The Discrete Fourier Transform

Moritz Diehl

Overview

- The ~~Frequency Response~~ Function (FRF)
- Laplace and Fourier Transforms
- Discrete Fourier Transform
- Aliasing and Leakage Errors
- Multisine Excitation Signals

The Frequency Response Function (FRF)

- Our aim: get transfer function $G(s)$ of LTI system
- the magnitudes and phases of $G(j\omega)$ for different positive frequencies ω form the Bode Diagram
- fundamental fact of LTI systems: sinusoidal inputs $\boxed{u(t) = \mathrm{Re}\{U \cdot e^{j\omega t}\}}$ lead to sinusoidal outputs $y(t)$ with a phase shift and a new magnitude described by $G(j\omega)$:

$$y(t) = \mathrm{Re}\{G(j\omega) \cdot U \cdot e^{j\omega t}\} = |G(j\omega)| \cdot U \cdot \mathrm{Re}\{e^{j[\omega t + \arg G(j\omega)]}\}$$

- for this reason, $G(j\omega)$ is called the "Frequency Response Function (FRF)"

Comments

Sine Wave Testing (Frequency Sweep)

- One way to obtain $G(j\omega)$ for a specific frequency ω is to use a sine wave $u(t) = U_0 \sin(\omega t)$ as input and record the magnitude Y_0 and phase shift ϕ of $y(t) = Y_0 \sin(\omega t + \phi)$ to form
$$G(j\omega) = \frac{Y_0}{U_0} e^{j\phi}$$
- a "frequency sweep" goes through all frequencies ω, waits until transients have died out, and records magnitude and phase for each frequency.
- The resulting estimate of the FRF might also be called "estimated transfer function (ETF)" (Robin) or "empirical transfer function estimate (ETFE)" (L. Ljung)
- note that for each new frequency, we have to wait until transients died out. Today, we want to find a more efficient way to estimate the FRF.

Comments

Laplace and Fourier Transforms

- Remember: $G(s) = \frac{Y(s)}{U(s)}$
- Laplace transform $G(s)$ defined for any $g(t)$ which is zero for $t < 0$:

$$G(s) := \int_0^\infty g(t)e^{-st}\mathrm{d}t = \int_{-\infty}^\infty g(t)e^{-st}\mathrm{d}t$$

- for FRF $G(j\omega)$, we only need imaginary values $\boxed{s = j\omega}$
- Here, we have

$$\mathcal{F}\{g\}(\omega) = \boxed{G(j\omega) = \int_{-\infty}^\infty g(t)e^{-j\omega t}\mathrm{d}t}$$

- This expression is identical to "Fourier Transform (FT)", defined for any function $f : \mathbb{R} \to \mathbb{R}$ by

$$\boxed{\mathcal{F}\{f\}(\omega) := \int_{-\infty}^\infty f(t)e^{-j\omega t}\mathrm{d}t}$$

Comments

Differences of Laplace and Fourier Transform

- both transformations basically contain the same information
- they transform a time signal $f(t)$ from "time domain" into "frequency domain"
- both transformations have inverse transformations that give the original time signal back
- both transformations generate complex valued functions
- Laplace transform has complex input argument $s \in \mathbb{C}$, while Fourier transform has real ω
- for Laplace transform, all input signals are by definition zero for $t < 0$, while Fourier transform deals with functions defined for any $t \in \mathbb{R}$ (i.e. functions with infinite support)
- Laplace transform often used by engineers, Fourier transform more often used by mathematicians and physicists

Comments

Inverse Fourier Transform

- if $F(\omega) = \mathcal{F}\{f\}(\omega)$, then $f(t)$ can be recoverd by inverse Fourier transformation $\mathcal{F}^{-1}$ given by:

