------
def invert_filon_exp(T, N, alpha, K0_est, fsing=None, eps=2e-3, F=None):
    n = np.arange(1, N); f_pos = n/(2.0*T); w_pos = 2.0*np.pi*f_pos
    if F is None: F = f_pos[-1]
    R = A_of_f(f_pos) + 1j*B_of_f(f_pos)
    Rsub = K0_est / (alpha + 1j*2.0*np.pi*f_pos)
    Rc = R - Rsub
    if fsing is not None:
        mask1 = f_pos <= (fsing - eps); mask2 = f_pos >= (fsing + eps)
        segs = [(f_pos[mask1], Rc[mask1]), (f_pos[mask2], Rc[mask2])]
    else:
        segs = [(f_pos, Rc)]
    t_full = np.linspace(0.0, T, N+1); K_true = np.sinc(t_full)
    K_rec = np.zeros_like(t_full); K_hat_partial = np.zeros_like(t_full)
    for j, tj in enumerate(t_full):
        I = 0j
```

```
        for fseg, Xseg in segs:
            if len(fseg) >= 3:
                if (len(fseg)-1) % 2 == 1:
                    fseg, Xseg = fseg[:-1], Xseg[:-1]
                I += filon_block_uniform(fseg, Xseg, tj)
        K_hat_partial[j] = 2*np.real(I)
        K_rec[j] = K_hat_partial[j] + K0_est*np.exp(-alpha*tj)
    S_pos = 2.0*B_of_f(f_pos) + (4.0*np.pi*K0_est*f_pos)/(alpha**2 + w_pos**2)
    return (f_pos, R.real, R.imag, S_pos), (t_full, K_true, K_rec, K_hat_partial)

def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("--method", choices=["dst","fft","filon_si","filon_exp","all"], de
    ap.add_argument("--T", type=float, default=40.3)
    ap.add_argument("--N", type=int, default=2048)
    ap.add_argument("--alphaT", type=float, default=8.0, help="alpha*T; alpha = alphaT
    ap.add_argument("--fsing", type=float, default=None, help="optional singular frequ
    ap.add_argument("--eps", type=float, default=2e-3, help="half-gap around fsing to
    ap.add_argument("--plot", action="store_true")
    args = ap.parse_args()

    T, N = args.T, args.N
    alpha = args.alphaT / T

    # Common RAO on positive sine grid (uniform Δf) for K0 estimate
    n = np.arange(1, N); f_pos = n/(2.0*T); Bpos = B_of_f(f_pos)
    K0_est = estimate_K0_from_tail(f_pos, Bpos)

    if args.method in ("dst","all"):
        rao_d, time_d = invert_dst(T, N, alpha, K0_est)
        save_standard(rao_d, time_d, prefix="dst" if args.method=="all" else "")
    if args.method in ("fft","all"):
        rao_f, time_f = invert_fft(T, N, alpha, K0_est)
        save_standard(rao_f, time_f, prefix="fft" if args.method=="all" else "")
    if args.method in ("filon_si","all"):
        rao_s, time_s = invert_filon_si(T, N, alpha, K0_est, fsing=args.fsing, eps=arg
        save_standard(rao_s, time_s, prefix="filon" if args.method=="all" else "")
    if args.method in ("filon_exp","all"):
        rao_e, time_e = invert_filon_exp(T, N, alpha, K0_est, fsing=args.fsing, eps=ar
        save_standard(rao_e, time_e, prefix="filon_exp" if args.method=="all" else "")

    if args.plot and plt is not None:
        if args.method == "all":
            # Overlay K_rec from all available outputs
            def load(name):
                import numpy as np
                return np.loadtxt(name, delimiter="\t", skiprows=1)
            data_dst  = load("impulse_response_comparison_dst.csv")
```

```
    data_fft  = load("impulse_response_comparison_fft.csv")
    data_si   = load("impulse_response_comparison_filon.csv")
    data_exp  = load("impulse_response_comparison_filon_exp.csv")
    plt.figure()
    plt.plot(data_fft[:,0], data_fft[:,1], label="K_true")
    plt.plot(data_dst[:,0], data_dst[:,2], "--", label="DST")
    plt.plot(data_fft[:,0], data_fft[:,2], ":", label="JR-FFT")
    plt.plot(data_si[:,0],  data_si[:,2],  "-.", label="Filon+Si-tail")
    plt.plot(data_exp[:,0], data_exp[:,2], "-",  label="Filon+exp-tail")
    plt.xlabel("t"); plt.ylabel("K(t)"); plt.title("All methods"); plt.legend(

if __name__ == "__main__":
    main()
```

---

## Why does the FFT method show slightly better accuracy here?

A few practical reasons specific to our setup:

1. **Quadrature rule & periodicity.**
    - The **FFT path** inverts the transform over a **period** $2T$ using the **periodic trapezoidal rule**, which is spectrally (super-algebraically) accurate for smooth periodic integrands. After the exponential subtraction, $\widetilde{R}(f)$ is smooth and periodic on our grid, so the trapezoid rule is extremely effective.
    - The **DST path** evaluates a **half-line integral** $0 to inf t$ via a **rectangular Riemann sum** on $f_n = n/(2T)$ and truncates at $a$. Even with the $1/f^3$ tail, this has an $O(\Delta f)$ (or $O(\Delta f^2)$ at best) quadrature error plus a truncation error $O(a^{-2})$.
2. **Using the full complex spectrum vs. imaginary-only.**
    - In exact continuous theory, using $B(f) = \Im R$ only (with the rational correction) equals using the full $R(f)$.
    - Discretely, the **full FFT** uses both $\Re R$ and $\Im R$ on a grid that is *exactly* dual to the IFFT, reducing discretization mismatch. The DST route discards $\Re R$ and relies on one-sided quadrature, which is slightly less balanced numerically.
3. **Grid consistency & scaling.**
    - The FFT uses frequencies $f_k = k/(2T)$ with $M = 2N$ samples, so $\Delta f = 1/(2T)$ and $\Delta t = 2T/M = T/N$, matching the sine-series time step. Crucially, the IFFT

gives the **exact** discrete inverse on that grid (up to floating-point), whereas our DST code performs a **numerical integral** (not an orthonormal DST-I/II with exact quadrature).

