信号処理とフーリエ変換 第10回

～音声信号の周波数を調べる～

かつらだ まさし
桂田 祐史

https://m-katsurada.sakura.ne.jp/fourier2024/

2024年12月11日

目次

本日の内容・連絡事項

- 2 回に渡って離散 Fourier 変換を説明した。今回はその応用編として、こちらが用意した音声データと Mathematica のノートブックを使って、音声信号の周波数を調べる実験を行う。(後日各自で用意した音声データで同じことをしてもらう課題を出す。) その後、音のデジタル信号表現の一つである PCM をざっと説明した後、実験の分析を行う。
- おおむね、講義ノート [1] の§4 に該当するが、プログラムについては授業で提供した方を参考にすること (講義ノートに載せたものから更新されている)。

1. この授業の WWW サイト から
   - ギターのドの音の WAVE ファイル `guitar-5-3.wav`
   - Mathematica のノートブック `20241211fourier.nb`

   を入手し、適当な場所 (デスクトップとか…以下デスクトップに置いた場合で説明する) に保存する [1]。

ターミナルで入手して (ためしに) デスクトップにコピー

```
curl -O https://m-katsurada.sakura.ne.jp/fourier2024/guitar-5-3.wav
curl -O https://m-katsurada.sakura.ne.jp/fourier2024/20241211fourier.nb
cp -p guitar-5-3.wav 20241211fourier.nb ~/Desktop
```

2. (1) で保存した `20241211fourier.nb` をダブルクリックして Mathematica を起動する。メニューの [評価] から [ノートブックを評価] を選択して実行する (ファイルを置いたのがデスクトップでなければ、実行前に `SetDirectory["~/Desktop"]` を適当に直すこと)。
3. この後は説明を聴き、時々指示に従い Mathematica を操作すること。

[1] 例えば、Safari では、control を押しながらクリックして、「リンク先のファイルを別名でダウンロード」を選択して、デスクトップを選択すれば良い。

4.1.2 WAVE ファイルを読み込み、標本化データを取り出す

```
fname="guitar-5-3.wav"
snd=Import[fname, "Sound"]
```

これでファイル guitar-5-3.wav を変数 snd に読み込む (snd は sound のつもり)。ボタンを押すと音が再生できる。

```
sr=Import[fname,"SampleRate"]
```

これは 44100 となるはずである (sr は Sample Rate のつもり)。音楽用 CD と同じ、44.1 kHz というサンプリング・レートで録音したことを示している。

```
sl=Import[fname,"SampledSoundList"];
samples = sl[[1,1]];
```

左チャンネルの音の標本化データを変数 samples に代入した。

右チャンネルが必要なら rsamples=sl[[1,2]] または {samples,rsamples}=sl[[1]] とする。(注: モノラルの場合は sl[[1,2]] は存在しない。)

