Tuesdays & Thursdays 12:00-13:30 ET

Topic 1

ME599-004: Data-Driven Methods for Control Systems

Winter 2025

Instructor: Uduak (*Who-dwak*) Inyang-Udoh

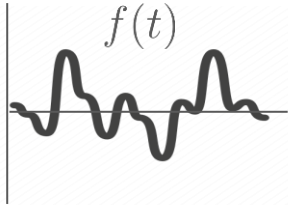
MECHANICAL ENGINEERING
UNIVERSITY OF MICHIGAN

FFT Video

FFT Video (Cont'd)

FFT Video (Cont'd)

Fourier Series & Transforms



$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{ik\pi t/L}$$

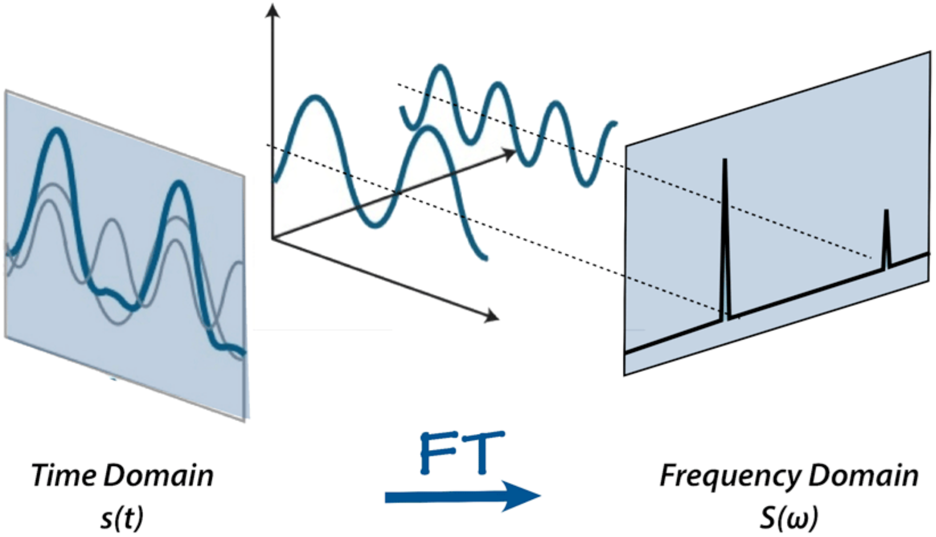
$$c_k = \frac{1}{2L} \left\langle f(t), e^{ik\pi t/L} \right\rangle = \frac{1}{2L} \int_{-L}^{L} f(t) e^{-ik\pi t/L}$$

$$f(t) = \lim_{\Delta\omega \to 0} \sum_{k=-\infty}^{\infty} \frac{\Delta\omega}{2\pi} \left\langle f(t), e^{i\omega t} \right\rangle_{[-\frac{\pi}{\delta\omega}, \frac{\pi}{\delta\omega})} e^{ik\Delta\omega t}$$

$$\hat{f}(\omega) = \mathcal{F}(f(t)) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} \, \mathrm{d}t$$

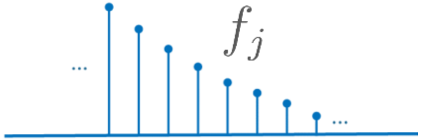
$$f(t) = \mathcal{F}^{-1}(\hat{f}(\omega)) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{i\omega t} \, \mathrm{d}\omega$$

Fourier Series & Transforms



Source

Discrete & Fourier Transforms



$$\omega_N = e^{-i\frac{2\pi}{N}}$$

$$\hat{f}_k = \sum_{j=0}^{N-1} f_j e^{-ik\frac{2\pi}{N}j}$$

$$f_j = \frac{1}{N}\sum_{k=0}^{N-1} \hat{f}_k e^{ij\frac{2\pi}{N}k}$$

$$\begin{array}{c} k \\ 0 \\ 1 \\ 2 \\ \vdots \\ N-1 \end{array}
\begin{array}{c} j \\ \begin{bmatrix} \hat{f}_0 \\ \hat{f}_1 \\ \hat{f}_2 \\ \vdots \\ \hat{f}_{N-1} \end{bmatrix} \end{array}
=
\begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & & N-1 \end{array} \\
\begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & \omega_N^1 & \omega_N^2 & & \omega_N^{N-1} \\
1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_N^{2(N-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)^2}
\end{bmatrix} \end{array}
\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{N-1} \end{bmatrix}$$

