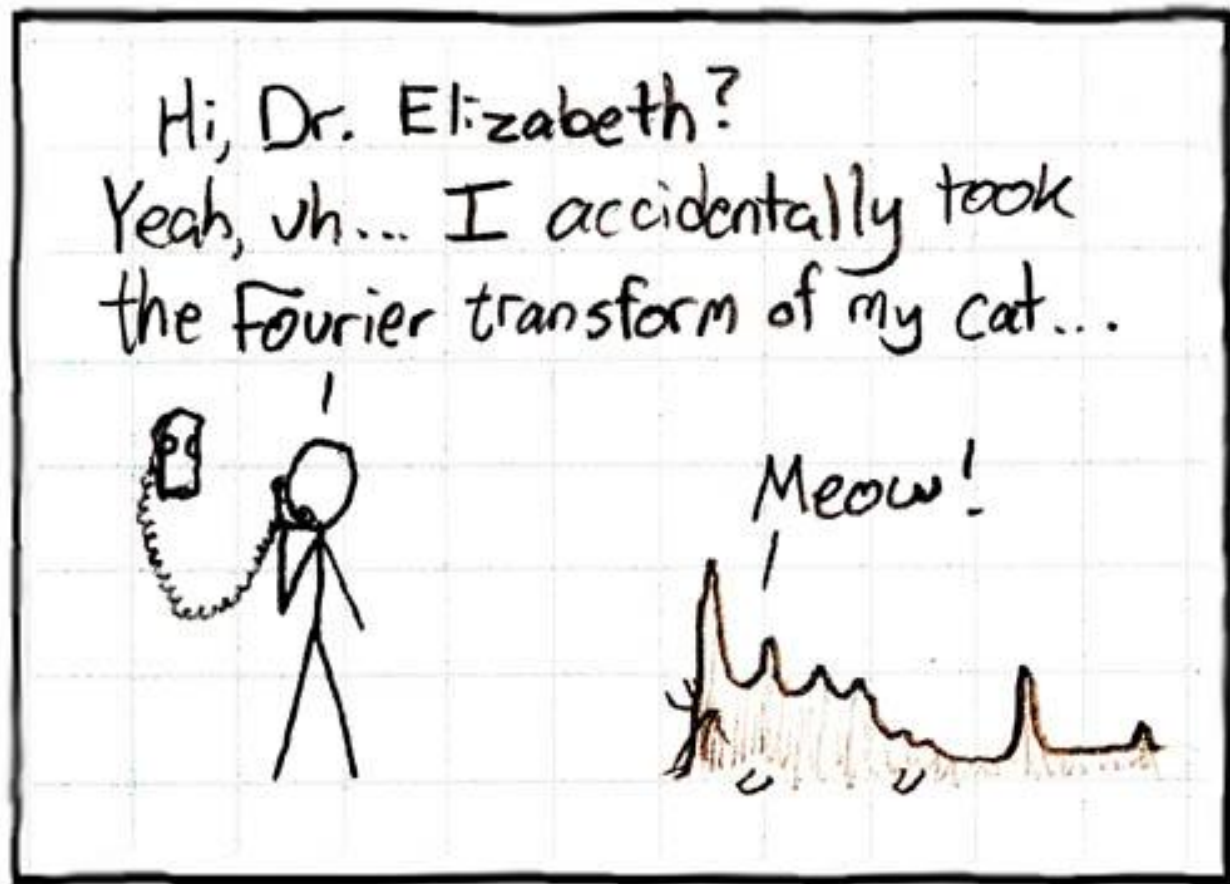


The Chirp
Z-transform;
Fourier transforms
over sensible ranges
for fun and profit.



I am not a signal theorist

this talk should not be construed
as legal advice and has not been
approved by the FDA for the
treatment of any medical condition

a refresher

$$\tilde{F}(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-i\omega t} dt$$

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \tilde{F}(\omega)e^{i\omega t} d\omega.$$

A scientist in possession of good data must be in want of a frequency-domain representation of that data.