ちなみに sl[[2]] はサンプリング周波数である。

4.1.2 WAVE ファイルを読み込み、標本化データを取り出す

`snd=Import[fname, "Sound"]` でインポートすると

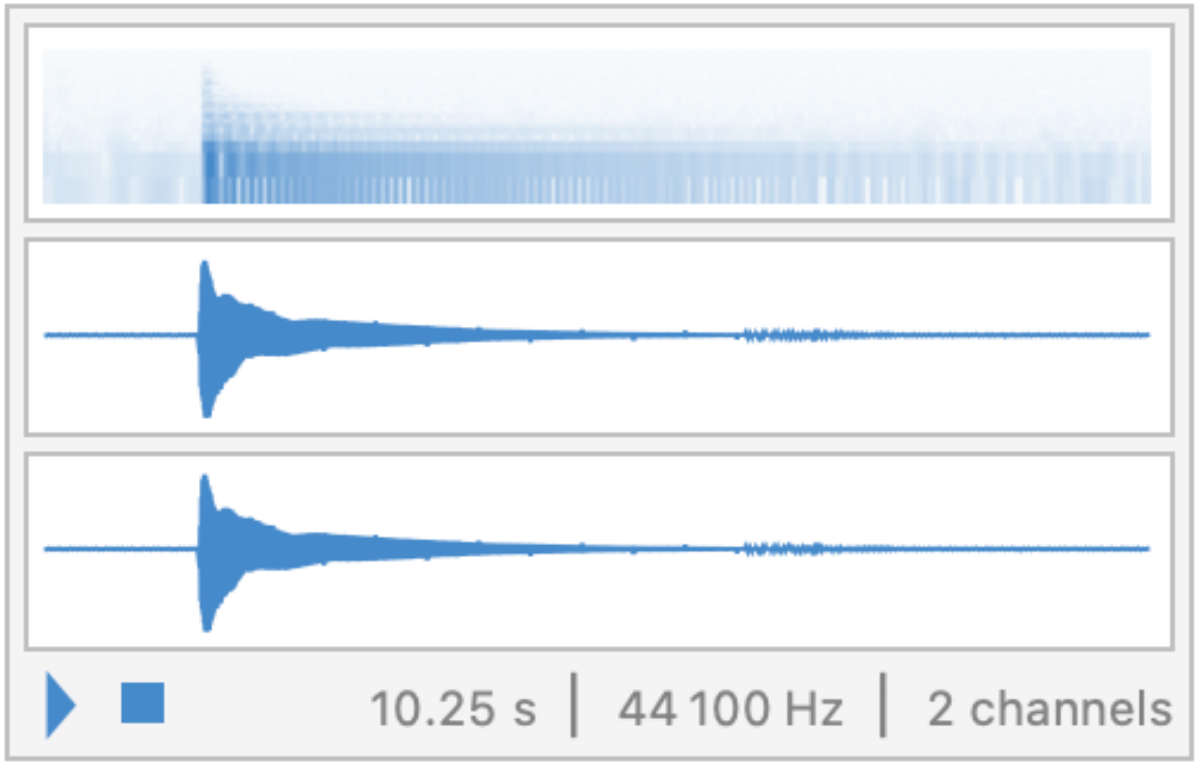


図 1: 再生ボタン ▷ を押すと音が聞こえるはず

2 チャンネル分の波形データが見える

4.1.2 WAVE ファイルを読み込み、標本化データを取り出す

```
s3 = Take[samples, {1, 3*sr}];
g1 = ListPlot[s3, PlotRange->All]
```

samples から 3 秒分のデータ (長さは Sample Rate の 3 倍) を取り出してプロットしてみた。

(サンプリング周波数が sr kHz なので、1 から 3*sr で、3 秒分のデータということになる。Take[] はリストから、指定した範囲のデータを取り出す関数である。)

```
x = Take[samples, {62800+1, 62800+sr}];
g2 = ListPlot[x, Joined->True, PlotRange->{{1, 1600}, {-0.3, 0.3}}]
```

音が鳴り始めるのは 62800 番目辺りからなので、そこから 1 秒分 (sr$\times$1 s = 44100 個のデータを) 取り出して、1600 個分 (1600/44100 $\fallingdotseq$ 0.036 秒分) プロットしてみた。ここは色々試してみると良い。

```
ListPlay[x, SampleRate->sr]
```