Fast Fourier Transforms

$$\hat{f}_k = \sum_{j=0}^{N-1} f_j e^{-ik\frac{2\pi}{N}j}$$

$$\hat{f}_k = \sum_{j=0}^{N/2-1} \underbrace{f_{2j}}_{\text{even}} e^{-ik\cdot\frac{2\pi}{N/2j}} + e^{-ik2\pi/N} \sum_{j=0}^{N/2-1} \underbrace{f_{2j+1}}_{\text{odd}} e^{-ik\frac{2\pi}{N}2j}$$

$$\hat{f}_{k+\frac{N}{2}} = \sum_{j=0}^{N/2-1} \underbrace{f_{2j}}_{\text{even}} e^{-ik\cdot\frac{2\pi}{N/2j}} - e^{-ik2\pi/N} \sum_{j=0}^{N/2-1} \underbrace{f_{2j+1}}_{\text{odd}} e^{-ik\frac{2\pi}{N}2j}$$

Algorithm 1: `fft`

Data: Signal vector $\mathbf{f} = \{f_0, \cdots, f_{N-1}\}$

Result: $\hat{\mathbf{f}} = \{\hat{f}_0, \cdots, \hat{f}_{N-1}\}$

if $N = 1$ **then**

  $\hat{\mathbf{f}} \leftarrow \mathbf{f}$

else

  $\hat{\mathbf{f}}_{even} \leftarrow \texttt{fft}(\{f_0, f_2, \cdots, f_{N-1}\});$

  $\hat{\mathbf{f}}_{odd} \leftarrow \texttt{fft}(\{f_1, f_2, \cdots, f_{N-2}\});$

  $\mathbf{n} \leftarrow \{1, 2, \cdots, \frac{N}{2} - 1\};$

  $\omega \leftarrow e^{i\frac{2\pi}{N}\mathbf{n}};$

  $\hat{\mathbf{f}} \leftarrow [\hat{\mathbf{f}}_{even} + \omega\hat{\mathbf{f}}_{odd}, \hat{\mathbf{f}}_{even} - \omega\hat{\mathbf{f}}_{odd}]$

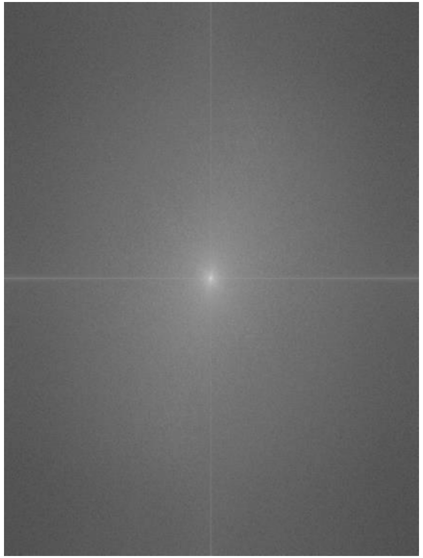
end

2D Fourier Transform

Fourier transforms do not only apply to temporal signals; applies to spatial signals

$$\hat{f}_k = \sum_{j=0}^{N-1} f_j e^{-ik\frac{2\pi}{N}j}$$

$$\hat{f}_{k_1,k_2} = \sum_{j_1=0}^{N_1-1} \sum_{j_2=0}^{N_2-1} f_{j_1,j_2} e^{-ik_1\frac{2\pi}{N_1}j_1} e^{-ik_2\frac{2\pi}{N_2}j_2}$$



Applications

1. Filtering
    a. Audio
    b. Images
2. Convolution:
    a. LTI systems
    b. Image processing
3. Differential Equations: (Laplace Transform also used)
4. Compression: DCT often used
    a. Audio
    b. Images
    c. Media
5. Time Series Analysis: trend analysis, anomaly detection
    a. audio signals
    b. finance
    c. Healthcare e.g. EEG
6. Others: Feature engineering/ Data augmentation, CNN’s, Natural Language Processing

