脳波解析のための数学シリーズ
基礎編

後藤 優仁

2021 年 6 月 26 日

目 次

0.1 本書の目的 3
0.2 本書のカバーする内容 4

第 1 章 掛け算 **6**
1.1 掛け算とは何か 6
1.1.1 数ってなぁに？ 6
1.1.2 単一の数スカラー 6
1.1.3 複数の数からなる積 7
1.1.4 ベクトルの積 8
1.1.5 内積の意味 10
1.1.6 ドット積 11
1.1.7 複素数の積 11
1.1.8 複素数の逆数 12
1.1.9 複素平面の掛け算の意味 12
1.1.10 大正義オイラーの公式 12

第 2 章 三角関数 **14**
2.1 三角比とは 14
2.2 三角比の公式 15
2.3 三角関数 15
2.4 加法定理 17
2.5 正弦と余弦 17
2.6 関数の内積 18
2.7 関数の直行性 19
2.8 直交性 19
2.8.1 フーリエ変換への一歩 20
2.9 大正義オイラーの利用 21

第 3 章 複素数 **22**
3.1 虚数 22
3.2 虚数の性質 23
3.3 複素数 23
3.4 数の大幅な拡張 24
3.5 複素数平面 25

3.6 極形式 27
3.7 大正義オイラー再び 27
3.8 複素数の掛け算 28
3.9 戻り学習 28
3.10 まとめ 28

第 4 章 微分・積分 **30**
4.1 極限とは 30
4.2 ε-δ 論法 30
4.3 微分 32
4.3.1 微小な変化率 32
4.3.2 三角関数の微分 34
4.3.3 ネイピア数 35
4.4 積分 37
4.4.1 足し算としての積分 38
4.4.2 関数の内積 39
4.4.3 畳み込み積分 39
4.4.4 つまり畳み込みとは 41

第 5 章 統計学 **45**
5.1 基本統計量 45
5.1.1 正規化 46
5.2 相関と共分散 47
5.2.1 相関と散布図 48
5.2.2 回帰 50
5.2.3 共分散 51
5.2.4 相関係数 52

0.1　本書の目的

本書は，主に脳波を用いた神経科学者を志す諸君のため，脳波解析に特化した数学の入門資料となることを望んで作成するものです．神経科学は学際的な学問領域であるため，数学的素養のある人間は勿論，高校以降数学に長らく触れていなかったもの，あるいは高校数学も物理も身を入れて学んでいない者と様々なバックグラウンドの人間が流入します．自分のような半端者と異なり医者なども多いが，彼らもやはり同様に数学的な知識については疎い事は往々にしてあります．ここで言う知識とは，単に計算が出来たり，定理を暗記している事ではありません．数学を言語的に解釈し,「つまり○○をしている」「○○に使える」といった発想が出来る事をさします．よって，この意味でも「数学くらいできるし」という考えの人も半分程度はつまずくはずです．本書の目的は，そんな諸君のため厳密な数学の議論は数学書に譲りつつ，各種数学的処理の「気持ち」の理解を提供する事にあります．本書の記述にもやもやしたものを感じる人はきっと，著者よりも数学的素養，素質のある人でしょう．申し訳ないが気になったところは適宜専門の数学書を参照されたい．

電気生理的な神経科学では，脳波という特殊な時系列データを解釈するために種々の数学的手段を用います．誹りを恐れず大雑把に，大きく，大きく分類するなればこれらは三種に分けられるでしょう．一つ目は信号処理，あるいは解析学．その基礎となるのが波の分解，周波数分解であり，またこの背景には偉大なオイラーの公式があります．オイラーの公式を理解すれば，一つ目の課題はほぼクリアしたと言っても良いはずです．実際，オイラーの公式と，あとは畳み込み積分させ理解していれば脳波解析に苦労することはなくなるはずです．

二つ目の分野は統計です．解析的手段によって得られた脳波の特徴量を，被験者間，タスク間，時間...様々な条件で比較し，どんな条件の時にどこにどんな違いが現れるのか，その違いは大事なものなのか...そんな議論をするために必要な学習です．これがなければ，解析して得られたデータなどたいていの場合は何の役にも立たちません．あとはまぁ，脳もなにかしらの統計的学習をしているよねという議論はよくなされているし，最近だと Free Energy Principle もそうですが脳の従っているなんらかのルールを記述するような研究をかじっていくにはどうしても必要な知識になります．やっぱベイズは大事ですよねえ，ベイズ．筆者も勉強しなければと思う次第です．

三つ目，簡単に言えば「その他」になります (怒られそう)．つまりこれらは，必ずしも全員に必要なものではないが，脳を理解するため，特に今後の脳神経科学研究においては必要になってくるであろう数学群で，自由エネルギーやら統合情報量やらに代表されるような脳の仕組みに関する理論や，グラフ理論や力学，情報学といった新しい脳情報解析に利用される数学などです．必要になった時に学べば良いと思いますが，やはりこうした議論を何も理解していない者とそうでない者とには今後大きく差がつくであろうし，日々勉強する事は大切であるように思います．本書は解析や統計に関する記述を優先的に補充していきますが，それ以外の数学については筆者の好みで足していこうと思います．(2021.5.15)

以上のような本書の目的に従い，文中では (特に後半)，数式のみでなくこれらの計算を実際に手元の PC にやらせるためのソースコードも記載していく事とします．言語は迷いましたが，ひとまずは MATLAB を使用し，必要を感じれば別途 Python のソースコードを用意します．理由としては，筆者の所属ラボ含め神経科学の多くの研究室は脳波や脳磁，核磁気共鳴画像の解析に MATLAB を使用している事，それによって公開されている論文のソースコードも MATLAB で実装されている事，EEGLAB などのツールボックスも充実しているため使うのが楽である事，などがあります．またコードは文中には minimum で載せますが完全版というか，資料に載せているような plot をだすコードは同じファイル名のスクリプト (.m) ファイルで用意しておきます．気になる人はダウンロードするなりして自分の環境で実行して

ください．なお筆者のネイティブはPythonなので宗教戦争には巻き込まないでください．いやほんと，ＭＡＴＬＡＢダサいしなんか気持ち悪いし嫌いです．ほんと．まじ嫌い．

さて筆者のプロフィールですが, 脳神経科学歴1年 (2018年9月), 数学歴半年のぺーぺーです. 数学に至っては半年前は虚数が何か分からなかったし三角関数の意味も知りませんでした.
よって変な理解だとか知識の抜けもあると思うのでご指摘お待ちしています.

2021/5現在は生理学研究所の博士課程です.

0.2　本書のカバーする内容

本書は筆者の気が向いた時に少しずつ更新していくので, タイミングによってはほとんど書いてないと思いますが, 最終的に目指している点をここに記します.

基本的には脳神経科学, 特に脳波を用いる際に必要になってくる数学を全体集合として, 部分集合を小出しにだしていく感じを考えています. 補集合については更新を待つか諦めて独学をする, あるいは筆者に書くよう申請をしてください. 従って本書のカバーする内容はおおまかに分けて

- 前提となる基礎的な数学
- 脳波を代表とした信号を処理する信号処理数学
- ANOVAやt検定, 簡単な機械学習などの統計的数学
- その他愉快な仲間たち

などといった感じになると思います. 初学者，特に実験系の人達の考える事は痛いほどに分かります.「別に数学の研究したいわけじゃないし」ですよね．しかしこれら基礎が分かっている人と分かっていない人とでは，まず先行研究に対する批判的な読み方に大きく差がつき，それが理解の差につながります. 当然，転じていざ自分の研究を計画する時の綿密さにも差が付き，手を動かして解析し統計的議論をする頃には覆しえない程の差がついている事でしょう.
うん，言われても納得しがたいとは思いますが，実際一度研究をしてみれば分かると思います．後から「え... これ意味ないですよね」とか「この解析，信号に○○性の仮定が必要だと思うんですが今回の実験で得られるデータは○○性を満たさないのではないですか？」などと学会の発表でマサカリを投げられて致命傷を負う読者諸君が見えます．飯うまい．僕もやられたし．ほんとあれしんどいです．勉強しとけ．まじで.

頑張って行きましょう.

いよいよ始まりますが，まず最初のこの basic.pdf はこれ以降の議論を展開するために最低限身に着けておいて欲しい数学的知識，考え方です．心して確認してください．

第1章　掛け算

はじめに，基礎を学ぶ上で必要になる基礎の勉強，というか確認から始めます．高校以降は文系よりだった人や，高校でも数学はやってたけど受験のための暗記数学をやっていて，正直数学についてあまり考えたことがない人向けです．以降の章でやっていく内容を読んでいく上で最低限覚えておいてほしい内容です．

1.1　掛け算とは何か

信号処理や統計といった難しい数学をこれから皆さんはやっていくわけですが，こういった数学は結局のところどこまで行っても足し算と掛け算です．脳波の解析も統計も，複雑な「数」に対応した，複雑な「足し算，掛け算」をやってあーだこーだしていく処理にすぎません．なのでまずはここで，掛け算とは何かを考えていきましょう．足し算については比較的簡単であまり違いを意識しなくても良いので，ここでは掛け算のみ触れます．

伝えたいメッセージは，世の中には色々な「数」があって，対応した色々な「掛け算」がある．という事です．まずはこの事をしっかり頭に入れておけば，解析とかの勉強もさして難しくないはずです．別に全て覚える必要はないので，とにかく色々あるんだなーとだけ感じてください．

1.1.1　数ってなぁに？

まず初めに，この書において扱う数はユークリッド空間のものに限ります．非ユークリッドを持ち出してねちねちと文句は言わないでください．だから嫌われるんですよ．あぁすみません，嫌われている自覚がなかったんですね．余計な事を言ってしまって申し訳ないです．さて，先程ユークリッド空間や非ユークリッド空間といった言葉が出てきましたが, 簡単にいうとこれらは全て別の「数」として考える必要があるため, 空間と名付けられています．そもそも,「数」ってなんだろう？最初はそんなところから考えてみましょう．
まずは単一の数と複数の数に分けるところからいきます．

1.1.2　単一の数スカラー

単一の数, スカラー数です．高校での数学を真面目に受けていた人間なら聞き覚えがえるでしょう．スカラー数とは最も簡単な数... すなわち僕達が昔から親しんできた「数」のことです！
簡単ですね！以上です．

スカラー数の掛け算は僕たちが直感的にやっている

$$3 \times 5 = 15 \tag{1.1}$$

の事です.

Listing 1.1: 式 (1.1) の MATLAB コード

```
3 * 5
```

以降，ソースコードはコード (1.1) のように記載していきます．式では変数としているものも，実際に数字がないと計算できないので好きに補完しつつ実装してみてください.

1.1.3 複数の数からなる積

複数の数からなる数には色々あります.

- ベクトル
- 複素数
- 行列

等です．これらはスカラー数の掛け算とは全く違う世界になります．心してかかりましょう．高校以上の数学は非常に難しいもので, 僕も全く分からずに置いて行かれましたが, どうにか理解するための最初の第一歩は，「世の中には (スカラー) 数じゃない数があるらしい」と知る事です．僕はこれを理解するまでに無駄に長い時間を費やしてしまいました.

では, そもそも何故スカラー以外の数を考える必要があるのでしょうか？
普通に生きていてベクトルや複素数といったものを自然の中に見る事はないでしょう (あるいは啓蒙を高めると見る事が出来るのやもしれない).

(追記: 2019 年 9 月現在の筆者は日常にたくさん見るようになった．啓蒙が高まったのやもしれん．)
(追記: 2021 年現在の筆者は上の記述が恥ずかしい．当たり前やんけ)

しかしスカラー数には限界があるのです．何故なら, 基本的にスカラー数とは直線 (上の点) であるからです！

0 があって 1 があり, その次に 2 が来るのは小学 2 年生の数直線ですが, 小学 4 年生頃には「どうやら 0 と 1 の間にも小数や分数があるらしい」と知ります．次に中学になり, 0 の向こう側には同じだけの数の負の数があり, 更には $\sqrt{}$ がつく数などもある事を知ります．そして少し飛びますが高校では, 数直線の右と左に終わりはなく, 無限という領域に飛んでいる事を知りますね．ここまでがスカラーの学習順序ですが, いずれも 1 本の直線を細かく見たり, あるいはずっと先を見ているだけです.

ただの直線であるスカラーでは記述できる情報が少ないため, 不都合があるから別の数を考える必要があったわけです. スカラーっぽいやつらを複数束ねてまとめた「数」を新しく定義してやれば表現できる情報量は各段に増え, 世界が広がります.

記述する要素が増えれば情報が増えるのは考えれば納得ですね. イメージは中学数学の 1 次関数のグラフとかです. 縦と横, 軸が増えれば情報は増えますね？

1.1.4 ベクトルの積

ベクトルとは, 二次元の場合は 2 つ, 三次元の場合は 3 つの数... を束ねたものです.
数を複数束ねると, その間に比が生まれ, それらによって定義される長さを持った線 (と基準線との角度) が生まれます.
これが先述した, 増える情報量であり, スカラー数との決定的な違いです.
ちなみに, この「比と角度 (と長さ)」ですが, これは脳波においては位相や電位の強度といったものを考える時に必要になります. というかそのままです. よく確認しておいてください.