- Jane Austen

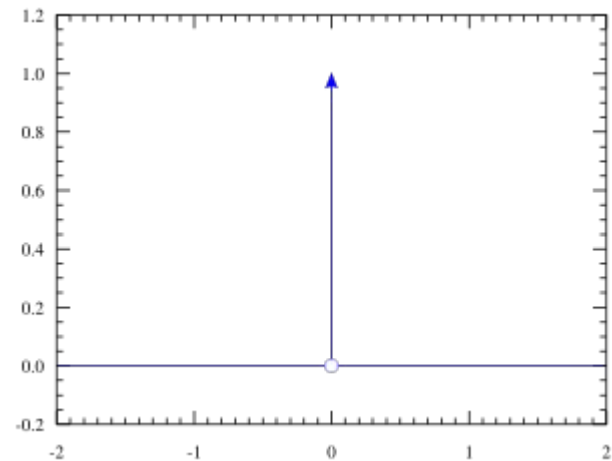
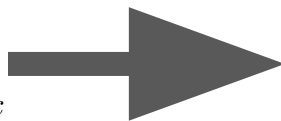
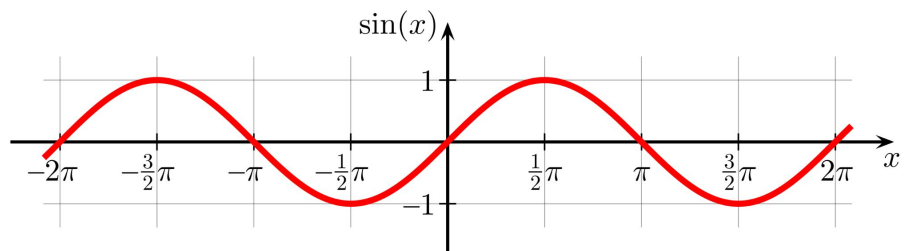
FourierParameters

FourierParameters

is an option to `Fourier` and related functions that specifies the conventions to use in computing Fourier transforms.

- Some common choices for $\{a, b\}$ are $\{0, 1\}$ (default), $\{-1, 1\}$ (data analysis), $\{1, -1\}$ (signal processing).

a refresher

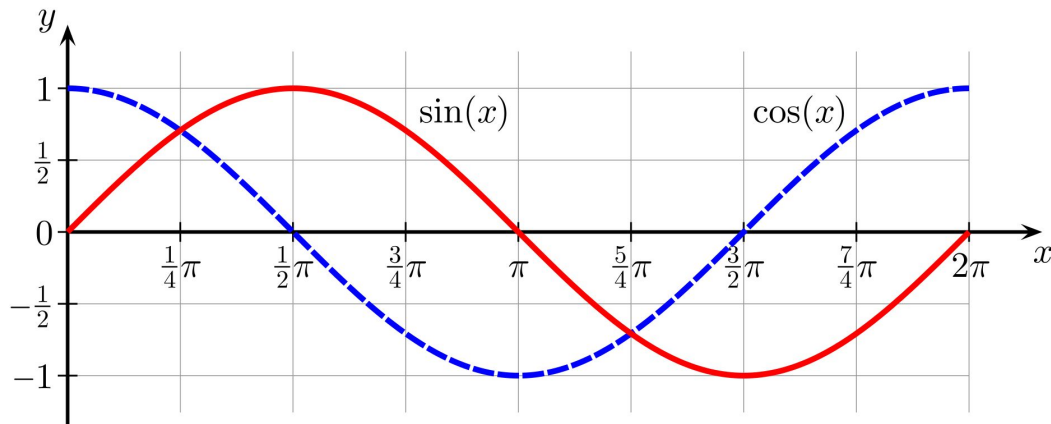


(wrong) intuition: two arrays

FFT gives you two arrays of coefficients α, β

reconstruct original signal using $[\alpha_{frequency} \sin() + \beta_{frequency} \cos()]$ for each component

α, β happen to be the real and imaginary parts of the same



<i>Continuous / Analytic:</i>		<i>Discrete version:</i>
Laplace transform	<p>“Good for damped pendulums.”</p> <p>I understand that this can be thought as fourier transform supporting exponential variation with time.</p>	Z-transform
Fourier transform	Constant sins and cosines, periodic over all time.	DFT (almost always FFT , Blackman-Tukey in the past)
Bunch of other stuff I guess (wavelet transform, STFT,)		

The Convolution Property

The convolution property states that:

$$x(t) * h(t) \leftrightarrow X(\omega)H(\omega)$$

More Properties of the Fourier Transform.