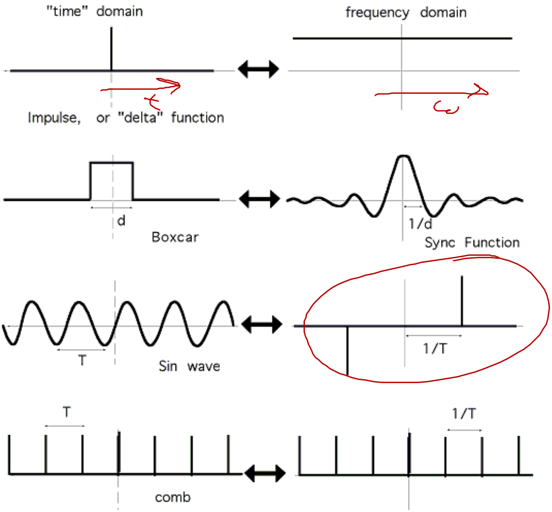
$$\boxed{f(t) = \mathcal{F}^{-1}\{F\}(t) := \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}\mathrm{d}\omega}$$

- Note the similarity of normal and inverse FT: just the sign in the exponent and the factor is different (some definitions even use twice the same factor, $\frac{1}{\sqrt{2\pi}}$, to make it symmetric)
- inverse FT can be used to construct inverse Laplace transform
- interesting related fact: Dirac-delta function is superposition of all frequencies with equal weight:

$$\boxed{\delta(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{j\omega t}\mathrm{d}\omega}$$

Comments

Fourier Transform: some transformed functions



Comments

Estimating the FRF with Fourier Transform

- if we have recorded two arbitrary time signals $u(t)$ and $y(t)$, we can use their Fourier transforms to estimate the frequency response function (FRF) by

$$G(j\omega) = \frac{\mathcal{F}\{y\}(\omega)}{\mathcal{F}\{u\}(\omega)}$$

- this fact is implicitly used in sine wave testing with frequency ω_0
- note: if $f_1(t) = \frac{e^{j\omega_0 t}}{2\pi}$ then

$$\mathcal{F}\{f_1\}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{j(\omega_0-\omega)t}\mathrm{d}t = \delta(\omega-\omega_0)$$

cos $\omega_0 t$ + j sin $\omega_0 t$

- thus, we have for a real sine: if $f_2(t) = \frac{e^{j\omega_0 t} - e^{-j\omega_0 t}}{2\pi \, j \, ?}$ then

$$\mathcal{F}\{f_2\}(\omega) = \delta(\omega-\omega_0) - \delta(\omega+\omega_0)$$

Comments

Estimating the FRF with Fourier Transform (cont.)

- in reality, even for sine waves of frequency ω_0 , the signals $u(t)$ and $y(t)$ will have finite duration, and thus the FT finite values $\mathcal{F}\{y\}(\omega_0)$ and $\mathcal{F}\{u\}(\omega_0)$. From these we can compute $G(j\omega_0)$ by
$$G(j\omega_0) = \frac{\mathcal{F}\{y\}(\omega_0)}{\mathcal{F}\{u\}(\omega_0)}$$
- Note: Fourier Transform works with continuous time signals on infinite horizons
- Two questions and answers:
  1. How to compute FT in practice? Answer: by the Discrete Fourier Transform.
  2. Can we use an input with many frequencies to get many FRF values in a single experiment? Answer: yes, we should then use "multisines".

Comments

The Discrete Fourier Transform (DFT)

- FT works with continuous time signals on infinite horizons
- Discrete Fourier Transform (DFT) works with discrete signals on finite horizons
- DFT takes any vector of N numbers $u(0), u(1), \ldots, u(N-1)$ and generates a new vector $U(0), \ldots, U(N-1))$ (here we start with index zero for convenience)
- DFT also has an inverse transformation that recovers the original vector

Comments

Fast Fourier Transform (FFT)