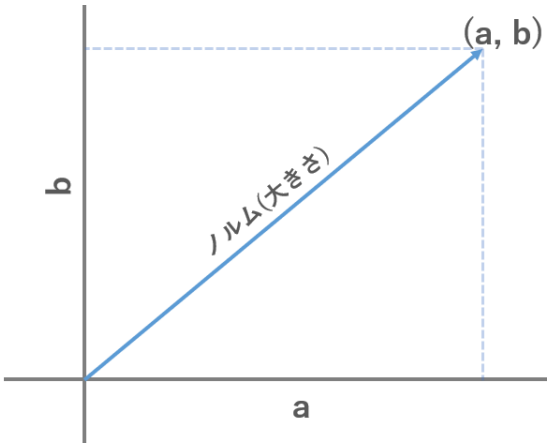

図 1.1: 2 次元ベクトルの例

ベクトル同士を掛け合わせるとき, どうするでしょうか？
一番素朴な形は下記の通りでしょう.

$$(x_1, y_1)(x_2, y_2) = (x_1x_2 + y_1y_2) \tag{1.2}$$

これをベクトルの内積と言います.

Listing 1.2: 1.2 の MATLAB コード

```
x = [1 2 3]
y = [4 5 6]
dot(x,y)
```

そして, ベクトルの内積はもう一つの算出方法がありましたね. 2 つのベクトルの絶対値の積に互いになす角の cos をかけたものです. 確認してみましょう.

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}|cos\theta \tag{1.3}$$

まず，式 (1.3) の見方ですが，太字は数学ではベクトルを表します．なので $\mathbf{a}, \mathbf{b}$ はスカラーじゃないことに注意してください.

さてここからが味噌ですが，普通は上の式を見ると絶対値の積に，$cos\theta$ をかけていると思うでしょうがそうではありません．考え方としては，$|\mathbf{a}|$ に $|\mathbf{b}|cos\theta$ をかけているという方が分かりやすくなります. あ，もちろん a と b を逆にしても大丈夫です.

では $|\mathbf{a}|$ は良いとして，もう一つの方はなんだよという事が当然気になるわけですが，そこで出てくるのが射影という考え方です.

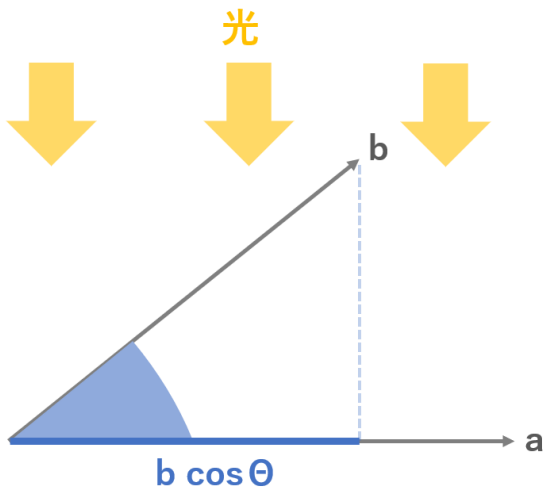

図 1.2: ベクトル b のベクトル a に対する正射影

よくある説明は，ベクトル a に対して垂直な光によって照らされたベクトル b の影の長さ，というものです．まぁようはベクトル b の，ベクトル a の方向に対してもっている影響力とかなんかそんな感じの量です．これでベクトル同士の向きをそろえた上でかけているわけですね．

ここからベクトルの内積というものが何を表しているのかが分かるのですが, いったん置いといて次へ進みます．

1.1.5 内積の意味

内積の定義を見てきましたが, 結局のところその意味は何でしょう？

先程，内積が $\mathbf{a}$ と射影した $\mathbf{b}$ の積である事は確認できました．ここから場合わけして考えます．まず $\mathbf{a}, \mathbf{b}$ が同じような方向を向いていた場合，θ の値は小さくなり，射影されて出来る $|\mathbf{b}|cos\theta$ の値はほぼ $|\mathbf{b}|$ から変わりません．そこから角度が大きくなるにつれ値は小さくなっていき，ちょうど 2 つのベクトルが直行する時 $(\theta = \frac{\pi}{2})$ に内積は 0 になります $(\because cos\frac{\pi}{2} = 0)$.

そしてそのまま角度を大きくしていくと，今度は射影ベクトルがベクトル a と全く重ならないようになってしまい，$cos\theta < 0$ となります．つまり，二つのベクトルが直角よりなお別の向きを向いている場

合，内積は負の値をとります.

この事をまとめると，以下のような関係が言えます.

内積の定義

$$\begin{aligned}\mathbf{a}\cdot\mathbf{b} &= a_1b_1 + a_2b_2 \\ &= |\mathbf{a}||\mathbf{b}|cos\theta\end{aligned} \tag{1.4}$$

内積の取る値

$$\begin{aligned}\mathbf{a}\cdot\mathbf{b} > 0 \quad &\text{(ベクトルの向きが似てる時)} \\ = 0 \quad &\text{(ベクトルが直行している時)} \\ < 0 \quad &\text{(ベクトルの向きが離れている時)}\end{aligned} \tag{1.5}$$

つまり内積とは，二つのベクトルの向きの関係性を示す指標として捉える事が出来，特に 0 になる場合にそれらのベクトルが直行している事を示せるものです.

1.1.6 ドット積

ベクトルの掛け算のうち, ドット積だとか点乗積と呼ばれるものがあります．ドット積とは, 同じ長さの数列の各要素を掛け合わせ, その値を足し合わせて一つの値として出力したものです.

$$\begin{aligned}a &= a_1, a_2, a_3, a_4 \\ b &= b_1, b_2, b_3, b_4 \\ a \cdot b &= a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4\end{aligned} \tag{1.6}$$

そう, つまり内積の事ですね.

ドット積は実質 (ユークリッド空間では) 内積と同じ扱いをされます.

こいつは代数学・幾何学において非常に大事になる上, 畳み込みの基礎となるものです！

そしてそれはつまり, フーリエ変換やウェーブレット変換 (参照: analysis) の基礎になる重要な掛け算です．ベクトルの掛け算を学ぶ意味は，そもそも我々がこの先扱う事になる脳波データなどの信号は全てベクトルであるからです．ベクトルを操作する方法をしっかりと学ぶ事が必要になります．そして内積はいずれ脳波の解析において最も基本となる処理，フーリエ変換を理解するために必要となります.

1.1.7 複素数の積

上記ベクトルと同様, 複素平面上では $r(\cos x + i\sin x)$ というものを考えます.
上記同様, 一般的な積を考えてみましょう.

$$
\begin{aligned}
&r_1(\cos x + i\sin x) r_2(\cos y + i\sin y) \\
&= r_1 r_2(\cos x\cos y - \sin x\sin y + i\sin x\cos y + i\cos x\sin y)
\end{aligned}
\tag{1.7}
$$

上記は加法定理より

$$
r_1 r_2(\cos(x+y) + i\sin(x+y)) \tag{1.8}
$$

となります.

ベクトルの内積と似てはいますが, 決定的な違いがあります.
掛け算をすると角度の足し算になっているのです！

1.1.8 複素数の逆数

地味に複素数の偉大なところです.
逆数... つまり $\frac{1}{x}$ が出来れば割り算はできますよね？
ベクトルや行列は複数の数を束ねただけなので逆数と言われてもこまりますが, 複素数は割り算が簡単にできます.
何故なら複素数は見た目上スカラーっぽいですからね！
逆数を取るという事は, 複素数の割り算は角度の引き算になりますね.

1.1.9 複素平面の掛け算の意味

複素平面で掛け算をすると角度の足し算になるようでした．つまり回転ですね．これの何がうれしいかって言うと, 計算が超楽になる事です．基本的に，数学では掛け算より足し算の方が扱いやすいので, 掛け算を足し算に変換できる性質は重宝されがちです.

そしてこの性質が今後とても大事になるオイラーの公式というものに結びついているのです．しっかりと理解しておきましょう.

1.1.10 大正義オイラーの公式

証明は今の段階では難しいので紹介だけになりますが，オイラーの公式をここで確認しておきましょう．オイラーの公式は脳波解析の基本の基本，全てにおいて重要になる公式です．basic.pdf の主だった目的の一つが，この公式を証明する武器を備えてもらうことであるといっても過言じゃない程のものです．まずは確認してみましょう.

オイラーの公式 (Euler's formula)

$$e^{ix} = \cos x + i \sin x \tag{1.9}$$

ふむふむ，三角関数の足し合わせが，指数関数っぽく表されてるのか．あとは何やら虚数単位 i もいますね．しかしこれは... 美しいですね！

更にこいつの特殊形がこうなります.

$$e^{i\pi} + 1 = 0 \tag{1.10}$$

美しすぎますね !! 数学の基本である指数関数，虚数，三角関数などといったものが濃縮され，数も基本の１と０のみでまとめられています.

この公式をもって, 我々は脳波と戦っていくことになります．Euler's formula を制するものは脳を制するのです.

つまり内積同様, フーリエ変換の基本の一つになっています．２章以降は統計以外の全てが，このオイラーの公式を証明するために必要な数学のおさらいになります．証明は Analysis の方に記載しています.

第2章　三角関数

さて，我々の最初の課題は三角関数を理解する事です．三角関数は様々な便利な性質をもっており，脳波に限らず信号の解析をしていく上で非常に重要，というより一番基礎になる概念かもしれません．しっかりとここで確認しておきましょう．

2.1　三角比とは

三角関数の前に, まずは三角比から復習しましょう．三角比とは, $\sin x$ や $\cos x$ $\tan x$ といった具合に三角形の辺の比を表す幾何学の方法でした．直角三角形を考えた時, 横の長さが cos, 縦の長さが sin, そして斜辺の長さが tan で表せるのでした．

あくまで辺の比としてだせるので, あとで絶対値分かける必要がありますが, それでも非常に便利なものです．高校数学では, 結局辺の長さや比を覚えているものでしか使えないため, 三角比の勉強をする意味に苦しんだことと思います．

$$\sin 45^\circ = \frac{1}{\sqrt{2}} = \frac{\sqrt{2}}{2} \tag{2.1}$$

$$\cos \pi = -1 \tag{2.2}$$

こんなやつですね！

sin が縦, cos が横, そして tan が傾きを表しているので, 基本的な直角三角形であればその場ですぐ計算できますね？

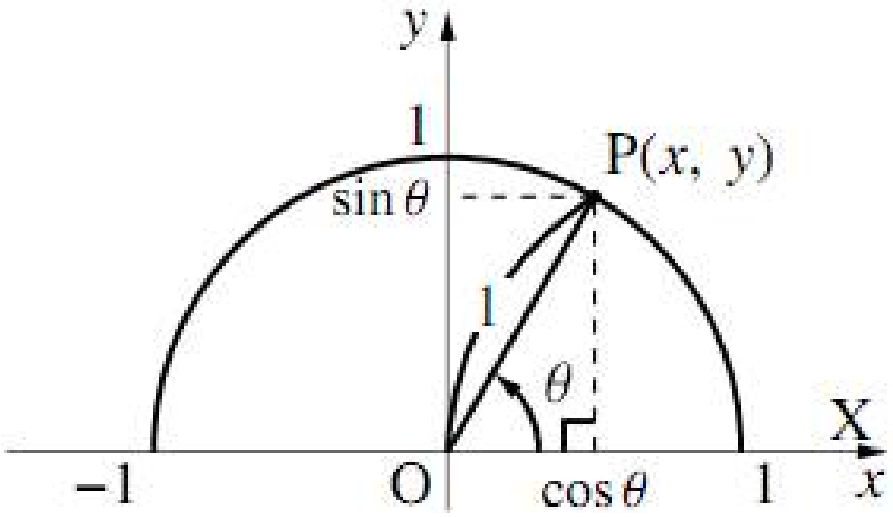

図 2.1: 三角比のイメージ図

2.2 三角比の公式

三角比には, いくつか公式があります. ここを暗記で片づけるのは阿呆のする事です. いったん確認してみましょう.

$$\tan\theta = \frac{\sin\theta}{\cos\theta} \tag{2.3}$$

$$\sin^2\theta + \cos^2\theta = 1 \tag{2.4}$$

$$\tan^2\theta + 1 = \frac{1}{\cos^2\theta} \tag{2.5}$$

式 (2.3) については自明ですね. 横の増加量分の縦の増加量なのだから, 傾きです.

式 (2.4) も自明です. ピタゴラスの定理です. 中 2 でやった.

式 (2.5) は一寸ややこしいですが, 式 (2.4) を用いれば秒で片づけられます. 式 (2.4) の両辺を $\cos^2\theta$ で割ったのが式 (2.5) です.

このように, 三角比の公式というのはおまじないではなく, 常識なのです. 感覚で理解し, すぐにその場で公式を作れるようにしましょう.

2.3 三角関数

さて, 三角比がどんなものか分かったうえで, それぞれの三角比を関数として考えてみます. 関数とは, ある変数の値によって定まる数の集合の事です. つまり x が 1,2,3,... と値をかえていくにつれて変化し

ていく y などを指し，$f(x)$ などと表します.