<https://class.ece.uw.edu/235dl/EE235/Project/lesson17/lesson17.html>

Correlation: Particle image velocimetry

*Discetti, Stefano, and Andrea Ianiro. 2017.
Experimental Aerodynamics. CRC Press.*

Multitaper

[the periodogram, $\text{magnitude}(\text{fft})^2$, is]...the **naive spectral estimate**, meaning it is the spectral estimate that you get **if you don't know that there is something better. Please do not use the periodogram in your publications.**

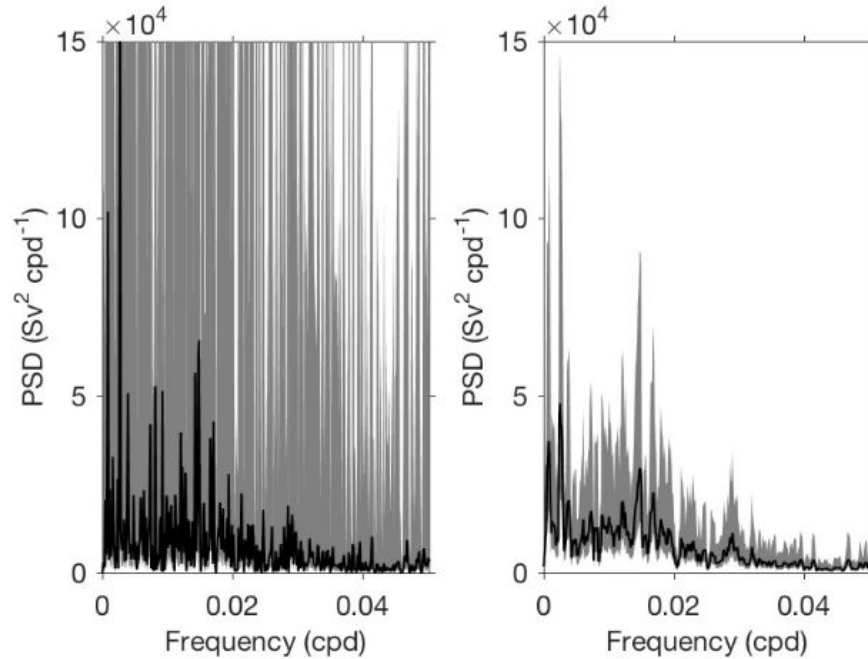
emph. mine

*Elipot, Shane. "Lecture 4: Time Series Analyses."
University of Miami.*

<https://selipot.github.io/talks/lecture4.pdf>

Uncertainty of the spectrum

Periodogram (left) and multitaper (right) estimate with CIs on linear-linear scales



Real example: Tracking down interference

- Connect an oscilloscope to a system.
- What is the source of interference?

Interested in frequency range

10 to 1000 Hz

60 Hz + harmonics



Oscilloscope setup looks sensible enough...

```
pretty_out(f"Input data: <br>Sample rate = {fs :.0f} samples per second [{fs/1e6} Ms/s (Hz)] <br> Length: {N} samples ({N/fs :.1f} seconds)")
```

Input data:

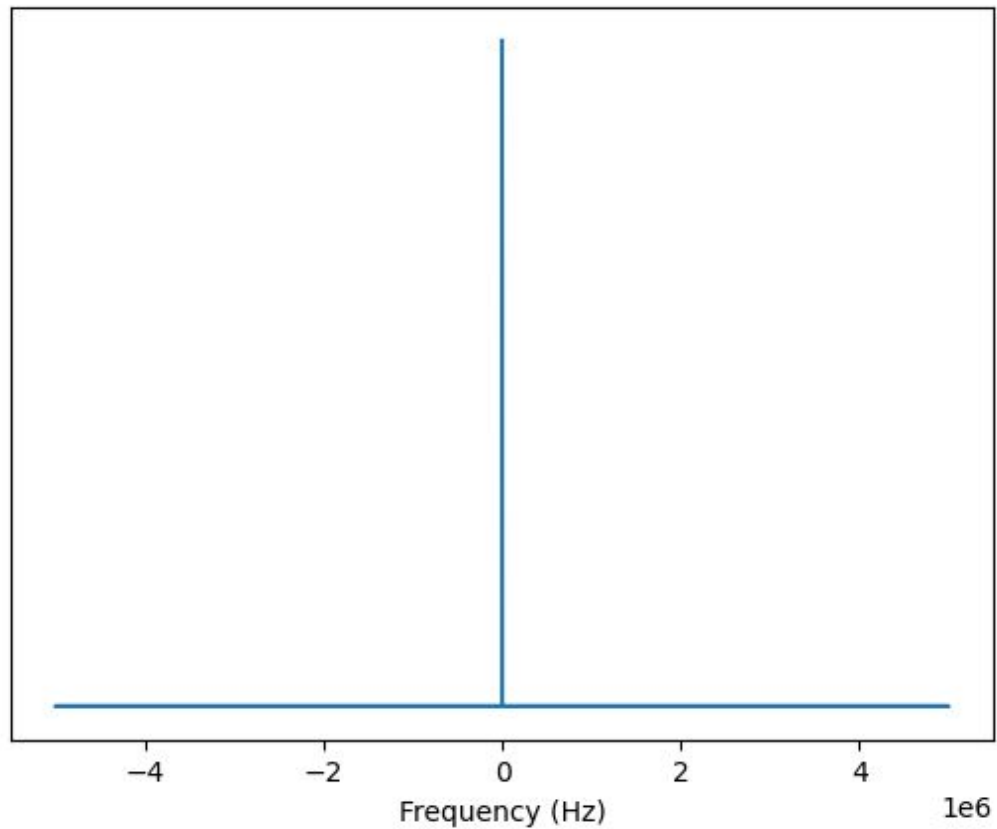
Sample rate = 10000000 samples per second [10.0 Ms/s (Hz)]

Length: 12000512 samples (1.2 seconds)

Frequency resolution, per output bin: $\Delta_f = (f_s/N) = 0.833$ Hz

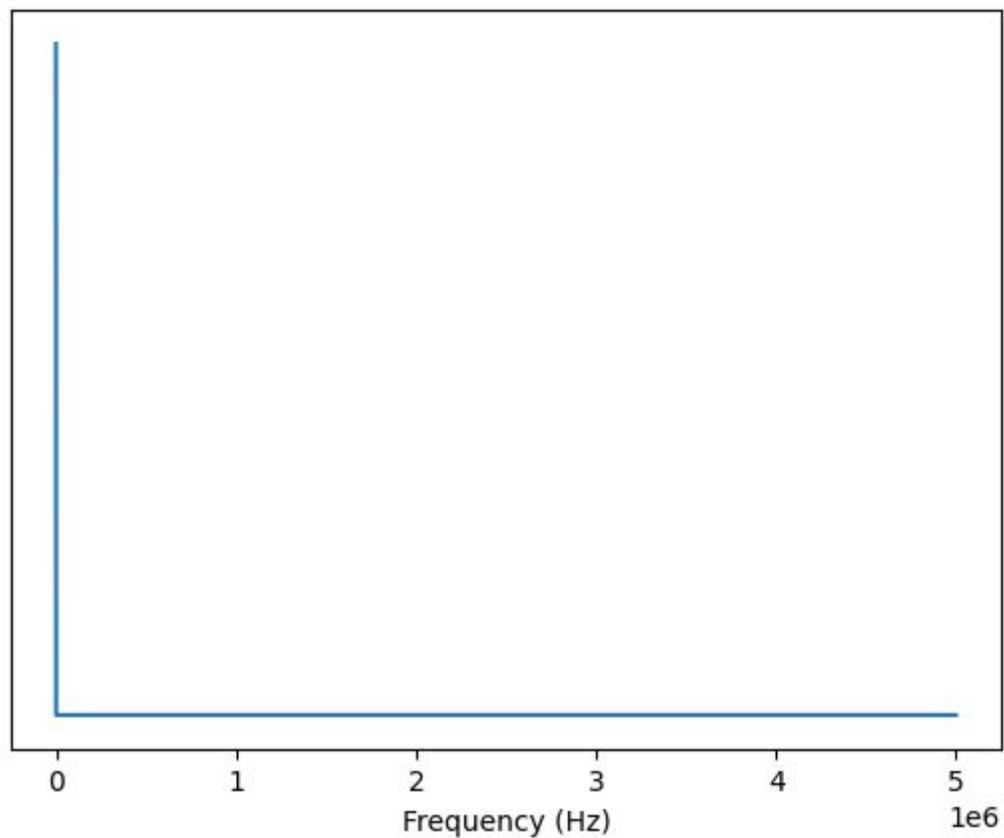
```
ft = np.fft.fft(y)
```

```
plt.plot(bin_frequencies, np.abs(ft)**2); ...
```



```
rft = np.fft.rfft(y)
```

```
plt.plot((np.arange(N//2+1) * fs/N), np.abs(rft)**2); ...
```