とすると、取り出したデータ x の音を鳴らすことが出来る。

4.1.2 WAVE ファイルを読み込み、標本化データを取り出す

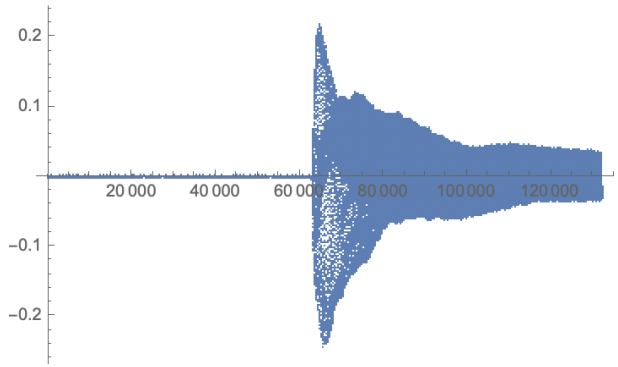


図 2: 最初から 3 秒分プロット

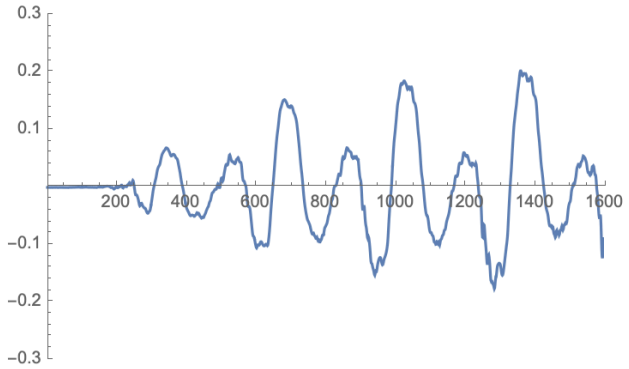


図 3: 62800 から 1600 個プロット

無音部分がある

1600/44100 ≒ 0.036 秒

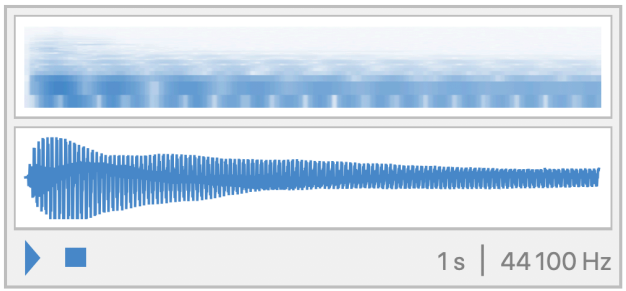


図 4: 62800 から 1 秒分の標本化データで再生

4.1.3 離散 Fourier 変換して周波数を調べる

sr= 44100 を N とおく。x= $(x_0, x_1 \cdots, x_{N-1})^\top$ とおく。x の離散 Fourier 変換 c $= (C_0, C_1, \cdots, C_{N-1})^\top$ を求めるには Fourier[] を使えば良い。

```
c = Fourier[x];
ListPlot[Abs[c], Joined->True, PlotRange->All]
```

絶対値 $|C_n|$ をプロットした。これから周波数成分の分布が読み取れるはず…

しかし、N は大きすぎて分かりにくいので、範囲を指定して表示すると良い。

```
(* n1〜n2 の範囲で |c[[n]]| をプロットする。 *)
graph[c_,n1_,n2_] := ListPlot[Abs[c], Joined -> True,
        PlotRange -> {{n1, n2}, {0, Max[Abs[c]]}}]

graph[c, 1, 1600]
graph[c, 120, 140]

Manipulate[graph[c,n1,n2],
  {n1,1,Min[Length[c],2000],20},  {n2,1,Min[Length[c],2000],20}]
```

4.1.3 離散 Fourier 変換して周波数を調べる

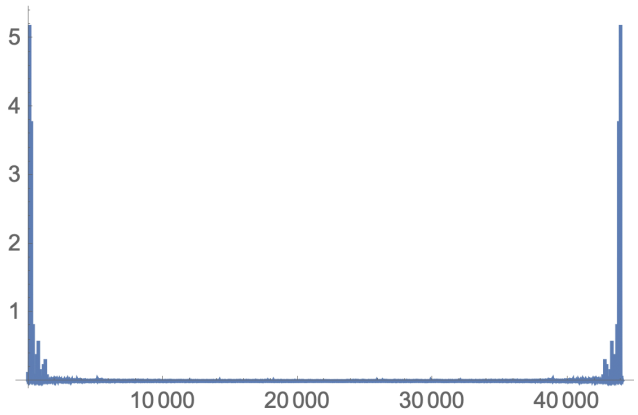


図 5: $|C_n|$ $(n = 0, 1, \cdots, 44099)$ をプロット

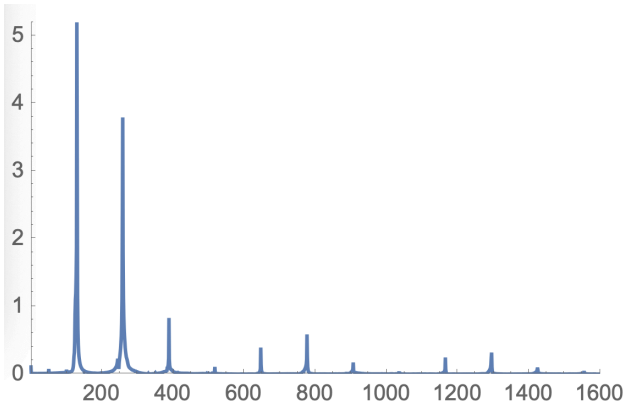


図 6: $|C_0|$, $|C_1|$, $\cdots$, $|C_{1599}|$

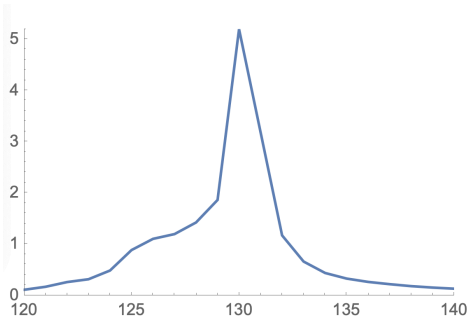


図 7: $|C_n|$ $(n = 119, \cdots, 139)$ をプロット

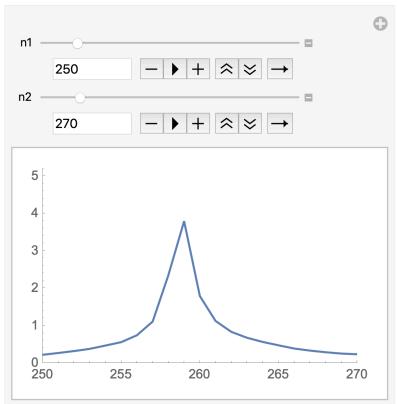


図 8: $|C_n|$ $(n = 249, \cdots, 269)$ をプロット

4.1.3 離散 Fourier 変換して周波数を調べる

範囲を区切って表示することで、ピークを探してみた (左右対称なので左側だけで探す)。素朴に目で見て探したが、プログラムを書いて自動化することも難しくないであろう。