$\sin x$ は x が $0+2\pi n$ の時に 0, $\frac{\pi}{2}+2\pi n$ の時に 1, $\pi+2\pi n$ の時に 0, $\frac{3}{2}\pi+2\pi n$ の時に-1 を常に通り, その間を滑らかな曲線で結んだ関数になります. n の値は何周目かを表します.

$\cos x$ も同様にして 1 ,0 ,-1 ,0 となっています.

$\tan x$ は一寸特殊で, 三角形の斜辺の傾きを表すという性質上, 端がありません. よって x = 0 の時には 0, で正負方向に対称で $\frac{\pi}{2}+\pi n$ で ∞ または $-\infty$ を極限としてとんでいきます.

ここまでは常識ですね？今時小学生でも知ってるかもしれません.

さて, こいつら三角関数の何がすごいかって, 超単純な周期関数である事です.

さらに, 実は sin と cos に限って言えば位相が違うだけでその形は全く同じなのです !!! みていてください.

$$\cos x = \sin(x+\pi/2) \tag{2.6}$$

こうなっているのでしたね.

Listing 2.1: 三角関数の MATLAB コード

```
1 t = (0:0.0005: 5)';
2 plot(sin(2*pi*1*t))
3 hold on
4 plot(cos(2*pi*1*t))
5 hold off
```

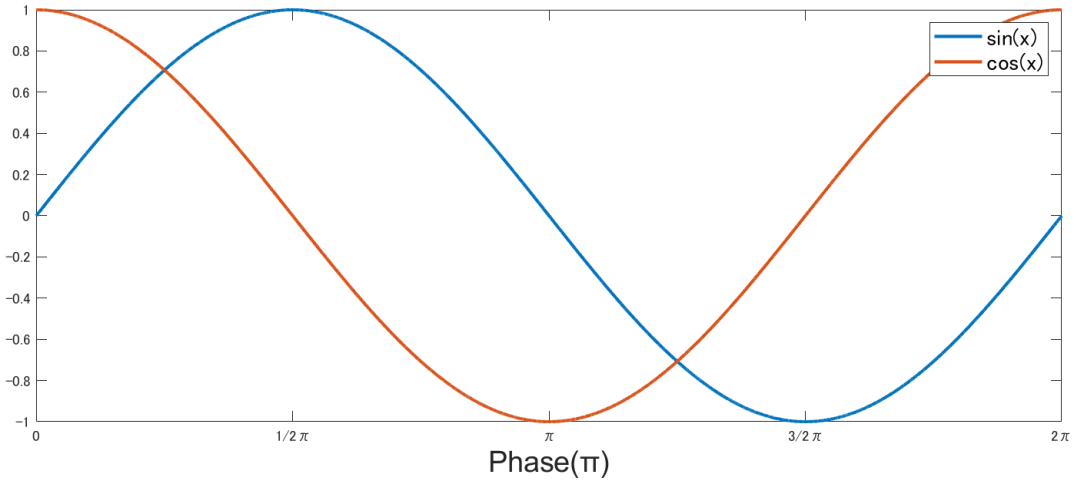

図 2.2: sin / cos の図

2.4 加法定理

これはオイラーの公式を用いる事で, 覚えていなくてもその場で導出できます. (オイラーの証明 (参照: analysis)) みててください. オイラーの公式より

$$e^{i(\alpha+\beta)} = \cos(\alpha+\beta) + i\sin(\alpha+\beta) \tag{2.7}$$

指数法則より,

$$\begin{aligned} e^{i(\alpha+\beta)} &= e^{i\alpha} + e^{i\beta} \\ &= (\cos\alpha + i\sin\alpha)(\cos\beta + i\sin\beta) \\ &= (\cos\alpha\cos\beta - \sin\alpha\sin\beta) \pm i(\sin\alpha\cos\beta + \cos\alpha\sin\beta) \end{aligned} \tag{2.8}$$

ここで再び式 (2.7) と見比べると, 式 (2.8) の実部が cos, 虚部が sin の加法になっている事が分かります. ここから三角関数の加法定理は

$$\sin(\alpha+\beta) = (\sin\alpha\cos\beta + \cos\alpha\sin\beta) \tag{2.9}$$

$$\cos(\alpha+\beta) = (\cos\alpha\cos\beta - \sin\alpha\sin\beta) \tag{2.10}$$

と証明できます. 例のごとく tan は一寸面倒なので, 自分で証明してみてください.

2.5 正弦と余弦

sin と cos の関数の定義は分かりましたが, それがなんで嬉しいのでしょうか？
これを知るためには, sin と cos の関係はどんなものであるのか考える必要があります.

ここで前回やった掛け算が出てきます.

任意の sin 関数と cos 関数の積を考えてみます.

$r_1 \sin mx$ と $r_2 \cos nx$ です.

ここで r_1, r_2 は振幅, m と n は角周波数を示していますので, 異なる周波数の三角関数になっています.

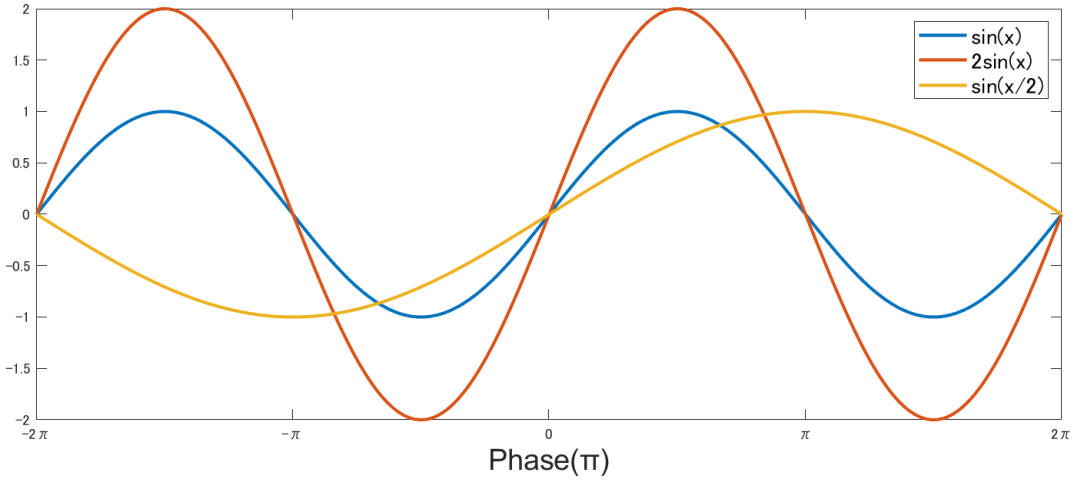

図 2.3: 様々な sin 波の図

こいつらの掛け算を考え, さらにそいつを積分してみます.

$$\int_{-\infty}^{\infty} r_1 \sin mx r_2 \cos nx dx = \int_{0}^{2\pi} r_1 \sin mx r_2 \cos nx dx = 0 \tag{2.11}$$

三角関数は 2π 周期で同じ波形の繰り返しになっているので, 範囲が ∞ から $-\infty$ じゃなくていいのは自明ですね.

三角関数は周期関数なので, それぞれが積分すると 0 なのも自明ですが, 更に積の積分も 0 になるのです !!!
これはすごい !! 波としては複雑になっても, 周期関数である事に変わりはなく, 積分しても 0 になるのは変わらないわけです !!

Listing 2.2: 式 (2.11) あたりのコード

```
1 syms x          // シンボリック変数
2 int(cos(x), [0 2*pi])        // 0 -- 2pi の積分, どちらも当然 0
3 int(sin(x), [0 2*pi])
4 int((2*(sin(3*x))*(3*cos(10*x))), [0 2*pi])  // [r1 r2 m n] = [2 3 3 10] の
      0--2pi の積分. これも 0
```

2.6　関数の内積

さて, ここで掛け算の復習です. ベクトルの演算, ドット積の定義を覚えていますか？

そう, 対応する次元の要素を掛け合わせたものの総和でしたね！

これ... 関数でも使えそうじゃないですか !?
同じ x の値の時の y 同士をかけ, それの足し合わせをするのもやってる事は同じなのです !!

これを使うと, すごい事がおきます. そう. 関数に内積を定義できるのです !! すばらしい !!
ただし注意する必要があるのは, ベクトルでは総和を Σ で表していましたが, 関数の場合は x が離散値ではなく連続値 (参照: 4.2) をとるため, 総和は Σ でなく ∫ で表す (参照: 4.4.3) という事です.

いずれにせよ, 掛け算の総和で関数の内積を定義できる事に変わりはないです. これは崇める必要がありそうですね !!
この節の話題から少し離れてしまうため, 細かい解説は 4.4.2 へ飛ばします. 大事なのは, 関数でも内積が定義できるという事です.

2.7　関数の直行性

話をもどします. ベクトルにおいて内積が 0 になるとどうなっていましたか？
そう, 直交です !!!

ベクトルの内積を関数の内積に拡張する事ができ, ベクトルの内積が 0 という事は直交を表す... すなわち関数についても直行性を定義する事が出来るのです.

sin と cos は直交する関数なんですね !!! すごい !!
ここでよくある勘違いについて言及しておくと, 直交と直角は別物です. 留意せよ.

2.8　直交性

ここでは関数の直行性については細かい話は省きますが, 直交する関数２つ (sin と cos) を用意できたなら, これを使えばすべての関数を表す事が出来ます.

そりゃそうですよねー. 複素数や直交座標系と同じノリです.

つまり... 全ての (周期) 関数は, 異なる振幅, 周波数の三角関数の足し合わせによって表す事ができるのです !!!

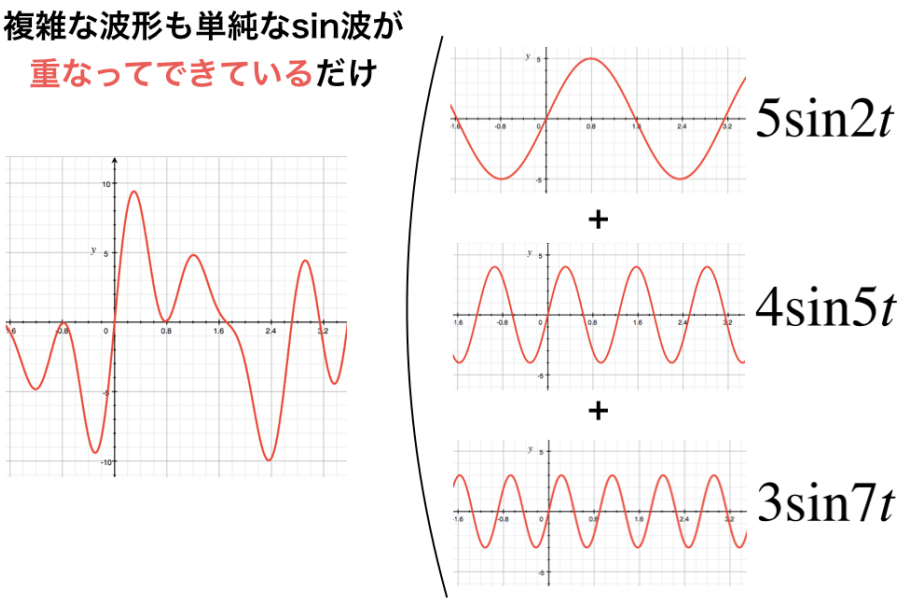

図 2.4: フーリエ変換の気持ち (宇宙一分かりやすいフーリエ変換より)

2.8.1 フーリエ変換への一歩

前項での内容を式に表すとこうなります. よく見てください.

$$f(x) = \sum_{n=1}^{\infty} (a_n \sin(\omega t) + b_n \cos(\omega t)) \tag{2.12}$$

ここで, 式 (4.19) はすべて sin,cos の値が 0 になる角度の際に吐き出される値も 0 になるものですが, 実世界においてはそううまくはいかず, そもそもこの関数の背景にある信号の影響によって基準値が上昇します. 電気回路系で言うなら直流電源などです. そこで式 (4.19) を改造します.

$$f(x) = a_0 + \sum_{n=1}^{\infty} (a_n \sin(nt) + b_n \cos(nt)) \tag{2.13}$$

ここで a_0 は DC 電源の分です.

脳波は時間によって変化する時間関数ととらえる事が出来るので, この方法で脳波を三角関数に分解して表現する事が出来るのです !!!

これが三角関数を崇める理由です.

そしてこの, 時間関数 f(t) を三角関数の和に変換する事をフーリエ変換といいます. 詳しくは今度（Advanced）.

2.9　大正義オイラーの利用

とはいえ, 三角関数を ∞ に足し合わせるというのも酷な話で, 計算が面倒なこと面倒なこと...

どうにかならんもんか...

ん？三角関数の足し合わせ？ どこかで聞きましたね？
そう, 式 (1.9) に表した, 我らが至宝オイラーの公式です !!!