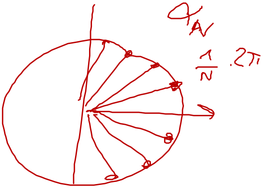
- one efficient algorithm to compute the DFT is called "fast fourier transform (FFT)"
- the DFT is nearly always computed by the FFT algorithm, therefore many people (and MATLAB) use the word FFT synonymously with DFT
- MATLAB commands `fft` and `ifft` work with any vector of N complex numbers and compute another vector of N complex numbers.
- example: `u=randn(10,1); U=fft(u); unew = ifft(U); plot(u,unew)`

Comments

DFT definition

- Definition of the DFT $U(0), \ldots, U(N-1)$ computed from a vector $u(0), \ldots, u(N-1)$:

$$U(\underline{m}) := \sum_{k=0}^{N-1} u(\cancel{t}k)\alpha_N^{-mk}$$



with

$$\alpha_N := e^{j\frac{2\pi}{N}}$$

- note that α_N is an N-th complex root of 1, i.e.

$$\alpha_N^N = 1$$

- also note that $\alpha_N^{-mk} = e^{-j\frac{2\pi}{N}mk}$ and $\overline{\alpha_N^{-mk}} = \alpha_N^{mk}$

Comments

DFT properties

.

- DFT of a real valued signal consists of N complex numbers, but second half of vector are complex conjugates of first half:

$$U(N-m) = \overline{U(m)}$$

Proof:

$$U(N-m) = \sum_{k=0}^{N-1} u(t)\alpha_N^{-(N-m)k} = \sum_{k=0}^{N-1} u(t)\alpha_N^{mk} = \sum_{k=0}^{N-1} u(t)\overline{\alpha_N^{-mk}}$$

- example: `u=sin(1:0.1:10.1); U=fft(u);`
`subplot(2,1,1);plot(real(U)); subplot(2,1,2);`
`plot(imag(U));`

Comments

Overview

- The Frequency Response Function (FRF)
- Laplace and Fourier Transforms
- Discrete Fourier Transform
- **Aliasing and Leakage Errors**
- Multisine Excitation Signals

Comments

Comparison of FT and DFT

- FT works on continuous time signals $u_c(t)$ with infinite support
- DFT introduces two approximations:
  1. **Sampling:** DFT works on sampled (discrete time) signals

     $$u_d(k) := u_c(k \cdot \boxed{\Delta t})$$

     with Δt the sampling time.
  2. **Windowing:** DFT only uses only N samples, i.e. limits the signal to a finite window of horizon length $\boxed{T} = N\Delta t$
- both approximations lead to characteristic errors.

Comments

Sampling can lead to Aliasing Errors

- Sampling can introduce so called **aliasing errors** if the continuous time signal contained too high frequencies
- example: `t=[0:0.1:10]'; u1=sin(6*t); u2=sin(20*t); u3=sin(60*t); subplot(3,1,1);plot(t,u1); subplot(3,1,2); plot(t,u2);subplot(3,1,3); plot(t,u3);`
- if we introduce sampling rate $f_s = \frac{1}{\Delta t}$, then any signal with frequencies higher than **half the sampling rate** will suffer from aliasing
- the limit is called the **Nyquist frequency**: $f_{\text{Nyquist}} = \frac{1}{2\Delta t}$ [Hz] or $\omega_{\text{Nyquist}} = \frac{2\pi}{2\Delta t}$ [rad / s]

Comments

Windowing can lead to "leakage"

- "leakage": DFT spectrum shows frequencies that were not present in original signal, but are close to the true frequencies
- example (leakage): `t=[0:49]'; u=sin(2*pi/50*20.5*t); U=fft(u); plot(abs(U));`
- example (no leakage): `t=[0:49]'; u=sin(2*pi/50*20*t); U=fft(u); plot(abs(U));`
- comparison of FT and DFT

$$\int_{-\infty}^{\infty} u_c(t)\cdot e^{-j\omega t}\,\mathrm{d}t \approx \sum_{k=0}^{N-1} u_d(k)\cdot \underbrace{e^{-j\omega(k\cdot\Delta t)}}_{=e^{-j\frac{2\pi}{N}km}}\cdot\Delta t$$