FFT has a non-negotiable relationship between input and output

Nyquist limit (sample frequency / 2)

- Need at least two points per cycle

(although if frequency components higher than the Nyquist are *present* during measurement, they may still be be *aliased* into your FFT range!)

Rayleigh frequency ($1/T$)

- need at least one cycle over the whole length of the recording

Intrinsic to the information theory - does not depend on the implementation

(in the past, FFT implementations often also wanted an even power-of-2 input length [or would pad it internally, which is expensive], but with modern libraries this is not usually a concern anymore.)

Using this dataset as an input,

we can easily see the part of the spectrum that we want if we truncate - trim off all the parts we don't want

But, this only represents
1188 bins of the 6000257
that `rfft()` gives us
(0.019%)

Need to compute and also
allocate memory for all
those wasted bins

Well, that's stupid

why did you record it at
10 Ms/s if you only want
a 1 KHz signal?

Well, that's stupid.

Just zoom out with
the oscilloscope.

Well, that's stupid.

Just zoom out with the
oscilloscope.

Why are we even talking
about this.

Problem occurs anytime the sample rate and/or record length are poorly matched to the frequency range.

transform (DFT) of $\{x(n)\}$ to get the spectrum, $X(f)$, of $x(t)$, where $x(n) = x(t)|_{t=n\Delta t}$ and Δt is the time step used in the FD-TD algorithm. For some applications [10], this method is not very efficient and it does not have sufficient accuracy. This is because the $N/2$ values of DFT are uniformly distributed over a very large frequency bandwidth, extending from 0 to $f_s/2$ Hz, where $f_s = 1/\Delta t$ is the sam-

Some settings where you don't have the flexibility; FDTD, MD

Common techniques

- *Decimate* the input

Throw out 9 points, keep every 10th. Artificially decreases the sample rate.

“I spent a month collecting that data, now you want me to throw 90% of it out?”

Common techniques

- *Pad* the input with zeros

Artificially increasing N to change the ratios

Works OK

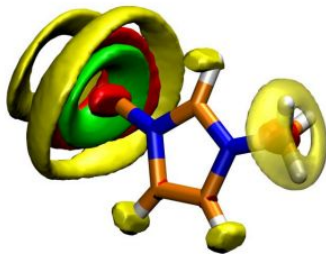
Enormous increase in computational cost

Common techniques

- *Pad* the input
 - Side effect of creating frequency bins
 - *Sometimes* that's useful, can help in interpreting

TRAVIS

Trajectory Analyzer and Visualizer

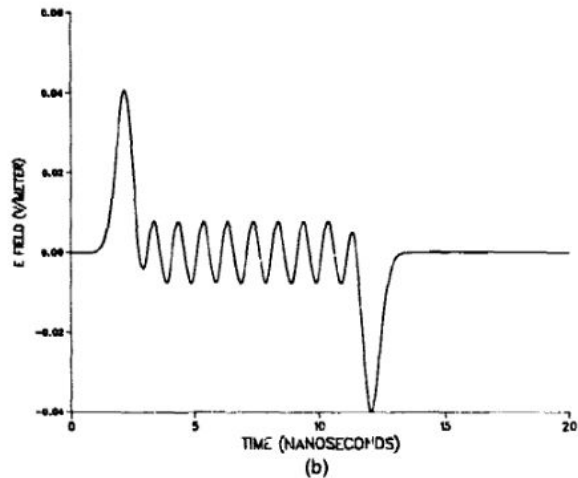
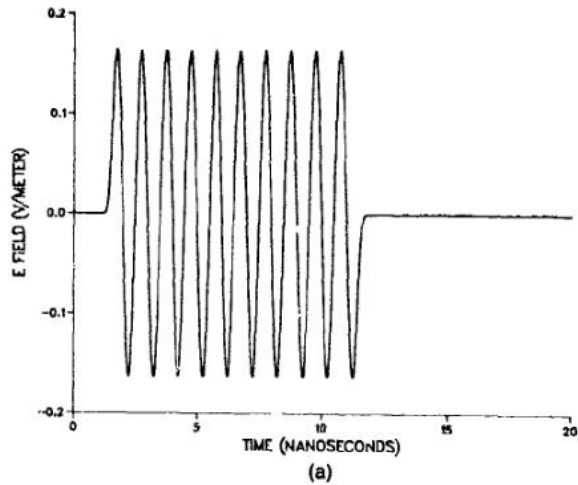