オイラーの公式を用いる事で, 三角関数の足し合わせは指数関数で表せます. これを使う事で劇的に計算を楽にする事ができるのです！
ただ, オイラーの公式には sin の前に謎の記号 i がついています. 一体こいつはなんなのでしょう. 次の章では, 虚数と複素数について学びます.

第3章　複素数

　複素数とは偉大なものでありますが, 最初は理解に非常に苦しむものです. ですが複素数をマスターするという事は, 整数から分数への拡張や, 正の数から負の数への拡張を遥かにしのぐ恩恵を与えてくれるのです. 高校で習う内容ですが, ここでもう一寸深く学んでみましょう！

3.1　虚数

　虚数の定義からおさらいをしましょう. 高校で習う領域の中でも, 比較的簡単だったはずです.

虚数とは, 平方すると-1 になる数を基底とした数の事です.

1 本の直線でしかない実数の問題は, 表現の幅が圧倒的に限られてしまう事です. そこで昔の人類は, 実数軸を 2 本直角に重ね合わせた直交座標系やベクトルといった考えを用いたわけです.

しかし, 虚数の登場がこの常識を大きく変える事になりました !!

ベクトルあるいは行列を少し復習しましょう.
これらにおいて, 次元はどうやって定義されていたでしょうか？ 直交性・独立性ですね !!!

実数ではない... 正確には，0 の時（つまり交点）以外に必ず実数と交わらず, 直交性をもった「数」を定義できないか？昔の偉い人はこう考えたのです. 頭おかしいですね !!

こうして生まれたのが虚数単位 i です. 実数において単位として利用される 1 の概念と直交する必要があったため, ここで 1 の平方が 1 である事に注目しました.

平方して負の値になる数は存在しない. これは中学で習う事ですがこれを厳密に言うと,「従来の数には存在しない」と解釈したわけです.

そこで, 平方して負の値になる数を考え, これを虚数と名付けました.
実数の単位が 1 なら, 虚数の単位は 1i となります. 虚数も 0 の平方にマイナスをつけようとしたところで 0 なので, 実数と虚数は 0 で直交する事になります.

これが虚数の定義ですね !! 高校数学では「存在しない数」などと習いますがこれは誤解を招きます. 虚

数の意義とは「y 軸」を定義できる事にこそあるのです !!!
存在しない数があったから, 虚数と名付けたのではなく, 実数と直交性を持った数 = 虚数として使えそうな定義はないものかと考えた時に, 平方すると　-1 になる数, を考え付いたわけです.

3.2　虚数の性質

虚数は単位 i がついてるだけなので, 基本的な足し算引き算はすべて実数と同様に扱えます. 簡単ですね！

$$3i + 5i = (3 + 5)i = 8i \tag{3.1}$$

掛け算になると一寸難しくなり, 虚数同士をかけると数字同士の積にマイナスをつける必要があります. それが虚数の性質でしたね！

$$3i \times 5i = (3 \times 5)i^2 = 15 \times -1 = -15 \tag{3.2}$$

Listing 3.1: 式 (3.1 3.2) あたりのコード

```
1 3*1i + 5*1i        // add
2 3*1i * 5*1i        // multiply
```

コード (3.1) ではそもそも虚数単位と数字を分けて書いてますね，こう書かれると数字と虚数単位を別に計算させてるのもより実感できます. 余談ですが，プログラムを書くときは変数の i とかと混ざらないようにこのような記法をして虚数単位を区別する書き方が推奨されています. この掛け算, 実は複素数の掛け算の性質を考えると自明になります.
この章の最後でそれを説明します！

3.3　複素数

虚数のおさらいができたところで, 複素数です. 実数も虚数も直線であるため, これらを軸として, 直交座標系っぽいものを考えます.

こうしてできた平面の事を, 複素平面やガウス平面といいます.

さて, ガウス平面を考えた際, とてもうれしい事が数多くあります. まずはそれぞれの軸における長さですが, 実数も虚数も単位が 1(i) であるため, 原点からの距離を考えると $|a| = |ai|$ になっているのです.

この性質を利用すると, ガウス平面上の任意の点をユークリッド平面（今まで見た事のある関数のグラフのあれ）の点に対応づける事ができ, (x, y) ≒ x + yi といった具合に表す事が出来ます.

何故ユークリッド平面では (x,y) なのにガウス平面だと足し算で表せるのか？ユークリッドは x も y も実数だけど，ガウスの場合は実数と虚数で独立しているからです．これがやりたいから虚数を考えたのでしたね！

このように, 実数部分と虚数部分の足し合わせによって, 複素平面（二次元）上の数を表現した数を複素数といいます.

こいつのうれしさは, 直交座標や行列のように複数の数の集まりではなく, ひとつの数として扱える点です !!!

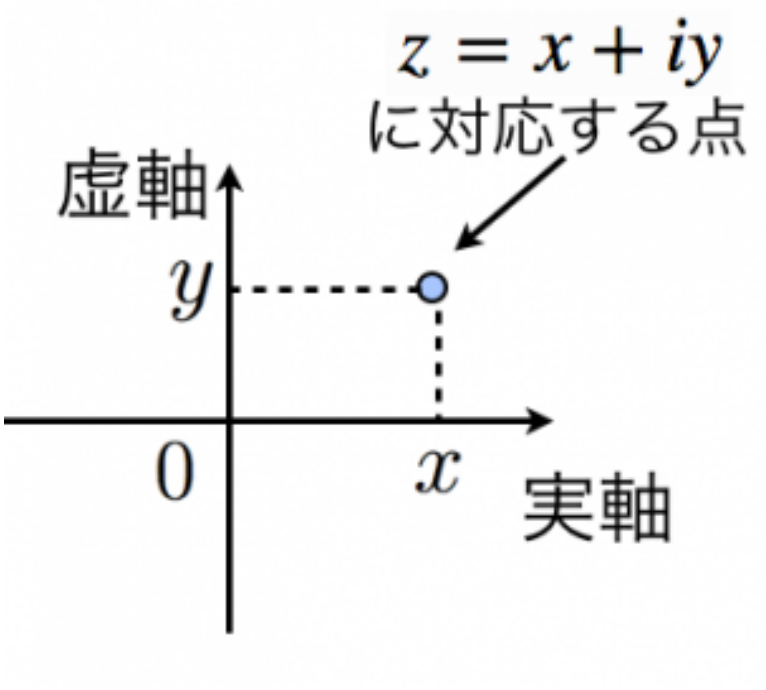

図 3.1: 虚数軸 (高校数学の美しい物語より)

3.4 数の大幅な拡張

複素数の定義が分かったところで, これが実際にどうやって影響してくるのかについて考えます. 一次元が二次元に拡張されたわけですからその恩恵は計り知れません.

さらに複素数の偉大な点は, 実数を複素数の一部として扱える点です.

たとえば 3 は, 複素数で表すと $3 + 0i$ となります.
そして注意しないといけないのは, 複素数を実数に変換した場合, その虚数成分は捨てられてしまうので

す !!
実軸に投射される過程で, 虚軸 (高さ) 成分は消えてしまうという事です.

つまり...
我々が今まで 3 だと思ってた数は, 実は 3 + 5i かもしれない... という考え方もできます !!

ちょっと違うような気もしますが, たとえば大学の単位で考えましょう.
成績が C でも S でも, 来る単位は 1 です.

実軸を単位, 虚軸を成績とすると, 実軸だけでは C の人も S の人も同じ優秀さぽく見えますが, 虚数成分も合わせれば同じ単位取得者でも上下関係があった事に気付けるわけです !!

まぁ，今のたとえ，成績と単位は線形独立でないので本当はだめですが，まぁイメージとしてはそんな感じという事で目を瞑ってください.
とかく，複素数は偉大ですね !!
実はこの考え方が後にウェーブレット変換を学ぶ際に非常に重要になるのです. 頭の片隅においておいてください.

3.5 複素数平面

複素数平面において, 任意の点 x + yi を表す方法について考えていきます.

実軸と虚軸の直行性より, 複素数 z = x + yi の横（実部 Re）の長さ (3.3) と, 縦（虚部 Im）の長さ (3.4), 原点との距離 (3.5), そして実軸との間になす角 (3.6) はそれぞれ

$$Rez = x \tag{3.3}$$
$$Imz = y \tag{3.4}$$
$$|z| = \sqrt{x^2 + y^2} \tag{3.5}$$
$$argz = \tan^{-1}\frac{y}{x} \tag{3.6}$$

のように表されます. 長さについては直交座標なので自明ですね. 絶対値に関してもピタゴラスの定理より自明. 式 (3.6) で表される角の事は, 偏角と言います. 実軸との角度の事であるので, tan を使って表す事が出来るというわけですね.

また, 複素数 z を実軸に線対称な点を取った点の事を $\bar{z}$ と表し, 複素共役な点といいます.

$$z = x + yi \tag{3.7}$$

$$\overline{z} = x - yi \tag{3.8}$$

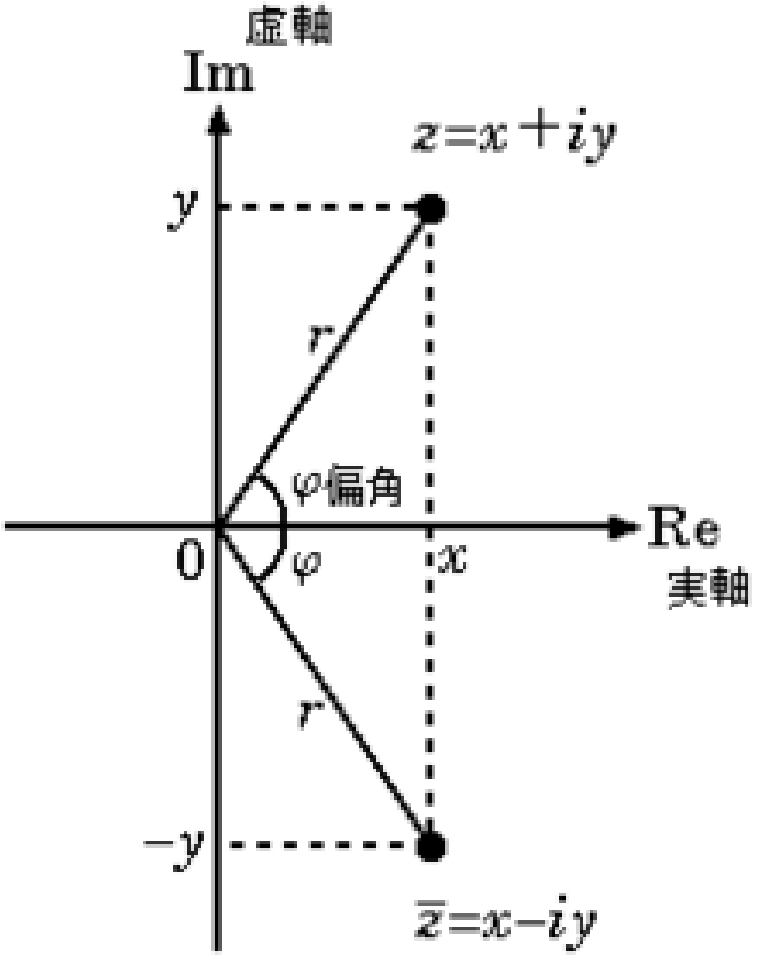

図 3.2: 複素平面 (wikipedia より)

MATLAB ではそれぞれ，以下のようにして求める事が出来ます.

Listing 3.2: 式 (3.33.7) あたりのコード

```
x = 3 + 4*1i
real(x) // 実部
imag(x) // 虚部
abs(x) // ノルム，絶対値
angle(x) // 偏角
conj(x) // 複素共役
```

3.6　極形式

複素数 z は x + yi のような表記の仕方以外に, もう一つの表し方ができます.

前節で確認した, 複素数 z の絶対値は, 原点との距離を表す実数でした. 範囲は 0 以上の実数になりますね !!

偏角の取りうる範囲はどうなるでしょうか？
原点を中心として 4 つの象限をぐるぐる回るので, -π ～ πですね！

この 2 つに注目して考えます. 原点からの距離 (r) と偏角 (φ) が分かるという事は, 平面上で一意に定まる点を定義できますよね？三角比で cos は横，sin は縦の長さでした.

$$z = x + iy = r(\cos\varphi + i\sin\varphi) \tag{3.9}$$

$r\sin\varphi$ で導出される数は実数 (y) になっているため i をかける事を忘れないように.