Noise Filtering

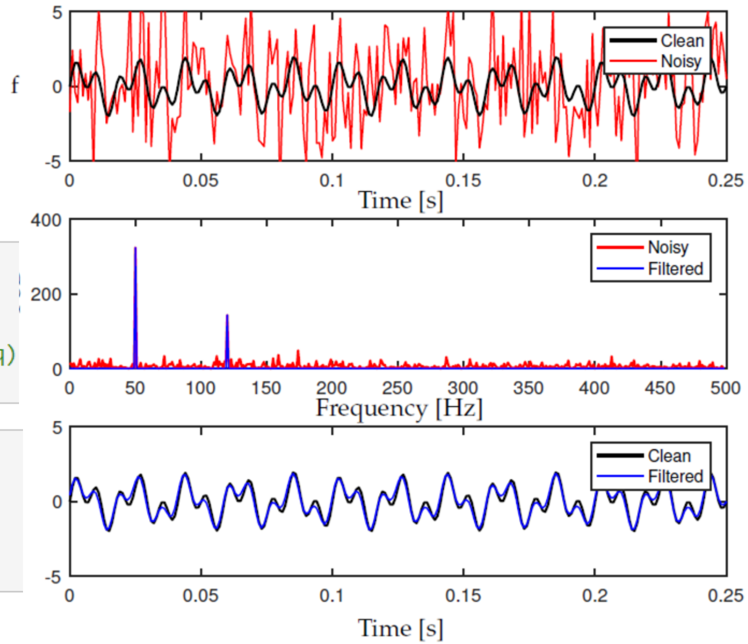
$$f(t) = \sin(2\pi f_1 t) + \sin(2\pi f_1 t)$$

$$f_1 = 50; f_2 = 120$$

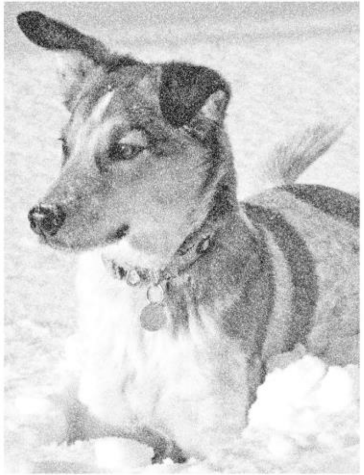
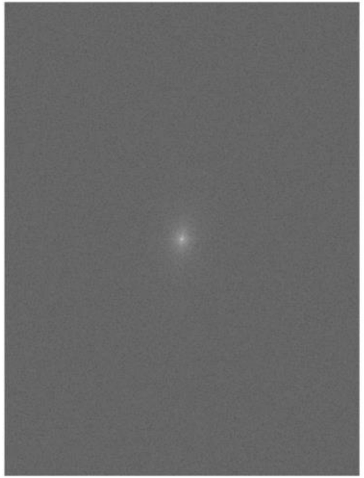
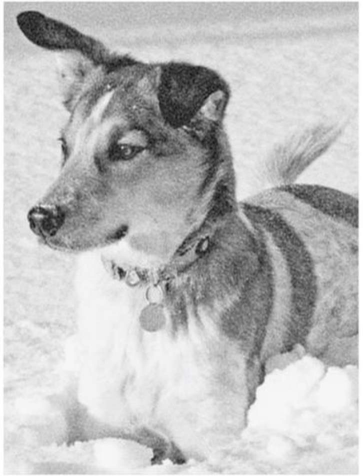
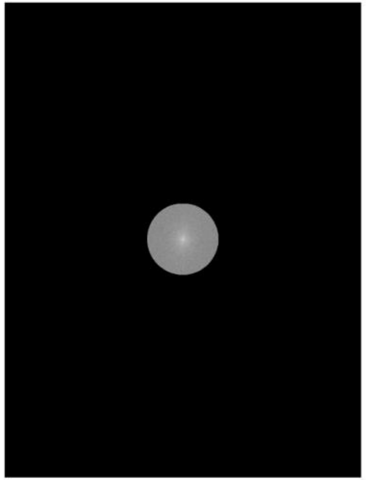
```
% Code is partly courtesy of Brunton and Kutz (2023) text
dt = .001;
t = 0:dt:1;
f = sin(2*pi*50*t) + sin(2*pi*120*t); % Sum of 2 frequencies
f_noisy = f + 2.5*randn(size(t)); % Add some noise
```

```
%% Compute the Fast Fourier Transform FFT
n = length(t);
fhat_noisy = fft(f_noisy,n); % Compute the fast Fourier transform
PSD_noisy = fhat_noisy.*conj(fhat_noisy)/n; % Power spectral density (power per freq)
L = 1:floor(n/2); % Only plot the first half of freqs
```

```
%% Use the PSD to filter out noise
indices = PSD_noisy>100; % Find all freqs with large power
PSDclean = PSD_noisy.*indices; % Zero out all others
fhat_noisy = indices.*fhat_noisy; % Zero out small Fourier coeffs. in fhat
ffilt = ifft(fhat_noisy); % Inverse FFT for filtered time signal
```