Common techniques

- *Pad* the input
 - Side effect of inventing new frequency bins between
 - *Sometimes* that's useful
 - ***Usually*** just dubious and concerning

New bins don't contain actual information

Just a *sinc()* interpolation.



Albanese, Richard, John Penn, and Richard Medina. 1989. "Short-Rise-Time Microwave Pulse Propagation through Dispersive Biological Media." *Journal of the Optical Society of America A* 6 (9): 1441. <https://doi.org/10.1364/JOSAA.6.001441>.

Seems inelegant.

Is there anything we
can do about it?

Bi et al 1992. 10.110922/149539

where N is the length of the sequence $\{x(n)\}$, is too coarse to accurately determine resonant frequencies. In practice, only the lower part of the band is of interest. One method that has been suggested here is to do the numerical integration of Fourier Transform of $x(t)$ directly in the interested frequency band

“Oh, you don't need the whole thing? Just do the part you want.”

pling frequency, and because the frequency resolution, which is given by

$$\Delta f = \frac{1}{N \cdot \Delta t}, \quad (9)$$

where N is the length of the sequence $\{x(n)\}$, is too coarse to accurately determine resonant frequencies. In practice, only the lower part of the band is of interest. One method that has been suggested here is to do the numerical integration of Fourier Transform of $x(t)$ directly in the interested frequency band

$$\begin{aligned} X(f) &= \int_0^{\infty} x(t) \exp(-j2\pi ft) dt \\ &\approx \int_0^{N\Delta t} x(t) \exp(-j2\pi ft) dt \\ &\approx \sum_{n=0}^{N-1} x(n) \exp(-j2\pi fn \Delta t) \Delta t. \end{aligned} \quad (10)$$

The advantage of this method is that it removes ambiguities sometimes encountered with the discrete Fourier Transform, due to narrowband signal components with center frequencies that lie in the gaps between the $N/2$

<https://dsp.stackexchange.com/questions/9251/fft-f-or-a-specific-frequency-range>

“This is a more direct and exact method of calculating a spectral zoom as opposed to zero padding a DFT and interpolating to desired point locations.”

*Martin, Grant. 2005. “Chirp Z-Transform Spectral Zoom Optimization with MATLAB.” SAND2005-7084, 1004350.
<https://doi.org/10.2172/1004350>.*



Chirp Z-transform and Zoom FFT

`czt` (x [, m , w , a , $axis$])

Compute the frequency response around a spiral in the Z plane.

`zoom_fft` (x , fn [, m , fs , $endpoint$, $axis$])

Compute the DFT of x only for frequencies in range fn .

Contributed to SciPy 1.8.0 (2022), original code by Paul Kienzle from NIST and pulled in by Nadav Horesh from Medtronic

github.com/garrettj403/CZT

```
import czt  
_, out = czt.time2freq(t, y, freq)
```

Sukhoy, Vladimir, and Alexander Stoytchev. 2019.

“Generalizing the Inverse FFT off the Unit Circle.” Scientific Reports 9 (1): 14443.

<https://doi.org/10.1038/s41598-019-50234-9>.

Basic algorithm only discovered in 2019

“An efficient algorithm for computing the forward chirp z-transform was described 50 years ago[1–5]. It was derived using an index substitution, which was originally proposed by Bluestein[1,5], to compute the transform...