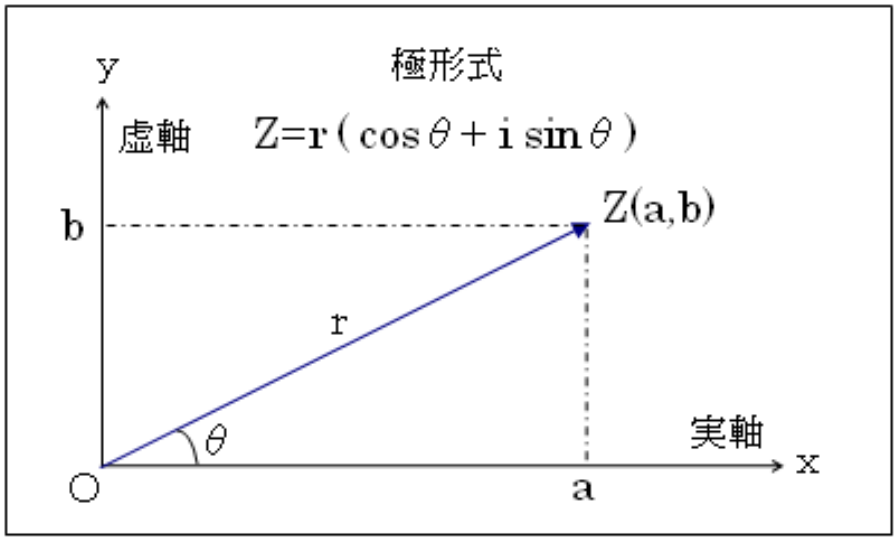

図 3.3: 極形式 (はてなフォトライフより)

3.7　大正義オイラー再び

式 (3.9) の最右辺をよく見てください. どこかで見た式ですね !!!
人類の至宝を利用する事により式 (3.9) はこのように変換する事が出来ます.

$$z = x + iy = r(\cos\varphi + i\sin\varphi) = re^{i\varphi} \tag{3.10}$$

3.8　複素数の掛け算

さて, 複素数を $re^{i\varphi}$ で表せたところで, 複素数の掛け算を改めて考えます.

$$
\begin{aligned}
& z_1 z_2 \\
& \quad = (x_1 + iy_1)(x_2 + iy_2) = (x_1x_2 - y_1y_2) + i(x_1y_2 + x_2y_1) \\
& \quad = r_1(\cos\theta_1 + i\sin\theta_1) r_2(\cos\theta_2 + i\sin\theta_2) \\
& \quad = r_1 r_2 \mathrm{e}^{i\theta_1}\mathrm{e}^{i\theta_2} \\
& \quad = r_1 r_2 \mathrm{e}^{i(\theta_1+\theta_2)}
\end{aligned}
\tag{3.11}
$$

こうなっているのです !!!
複素数の掛け算は絶対値をかけた上の回転を表すのですね !!

3.9　戻り学習

ここで再び, 虚数とは何か, 何故平方すると-1 になるのかを考えます.
任意の複素数を考えます. ここでは面倒なので $3 + 0i$ としましょうか.

$$3 + 0i = 3(\cos 0 + i\sin 0) = 3(1 + 0) = 3 \tag{3.12}$$

こんな感じに表せますね.
こいつに 0+1i をかけます. よく見ててください.

$$i(3 + 0i) = 3i(\cos 0 + i\sin 0) = 3i(1 + 0) = 3i \tag{3.13}$$

$$\tag{3.14}$$

そしてこいつと...

$$3(\cos(0 + \frac{\pi}{2}) + i\sin(0 + \frac{\pi}{2})) = 3(\cos\frac{\pi}{2} + i\sin\frac{\pi}{2}) = 3i \tag{3.15}$$

こいつを見比べるのです. 同じ値になってますね !!
複素数の積は回転を表すので, つまり i をかけるという事は $\frac{\pi}{2}$ 回転するという事なのです !!

なのでもちろん式 (3.15) に i をかけると, -3 になります. 虚数単位 i を 2 回かけると-1 を掛ける事になる, つまり π 回転するのですね！
まだしっくりこないなら，たとえば $cos0\pi = 1$ は $\frac{\pi}{2}$ 回転すると 0 になり, もう一度かけると -1 になりましたね？実数である 1 が消え (虚数では存在してる)，再び実数にもどってきたというわけです.

3.10　まとめ

このように三角関数と虚数は非常に親和性が高く, そしてオイラーの公式によって指数関数ともつながる非常に重要な概念になっています.

三角関数と虚数を理解すれば, 脳波解析を学ぶ基礎となる数学はほぼ出そろったという事になります !!
しっかりと理解していってください !!

第4章　微分・積分

この章ではみんな大好き微分積分いい気分の気持ちを考えていきます.
高校数学で習うなかでも, とびっきり意味が分からない分野ですね.

4.1　極限とは

まず, 微分や積分といった計算は超極小の範囲で考える数学です. この超極小というのがなんなのかから考えます.

超極小... 曖昧な響きですね. 我々の身長の話をしている際に mm を持ってきたら極小と言えますが, 蟻の体長の話をしている時には大きすぎますね.

このように, どんなスケールでの話をしていても十分に小さいといえるようなスケールでの話が微分積分です.

実用では「0 に限りなく近づく」などといった表現で表され, 数式だとこうなります.

$$\lim_{x \to 0} f(x) \tag{4.1}$$

この意味するところは, 関数 f(x) の x を限りなく 0 に近付けた時の関数の返す値という事になります. つまり f(x) が 2x とかであれば

$$\lim_{x \to 0} 2x \fallingdotseq 0 \tag{4.2}$$

になります. しかしここで x は限りなく 0 に近い値なので実質的にはこの式が返す値も 0 と見做せるので, 等号は成り立ち, 点々は外れます.

$$\lim_{x \to 0} 2x = 0 \tag{4.3}$$

$$\lim_{z \to \infty} 2x = \infty \tag{4.4}$$

4.2　ε - δ 論法

限りなく 0 に近いって曖昧な表現ですよね. さっきも言ったように mm なのか μ m なのか, どこからが限りないと言っていいのかわかりません. ここで極限

$$\lim_{x \to a} f(x) = b \tag{4.5}$$

は数学語では

ε - δ 論法

$$\forall \varepsilon > 0, \exists \delta > 0 s.t. \forall x \in \mathbb{R}, |x - a| < \delta \Rightarrow |f(x) - f(a)| < \varepsilon \tag{4.6}$$

のように定義されます. 有名な ε - δ 論法というやつです.

この式の意味はこうです.

「任意の正の数 ε に対し, ある適当な正の数 δ が存在し, $0 < |x - a| < \delta$ を満たす全ての実数 x に対し、 $|f(x) - b| < \varepsilon$ が成り立つ」

f(x) と b の間の距離が任意の正の数 ε より狭い範囲において, 必ず対応する x の値が存在しているといった具合で, 「お前がどんなに頑張ろうと俺はその上をいく」みたいなやつです. これが連続値の定義になるわけですね.

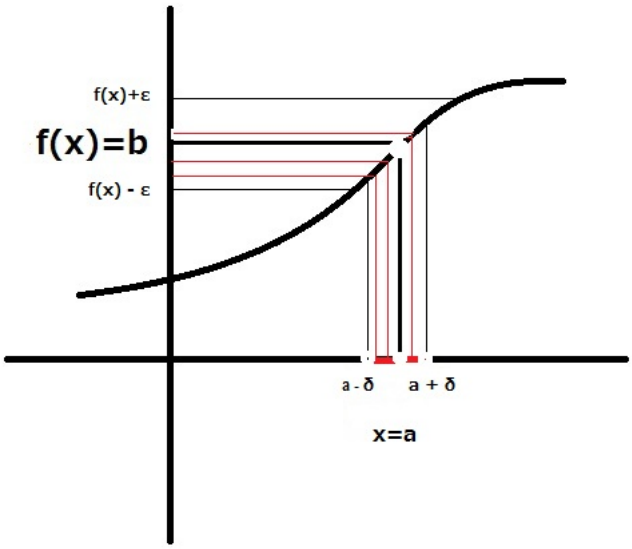

図 4.1: ε - δ 論法. a の左側は連続だが右は連続じゃない

つまり極限とは mm なのか μ m なのか？という問いに対する答えは,「お前が m の話をするなら cm, cm の話をするなら mm, mm の話をするなら...」を死ぬまでやるイタチごっこだということです !!

ちなみに式 (4.6) はついでに関数の連続性についても定義していて, どんなに小さい範囲で見てもそれより小さい値が存在するという事は切れ目なくしっかりとつながっている＝連続であるという定義にもなります.

4.3 微分

4.3.1 微小な変化率

さて, 極限とはある一点に限りなく小さい範囲を考える事でした.
ここで, 関数でも同じ事を考えます.

まずはざっくりとした範囲で考えてみましょう.

$$\lim_{0\to 5} x = 5 \tag{4.7}$$

この式の意味を考えてみます. 「x の値が 0 から 5 に限りなく近づくとき, y の値は何になっていくか」ですね！

ぶっちゃけて言えば x が 5 増えたら y はどんだけ増える？という問いで, それに対する答えが「5」だったわけです.

変化する値が 2 つあったら, その間の比を求めたくなるのが健常な脳の持ち主です. え，求めたくない？..... やだ，この人頭おかしい.

とりあえず黙って，x が増えた分に対して y がどれだけ増えたかを考えましょう.

$$\frac{\Delta y}{\Delta x} = \frac{5}{5} = 1 \tag{4.8}$$

こうですね. Δ は増加分という意味です.
これ, あれですよね. 中学で習った変化の割合, またの名を平均変化率, またの名を傾きでした.
今回は傾き 1 の一次関数を例にしたので, 変化の割合もしっかり 1 になっています.
しかしこれが二次以上の関数だった場合どうなるでしょう？ 元が直線でないので, 重なる事はないはずです.

$$\lim_{x\to a} f(x) \tag{4.9}$$

このうち, f(x) が二次以上の関数の場合は $|x-a|$ の値が大きければ大きい程グラフの形を無視して串刺しにする線が引かれ, 小さい程グラフの縁の角度に近い線が引かれていきます.

　そしてこの値が一点に集中したとき, つまり $|x-a|$ が 0 となったとき, それは点 a における接線の式になるのでした.

これはつまり, 超微小で見た変化率＝接線を求めているという事で, この操作をする事を微分と言います.

微分とは, 局所的な微小変化率を求める事なのです !!!
「あたりまえだろお前高校数学やったか？」と思っただろ !! やってねえんだよ !!

実際この微分の捉え方, よく理解できている人は多くないように思えます. かみしめてください.

よく,「微分とは接線を求める事です」という人がいますがそれは不適切. あくまで微小変化率を求めるものです. それが 2 次元で行われた場合接線になるだけにすぎません.

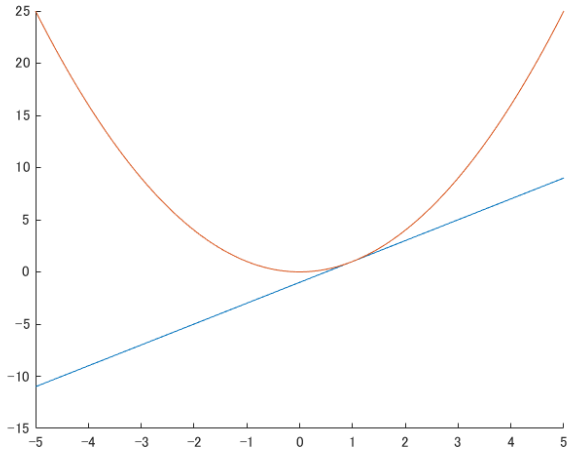

図 4.2: 局所的な微小変化率 = 接線？

4.3.2 三角関数の微分

上記の性質が分かっていれば, 簡単な公式達です. 覚えるまでもありません.

$$(\sin x)' = \cos x \tag{4.10}$$
$$(\cos x)' = -\sin x \tag{4.11}$$
$$(-\sin x)' = -\cos x \tag{4.12}$$
$$(-\cos x)' = \sin x \tag{4.13}$$

Listing 4.1: いろんな微分

```
1 syms x
2 diff(cos(x))
3 diff(cos(x)*(sin(x)))
4 diff(exp(x))
```

自明ですね. sin, cos 関数それぞれがどういった挙動をしていて, その傾きをすべての点で取ってつなげていったらどうなるかを考えれば, 自ずと分かるはずです.

「微分とは, 局所的な微小変化率を求める事」です.

4.3.3 ネイピア数

三角関数の次, 微分といったらこいつですよね. π と並び超越数として名高い e の出番です.
ネイピア数は別名オイラー数です. これだけで崇めないといけない気がしてきますね !?

$(\mathrm{e}^x)' = \mathrm{e}^x$ つまり, 指数関数にしたときに微分しても値が変わらないという変態な数です.

微分しても値が変わらないって, わけがわからないですよね. 微分とは傾きを求める事だとか, 体積から面積を求める事, といった理解をしていると全く訳がわからなくなります.

$$e = \lim_{x \to 0}(1+x)^{\frac{1}{x}} = \lim_{x \to \infty}(1+\frac{1}{x})^x \tag{4.14}$$

定義としてはこんな感じです.
まぁここは良いです. 大事なのは指数関数を微分しても値が変わらないという性質の確認だけです.