ピークは 130 番目である。つまり $|C_{129}|$ が最大ということである。(リスト c の要素 c[[1]], c[[2]], $\cdots$, c[[N]] は、C_0, C_1, $\cdots$, C_{N-1} であり、リストの要素の番号と Fourier 係数のインデックスが 1 ずれていることに注意する。)

これはこのギターの音の基本周波数が 129 Hz であることを意味する (1 秒分のデータを取って離散 Fourier 変換したから。つまり周期を 1 秒と仮定して分析している。)。これはドの周波数 131 Hz に近い。

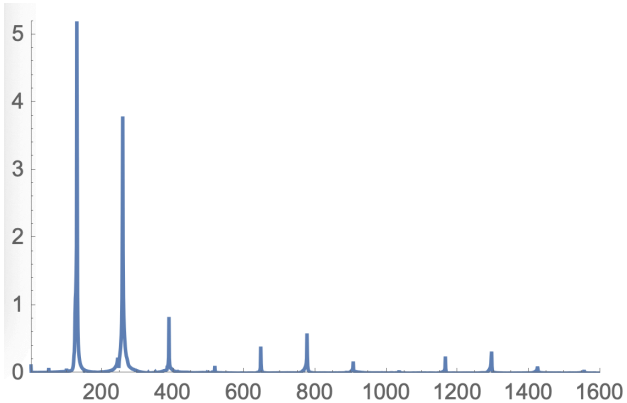


図 6: $|C_0|$, $|C_1|$, $\cdots$, $|C_{1599}|$

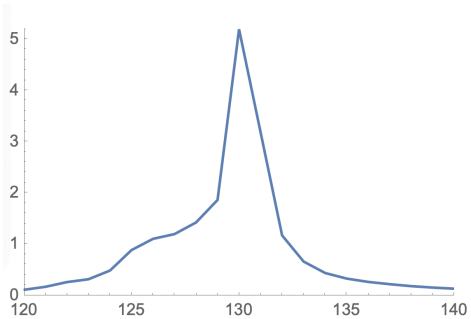


図 7: $|C_n|$ $(n = 119, \cdots, 139)$ をプロット

4.1.4 逆離散 Fourier 変換を試す

逆離散 Fourier 変換を計算する `InverseFourier[]` という関数が用意されている。

逆離散 Fourier 変換して元に戻るか？

```
x2=Re[InverseFourier[c]];
Norm[x-x2]
ListPlay[x2,SampleRate->sr]
```

元の `x` は $\mathbb{R}^N$ の要素 (つまり成分が実数のベクトル) であるが、丸め誤差が発生するため、本来は 0 であるはずの虚数部分が 0 にならず、`InverseFourier[c]` は $\mathbb{R}^N$ には属さないかもしれない。そこで `Re[]` を使っている。(`Re[]` を施さなくても `x2` は十分 `x` に近いが、`ListPlay[]` で音が出ない。)

特定の周波数に対応する C_n を 0 に変更してから逆離散 Fourier 変換すると、その周波数の音をカットすることが出来る。

今回の授業では「おまけ」としておくが、次のノートブックで実験できる。

`https://m-katsurada.sakura.ne.jp/fourier2024/piano-cutoff.nb`

(Mathematica のバージョンによっては、用意したプログラムが正常に動かなくなったりします。動かなかったら報告して下さい。)

Mathematica メモ

(第 2 回の授業でも簡単なメモを提供しました。)

- `In[数] :=` と言うプロンプトに対して、コマンドを入力して、最後に shift + ↵
- `??`関数名 でオンライン・ヘルプが呼び出せる。
- Mathematica で定義済みの関数・定数の名前は「大文字で始まる」というルールを守って決めてある。`Pi`, `Plot[]`, `Fourier[]` などなど。複数の単語からなる名前は、各々の単語の先頭の文字を大文字にする。`InverseFourier[]`, `ListPlot[]` などなど。ユーザーが自分で変数・関数を定義するとき、名前の先頭の文字を小文字にすると、名前の衝突が防げる。