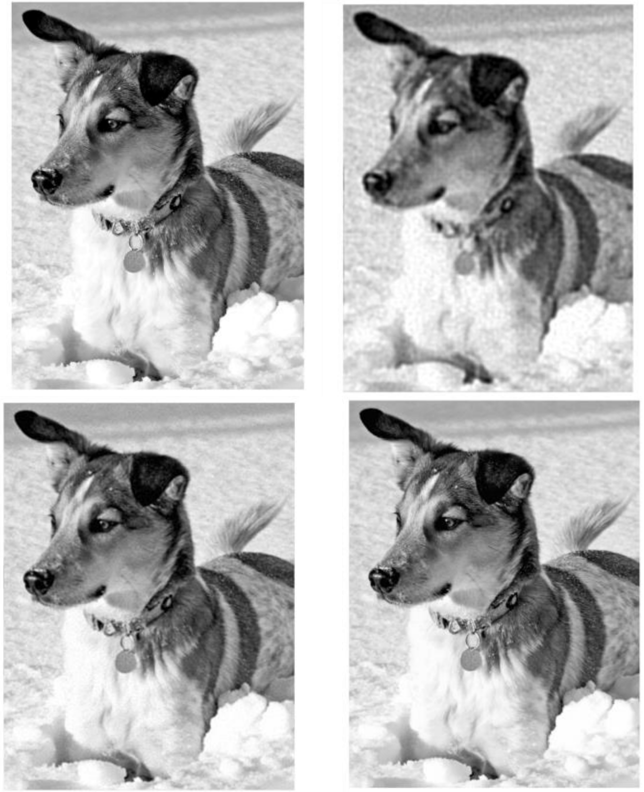


Noise Filtering (Cont'd)

Compression

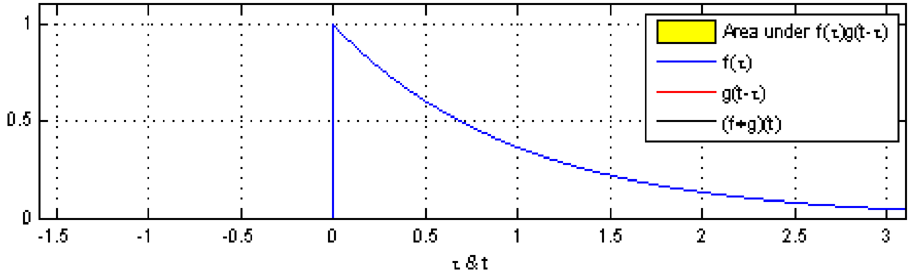
```
figure
Bt=fft2(B);
Btsort = sort(abs(Bt(:)));  % Sort by magnitude

% Zero out all small coefficients and inverse transform
percentvec = [.1 .05 .01 .002];
for k=1:4
    keep = percentvec(k);
    thresh = Btsort(floor((1-keep)*length(Btsort)));
    ind = abs(Bt)>thresh;
    Atlow = Bt.*ind;          % Threshold small indices
    Flow = log(abs(fftshift(Atlow))+1); % put FFT on log-scale
end
```



Convolution

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\, d\tau$$

To convolve a kernel with an input signal:
flip the signal, move to the desired time,
and accumulate every interaction with the kernel