さて, 一般に正の 1 以上の数を底とする指数関数は, 第一象限で二次関数的な上がり方をし, (0,1) を通って, 第二象限では x 軸に漸近しつつ徐々に横ばいな形をとりますね？

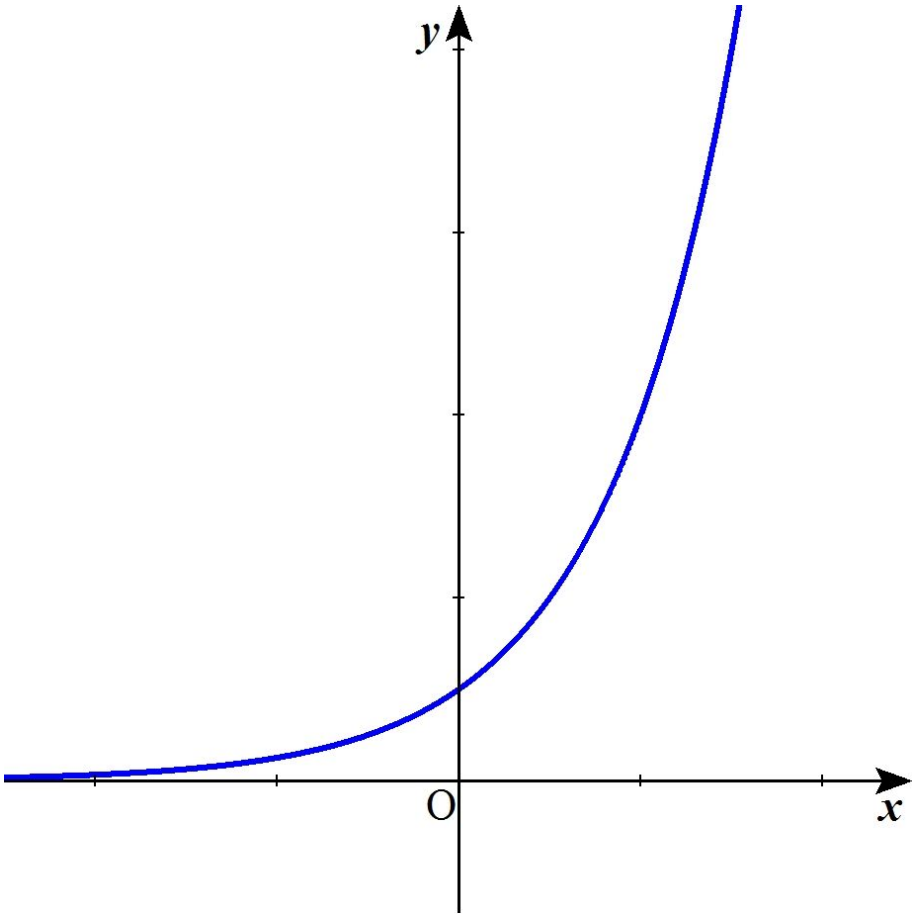

図 4.3: 指数関数

こいつを, x 軸方向に様々な点で接線ぽいのを引き, 傾きを推測してみてください.

面白い事に, こうやって求めた傾きの値を y 成分として新しくグラフを書いた f(x)' も指数関数の形をしているのです !!

察しの悪い人用に言うと，限りなく負の方向に行ってるとほぼ横に伸びてるので，その接線の傾きもほとんど変化しませんね？一方で正の方向は逆に進めば進むほど傾きが急になります．傾きの変化を関数としたのが微分なんだから，これが元の関数と同じ形になるってのも不思議じゃない気がしてきましたね？

そしてこの形は, (0,1) だけは固定としてその開き方が底の値によって異なります.
底の値次第で, 微分したら開いた指数関数がでてくるものや, 細くなった指数関数がでてくるのです.

もう分かりますね？ネイピア数とは, 丁度この開いたり閉じたりする境界にある数なのです !!
よって微分しても出てくる指数関数は開き具合を変えず, 重なった関数となるのです.

ネイピア数の何がすごいかって？
いろんな数学に利用できるところです !!!

$$\mathrm{e}^{i\theta} = \cos\theta + i\sin\theta \tag{4.15}$$

$$F(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \mathrm{d}x \tag{4.16}$$

書くのが面倒なのでいかついのは 1 個だけにします. ガウス関数ですね. 統計とか信号処理やるときにでてきます. 上は言わずもがなのオイラーの公式です.

4.4 積分

さて, 微分がわかったとこで積分の気持ちです. パパっといきます.
まず, 計算の上でいえる事としては, 微分の逆の操作をするという事です.

$$\int 2x dx = x^2 + C \tag{4.17}$$

こんな具合でしたね.

Listing 4.2: 4.17 のコード

```
1 int(2*x)
```

微分とは微小な増分である dx と dy を用いて変化率を求めるものでしたが, 積分の場合は, dx と y を用いて面積を求める操作になります.

関数 f(x) は x = a の時, f(a) という値を取り, x = b の時, f(b) という値を取ります. この時 x の増分 dx は, x 軸での距離を表すので横の長さととらえる事ができ, f(a) あるいは f(b) は縦の長さと考える事ができます.

こうして定義される横と縦を用いて作られる長方形を, a〜b, b〜c, c〜d..., m〜n といった範囲で同じように作っていき, それぞれの面積をたすと, 関数 f(x) と x 軸の間の, a〜n における面積っぽいものがだせますね！
しかしこのとき, 横はまだしも縦の長さが問題です. 左端に合わせるのか右端に合わせるのか, あるいはその中点にするのか... これによって求められる面積は理論値から離れた値になってしまいます.

この問題を解決するにはどうしたらよいでしょうか？
簡単ですね, x 軸上での幅を限りなく小さくしていけばよいのです !!

横幅が限りなく 0 に近づいた時, 左端と右端のとる高さは同じ値となるため, 微小長方形の形も関数 f(x)

にそったものになるはずです.

そしてそれらを足し合わせれば, デコボコではなく滑らかな, 関数 f(x) の凹凸にあった面積を導出する事が出来るのです !!!

$$\int_a^b f(x)dx \tag{4.18}$$

とはつまり...
x が a から b の範囲において, それぞれの f(x) = 高さと 微小な dx = 幅をかけたもの = 面積を足し合わせる
といった意味になっているのです !!!

Listing 4.3: 4.18 のコード

```
1 int(fx, [a b])
```

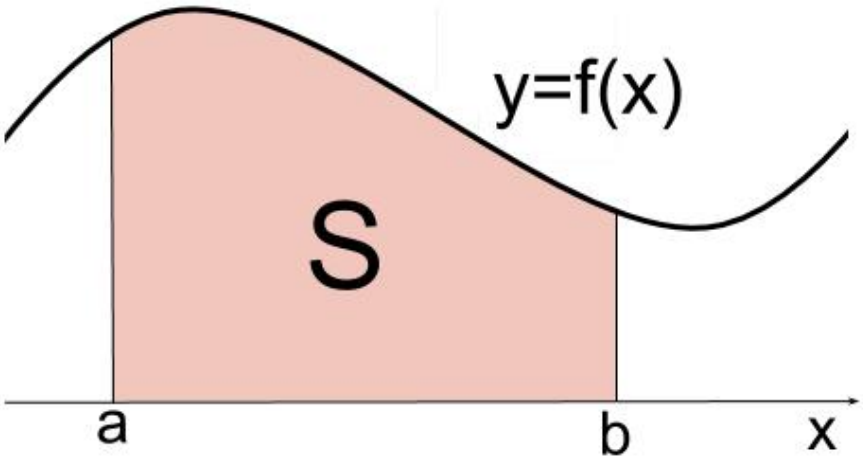

図 4.4: 積分と面積の関係

4.4.1 足し算としての積分

積分とは微小面積の足し合わせと言いましたが, 足し算は Σ で表されたはずです. 何故 ∫ を使うのでしょうか？

答え. Σ は離散値の計算にしか使えないからです！

$$\sum_{k=1}^{n} k \tag{4.19}$$

Listing 4.4: 4.19 のコード

```
K = [k1 k2 k3 ... kn]     // 数列の定義．実際は数字とか好きなのいれてください．
sum(K)
```

この式 (4.19), コード (4.4) では, k に入る数字は 1 から n までの整数になります. 無限小の幅による微小面積を考えるときにこれでは, 全く意味がないですよね.

そこで ∫ を使うと, 整数に限らずすべての数を足し合わせる事が出来るのです !!
素晴らしいですね !!! 足し算のアップデートです !!!

足し算として積分を使うことは, ここからの数学では, 頻繁に出てくるので是非感覚として理解してください.

$$X(\omega) = \mathcal{F}[x(t)] = \int_{-\infty}^{\infty} x(t)\exp(-j\omega t)dt \tag{4.20}$$

$$x(t) = \mathcal{F}^{-1}[X(\omega)] = \int_{-\infty}^{\infty} X(f)\exp(j\omega t)dt \tag{4.21}$$

たとえばこいつらは複素フーリエ変換とその逆変換の式ですが, これらも x(t) と exp なんたらの掛け算の-∞から∞での総和とかのように解釈する事ができるわけです !!!
とは言っても，実際 PC に計算させる時には計算を楽にさせるためだったり，そもそも計測されたデータが離散だったりするために積分でなく足し算だったりしますが...

足し算の拡張として積分が扱えるようになったところで, その応用法について考えます.

4.4.2 関数の内積

2.6 では, 関数の内積を定義する際に Σ の代わりに ∫ を使いました. その理由は, 上記の通り積分を使う事で, Σ では表現できなかった非整数も演算に組み込む事が出来るからです.

$$z = \int_{-\infty}^{\infty} f(x)g(x)dx \tag{4.22}$$

式 (4.22) が関数の内積の定義です. 日本語では, 同じ点 x についての関数 f(x),g(x) それぞれの値を算出し積を計算し, それをすべて足し合わせるという意味です. 当然, 直交していれば 0 の値を取るのは変わりません.

4.4.3 畳み込み積分

お次はいよいよ, ウェーブレットやヒルベルトを勉強する上でも必須になる, 畳み込み積分について学びます.
先に定義式を置いておきます.

$$f(t), g(t) \tag{4.23}$$

$$h(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{4.24}$$

式 (4.23) から読み取れる事ですが, まず同じ変数（ここでは時間 t) に対する関数である, f(t) , g(t) が定義されています. こいつらの式は任意ですが, 同じ変数に対応している事は重要です.
次に, 計算結果として求められるのも関数になっていますね. 計算結果である h(t) の引数も, f(t)g(t) 同様に t ですね.
そして最後に, 右辺では関数 f の引数は (τ) となっていて, g の引数も $(t-\tau)$ となっていますね. ここが内積と異なる点です. また, それぞれの関数の引数を足し合わせる事で常に一定の値を取る事も分かります $(\tau + t - \tau = t)$

t を時間とした場合でこの式を説明すると, τ は定義域の中のある瞬間の時間を指します. よって $f(\tau)$ は「ある時間 τ における f(t) の値」ですね.
という事は $g(t-\tau)$ は, 時刻 t(つまり現在) から, ある時刻 (関数 f による作用が起きた時) までの時間差を指します. それらの積を τ で積分するというのがこの式の意味ですね.

陰キャの皆さんは, いつも悲しい事ばかり経験している事と思います. 本書に載ってるレベルの数学も理解できない絶望感, 何故か自分だけ新歓されない孤独感, 笑顔で挨拶してくれたから脈ありかなと思った女の子に告ったらブロックされた無力感...

可哀想に. 僕みたいにイケメンの陽キャと違って, 陰キャの皆さんは人生ハードモードですね.

ですが大丈夫です. どんな心の痛みも, 数日寝てゲームして数学してれば..... まぁ, 時間が解決してくれるはずです.

さて, 人生を時間軸とすると, 陰キャの暗く惨めでむごたらしい人生では絶え間なく心に傷を負う「イベント」が起きるはずです. このイベントで受ける心の傷の大きさは場合によりけりなので, 時間依存の変数, つまり時間関数と捉えられます
(このたとえだと普通はとびとびの離散値になりますが, 究極の陰キャを想定して連続値を取る関数にしておきましょうか. 頑張れ陰キャ, 負けるな陰キャ. 君には数学がついている)

つまり, 関数 f(t) ですね !!

しかし先程も言ったように, 普通は心の傷は時間経過で減少していきます. ここで, その減少の仕方を関数 g(t) としましょうか. こいつも同じく時間関数ですよね.
勿論, 中には 1 年くらいしてから心の傷がぶり返す面倒なオタクもいますので, 関数 g(t) の形はなんでもいいです.

ここで, とある瞬間 (τ) に陰キャが自分だけクラス会に呼ばれなかった時に受けた心の傷を $f(\tau)$ とおきます.
今日は時間 t の時点です. つまり陰キャが心に傷を負ってから $(t-\tau)$ 日が経過しているわけですね. 傷が癒えるのは時間依存の関数 g(t) によりますから, 今日陰キャ君が感じている, クラス会による心のダメージ h(t) は

$$h(t) = f(\tau)g(t-\tau) \tag{4.25}$$

で表せますね !! 式 (4.23) に近づいてきました.
改めて, 式 (4.23) を確認してみます.

$$h(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{4.26}$$

式 (4.25) を $-\infty$ から ∞ の範囲で, 時間 τ について積分したものが h(t) になっていますね.
積分が何をする操作だったのかを考えれば, この式の残酷さが分かるはずです.
関数 f(x) について, 範囲 a から b までを x で積分というのはつまり...
「x が a から b の範囲の全ての数を取った時の f(x) の値を足し算する」事でした.