  例: `N=1` としたらおかしい。 →大文字はやめて `n=1` のように小文字にする等。
  (注意: `N[式]` で式が表す数の近似値を求める関数 `N[]` と名前が衝突している。)
- 変数名 `= Import["Wave ファイル名", "Sound"]` で Wave ファイルを読み込んで変数に代入できる。
- Mathematica では、ベクトルや行列はリストで表す。
  `x={1,2}` とか `a={{1,2},{3,4}}` など。
  `x` の第 1 成分は `x[[1]]` で、`a` の第 $(2,1)$ 成分は `a[[2,1]]` で表せる。
- リスト `list` の第 `n1`〜`n2` 要素を取り出すには、`Take[list, {n1,n2}]` とする。

Mathematica メモ

- `c := Fourier[x]` で、ベクトル (数値のリスト) `x` を離散 Fourier 変換したベクトル (数値のリスト) が `c` に得られる。
  実は Mathematica における離散 Fourier 変換の定義は、この講義の定義とは異なる。この講義の流儀に合わせるには、次のようにすれば良い。

```
c = Fourier[x, FourierParameters->{-1,-1}]
```

- `x = InverseFourier[c]` で、ベクトル (数値のリスト) `c` を逆離散 Fourier 変換したベクトル (数値のリスト) が `x` に得られる。
  この講義の流儀に合わせるには、

```
c = InverseFourier[x, FourierParameters->{-1,-1}]
```

- `ListPlot[数値リスト]` でプロット可能。`,Joined->True` は良く使う。
  逆離散 Fourier 変換で得た数値リストを扱うとき、丸め誤差により、本来実数であるものが丸め誤差により (虚部の絶対値の小さい) 虚数になってうまく処理できない (音が鳴らせない等々)。そのときは

```
x = Re[InverseFourier[c]]
```

  のように実部を取る (虚部を 0 にクリアする) と良い。
- `ListPlay[数値リスト, SampleRate->サンプリング周波数]` で再生できる。

Mathematica メモ

細かいところは、Mathematica のバージョンによって動作が異なります。うまく行かない場合、一人で悩まず、気軽に質問して下さい。

4.2 PCM による音のデジタル信号表現

音とは空気中を伝播する縦波である。音があるとき、気圧が時間変化する。音がないときの気圧 (基準圧力) からの変位を**音圧**と呼ぶ。音圧の時間変化を記録することで音を記録 (録音) 出来る。

PCM (pulse code modulation, パルス符号変調) とは、アナログ信号 (連続変数の関数) をデジタル信号 (離散変数の関数 — 数列) で表現するための 1 つの方法であり、音楽用 CD, コンピューターのデジタル・オーディオ, デジタル電話等で標準的な形式となっている。具体的には、次の二つに基づく。

(a) 一定の時間間隔で信号の値を記録する　(**サンプリング (標本化)** する、という)

(b) 信号の値は有限桁の数で表現する (**量子化**する、という)
特に値の属する区間を等間隔に区切って、もっとも近い値に丸めることで実現するとき、**LPCM** (linear PCM) という。

コンピューターで処理することを考えると、「有限桁の数」は、「2 進法の有限桁の数」ということになるが、その際の桁数 (ビット数) を**量子化ビット数**と呼ぶ。

8 ビットの場合は $2^8 = 256$ 段階、16 ビットの場合は $2^{16} = 65536$ 段階で表現することになる。音楽用 CD (1980 年に SONY と Phillips により規格化された [2]) では、量子化ビット数として、16 ビットが採用された。

[2] 音楽用の CD プレーヤーが初めて販売されたのは 1982 年である。当時ようやく 16 ビット CPU を用いたパソコンが市販された頃であった。外部記憶装置としては、1.2 MB の容量のフロッピー・ディスクが広く使われていた。

4.2 PCM による音のデジタル信号表現

コンピューターで音を記録 (録音) する場合は、電気的な信号に変換された音声を数値化して (アナログ信号からデジタル信号に変換するので **AD 変換** (analog-to-digital conversion) という)、数値を記録することになる。

サンプリング周波数とは、1 秒間に何回サンプリングするかを意味している。サンプリング周波数が高いほど、より高い周波数の音が記録できるようになる。

音楽用 CD では、サンプリング周波数 44.1 kHz が採用された。これは、1 秒間に 44100 回データを記録するということになる。この値が採用された理由は主に次の理由による。

(a) 人間が普通聞くことが出来る音の周波数は 20 Hz ～ 20 kHz と言われている。

(b) サンプリング周波数は、再生したい最も高い音の周波数の 2 倍より高くする必要がある (これは後で解説する**サンプリング定理**を根拠とする、と説明されるが、前々回の授業で紹介した Fourier 係数に関するサンプリング定理で説明することも出来るだろう。)。