4. **Singularity handling.**
    - $B(f)$ has logarithmic singularities at $f = \pm 1/2$. By choosing $T$ so the grid avoids sampling exactly at $\pm 1/2$, the FFT sees a smooth periodic function. The DST's one-sided Riemann sum still feels more of the near-singularity curvature on one side, which can slightly raise the error.

**Bottom line.** With our choices ($T = 40.3$, $N = 2048$, $\alpha T \approx 8$), the FFT path showed:

- DST: $|K_{\text{rec}} - K_{\text{true}}|_\infty \approx 6.53 \times 10^{-3}$, RMS $\approx 3.29 \times 10^{-3}$
- FFT: $|K_{\text{rec}} - K_{\text{true}}|_\infty \approx 3.09 \times 10^{-3}$, RMS $\approx 1.76 \times 10^{-3}$

Note: tThis doesn't mean FFT is *always* better; accuracy also depends on $T$, $N$, $\alpha$, and how the integrals are discretized. If ones needs the DST to match the FFT more closely, these suggestions may help:

.* switch to a true **DST-I** with orthogonal weights (or use Clenshaw–Curtis / Filon-type quadrature for oscillatory integrals)
2. slightly increase $N$ or use **Richardson extrapolation** on $N$;
3. adjust $\alpha$ so that $e^{-\alpha T}$ is smaller, reducing endpoint mismatch further.

# DST and FFT Discretizations and index ranges.

## 0) Continuous setup and goal

We use the Fourier pair

$$R(f) = \int_{-\infty}^{\infty} K(t) e^{-i2\pi f t} \mathrm{d}t, quad K(t) = \int_{-\infty}^{\infty} R(f) e^{i2\pi f t} \mathrm{d}f.$$

For a **real, causal** impulse response $K(t)$, write $R(f) = A(f) + iB(f)$ with $A$ even and $B$ odd. If $K$ has a jump $K_0 = K(0^+)$, then

$$\Im R(f) = B(f) \sim -\frac{K_0}{2\pi f} \quad (|f| \to \infty).$$

To accelerate inversion we remove the jump by subtracting an exponential and invert the **modified** signal:

$$\hat{K}(t) := K(t) - K_0 e^{-\alpha t} u(t), \quad \hat{R}(f) := R(f) - \frac{K_0}{\alpha + i2\pi f}.$$

- In the **sine/DST** route we only need the **odd spectrum** (purely imaginary):

$$\hat{R}_{\text{odd}}(f) = 2iB(f) + i\frac{4\pi K_0 f}{\alpha^2 + (2\pi f)^2}.$$

  Define the real multiplier

$$S(f) := 2B(f) + \frac{4\pi K_0 f}{\alpha^2 + (2\pi f)^2}.$$

- In the **FFT** route we invert the **full complex** $\hat{R}(f)$.

## 1) Grids and symbols (one notation for both methods)

Let

- $T > 0$ = time window where we reconstruct $K(t)$;
- $M$ = number of **interior** time nodes in $(0, T)$;
- interior time nodes

$$t_j = \frac{jT}{M+1}, \qquad j = 1, 2, \ldots, M;$$

- "natural" positive sine frequencies

$$f_n = \frac{n}{2T}, \qquad n = 1, 2, \ldots, M;$$

so $2\pi f_n t_j = \dfrac{jn\pi}{M+1}$ and $\Delta f = \dfrac{1}{2T}$.

> **Code mapping.** In the code we often choose a number of **subintervals** $N$ on $[0, T]$. Then $M = N - 1$ (there are $N - 1$ interior nodes), and $M + 1 = N$.
> That is why we see $\sin(jn\pi/N)$ with $j, n = 1, \ldots, N - 1$.

## 2) Discrete Sine Transform (DST-I) on $[0, T]$

## 2.1 Kernel and orthogonality

The DST-I kernel on the interior grid is

$$\sin\left(\frac{\pi jn}{M+1}\right), \qquad j,n = 1,\dots,M,$$

with discrete orthogonality

$$2\sum_{j=1}^{M} \cos\frac{\pi jm}{M+1}\cos\frac{\pi jn}{M+1} = (M+1)\delta_{mn}(1+\delta_{m0})$$
$$2\sum_{j=1}^{M} \sin\frac{\pi jm}{M+1}\sin\frac{\pi jn}{M+1} = (M+1)\delta_{mn}.$$

## 2.2 Transform pair (algebraic form)

With **1-based** indexing:

$$\widehat{x}_n = \sum_{j=1}^{M} x_j \sin\left(\frac{\pi jn}{M+1}\right), \quad x_j = \frac{2}{M+1}\sum_{n=1}^{M} \widehat{x}_n \sin\left(\frac{\pi jn}{M+1}\right).$$

With **0-based** indexing $k = j-1$, $n' = n-1$, $k, n' = 0,\dots,M-1$:

$$X_k = \sum_{n'=0}^{M-1} x_{n'} \sin\left(\frac{\pi(n'+1)(k+1)}{M+1}\right).$$

These are **the same** transform, just different index labels.

## 2.3 How the inversion appears in the code (quadrature form)

We approximate the continuous sine inversion

$$\hat{K}(t) = -2\int_0^\infty S(f)\sin(2\pi ft)\mathrm{d}f$$

by the **Riemann sum** on the natural sine grid:

$$\boxed{\hat{K}(t_j) \approx -2\Delta f \sum_{n=1}^{M} S(f_n) \sin\Big(\frac{\pi j n}{M+1}\Big), \quad \Delta f = \frac{1}{2T}.}$$

This uses the **same** sine kernel as DST-I; the extra factor $-2\Delta f$ comes from **quadrature** (integral approximation of $\mathrm{d}f$), not from the algebraic transform normalization.

> **Practical tip.** One can replace the explicit sum by a library **DST-I** (e.g. SciPy `dst(type=1)`) and multiply by the quadrature weight $2\Delta f$ to match the integral scaling; this is often faster and more accurate.