上記の陰キャの例で考えるなら, 式 (4.23) は「時刻 t の時, 陰キャ君が感じている苦痛 ($-\infty$ から ∞ の間に受けた心の傷の足し算)」です.

時刻 τ_1 の時にクラス会からハブられた $f(\tau_1)$ も...
時刻 τ_2 の時に可愛い女の子に LINE のスクショを Twitter に晒された $f(\tau_2)$ も...
時刻 τ_3 の時に洗車したての車のフロントに鳥の糞が落ちた $f(\tau_3)$ も...
時刻 τ_4 で未保存の tex ファイルを誤って閉じちゃった $f(\tau_4)$ も...

その全てを足し算したものが h(t) なわけです. ここで t は現在の時刻です.
$g(t-\tau)$ によってダメージは小さくなってるので, 実際は τ_4 のダメージは計り知れず, τ_1 のダメージはそれほどではないかもしれませんが, とりあえずそういう事です.

h(t) には陰キャくんの辛く惨めで凄惨な人生が詰まっているのです !!
..... おい待て，早まるな．h(t) は時間関数だ．今はつらくてもいつか楽になれるさ．頑張れ.

4.4.4 つまり畳み込みとは

さて, 皆の心を犠牲にして流れを理解できたところで, 最後に数学的な言葉で畳み込みの確認をします.

$$h(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{4.27}$$

式 (4.27) は,「全ての瞬間 (τ) に入力 (関数 f) されたそれぞれの信号が，処理システム (関数 g) によって変換されていて, その総和が現在出力されている信号 h(t) である」を意味します. これを各時間につ

いて計算したものが関数 h(t) なので...

畳み込みとは, 「関数 f が関数 g によって変化させられた関数 h を求める処理」というわけですね !!! 脳波の話をするなら, band-pass-filter とかを考えてみて下さい. 生データが, フィルターをかける事で特定周波数帯域のみの信号に変換されますね.

信号処理の話を理解するためには, 畳み込みはかかせない概念です. かなり丁寧に説明したので, 頑張って理解してください.

余談ですが, 畳み込みは可換なので, 以下は同じことを指す式です. 注意.

$$h(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau \tag{4.28}$$

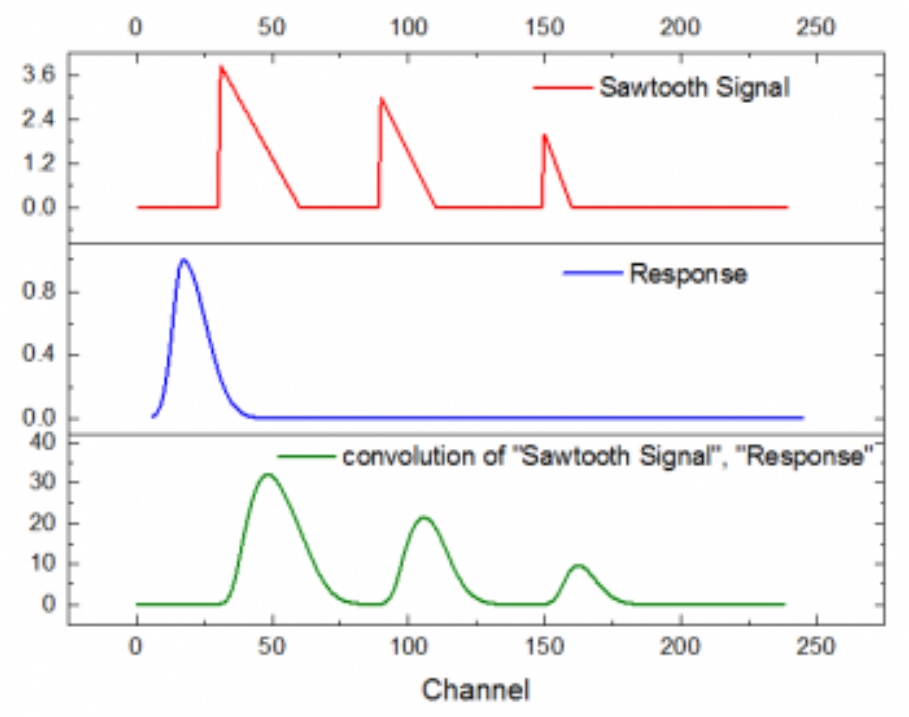

図 4.5: 畳み込みのイメージ (OriginLab より)

Listing 4.5: 畳み込みのコード

```
x = cos(3 * [0:0.1:20]);
y = [zeros(1,100) ones(1,20) zeros(1,81)];
z = conv(x,y);

subplot(2,1,1);
```

```
6  plot(x)
7  hold on
8  plot(y)
9
10 xlim([0 200])
11 ylim([-1.2 1.2])
12 legend三角波 (’’, フィルター’’)
13 title(’Raw signal and filter’)
14 hold off
15
16 subplot(2,1,2);
17 plot(z)
18 xlim([0 400])
19 title(’Filterd signal’)
```

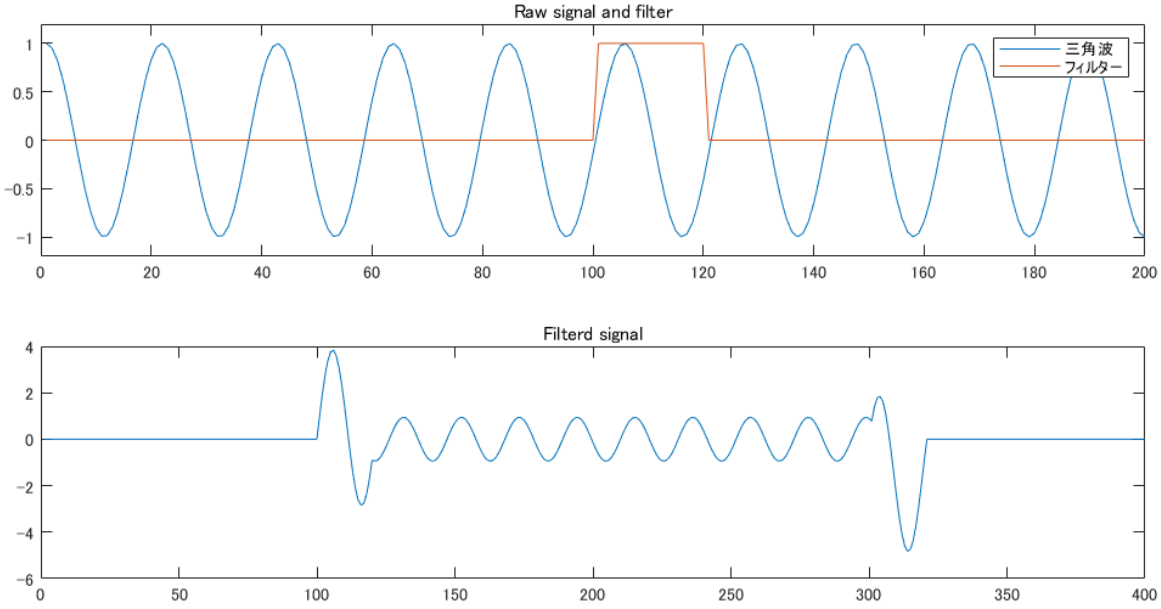

図 4.6: コード (4.5) の図

上にあげたコードによって計算された結果が図 (4.4.4) になります．いくつか気づく事があると思います．まず青の三角関数は 0 から 100 までの間 0 を保つように変形されましたね．同じくお尻の方もなっています．こんな使い方をするのがフィルターです！
今回は時間方向でそのままやっていますが，これを例えば一度周波数変換した信号に対しかけて，もう一度元の時間関数に戻す作業をしてあげれば，ここでいうオレンジのフィルターにかかっていた周波数帯域だけが元に戻ることになります．そう，周波数フィルターですね !!!

他にも気付くことがありますね，まずフィルタの長さよりも変形された信号の方が長くなります．これは畳み込みの計算上仕方ないですね．重なっている部分の積の和をとるわけですから．

そしてこれによって生じるもう一つの問題，それが畳みこまれた信号の両端に生じている謎の突起です．これも計算を考えれば道理ですが，フィルタの一部だけが信号にひっかかっている状態だとすべての要素の積が計算されるわけではありません (正確には 0 との積が行われている)．つまり本来であれば積の結果が負になったりするはずの部分が計算されていないわけですね！これが原因で，フィルタが信号に全部かかるまでの範囲は計算結果が一寸おかしな挙動をします．
これは analysis とかでいずれやると思いますが，畳み込みの際に気を付けないといけない事で，edge artifact なんて呼ばれていたりするものです．心の片隅においておいてください．

第5章　統計学

5.1　基本統計量

この節では統計の基礎を学びます. 統計は脳波解析に直接的に使うわけではありませんが, 解析した結果がいいデータなのかゴミデータなのかを判断したりするために必要です. また, 分野問わずどんな論文も統計の知識なくては読むことが出来ません. 幾何学などに比べれば大した事ない量とレベルなので, パパっと身に着けちゃいましょう !! (脳にフィットするとは言っていない)

さて, まずは冗談じゃなく小学生でも知ってる統計学から始めましょう.

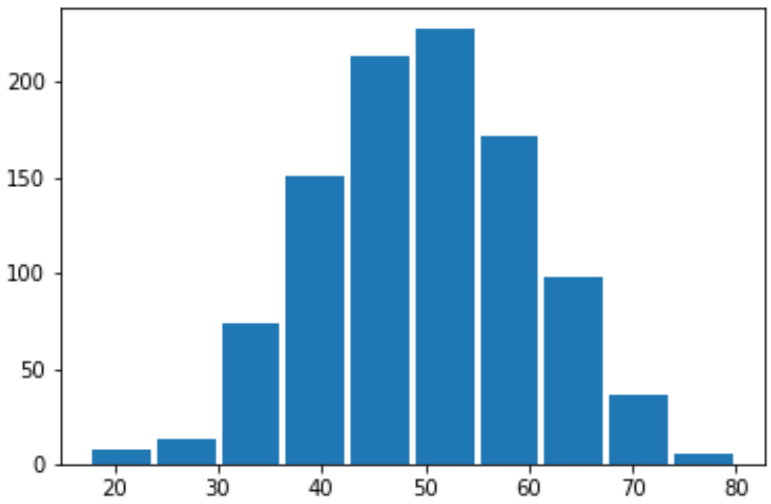

図 5.1: 平均 50, 標準偏差 10 の 1000 個のランダムデータ

複数の数値データ $x_i = [x_1, x_2, x_3, ...]$ が得られた際, とりあえず平均と分散, ついで標準偏差を求めたくなるのは人間の性ですね.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{5.1}$$

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} \tag{5.2}$$

$$s = \sqrt{s^2} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}} \tag{5.3}$$

Listing 5.1: 式 (5.15.25.3) のコード

```
x = [1 2 3 4]
mean(x)
var(x)
std(x)
```

とりあえず, 平均 (5.1) と分散 (5.2), 標準偏差 (5.3) の式を書いてみました. 平均は良いですね？全要素を足して要素数で割るだけです.
分散は, 一見ややこしくなってるかもしれませんが各要素と平均の差 (偏差) を足して, 同じく要素数で割っています. 標準偏差はその平方根です.

平均とは, 得られたデータのだいたい「真ん中あたり」を指す指標です. 代表値として, 他には中央値とか最頻値とかがいますが, まあ平均値だけ覚えていれば大丈夫です. モブは放置.

分散とは, 偏差の平均的な奴ですね？つまり, 得られたデータはどれだけ平均付近に固まっているか, あるいはバラバラになっているかを指します. 各要素が平均から遠い値であるほど, 偏差とその二乗は大きな値を取ります. 2 乗するのは, 符号の影響をなくすためです. そいつらの平均なので, やはり値が大きい程データがばらついているという事になります.

標準偏差は, 分散の平方根です. それだけ. 偏差の特性上, 分散を導出する際にデータを 2 乗してしまっているため, 分散は元のデータとスケールが違います. 例えば元データが長さだったら, 分散は面積を表してしまいます. なので平方根をとって次元をそろえてあげるわけですね. これで分かりやすくなります.

図 (5.1) に設定した統計量とヒストグラムを見比べて, 感覚を確認してください.
こいつらは後程アップデートされたりしますが, 基本的にはデータを扱う上で普遍的に利用されるものです. まず確実に理解しておきましょう.

5.1.1 正規化

ちなみに, 標準偏差と平均をどう使ってデータを読み解くのかという例として, こんなのもあります.

$$z_i = \frac{x_i - \bar{x}}{s} \tag{5.4}$$

Listing 5.2: 式 (5.4) のコード

```
x = [1 2 3 4]
for i=x
    z([i]) = ((x([i]) - mean(x)) / std(x))
end
z = normalize(x)          // 関数使うとこう
```

正規化 (標準化) と言われる処理です. 各要素と平均との偏差を標準偏差で割っていますね. 標準偏差はざっくり言うなら偏差の平均的なやつですので, それと各偏差の比を見ているわけです. こうする事で, 平均 0, 分散 1 のデータに置き換える事が出来るので, それぞれの要素がどれくらい平均から外れているかが分かりやすくなります. z の大きさが 1 を超えれば平均からかなり (正なら正方向, 負なら負方向に) 離れているってわけですね !!