つまり、人間が普通に聞ける音の記録・再生のためには、サンプリング周波数は $2 \times 20\ \mathrm{kHz} = 40\ \mathrm{kHz}$ より高くする必要がある。

サンプリング周波数と量子化ビット数の値が大きいほど、元の信号により忠実なデータが得られるが、もちろんデータの量はそれだけ増大する。

余談 1 (CD プレーヤー発売当時のパソコン技術の相場)

音楽用 CD では、ステレオ (2 ch) が普通なので、1 秒間あたり、

$$44.1\ \mathrm{k} \times 16\ \mathrm{b} \times 2 = 1411.2\ \mathrm{kb} = 176.4\ \mathrm{kB} = 172.266\ \mathrm{KB}$$

のデータが流れることになる (b はビット、B はバイト (=8 ビット)、$\mathrm{K} = 2^{10} = 1024$)。1 分間では、その 60 倍の 10.0937 MB (ここで 1 MB = 1024 KB という意味) が流れることになる。

1982 年当時に普及していたリムーバブルな外部記憶媒体であるフロッピー・ディスクの容量は 1.2 MB 程度だったので、1 分の音声信号を記録するのに、10 枚近いフロッピー・ディスクが必要だったことになる。これでは全然実用的ではない。CD という新しいメディア (74 分程度記録出来るようにするため、740 MB の容量となった) が必要になったのは当然のことである。

音楽用 CD では LPCM で得られたデータをそのまま記録しているが、その後は、圧縮技術が進歩した。1993 年に登場した **MP3** (MPEG-1 Audio Layer-3) では、圧縮によって、データのサイズを $\frac{1}{10}$ 程度まで小さくすることが出来るようになった。 □

4.2 PCM による音のデジタル信号表現

WAVE (WAV) は、Microsoft と IBM により策定された音声データ用のフォーマットである。ファイルの拡張子は `.wav` である。通常は圧縮なしの、LPCM でサンプリングしたデータが使われる (規格上はデータ形式は自由で、圧縮データも格納しうるそうであるが…)。

4.3 結果の分析　4.3.1 一般論の復習 (ゆっくり)

音声信号を扱うとき、独立変数は (時刻なので) t で表し、信号の値そのものは x で表す、つまり信号を $x(t)$ で表すことが多いので、ここでもそれに従う。

周期 T の周期関数 $x\colon \mathbb{R} \to \mathbb{C}$ は次のように Fourier 級数展開出来る。

$$x(t) = \sum_{n=-\infty}^{\infty} c_n e^{in\frac{2\pi}{T}t} \quad (t \in \mathbb{R}), \tag{1}$$

$$c_n = \frac{1}{T}\int_0^T x(t) e^{-in\frac{2\pi}{T}t}\, dt \quad (n \in \mathbb{Z}). \tag{2}$$

基音 ($n = \pm 1$) の周波数は、周期の逆数 $f = \dfrac{1}{T}$ である。第 n 項の周期は $\dfrac{T}{|n|}$, 周波数は $|n|f$. n_0 倍音の周波数 $n_0 f$ に対応するのは、$n = \pm n_0$ の項である。

音声信号では、x が実数値関数なので、$\overline{c_n} = c_{-n}$ が成り立つ。特に $|c_{-n}| = |c_n|$. 実際

$$\overline{c_n} = \overline{\frac{1}{T}\int_0^T x(t) e^{-in\frac{2\pi}{T}t}\, dt} = \frac{1}{T}\int_0^T \overline{x(t)}\, \overline{e^{-in\frac{2\pi}{T}t}}\, dt = \frac{1}{T}\int_0^T x(t) e^{-i(-n)\frac{2\pi}{T}t}\, dt = c_{-n}.$$

(Cf. 実数値関数 f の Fourier 変換 $\widehat{f}$ に対して、$\overline{\widehat{f}(\xi)} = \widehat{f}(-\xi)$ が成り立つ。)

4.3.1 一般論の復習 (ちゃっちゃと)

一周期区間 $[0, T]$ に N 回測定 (サンプリング) すると、サンプリング周期 $T_s = T/N$, サンプリング周波数 $f_s = \frac{1}{T_s} = \frac{N}{T}$ でサンプリングすることになる。

区間 $[0, T]$ の N 等分点 $t_j := jT_s$ での x の値 $x_j = x(t_j)$ を用いる。このとき離散 Fourier 係数 C_n は次式で与えられる。

$$C_n = \frac{1}{N} \sum_{j=0}^{N-1} x_j \omega^{-nj} \quad (n \in \mathbb{Z}), \quad \omega = e^{2\pi i/N}. \tag{3}$$

