計算機概論 I
数式処理系 Maple 入門

目次

1 ご託宣

今回から 3 回にわたって, 数式処理系 Maple の使い方を解説します.

この資料は, 過去に Maple について講義した内容を集めたもので, 3 回の講義で全てを取り上げるのは不可能です. 講義では, 今回と次回でできる限り内容を進めた後, 3 回目は一番最後の節の RSA 公開鍵暗号をとりあげます. 講義の最終回は, プログラミング言語入門を予定しています. 講義で取り上げることができなかった内容も, 時間がある時に自習してみて下さい.

- 本資料中の問は, レポート課題ではありません. ただし, レポートとして解答を提出した方には, 評価の際に加点します. これまでのレポートの出し忘れとかがある人は, 積極的に提出して下さい. 内容が数学なものは, レポートを送る時は, LaTeX ソースを送って下さい. 件名は, 適当につけて下さい. 締め切りは, 1 月 29 日 (月) とします.
  注意: ネットにある内容をコピペしただけの様な雑なレポートを送られても, 加点はありません.
- コンピュータの入力は, 日本語の文書作成以外のときには, 英語が基本となります. Maple のようなソフトウェアの場合, 世界的な利用者がいる中で, 日本でしか通用しないソフトウェアの仕様を考えるのは無駄です. これは LaTeX でも HTML でも経験していると思います. ほんの少しの専門的な英単語を知るだけで, コンピュータ利用のレベルを高める事ができますので, 英語にアレルギーを持たないようにして下さい.
- リモートでセンターのグラフィカルな maple を使うには, X サーバソフトと ssh + portforwarding で cc.u-ryukyu.ac.jp に接続して, 端末から次のコマンドを入力します.

```
bash-4.4$ xmaple
```

- 基盤情報統括センターの Linux の maple では, X Window で 3D グラフィックスが表示できません. Windows にも maple は入ってますので, 3D グラフィックスを見たい人は, Windows の方を使ってください.

2 Maple とは

Maple は数式処理系です. 数式処理とは, 数学で行う計算, 即ち, 式の展開, 因数分解, 微分, 積分, 方程式を解く, 逆行列を求める, 固有値計算等を数式のままコンピュータでする事を指します[*1]. 実際 Maple, Mathematica, Macsyma(MAXIMA), SageMath 等の数式処理ソフトは, 大学初年級以上の数学を処理する能力を持っており, 数学教育で用いるソフトとして便利ですし, 研究上の実験ソフトとしても役に立ちます.

Maple は, Canada の Waterloo 大学で開発された数式処理ソフトです. Maple は, 沖縄では見ませんが, 楓という木の事です. Canada 国旗の中央部には楓の葉がデザインされています[*2]. 現在, Waterloo Maple Inc. が販売しております. サポートされている OS は, MacOS, Windows, Solaris, Linux (x86) 等です. Maple の開発は, 1980 年頃に始まったようです. この当時は, Maple が動作する PC は存在せず, Work station と呼ばれていた, 少し大きめのシステムで開発さてていました. 1980 年代の後半に, ようやく PC(Mac) で, Maple

[*1] 数式を美しく印刷する作業を数式処理と言う人もいますが, 通常はこちらは組版 (くみはん) 処理といいます.
[*2] 個人的には, パンケーキのシロップは, メイプルシロップ.

が動くようになりました. センターの実習室のマシンに, Maple が入っています. Maple は一般ユーザ版は 20 万円以上しますが, 学生版は格安で購入できるようです. Maple の Web page(https://jp.maplesoft.com/) を見てください. (学生版と言っても機能が劣っているわけではなく, 学生でなくなるとバージョンアップなどのサービスが受けられなくなるもの.)

3 Maple の起動と初期設定

グラフィカルな maple を起動します. アクティビティをクリックして, 全てのアプリケーションを選ぶと, Maple アイコンがあるので, それをクリックします.

このテキストで記述したような入力ができるように, 設定をします.