---

## 3) Periodic FFT on the doubled window

### 3.1 Periodization and grids

We treat $\hat{R}(f)$ as sampled on a **periodic** frequency grid of period

$$L := 2T \qquad \Delta f_{\mathrm{FFT}} = \frac{1}{L} = \frac{1}{2T},$$

with $M_{\mathrm{FFT}}$ samples (in code we pick $M_{\mathrm{FFT}} = 2N = 2(M+1)$ so $\Delta t$ matches the DST). The time step is

$$\Delta t_{\mathrm{FFT}} = \frac{L}{M_{\mathrm{FFT}}} = \frac{2T}{M_{\mathrm{FFT}}}.$$

Frequencies are $f_k = \dfrac{k}{L}$ (via `fftfreq + fftshift` / `ifftshift`).

### 3.2 Full complex subtraction and IFFT scaling

We build

$$\hat{R}(f) = R(f) - \frac{K_0}{\alpha + i2\pi f} \quad \text{on the FFT grid},$$

and approximate

$$\hat{K}(t_n) \approx \sum_k \hat{R}(f_k), e^{i2\pi f_k t_n} \Delta f_{\mathrm{FFT}}.$$

If `ifft` implements $\frac{1}{M_{\mathrm{FFT}}} \sum_k X_k e^{i2\pi kn/M_{\mathrm{FFT}}}$, we pass

$$X_k = M_{\mathrm{FFT}} \Delta f_{\mathrm{FFT}} \hat{R} f_k,$$

so that `ifft(X)` returns $\hat{K}(t_n)$ at the FFT time nodes $t_n = n\Delta t_{\mathrm{FFT}}$. Finally,

$$K(t) = \hat{K}(t) + K_0 e^{-\alpha t}.$$

> **Interpretation.** This is the **periodic trapezoidal rule** on the frequency torus of length $L = 2T$. After the exponential subtraction, $\hat{R}$ is smooth enough that the trapezoidal rule is highly accurate.

---

## 4) Why the "natural" sine frequencies are $f_n = \dfrac{n}{2T}$

Sine modes for the odd $2T$-periodic extension are

$$\sin\left(\frac{n\pi t}{T}\right) = \sin\left(2\pi f_n t\right), \qquad f_n = \frac{n}{2T}.$$

Evaluating them at the **interior** grid $t_j = \dfrac{jT}{M+1}$ gives

$$2\pi f_n t_j = \frac{n\pi}{T} \cdot \frac{jT}{M+1} = \frac{\pi jn}{M+1},$$

which is exactly the DST-I kernel angle. That is why $\Delta f = \dfrac{1}{2T}$ and $\Delta t = \dfrac{T}{M+1}$ are the "natural" spacings.

---

## 5) Putting it together with the code variables

- In the **DST** code:
    - we set the number of **subintervals** $N$, hence $M = N - 1$;

- time nodes: $t_j = jT/N, j = 1, \dots, N-1$;
- sine frequencies: $f_n = n/(2T), n = 1, \dots, N-1$;
- kernel: $\sin\left(\pi jn/N\right)$ (DST-I with $M = N-1 \Rightarrow M+1 = N$);
- inversion: $\hat{K}(t_j) \approx -2\Delta f \sum_n S(f_n)\sin(\pi jn/N)$;
- restore: $K(t) = \hat{K}(t) + K_0 e^{-\alpha t}$.

- In the **FFT** code:
  - period $L = 2T$, choose $M_{\mathrm{FFT}} = 2N$;
  - frequency grid: $f_k = k/L$, $\Delta f = 1/L = 1/(2T)$;
  - compute $\hat{R}(f_k) = R(f_k) - K_0/(\alpha + i2\pi f_k)$;
  - inverse: `ifft(` $M_{\mathrm{FFT}}\Delta f$ `)*ifftshift(` $\hat{R}$ `)` gives $\hat{K}$;
  - restrict to $t \in [0, T]$, then $K(t) = \hat{K}(t) + K_0 e^{-\alpha t}$.

---

## 6) Normalization cheat-sheet

- **DST-I algebraic pair**

$$\widehat{x}_n = \sum_{j=1}^{M} x_j \sin\left(\frac{\pi jn}{M+1}\right), \quad x_j = \frac{2}{M+1}\sum_{n=1}^{M} \widehat{x}_n \sin\left(\frac{\pi jn}{M+1}\right).$$

**Orthogonality**

$$x_j = \frac{1}{M+1}\sum_{k=1}^{M} x_k \sum_{n=1}^{M}\left(\cos\frac{\pi(k-j)n}{M+1} - \cos\frac{\pi(k+j)n}{M+1}\right)$$

$S_{kj} = \sum_{n=1}^{M}\left(\cos\frac{\pi(k-j)n}{M+1} - \cos\frac{\pi(k+j)n}{M+1}\right)$

**The cosine summation formula**:

$$\sum_{n=1}^{M}\cos(n\theta) = \frac{\sin\left((M+\frac{1}{2})\theta\right)}{2\sin(\theta/2)} - \frac{1}{2}.$$

- If $k = j$,

$$\sum_{n=1}^{M}\cos\frac{\pi(k-j)n}{M+1} = \sum_{n=1}^{M} 1 = M, \quad \sum_{n=1}^{M}\cos\frac{\pi(k+j)n}{M+1} = \sum_{n=1}^{M}\cos\frac{2\pi jn}{M+1} = -1$$

$$\therefore\ S_{jj} = M - (-1) = M+1.$$

- If $k \neq j$, let $a = k - j, b = k + j$. Since $k, j$ are integers, $a$ and $b$ has the same parity:
  - When $a$ is even (so is $b$):

$$\sum_{n=1}^{M} \cos \frac{\pi a n}{M+1} = -1, \quad \sum_{n=1}^{M} \cos \frac{\pi b n}{M+1} = -1 \implies S_{kj} = (-1) - (-1) = 0$$

  - When $a$ is odd (so is $b$):

$$\sum_{n=1}^{M} \cos \frac{\pi a n}{M+1} = 0, \quad \sum_{n=1}^{M} \cos \frac{\pi b n}{M+1} = 0 \implies S_{kj} = 0 - 0 = 0$$

$$\therefore\ 2 \sum_{n=1}^{M} \sin \tfrac{\pi j n}{M+1} \sin \tfrac{\pi k n}{M+1} = (M+1)\delta_{kj},$$

$$x_j = \frac{1}{M+1} \sum_{k=1}^{M} x_k (M+1)\delta_{kj} = x_j$$

- **Sine integral quadrature** (what the code uses)

$$\hat{K}(t_j) \approx -2\Delta f \sum_{n=1}^{M} S(f_n) \sin\left(\frac{\pi j n}{M+1}\right), \quad \Delta f = \frac{1}{2T}.$$

- **IFFT scaling for the inverse integral**

$$\hat{K}(t_n) \approx \sum_{k} \hat{R}(f_k) e^{i2\pi f_k t_n} \Delta f \quad \Longrightarrow \quad \texttt{ifft}\left(M_{\mathrm{FFT}} \Delta f \hat{R}\right).$$

---

## 7) Accuracy notes (why FFT may look better)

- The **FFT** implements the **periodic trapezoidal rule** on the frequency torus; after subtracting the pole, $\hat{R}$ is smooth and the trapezoid rule is spectrally accurate.