This paper describes the first algorithm for computing the inverse chirp z-transform (ICZT) in $O(n \log n)$ time. This matches the computational complexity of the chirp z-transform (CZT) algorithm that was discovered 50 years ago. Despite multiple previous attempts, an efficient ICZT algorithm remained elusive until now.”

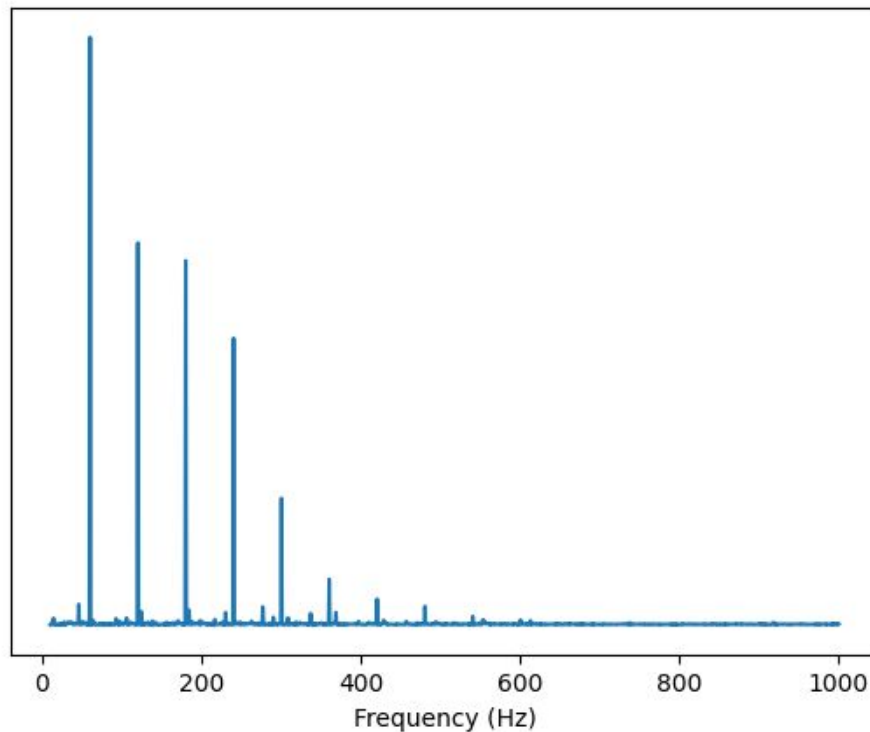
Sukhoy, Vladimir, and Alexander Stoytchev. 2019.

“Generalizing the Inverse FFT off the Unit Circle.” *Scientific Reports* 9 (1): 14443. <https://doi.org/10.1038/s41598-019-50234-9>.

```
start_frequency, end_frequency = 10, 1000
No = int((end_frequency-start_frequency) // delta_f)
# don't want to make up data, use the same # of bins as the fft

zft = signal.ZoomFFT(len(y), [start_frequency,end_frequency], m=No, fs=fs)

segment_bin_frequencies = np.linspace(f1, f2, No) ●●●
```



```
%%time
zft = signal.ZoomFFT(len(y), [start_frequency,end_frequency], m=No, fs=fs)
_zft()
```

```
CPU times: user 2.43 s, sys: 504 ms, total: 2.93 s
Wall time: 2.93 s
```

```
print(f"mem: {max(memory_usage(_zft)) :0.2f} MiB")
```

```
mem: 7385.28 MiB
```

```
%%time
fft()
```

```
CPU times: user 2.78 s, sys: 1.02 s, total: 3.79 s
Wall time: 3.79 s
```

```
print(f"mem: {max(memory_usage(fft)) :0.2f} MiB")
```

```
mem: 8381.08 MiB
```

```
%%time
rfft()
```

```
CPU times: user 2.83 s, sys: 752 ms, total: 3.59 s
Wall time: 3.58 s
```

```
print(f"mem: {max(memory_usage(rfft)) :0.2f} MiB")
```

```
mem: 8289.33 MiB
```

```
%%time
_, sig_f = czt.time2freq(t, y, freq)
```

```
CPU times: user 9.43 s, sys: 1.5 s, total: 10.9 s
Wall time: 10.9 s
```