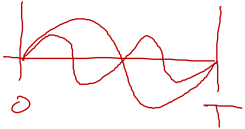
here, the FT and DFT expressions are only similar, if

BASE FREQ

$$-j\omega(k\cdot\Delta t) = -j\frac{2\pi}{N}km \quad \text{i.e.} \quad \omega = m\frac{2\pi}{\underbrace{\Delta t\cdot N}} = m\frac{2\pi}{T}$$

Comments

The Base Frequency and its Harmonics



- Let us define the “base frequency”

$$\omega_{\text{base}} := \frac{2\pi}{N \cdot \Delta t} = \frac{2\pi}{T}$$

- corresponds to the slowest sine that fits exactly into the window
- a sine signal $\sin(\omega t)$ with $\omega = m \cdot \omega_{\text{base}}$ is called the “m-th harmonic”
- the DFT contains only the first $N/2$ harmonics of the base signal
- the frequency resolution (difference of two frequencies that are distinguished by the DFT) is equal to the base frequency
- the finite length of the window limits the frequency resolution: the longer the window, the finer the frequencies can be resolved in the signal

Comments

Visualization of Harmonics

- example in time domain: `deltat=0.1; T=10;`
  ```
  t=[0:deltat:T-deltat]';wbase=2*pi/T;
  u1=sin(wbase*t); subplot(4,1,1); plot(t,u1);
  u2=sin(2*wbase*t); subplot(4,1,2); plot(t,u2);
  u3=sin(3*wbase*t); subplot(4,1,3); plot(t,u3);
  u4=sin(3.5*wbase*t); subplot(4,1,4); plot(t,u4);
  ```
- same example in frequency domain: `U1=fft(u1);`
  ```
  subplot(4,1,1); plot(abs(U1)); U2=fft(u2);
  subplot(4,1,2); plot(abs(U2)); U3=fft(u3);
  subplot(4,1,3); plot(abs(U3)); U4=fft(u4);
  subplot(4,1,4); plot(abs(U4));
  ```

Comments

Multisines: the perfect excitation signal?

- we can choose $u(t)$ as a "multisine", i.e. a superposition of specially chosen sine waves
- we can avoid both aliasing and leakage if the following three conditions are met:
  1. we choose a DFT window length T that is an integer multiple of the sampling time Δt, i.e. $T = N \cdot \Delta t$
  2. the multisine contains only harmonics of the base frequency $\omega_{\mathrm{base}} = \frac{2\pi}{T}$ i.e. it is periodic with period T (or an integer fraction of T)
  3. the multisine does not contain any frequency higher than the Nyquist frequency $\omega_{\mathrm{Nyquist}} = \frac{\pi}{\Delta t}$
- in order to achieve optimal excitation without too large input amplitudes, one chooses the phases of the multisine carefully to avoid positive interference

Comments

-

The Crest Factor

- The "crest factor" is the ratio between the highest peak $u_{\max}$ and the root mean square u_{rms} of the input signal:

$$u_{\max} := \max_{t \in [0,T]} |u(t)|$$

and

$$u_{\mathrm{rms}} := \sqrt{\frac{1}{T} \int_0^T u(t)^2 \mathrm{d}t}$$

- example for bad crest factor: `N=20; U= zeros(N, 1); U(2:N/2) = 1; U(end:-1:N/2+2) = conj(U(2:N/2)); u= ifft(U); plot([u;u;u]);`
- example for better crest factor: `N=20; U= zeros(N, 1); U(2:N/2) = exp(i*2*pi*rand(N/2-1,1)); U(end:-1:N/2+2) = conj(U(2:N/2)); u= ifft(U); plot([u;u;u]);`

Comments