- The **DST** line uses a straightforward **Riemann sum** for a half-line oscillatory integral: it is accurate and convergent (thanks to $S(f) = O(f^{-3})$) but typically not as sharp as the periodic trapezoid rule unless a higher-order oscillatory quadrature or a true orthonormal DST with proper weights are used.

Both routes converge to the same answer; the small observed differences are discretization/normalization effects, not theory.

---

## 8) Minimal “dictionary” between the two notations

- If a reference shows

$$X_k = \sum_{n=0}^{M-1} x_n \sin\left(\frac{\pi(n+1)(k+1)}{M+1}\right),$$

  then set $j = k+1$, $n \leftarrow n+1$ and $M+1 \leftarrow N$ to get

$$\sum_{n=1}^{N-1} x_n \sin\left(\frac{\pi j n}{N}\right),$$

  which is exactly the kernel used in the code (with $N = M+1$).

---

### Remarks about the indices

- Use $M$=**interior points**, $t_j = jT/(M+1)$, $f_n = n/(2T)$, kernel $\sin(jn\pi/(M+1))$.
- **DST path**: evaluate $\hat{K}$ by a sine-weighted Riemann sum on $f_n$, then add back $K_0 e^{-\alpha t}$.
- **FFT path**: subtract the full rational term in frequency, apply the IFFT with $M_{\mathrm{FFT}}\Delta f$ scaling, restrict to $[0, T]$, and add back $K_0 e^{-\alpha t}$.

## Logarithmic Frequency Sampling(NOT implemented)

Logarithmic frequency sampling is an efficient strategy for calculating impulse response functions (IRFs) or performing frequency domain analysis, especially suitable for scenarios where the system response changes slowly in the low-frequency region and decays rapidly in the high-frequency region (e.g., frequency responses in vibration systems, acoustic or electromagnetic models). Its core idea is to use denser sampling points in the low-frequency region (to capture resonance and slowly varying characteristics) and sparser sampling points in the high-frequency region (utilizing the smooth decay characteristics of the high-frequency response), thereby reducing the total number of sampling points while maintaining accuracy. Below, I will demonstrate the effectiveness of this method step by step from a theoretical perspective.

- The disadvantages of linear sampling: If equally spaced linear sampling is performed on the frequency axis (e.g., $\Delta\omega$ = constant), extremely high sampling density is required to capture

high-frequency details, resulting in a huge computational burden (the number of points is proportional to the frequency range).

- The advantages of logarithmic sampling: Applying a logarithmic scale to the frequency axis (i.e., sampling points are distributed on $\log(\omega)$ makes:
  - Low-frequency region: $\Delta\omega$ is small, sampling is dense, and formants or slowly changing phases can be accurately captured.
  - High-frequency region: $\Delta\omega$ is large, sampling is sparse. Because the high-frequency response often exhibits smooth decay (e.g., in the form of $\omega^{-n}$), sparse sampling can still approximate the frequency response.

## Impulse Response Function Definition

### WAMIT Definition

$$\bar{F}_c(\omega) = \omega\bar{B}(\omega) + i\omega[\bar{A}(\omega) - \bar{A}(\infty)]?$$

To be consistent with WAMIT 7.5 manual eq.(13.3)-(13.4), the Forward Fourier Transform of the Impulse Response Function (IRF) denoted by $L_{ij}(t)$ with conditions $L(0) = 0,\ L(\infty) \to 0$:

$$[A_{ij}(\omega) - A_{ij}(\infty)] \quad + \quad \frac{B_{ij}(\omega)}{i\omega} = \int_0^\infty L_{ij}(t)e^{-i\omega t}\mathrm{d}t \tag{1}$$

$$= \int_0^\infty L_{ij}(t)\cos(\omega t)\mathrm{d}t - i\int_0^\infty L_{ij}(t)\sin(\omega t)\mathrm{d}t \tag{2}$$

The inverse-transforms of (1) and (2) give complementary relations for the IRF:

$$L_{ij}(t) = \frac{2}{\pi}\int_0^\infty [A_{ij}(\omega) - A_{ij}(\infty)]\cos(\omega t)\mathrm{d}\omega \tag{3}$$

$$L_{ij}(t) = \frac{2}{\pi}\int_0^\infty \frac{B_{ij}(\omega)}{\omega}\sin(\omega t)\mathrm{d}\omega \tag{4}$$

Where the boundary conditions $L_{ij}(0) = 0$ and $L_{ij}(\infty) \to 0$ can be naturally derived.

### SIMA/SIMO Conventions (based on SIMO theory manual 4.20.4, 2021)

In SIMA/SIMO, the IRF is also called retardation function (w.r.t. convolution integral) and is denoted by $h(t)$.

#### 1) SIMA/SIMO Defintions

- Let the **frequency-dependent radiation coefficients** be written

$$A(\omega) = A_\infty + a(\omega), \quad C(\omega) = c(\omega), \quad C_\infty \equiv 0,$$

  with $a(\omega), c(\omega)$ **real** and **even** in $\omega$.