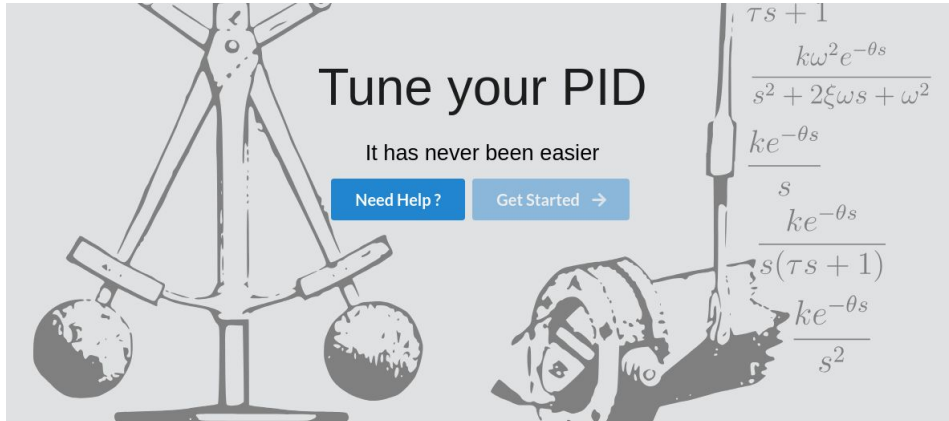
```
print(f"mem: {max(memory_usage(czt1)) :0.2f} MiB")
```

```
mem: 8211.79 MiB
```

Thanks for your
patience :)

Bonus Laplace Transform content

The PID loop is a conspiracy by Big
Control to keep the masses docile



Tune your PID

It has never been easier

[Need Help ?](#) [Get Started →](#)

$$\frac{\tau s + 1}{k\omega^2 e^{-\theta s}} \frac{1}{s^2 + 2\xi\omega s + \omega^2}$$

$$\frac{k e^{-\theta s}}{s}$$

$$\frac{s(\tau s + 1)}{k e^{-\theta s}} \frac{1}{s^2}$$

[Save](#) [Save As](#)

- Import Data
- Select Step
- Select Model
- Tune PID

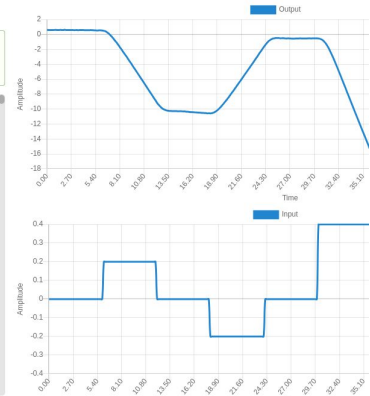
Clear Table

Valid Data
The data is correct. Click on the green 'Next' button to continue.

	Time	Input	Output
1	0	0	0.61175344
2	0.06	0	0.60157694
3	0.12	0	0.60976509
4	0.18	0	0.58817355
5	0.24	0	0.59133825
6	0.3	0	0.60408661
7	0.36	0	0.59860008
8	0.42	0	0.61317274
9	0.48	0	0.59705234
10	0.54	0	0.58565935
11	0.6	0	0.6045133
12	0.66	0	0.5854639
13	0.72	0	0.59172816
14	0.78	0	0.58402341
15	0.84	0	0.57626044
16	0.9	0	0.6059277
17	0.96	0	0.59374551
18	1.02	0	0.58475204
19	1.08	0	0.59831146
20	1.14	0	0.58064615

Import your step response data
Here. Copy and paste to the table below using the ctrl+v shortcut. Match each column of the table with the corresponding data. All three columns must be of the same length. A minimum of 50 samples is required. Right click over the table headers or rows to get a context menu.

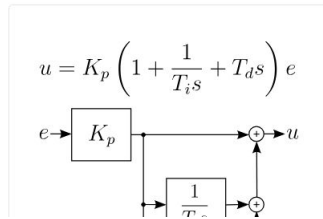
1. Import Data



Why

Tuning a PID controller can be easily done using a systematic procedure. We just put it in a web application. No more trial and error.

How



PidTuner - <https://github.com/pidtuner/pidtuner.github.io>.