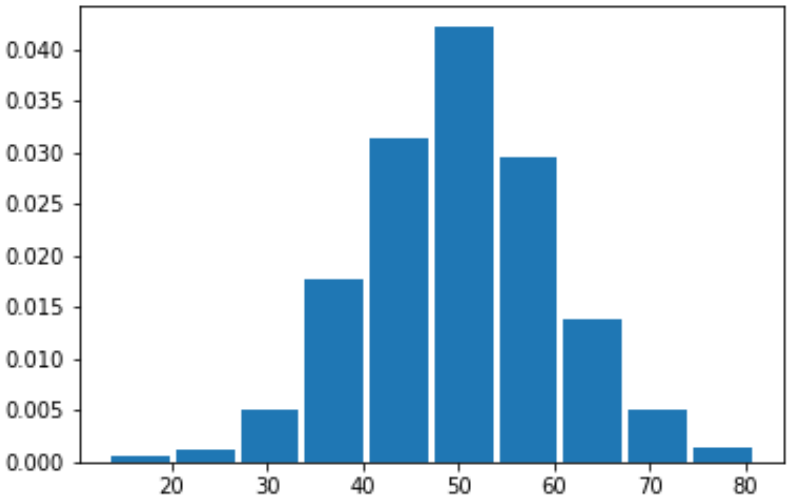

図 5.2: 正規化した, 同じ条件の図 (5.1

まぁ, 勿論こんな事しないでそのまま眺めるだけでも十分だったりしますが.

5.2 相関と共分散

次もよくつかわれる基礎です. 2 つ以上の変数を持つデータ (たとえば A さん,B さん ... の身長と体重) の性質を検証する術です.

5.2.1 相関と散布図

相関は非常に簡単です.

Listing 5.3: 図 (5.2.15.2.1) のコード

```
// 正の相関あり
x = [0:0.1:10]+ rand(1,101)
y = [0:0.1:10]
scatter(x,y)

// 負の相関あり
x = [0:0.1:10] + rand(1,101)
y = [0:-0.1:-10]
scatter(x,y)

// 相関なし
x = rand(1,100)
y = rand(1,100)
scatter(x,y)
```

直交座標軸横軸に変数 1, 縦軸に変数 2 をおいて, あとは得られたデータをすべて対応する位置にデータを点でプロットしていくだけです. こうして得られた, 点を散りばめられた図を散布図だとか相関図と言います.
この図の特徴ですが, データによって点の散らばり方が変わります. たとえば先程の身長と体重の例の場合, 一般的に身長が高い人の方が体重が重い傾向にありますね？これは図では点が右肩上がりの斜めライン付近に多く集まる事を意味します.
逆に, 右肩上がりの斜めに点が集まるという事は, 一方が上がると他方も上がる, 比例に似た関係にあると言えます. 逆に右肩下がりの場合, 一方が上がると他方が下がる関係ですね !!

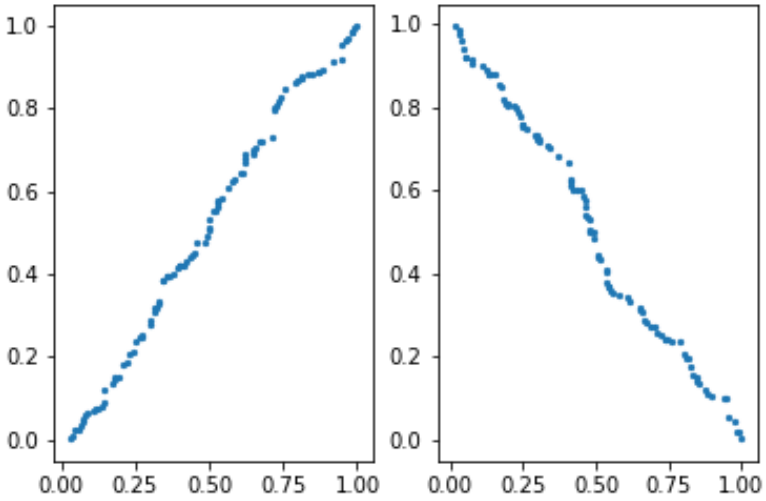

図 5.3: 左: 正の相関図 右: 負の相関図

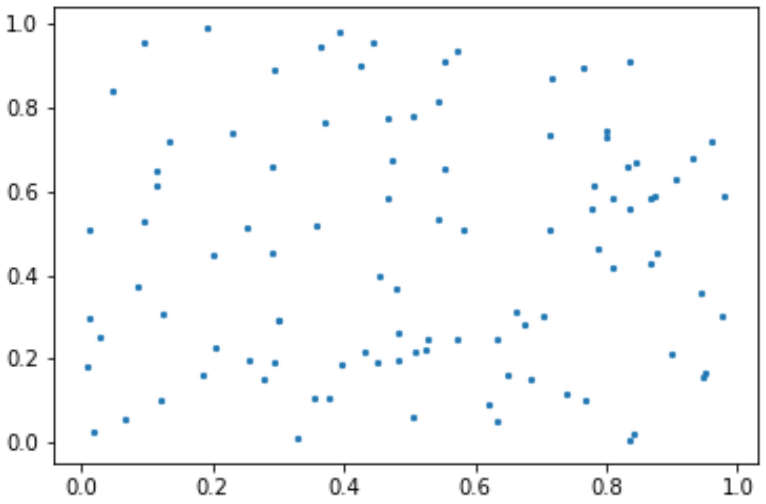

図 5.4: 相関なし

前者を正の相関, 後者を負の相関があるといいます. 散布図を描いた時の見た目でなんとなくデータ間の関係が分かるわけです. 見た目ではっきり分かるほど点が線状に並ぶ程, 相関が強いと表現されます (図 5.2.1 はかなり強い). 逆に見た目で何も分からん場合 (図 5.2.1) は, データ間には相関がないと言えます.
非常に便利なので, アンケートデータとか取ったらとりあえずプロットしてみると良いです.

5.2.2 回帰

ちなみにですが, 今は相関を見る際になんとなくで評価しましたが, それっぽい位置に線を引いてあげる (回帰) とよりデータの傾向を強く感じる事が出来るはずです. この, 「それっぽい線」というのが実は曲者で, ひとまず目算でも良いのですが正確に書く場合, メジャーな方法では最小二乗法などが使われます.
そう, 近年では意識スカイツリー系イケイケ起業マンたちに「AI」と呼ばれていたりもするあれです !!! 散布図が描けて, あとは最小二乗法 (図 5.2.2) が使えちゃえば AI エンジニアが名乗れるのです !!! 素晴らしいですね !!

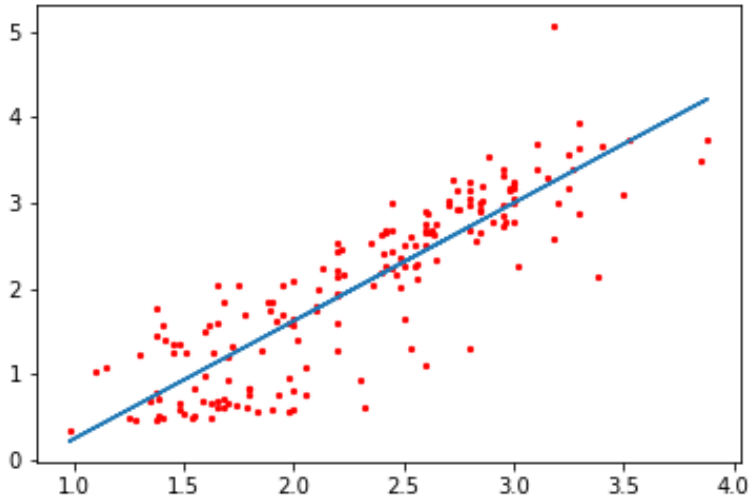

図 5.5: AI エンジニアわい迫真の単回帰分析

Listing 5.4: 単回帰のコード

```
x = [0:0.1:10]+ rand(1,101)
y = [0:0.1:10]
mdl = fitlm(x,y)
scatter(x,y)
hold on
plot(mdl)
hold off
```

5.2.3　共分散

変数間の関係を探るのに非常に便利な相関ですが, やや客観性にかけていた事に気付くでしょう. 「なんとなく」「それっぽい」などの語彙を多用していたはずです. これでは困ってしまいますので, より客観的に評価する方法を考えます.

それが共分散というやつで, 普通の分散は 1 変数において偏差の二乗平均を取るわけですが, 共分散では 2 変数の偏差同士を掛け合わせます. こうする事で, それぞれの偏差の特徴を反映させた新しい分散を定義する事が出来ます.

それぞれの特徴とは, 偏差の符号です. 1 変数分散の場合は単に二乗してしまうため符号は外れますが, 共分散の場合はそれぞれの偏差の符号が異なった場合には負の値を取ります. 偏差の符号が異なるというのは負の相関が, 同じなら正の相関があるという事です.
さらに, 絶対値の方も有用な情報をくれます. それぞれの偏差が大きい程, 共分散の絶対値も大きくなるわけです. これは相関の強さを表します.

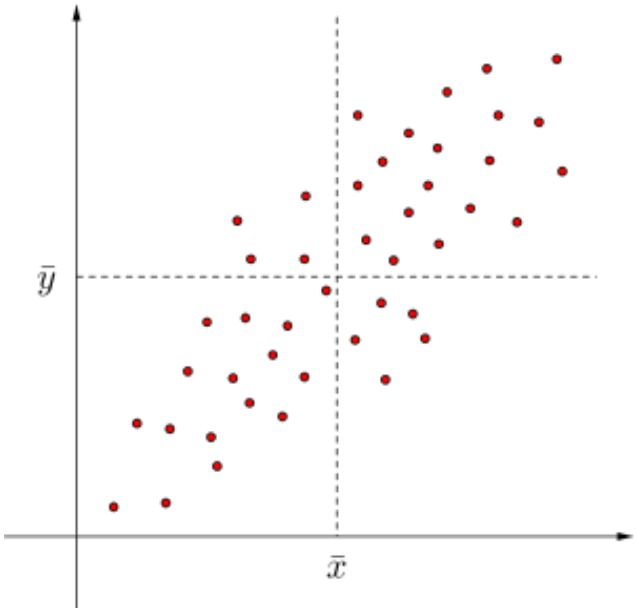

図 5.6: 適当な散布図. どの象限に点が多いかで共分散の値が決まる (高校数学の美しい物語より)

この 2 つを合わせる事で, 共分散は次のような性質を持ちます.

共分散は S_xy, 変数 x と変数 y の相関が強い程絶対値が大きくなり, その符号は相関の正負に一致する. 式は以下です.

$$S_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n} \tag{5.5}$$

Listing 5.5: 単回帰のコード

```
x = [0:0.1:10]+ rand(1,101)
y = [0:0.1:10]

Sxy = (sum((x - mean(x)).*(y - mean(y))))/length(x)
cov(x,y) // 関数を使うとこう共分散行列 ()
```

共分散は，式 (5.5) にもあるように S_{xy} などに求められるものです．2 つの行列の場合，この組み合わせで 4 つ共分散が計算される事になります．つまり $S_{xy}, S_{xx}, S_{yy}, S_{yx}$ です．これらを並べたものが共分散行列と言います．関数 cov はこれを求めるものです．
いうまでもなく，$S_{xx} = var(x)$ です．

5.2.4 相関係数

ただ, 共分散は分散と違い, かけている変数の単位が違うため, 数字自体に意味はない事に注意が必要です．例えば mm と mg で算出したときと, km と kg で算出したときとでは同じデータなのに共分散の値が大きく変わってしまいますし, 単純に平方根を取ったところで違う単位系の積になってるので読み取れる情報がありません．

そこで, 共分散版の標準偏差的なものを定義してあげます．それが相関係数です．

$$r_{xy} = \frac{S_{xy}}{S_x S_y} = \frac{\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{n}}{\sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n}}\sqrt{\frac{\sum_{i=1}^{n}(y_i-\bar{y})^2}{n}}} \tag{5.6}$$

Listing 5.6: 単回帰のコード

```
x = [0:0.1:10]+ rand(1,101)
y = [0:0.1:10]

// ひ，ひぇぇ...
cov(x,y) / (std(x) * std(y)) //
// つまり...
((sum((x - mean(x)).*(y - mean(y))))/length(x)) / (sqrt(sum((x - mean(x)).^2)/
    length(x)) * sqrt(sum((y - mean(y)).^2)/length(y)))

// 関数を使うとこう．相関係数行列
corrcoef(x,y)
```

分子が共分散になっているので, 分母にはそれぞれの標準偏差を入れているのですね．これにより相関係数はその単位に関係ない評価が出来るようになります．当然, 共分散が標準偏差同士をかけたやつよりも上回る事はないので, 相関係数は -1 から 1 の間の値を取ります．
相関係数の値が-1 に近い程負の相関が強く, 1 に近い程正の相関が強いわけですね !!! 便利です !!
コードではやはり，corrcoef で相関係数行列を求めています．