- Define the **retardation (impulse-response) function** $h(\tau)$ and its spectrum

$$H(\omega) = c(\omega) + i\omega a(\omega) = \int_{-\infty}^{\infty} h(\tau) e^{-i\omega\tau} \mathrm{d}\tau, \quad h(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} H(\omega) e^{i\omega\tau} \mathrm{d}\omega \tag{T}$$

- Causality (memory force depends only on the past) implies $h(\tau) = 0$ for $\tau < 0$.
- The radiation equation of motion in time domain is

$$A_\infty \ddot{x}(t) + \int_0^t h(t-\tau)\dot{x}(\tau)\mathrm{d}\tau = f(t).$$

  Which is exactly SIMO (4.11).

**2) From (T) to the cosine/sine formulas (4.15)–(4.16)**

Because $a(\omega)$ and $c(\omega)$ are both **even**, $H(-\omega) = H(\omega)^*$ and $h(\tau)$ is real. Split the inverse transform into cosine and sine parts:

$$\begin{aligned} h(\tau) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} [c(\omega) + i\omega a(\omega)] e^{i\omega\tau} \mathrm{d}\omega \\ &= \frac{1}{\pi} \int_0^{\infty} c(\omega) \cos(\omega\tau) \mathrm{d}\omega - \frac{1}{\pi} \int_0^{\infty} \omega a(\omega) \sin(\omega\tau) \mathrm{d}\omega. \end{aligned} \tag{4.15}$$

Evaluate the same expression at **negative** time $\tau \mapsto -\tau$:

$$h(-\tau) = \frac{1}{\pi} \int_0^{\infty} c(\omega) \cos(\omega\tau) \mathrm{d}\omega + \frac{1}{\pi} \int_0^{\infty} \omega a(\omega) \sin(\omega\tau) \mathrm{d}\omega.$$

Causality gives $h(-\tau) = 0$ for $\tau > 0$. Hence, for every $\tau > 0$,

$$\int_0^{\infty} c(\omega) \cos(\omega\tau) \mathrm{d}\omega = -\int_0^{\infty} \omega a(\omega) \sin(\omega\tau) \mathrm{d}\omega.$$

Insert this into (4.15) then (4.16) is obtained:

$$\boxed{h(\tau) = \frac{2}{\pi} \int_0^{\infty} c(\omega) \cos(\omega\tau) \mathrm{d}\omega = -\frac{2}{\pi} \int_0^{\infty} \omega a(\omega) \sin(\omega\tau) \mathrm{d}\omega, \quad \tau > 0.} \tag{4.16}$$

**3) Inversion formulas (4.17)–(4.18)**

Use the standard one-sided cosine/sine transform pairs (for $\tau > 0$):

- If $f(\tau) = \frac{2}{\pi}\int_0^\infty F(\omega)\cos(\omega\tau)\mathrm{d}\omega$, then $F(\omega) = \int_0^\infty f(\tau)\cos(\omega\tau)\mathrm{d}\tau$.
- If $f(\tau) = \frac{2}{\pi}\int_0^\infty G(\omega)\sin(\omega\tau)\mathrm{d}\omega$, then $G(\omega) = \int_0^\infty f(\tau)\sin(\omega\tau)\mathrm{d}\tau$.

Apply these to (4.16):

- From the **sine** identity (second equality): $G(\omega) = \omega a(\omega) = -\int_0^\infty h(\tau)\sin(\omega\tau)\mathrm{d}\tau$, i.e.

$$\boxed{a(\omega) = -\frac{1}{\omega}\int_0^\infty h(\tau)\sin(\omega\tau)\mathrm{d}\tau.} \tag{4.17}$$

- From the **cosine** identity (first equality):

$$\boxed{c(\omega) = \int_0^\infty h(\tau)\cos(\omega\tau)\mathrm{d}\tau.} \tag{4.18}$$

> ⚠️ Sign error corrected from SIMO theory manual 4.20.4.

At $\tau = 0$ in (4.15) the sine term vanishes, giving:

$$\boxed{h(0) = \frac{1}{\pi}\int_0^\infty c(\omega)\mathrm{d}\omega.} \tag{4.19}$$

Finally, set $\omega = 0$ in (4.18): $c(0) = \int_0^\infty h(\tau)\mathrm{d}\tau$. If (as in potential-flow radiation at zero forward speed) **there is no radiation at zero frequency**, then $c(0) = 0$ and therefore

$$\boxed{\int_0^\infty h(\tau)\mathrm{d}\tau = 0.} \tag{4.20}$$

But how the **zero-frequency** data should be used in numerical inversions is still unknown.

---

**4) "Cummins kernel"**

The Fourier transform of the IRF in the notes is $H(\omega) = c(\omega) + i\omega a(\omega)$.
If we multiply this by $i\omega$ (because the convolution acts on $\dot{x}$), get the **operator spectrum**

$$i\omega H(\omega) = \underbrace{c(\omega)}_{\text{damping}} + i\omega \underbrace{a(\omega)}_{\text{added-mass shift}}.$$

This is the form that appears in the **frequency-domain** equation

$$[i\omega A(\omega) + C(\omega)]i\omega X(\omega) = F(\omega) \tag{4.7}$$

and

$$\boxed{\text{“Cummins operator spectrum”} = \underbrace{c(\omega)}_{\bar{B}} + i\omega \underbrace{a(\omega)}_{\bar{A}-\bar{A}_\infty}.}$$

So:

- $H(\omega) = c + i\omega a$ is the **transform of the IRF** itself.
- $i\omega H(\omega) = c + i\omega a \cdot i\omega$ (with the **plus** in front of $i\omega a$) is the spectrum that multiplies **displacement** in frequency, or **velocity** in time.

Both are standard; they’re just used in two different places in the formulation.

---

**5) Why does $c(0) = \int_0^\infty h(\tau)\mathrm{d}\tau = 0$ hold physically?**

From (4.18) at $\omega = 0$, the integral $c(\omega) = c(0)$. For **radiation** in linear **potential flow** at zero forward speed:

- A body moving at **strictly constant velocity** does **not** radiate progressive waves; there is no energy flux to infinity (no wave resistance). Or in another word, the free surface is constant, independent on time. Hence the **radiation damping at zero frequency vanishes**: $c(0) = 0$.