Source

Fourier Transform of Convolution

$$\mathscr{F}(f^* g) = \mathscr{F}(f)\mathscr{F}(g)$$

$$\mathscr{F}^{-1}(\hat{f}\hat{g}) = f * g$$

Proof (1):

$$\begin{aligned}
\mathscr{F}(f^* g) &= \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau) \int_{-\infty}^{\infty} e^{-i\omega t} dt d\tau \\
&= \int_{-\infty}^{\infty} f(\tau)\, e^{-i\omega\tau} d\tau \int_{-\infty}^{\infty} g(t-\tau)\, e^{-i\omega(t-\tau)} dt \\
&= \int_{-\infty}^{\infty} f(\tau)\, e^{-i\omega\tau} d\tau \int_{-\infty}^{\infty} g(t'')\, e^{-i\omega t''} dt'' \\
&= \mathscr{F}(f)\mathscr{F}(g)
\end{aligned}$$

Convolution

Discrete Convolution

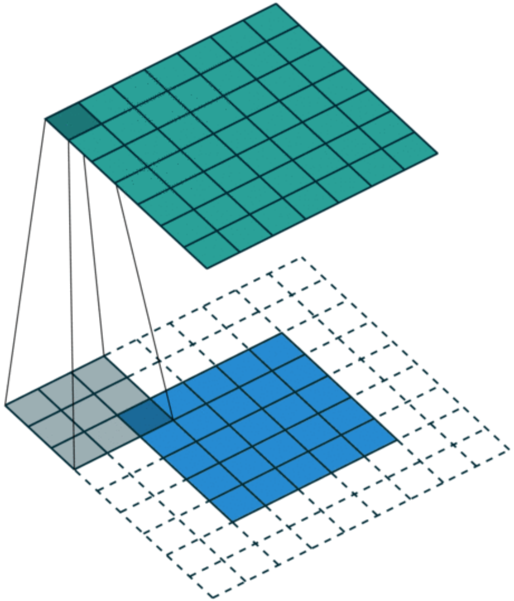
$$(f * g)[n] = \sum_{k=-\infty}^{\infty} f[k]g[n-k]$$

Discrete 2D Convolution

$$(F * G)[n_1, n_2]$$

$$= \sum_{k_2=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} F[k_1, k_2] \cdot G[n_1 - k_1, n_2 - k_2]$$

- Edge detection
- Blurring
- Sharpening
- Neural networks



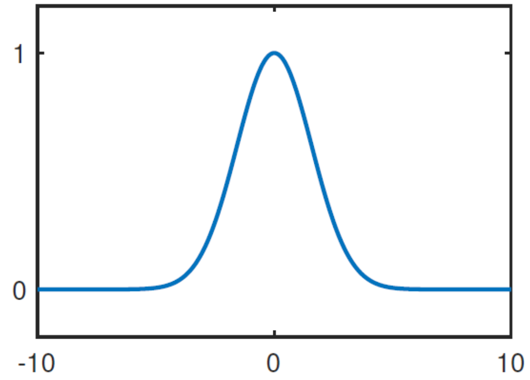
Laplace Transform

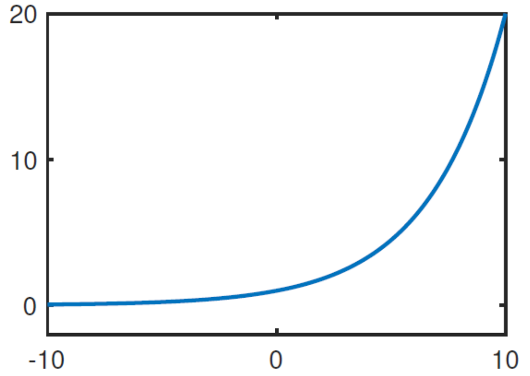
- ODE to algebraic equations
- Control theory

Laplace Transform

- ODE to algebraic equations
- Control theory

$$\hat{f}(\omega) = \mathcal{F}(f(t)) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} \, \mathrm{d}t$$