離散フーリエ係数 $\{C_n\}$ は周期数列なので、連続する N 項 $\{C_n\}_{n=0}^{N-1}$ を求めれば良い。

離散フーリエ係数 $\{C_n\}_{n=0}^{N-1}$ は、$\{c_n\}$ と次の関係を持つ:

$$C_n = \sum_{p \equiv n} c_p.$$

$\{C_n\}$ から $\{x_j\}$ を求めるには、逆離散 Fourier 変換すれば良い:

$$x_j = \sum_{n=0}^{N-1} C_n \omega^{jn} \quad (j = 0, 1, \ldots, N-1). \tag{4}$$

4.3.2 参考: 1 次元の弦の振動

長さ L の弦の (微小な) 振動は、次の波動方程式の初期値境界値問題をモデルに持つ。

$$\frac{1}{c^2}u_{tt}(x,t) = u_{xx}(x,t) \quad (0 < x < L,\ t > 0)$$
$$u(0,t) = u(L,t) = 0 \quad (t > 0)$$
$$u(x,0) = \phi(x), \quad u_t(x,0) = \psi(x) \quad (0 \le x \le L).$$

$u = u(x,t)$ は、釣り合いの位置 x にあった点の時刻 t における変位を表す。

T を張力、ρ を線密度 (単位長さあたりの質量) として、c は $c = \sqrt{T/\rho}$ で与えられるが、これは実は弦を伝わる波の速さに等しい。ヴァイオリンやギターなどでは、張力を変えることで音の高さを調整することが出来る。

この問題の解は次式で与えられる。

$$(\sharp) \qquad u(x,t) = \sum_{n=1}^{\infty} \sin\frac{n\pi x}{L}\left(a_n \cos\frac{cn\pi t}{L} + b_n \sin\frac{cn\pi t}{L}\right),$$

$$a_n = \frac{2}{L}\int_0^L \phi(x)\sin\frac{n\pi x}{L}\,dx, \quad b_n = \frac{2}{cn\pi}\int_0^L \psi(x)\sin\frac{n\pi x}{L}\,dx.$$

$(\sharp)$ の第 n 項の基本周期は $\dfrac{2L}{nc}$, その周波数は $\dfrac{nc}{2L}$. これらは一番低い周波数 $\dfrac{c}{2L}$ ($n = 1$ に対応する) の整数倍である。 □

4.3.3 今回の実習では

サンプリング周波数 fs = 44.1 kHz でサンプリングしたデータから、$T = 1$ s (1 秒) 分の信号 ($N = f_s T = 44100$ 個の数値) を取り出して離散フーリエ変換した。

周期 $T = 1$ s の周期信号とみなして Fourier 級数展開したことになる。

実信号の Fourier 係数には $\overline{c_n} = c_{-n}$, $|c_n| = |c_{-n}|$ という関係があるが、離散フーリエ係数については、次の関係がある。

$$\overline{C_n} = C_{-n} = C_{N-n}, \quad |C_n| = |C_{-n}| = |C_{N-n}|. \tag{5}$$

横軸 n ($0 \leq n \leq N$), 縦軸 $|C_n|$ でプロットすると、左右対称になる。

Fourier 級数展開 (1) の第 n 項の周期は $\dfrac{T}{|n|}$, 言い換えると周波数は $\dfrac{|n|}{T} = |n|$ Hz である ($T = 1$ s としてあることを思い出そう)。

c_1 と c_{-1} は (C_1 と C_{N-1} は) 1 Hz の成分、c_2 と c_{-2} は (C_2 と C_{N-2} は)2 Hz の成分、$\cdots$

$|C_{129}| = |C_{N-129}|$ が最大ということは、周波数が 129 Hz の成分が最大ということを意味している。

その次に大きいのは $|C_{258}| = |C_{N-258}|$ であった。258 Hz の成分がその次に大きいことを意味している。一番低い 129 Hz の整数倍になっているのは、ギターが 1 次元の振動現象であることから、もっともである。

4.3.4 参考　音階

1 オクターブ高い音の周波数は 2 倍である。

西洋の音階では、1 オクターブは半音 12 個分に相当する (C, C#, D, D#, E, F, F#, G, G#, A, A#, B)。

平均律では各音の周波数が等比数列になる (というよりも、そうするのが平均律である)。よって、半音高いと周波数は $2^{1/12} = 1.05946\cdots$ 倍になる。ピアノの鍵盤の中央付近にある $\overset{アー}{\mathrm{A}}$ (ラ) の音は、普通 440 Hz に調律されるので、その下の $\overset{ツェー}{\mathrm{C}}$ (ド) の音 (半音 9 個分低い) の周波数は

$$\frac{440}{2^{9/12}} = 261.6255653\cdots \text{ Hz}.$$

上の実験のギターの音は、これより 1 オクターブ低い、$\frac{261.6}{2} = 130.81\cdots$ を目安に調律したのであろう。実際に $|C_{129}|$ が最大になったのは、まあまあのチューニングだったのであろうか (ゼミ生にやってもらいました)。 □