- In the time domain, take a **step in velocity**: $\dot{x}(t) = V, H(t)$. The radiation force from the memory term is

$$F_{\text{rad}}(t) = \int_0^t h(\tau)V\mathrm{d}\tau = V\int_0^t h(\tau)\mathrm{d}\tau.$$

  At long times the motion has become a steady translation, so $F_{\text{rad}}(t) \to 0$. Therefore

$$\int_0^\infty h(\tau)\mathrm{d}\tau = 0.$$

This is not “energy conservation” in the abstract; it is the **absence of wave radiation at steady translation** in inviscid potential flow. (If the **viscous** effects were included, there will have an effective low-frequency damping; then $\int_0^\infty h \neq 0$.)

---

**6) Practical notes for numerics (AB data)**

- Keep the $\omega = 0$ data.
  – It **anchors** the integral and directly sets $h(0)$ via (4.19).
  – In the sine integral the $\sin(\omega t)$ weight kills the contribution at $\omega = 0$, but the point still stabilizes interpolation.
- Use the **cosine path** (from $\overline{A} - \overline{A}_\infty$) as the *primary* route for $h$; it is typically more stable than the sine path, especially on a finite band. If with the sine path, add a high-frequency **tail model** ($\overline{B} \sim c_B/\omega^3$).

---

**One-line summary**

- The IRF transform is $H(\omega) = c(\omega) + i\omega a(\omega)$.
- The identity $\int_0^\infty h(\tau)\mathrm{d}\tau = 0$ is equivalent to $c(0) = 0$, i.e. **no radiation at zero frequency** (no wave power for steady translation).

---

**1. High frequency truncation**

Starting from (1)-(2), we have

$$\begin{aligned} A(\omega) - A(\infty) &= \int_0^\infty L(t)\cos(\omega t)\mathrm{d}t \\ B(\omega) &= \omega \int_0^\infty L(t)\sin(\omega t)\mathrm{d}t \end{aligned} \tag{5}$$

**About (13.9) in the WAMIT manual**

Integrate (5) by parts once:

$$A(\omega) - A(\infty) = \Big[\frac{L(t)\sin(\omega t)}{\omega}\Big]_0^\infty - \frac{1}{\omega}\int_0^\infty L'(t)\sin(\omega t)\mathrm{d}t = -\frac{1}{\omega}\int_0^\infty L'(t)\sin(\omega t)\mathrm{d}t \tag{15}$$

because the boundary term vanishes $L(0) = 0$ and $L(\infty) \to 0$.
Integrate by parts again in the sine–integral (or use the standard asymptotic for oscillatory integrals) to get, as $\omega \to \infty$,

$$\int_0^\infty L'(t)\sin(\omega t)\mathrm{d}t = \frac{L'(0)}{\omega} - \frac{1}{\omega^2}\int_0^\infty L'''(t)\sin(\omega t)\mathrm{d}t = \frac{L'(0)}{\omega} + O(\omega^{-2}),$$

hence

$$\boxed{A(\omega) - A(\infty) = -\frac{L'(0)}{\omega^2} + O(\omega^{-3})}.$$

So the formula in (13.9) is correct under the assumptions ($L, L', L''$ and $L''' \in L^1$ integrable, and $L(0) = 0$).

> ✅ $L''' \in L^1$ integrable means: $\int_0^\infty |L'''|\mathrm{d}t < \infty \implies L''$ has a finite limit as $t \to \infty$, together with $L'' \in L^1$ this limit must be 0.
> ✅ WAMIT definitions (7) and (8) correspond to the SIMO definitions (4.18) and (4.17), respectively. Either pairs (7) and (8) or (4.18) and (4.17) form the Hilbert transform pair. Therefore, the relation between the two definitions for response function is $L'(t) = h(t)$.

Since the response RAO is

$$H(\omega) = c(\omega) + i\omega a(\omega)$$

the boundary terms after integration by parts will have

- The term $K_0/(i\omega) = -iK_0/\omega$ is **purely imaginary**.
- The real part comes from $-K_1/\omega^2 + o(1/\omega^2)$.

So:

- **Real part**:

$$\Re H(\omega) = c(\omega) = -\frac{K_1}{\omega^2} + o\left(\frac{1}{\omega^2}\right).$$

- **Imaginary part**:

$$\Im H(\omega) = \omega a(\omega) = -\frac{K_0}{\omega} + o\left(\frac{1}{\omega}\right).$$

Therefore, we have the rigorous asymptotics:

$$\boxed{a(\omega) \sim \frac{C_a}{\omega^2}, \quad c(\omega) \sim \frac{C_c}{\omega^2}, \qquad |\omega| \to \infty,}$$

with $C_a = -K_0 = -h(0^+), \quad C_c = -K_1 = -h'(0^+).$

**2. Why the "$\mathrm{Si}$ tail" is only $\sim K_0/2$ at $t = 0$, yet it "has the same effect" as the JR–FFT exponential tail**

Let $R(f) = A(f) + iB(f)$ be a general complex RAO and suppose the time-domain impulse response $K(t)$ has a **jump** $K_0 = K(0^+)$. Then

$$B(f) = \Im R(f) \sim -\frac{K_0}{2\pi f} \quad (|f| \to \infty).$$

There are **two equivalent ways** to handle this when inverting numerically.

---

## A. JR–FFT (jump removal + exponential)

Subtract the simple pole *for all frequencies*:

$$R_{\mathrm{sub}}(f) = \frac{K_0}{\alpha + i2\pi f}, \quad \widetilde{R}(f) = R(f) - R_{\mathrm{sub}}(f).$$

Invert $\widetilde{R}$ on a period $2T$ by IFFT (periodic trapezoid). Then **add back the exact time inverse** of $R_{\mathrm{sub}}$:

$$\boxed{K(t) = \underbrace{2\Re \int_0^\infty \widetilde{R}(f) e^{i2\pi f t} \mathrm{d}f}_{\text{by IFFT}} + \underbrace{K_0 e^{-\alpha t}}_{\text{exponential tail}}}.$$

Note tThis "exponential tail" is *not* a truncation correction as in WAMIT approach; it is the full time-domain contribution of the subtracted rational term.