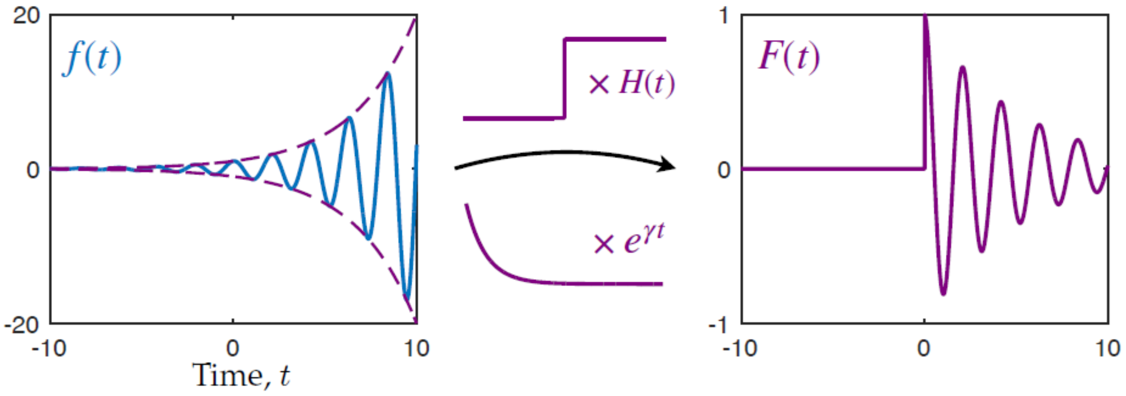


Laplace Transform

- ODE to algebraic equations
- Control theory

$$\hat{f}(\omega) = \mathcal{F}(f(t)) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}\,\mathrm{d}t$$

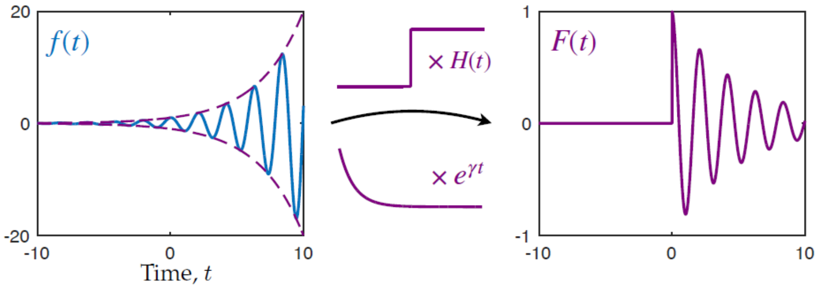
$$F(t) = f(t)e^{-\gamma t}H(t) = \begin{cases} 0 & \text{for } t \leq 0, \\ f(t)e^{-\gamma t} & \text{for } t > 0. \end{cases}$$



Laplace Transform

- ODE to algebraic equations
- Control theory

$$\hat{f}(\omega) = \mathcal{F}(f(t)) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}\,\mathrm{d}t$$



$$F(t) = f(t)e^{-\gamma t}H(t) = \begin{cases} 0 & \text{for } t \leq 0, \\ f(t)e^{-\gamma t} & \text{for } t > 0. \end{cases}$$

$$\begin{aligned} \hat{F}(\omega) = \mathcal{F}(F(t)) &= \int_{-\infty}^{\infty} F(t)e^{-i\omega t}\,\mathrm{d}t = \int_{0}^{\infty} f(t)e^{-\gamma t}e^{-i\omega t}\,\mathrm{d}t \\ &= \int_{0}^{\infty} f(t)e^{-(\gamma+i\omega)t}\,\mathrm{d}t = \int_{0}^{\infty} f(t)e^{-st}\,\mathrm{d}t = \bar{f}(s) \end{aligned}$$

Laplace Transform

Inverse Laplace Transform

$$F(t) = \mathcal{F}^{-1}(\hat{F}(\omega)) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{F}(\omega)e^{i\omega t}\,\mathrm{d}\omega$$

$$\begin{aligned} f(t)H(t) = e^{\gamma t}F(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{\gamma t}\hat{F}(\omega)e^{i\omega t}\,\mathrm{d}\omega \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{F}(\omega)e^{(\gamma+i\omega)t}\,\mathrm{d}\omega \end{aligned}$$

$$f(t)H(t) = \frac{1}{2\pi i}\int_{\gamma-i\infty}^{\gamma+i\infty} \bar{f}(s)e^{st}\,\mathrm{d}s$$

Laplace Transform

$$\mathcal{L}\left(\frac{\mathrm{d}^n}{\mathrm{d}t^n} f(t)\right) = -f^{(n-1)}(0) - s f^{(n-2)}(0) - \cdots - s^{n-1} f(0) + s^n \bar{f}(s)$$