4.3.5 より精密に

(この項書きかけです。すみません…)

実際には、周波数は自然数に限られるわけではない。その場合は $T = 1\,\mathrm{s}$ は、その信号の周期にならない可能性がある。

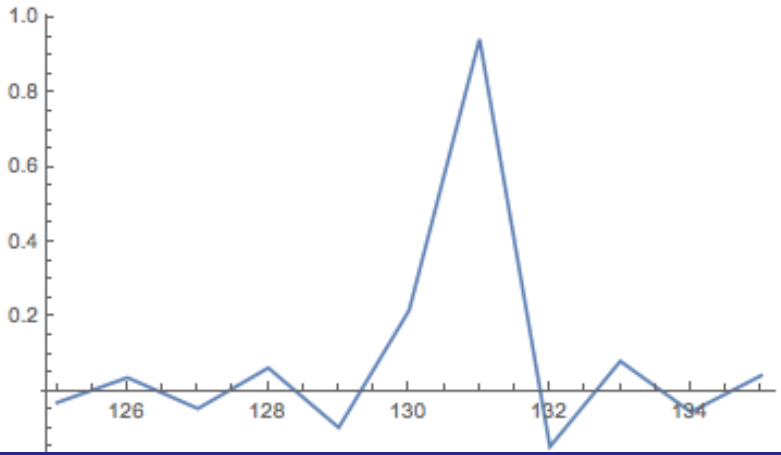
簡単のため、周波数 f の信号 $u(t) = e^{2\pi i f t}$ を考える。$0 \le t \le T$ で記録して、(周期 T の周期関数としての) Fourier 係数を求めると、

$$c_n = \frac{1}{T}\int_0^T u(t)e^{-in\frac{2\pi}{T}t}dt = \frac{1}{T}\int_0^T e^{2\pi i(f-n/T)t}dt = \frac{1}{T}\int_0^T e^{iA_n t}\,dt = \frac{1}{iA_n T}\left(e^{iA_n T} - 1\right).$$

ただし $A_n := 2\pi\,(f - n/T)$. これから

$$|c_n| = \mathrm{sinc}\,\frac{A_n T}{2}.$$

$T = 1\ \mathrm{s},\ f = 130.813\ \mathrm{Hz}$ のとき、$n = 125, \cdots, 135$ の範囲で $\mathrm{sinc}(A_n T/2)$ を調べると、



参考: Mathematica の Fourier[] における離散 Fourier 変換の定義

実は Mathematica の Fourier[ ] がデフォールトで計算するのは (C_n ではなく)

$$C'_n := \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega^{+nj}$$

である。C_n を計算させるには、FourierParameters->{-1,-1} というオプションを与えれば良い。

```
c=Fourier[tb, FourierParameters->{-1,-1}];
```

ところで、信号が実数値であることから、

$$C_n = \frac{1}{\sqrt{N}} \overline{C'_n}$$

という関係が成り立つ [3]。ゆえにパワースペクトル (絶対値の 2 乗のこと) については、$|C_n|^2 = \frac{1}{N} |C'_n|^2$ であるから、周波数を求めたりする場合は (パワースペクトルが大きくなる n はどこか調べる)、デフォールトのまま使っても良い。

[3] $\overline{x_j \omega^{nj}} = \overline{x_j}\,\overline{\omega^{nj}} = x_j \omega^{-nj}$ であることに注意せよ。

Python での実験

最近は Python に慣れている学生も多いので、Python でするための情報をつけておきます。

「音の取り扱いに関するメモ　Python」
https://m-katsurada.sakura.ne.jp/labo/text/memo-sound/node34.html

特に
「DFT してスペクトルを調べる」
https://m-katsurada.sakura.ne.jp/labo/text/memo-sound/node37.html

には、今回の実験とほぼ同じことを Python でするためのプログラムが載っています。

https://m-katsurada.sakura.ne.jp/lecture/fourier-2024/spectrum.py

参考文献

[1] 桂田祐史：「信号処理とフーリエ変換」講義ノート, https://m-katsurada.sakura.ne.jp/fourier/fourier-lecture-notes.pdf, 以前は「画像処理とフーリエ変換」というタイトルだったのを変更した。 (2014〜).