---

## B. Filon + truncation tail on $[0, \Omega]$ (WAMIT approach, Lee–Newman)

Integrate the **unsubtracted** integrand only up to $\Omega$, and add the **closed-form contribution of the missing high frequencies**. Approximating the tail by the dominant $-iK_0/(2\pi f)$ term gives, for cycles-per-second $f$,

$$\boxed{K(t) \approx 2\Re \int_0^\Omega R(f) e^{i2\pi f t} \mathrm{d}f + \frac{K_0}{\pi}\left(\frac{\pi}{2} - \mathrm{Si}(2\pi\Omega t)\right)}.$$

At $t \downarrow 0$, $\mathrm{Si}(0) = 0$, hence the tail tends to $K_0/2$. This is **not a contradiction**: for a discontinuity, the Fourier inversion of the unsubtracted signal converges (pointwise at the jump) to the **half-sum** of the left/right limits. The *high-frequency* part contributes $K_0/2$; the *low-frequency* part contributes the rest. So the $\mathrm{Si}$ tail is only "half of the jump" at $t = 0$ because it represents only the **missing high-frequency piece**.

---

## How the two are actually equivalent (for $t > 0$)

Write $R = \widetilde{R} + R_{\mathrm{sub}}$ inside the Filon formula:

$$
\begin{aligned}
&2\Re \int_0^{\Omega} R e^{i2\pi f t} \mathrm{d}f + \frac{K_0}{\pi}\left(\tfrac{\pi}{2} - \mathrm{Si}(2\pi\Omega t)\right) \\
&= 2\Re \int_0^{\Omega} \widetilde{R} e^{i2\pi f t} \mathrm{d}f + \underbrace{\left[2\Re \int_0^{\Omega} R_{\mathrm{sub}} e^{i2\pi f t} \mathrm{d}f + \frac{K_0}{\pi}\left(\tfrac{\pi}{2} - \mathrm{Si}(2\pi\Omega t)\right)\right]}_{\star}.
\end{aligned}
$$

For the simple pole $R_{\mathrm{sub}}$ one has the exact identity

$$
\star = K_0 e^{-\alpha t} - 2\Re \int_{\Omega}^{\infty} \left(R_{\mathrm{sub}}(f) - \frac{-iK_0}{2\pi f}\right) e^{i2\pi f t} \mathrm{d}f,
$$

and the remainder (the term in the last integral) is $O(\Omega^{-2})$ uniformly on bounded $t$.
Hence, as $\Omega \to \infty$,

$$
\boxed{2\Re \int_0^{\Omega} R e^{i2\pi f t} df + \frac{K_0}{\pi}\left(\tfrac{\pi}{2} - \mathrm{Si}(2\pi\Omega t)\right) \longrightarrow 2\Re \int_0^{\infty} \widetilde{R} e^{i2\pi f t} df + K_0 e^{-\alpha t}.}
$$

That is exactly the **JR–FFT** formula. In words:

- The $\mathrm{Si}$ **tail** supplies the **high-frequency part** of the step ($\approx K_0/2$ at $t = 0$), complementing the truncated **low-frequency** integral.
- The **exponential tail** $K_0 e^{-\alpha t}$ is the **full** time-domain contribution of the subtracted simple pole across **all** frequencies.
- Once it was accounted for the low-frequency portion of the pole inside $0 \le f \le \Omega$, the two reconstructions agree (and we observed this numerically).

## TL;DR

- **(13.9)** is correct: $A(\omega) - A(\infty) = -\omega^{-1} \int_0^\infty L'(t) \sin(\omega t) \mathrm{d}t \sim -L'(0)\omega^{-2}$.
- The $\mathrm{Si}$ **tail** is only "half the jump" at $t = 0$ because it represents the **missing high-frequency part** of the unsubtracted inversion, which converges to the **half-sum** at a discontinuity.
- **JR–FFT** adds the **full exponential** because it subtracts the simple pole **everywhere in frequency** and must therefore add its **entire** time-domain inverse back.
- For $t > 0$, **Filon +** $\mathrm{Si}$ **tail** and **JR–FFT** are equivalent up to $O(F^{-2})$ and match in practice; they only differ in the conventional value right at the discontinuity.

UAbsolutely—let me unpack that identity and showcarefully and show exactly how it connects the **Filon + Si-tail** formula to the **JR-FFT (jump-removal + exponential)** formula.

---

## Setup and the two "tails"

Work in **cycles per second** $f$ with the pair

$$R(f) = \int_{-\infty}^{\infty} K(t) e^{-i2\pi f t} \mathrm{d}t \quad K(t) = \int_{-\infty}^{\infty} R(f) e^{i2\pi f t} \mathrm{d}f = 2\Re \int_0^{\infty} R(f) e^{i2\pi f t} \mathrm{d}f.$$

Assume $K(t)$ has a jump $K_0 = K(0^+)$. Then $\Im R(f) \sim -K_0/(2\pi f)$ as $|f| \to \infty$.

- **JR-FFT tail** (global subtraction):
  Define the *simple pole*

  $$R_{\mathrm{sub}}(f) := \frac{K_0}{\alpha + i2\pi f},$$

  whose exact time inverse is $K_0 e^{-\alpha t} u(t)$. In JR-FFT we subtract $R_{\mathrm{sub}}$ **for all** $f$ and then add back $K_0 e^{-\alpha t}$.

- **Filon+Si truncation tail** (finite cut–off):
  If the integration only goes up to $\Omega$, the missing high-frequency contribution of the $1/f$ part is

  $$K_{\mathrm{Si\ tail}}(t) = \frac{K_0}{\pi} \left( \frac{\pi}{2} - \mathrm{Si}(2\pi\Omega t) \right) = 2\Re \int_{\Omega}^{\infty} \left( \frac{-iK_0}{2\pi f} \right) e^{i2\pi f t} \mathrm{d}f,$$

  where $\mathrm{Si}(z) = \int_0^z \frac{\sin u}{u} \mathrm{d}u$ and the equality on the right is a standard sine–integral identity.

---

## The identity to bridge the two methods

Consider the "bracket" term that appears when splitting the Filon formula:

$$\star := 2\Re \int_0^{\Omega} R_{\mathrm{sub}}(f) e^{i2\pi f t} \mathrm{d}f + \frac{K_0}{\pi}\Big(\frac{\pi}{2} - \mathrm{Si}(2\pi\Omega t)\Big).$$

**Claim (exact):**

$$\boxed{\star = K_0 e^{-\alpha t} - 2\Re \int_{\Omega}^{\infty} \Big(R_{\mathrm{sub}}(f) - \frac{-iK_0}{2\pi f}\Big) e^{i2\pi f t} \mathrm{d}f.} \qquad (\blacklozenge)$$

**Why this is true (step-by-step).**

1. The *global* inverse of $R_{\mathrm{sub}}$ is exact:

$$K_0 e^{-\alpha t} = \int_{-\infty}^{\infty} R_{\mathrm{sub}}(f) e^{i2\pi f t} \mathrm{d}f = 2\Re \int_0^{\infty} R_{\mathrm{sub}}(f) e^{i2\pi f t} \mathrm{d}f.$$

2. Split the integral at $\Omega$:

$$2\Re \int_0^{\infty} R_{\mathrm{sub}} e^{i2\pi f t} \mathrm{d}f = 2\Re \int_0^{\Omega} R_{\mathrm{sub}} e^{i2\pi f t} \mathrm{d}f + 2\Re \int_{\Omega}^{\infty} R_{\mathrm{sub}} e^{i2\pi f t} \mathrm{d}f.$$

3. Add and subtract the leading asymptotic $\frac{-iK_0}{2\pi f}$ in the **tail** $f \geq \Omega$:

$$2\Re \int_{\Omega}^{\infty} R_{\mathrm{sub}} e^{i2\pi f t} \mathrm{d}f = 2\Re \int_{\Omega}^{\infty} \Big(R_{\mathrm{sub}} - \frac{-iK_0}{2\pi f}\Big) e^{i2\pi f t} \mathrm{d}f + 2\Re \int_{\Omega}^{\infty} \Big(\frac{-iK_0}{2\pi f}\Big) e^{i2\pi f t} \mathrm{d}f.$$

4. Recognize the **Si tail**:

$$2\Re \int_{\Omega}^{\infty} \Big(\frac{-iK_0}{2\pi f}\Big) e^{i2\pi f t} \mathrm{d}f = \frac{K_0}{\pi}\Big(\frac{\pi}{2} - \mathrm{Si}(2\pi\Omega t)\Big).$$

5. Rearrange to get (♦).

That's all—no approximation has been used in deriving (♦). It is an *exact* algebraic identity.

---

## What (♦) tells about the two methods

Start from the **Filon + Si tail** formula (for a general $R$):

$$K_{\text{Filon+Si}}(t) = 2\Re \int_0^{\Omega} R(f) e^{i2\pi f t} \mathrm{d}f + \frac{K_0}{\pi}\Big(\frac{\pi}{2} - \mathrm{Si}(2\pi F t)\Big).$$

Insert $R = \big(R - R_{\text{sub}}\big) + R_{\text{sub}}$ and use (♦) on the part involving $R_{\text{sub}}$. We get

$$K_{\text{Filon+Si}}(t) = \underbrace{2\Re \int_0^{\Omega} \big(R - R_{\text{sub}}\big) e^{i2\pi f t} \mathrm{d}f + K_0 e^{-\alpha t}}_{\text{JR–FFT structure on } [0,\Omega]} - 2\Re \int_{\Omega}^{\infty} \Big(R_{\text{sub}}(f) - \frac{-iK_0}{2\pi f}\Big) e^{i2\pi f t} \mathrm{d}f.$$

Thus the two reconstructions differ by **only one tail**:

$$\boxed{K_{\text{Filon+Si}}(t) = K_{\text{JR–FFT (with band } \Omega)}(t) - 2\Re \int_{\Omega}^{\infty} \Big(R_{\text{sub}}(f) - \frac{-iK_0}{2\pi f}\Big) e^{i2\pi f t} \mathrm{d}f.}$$

### Size of the difference

For large $f$,

$$R_{\text{sub}}(f) = \frac{K_0}{\alpha + i2\pi f} = \frac{-iK_0}{2\pi f} + \frac{K_0 \alpha}{(2\pi f)^2} + O(f^{-3}).$$

Therefore the integrand in the *difference tail* behaves like $O(f^{-2})$.

- **Uniform bound in $t$**:
  $\big|\text{difference tail}\big| \le 2\int_{\Omega}^{\infty} C f^{-2} \mathrm{d}f = O(\Omega^{-1})$.
- **Sharper bound for $t \ge t_0 > 0$** (one integration by parts in $f$):
  $\big|\text{difference tail}\big| = O\Big(\frac{1}{t\Omega^2}\Big)$.

So as $\Omega \to \infty$, **Filon+Si** and **JR-FFT** agree for all $t > 0$; the only visible discrepancy is near $t = 0$ if we keep $\Omega$ modest (the well-known "half-jump" phenomenon when truncating an unsubtracted $1/f$ tail).

---

## Intuition in one sentence

- **JR-FFT**: subtract the simple pole **everywhere** in $f$, invert the smooth remainder, and add the **full exponential** $K_0 e^{-\alpha t}$.

- **Filon+Si**: keep the unsubtracted integrand only up to $F$, and add the **missing high-frequency part** of the $1/f$ term—which is the **Si tail**.
- Identity (♦) shows these are the *same* up to a small $f \geq \Omega$ remainder built from the **difference** between the exact pole and its leading $1/f$ asymptote.

That's exactly why, once we include the Si tail included, Filon matches JR-FFT in practice; and why using the **exponential** as the tail (i.e., preconditioning then Filon on the remainder) makes them nearly indistinguishable even for moderate $\Omega$.