メニュー → ツール → オプション

と選び,「表示」タブを選択します. 一番上にある「入力表示」の欄で,「Maple 表記」を選びます. そのあと一番下にある「全体に適用」のボタンを押します.

このテキストでの約束: このテキストでは, > で始まっている行は, Maple の入力です.
本文中の専門用語や記号は,「数学系 > 物理学系 > それ以外」の基準で用います. 例えば,「電界」ではなく「電場」,「演算子」ではなく「作用素」等です. 固有名詞については, 正式な日本語表記を知らないものが多いので, 原語表記を基本にしました.

maple に対する情報

- Maple の使い方の On line tutorial があります. 現在の画面の中央部分に,「Get to Know Maple, Fast!」というビデオへのリンクとか, Maple 基本ガイドとかがそれです. ただし, 内容は英語で書かれているようです. また, 入力表記が数学表記になっています. (数学表記は, 見た目は良いのですが, 入力はむしろ面倒になる.)
- Maple に関するプログラミングの本もいくつか出ています. 興味のある人は図書館を探すとか Web で検索するとかして下さい. Web で検索すると英語版のいくつかの本が, 無料でダウンロードできます.
- 以前に Maple のプログラミングを講義しました. そのときの内容は次の場所にあります. 記述が古くなっている部分もありますが, 興味のある人は参考にしてください. `ftp://ftp.math.u-ryukyu.ac.jp/pub/gengo/2002/`

4 最初の一歩

スタートアップウィンドウの画面中央左寄りにある「新規ワークシート」のアイコンをクリックしてください. ウィンドウ内に赤く不等号 > が出 てきます. ここに数式を入力する事で計算ができます. メニューバーのファイルメニューの新規作成からも, このウィンドウを新たに作る事もできます.

4.1 4 則演算など

Maple の四則演算の記号はそれぞれ, $+, -, *, /$ です. Maple では冪乗と階乗が定義されており, それぞれ, ^と! を用います. Maple では, 文の最後をセミコロン ; で終ってエンターキーを押しますと, 文の評価結果が出力されます. エンターキーだけでは単なる改行となります. 入力中の改行は無視されます. 入力の括弧 ( ) は数学と同じ意味になります. 4 則演算の優先順位も数学と一致します. 次を実行してみて下さい.

```
> 1 + 2;
> 10/3 + 2;
> 10/3.0;
> 2^10;
> 50!;
> (a+a-b)*c/d;
> a^2 + a;
```

有理数の扱い, 文字式の扱いが数学と一致します. 小数が式に含まれていれば, 自動的に小数扱いされます.

4.2 様々な定数, 手続き (函数)

Maple では, 数式処理のための手続きが 2700 以上定義されています. それらを全部解説する事は不可能ですので, ここではそれらの例をあげます. これらの例の中に初等函数が用いられていますが, それらの意味は容易に類推できると思いますので, これについての解説はいちいちしません.

数式処理に欠かせない定数 (円周率等) が既に定義されています. 円周率は, Pi という記号を使います. 関数に値を代入する時には, 必ず括弧 () が必要で, しかも数式計算上の括弧は, これ以外には使えません. {}, [ ] は別の意味になります.

```
> Pi;
> cos(Pi/4);
> tan(Pi/2);
> arctan(-infinity);
```

$\sqrt{2}$ がそのまま出て来る事, tan の不定値に対するエラーメッセージに注意して下さい. arctan は逆正接函数, infinity は無限大の事です. 最後の答は, 極限値を出力しています.

Maple では, 有理数, 冪根, 円周率等の定数は, そのまま出力されます. 上の逆正接函数の計算でもそうですし, 例えば, $1 + \dfrac{1}{2^2} + \dfrac{1}{3^2} + \cdots = \displaystyle\sum_{n=1}^{\infty} \frac{1}{n^2} = \zeta(2) = \frac{\pi^2}{6}$ も次で計算させると, 円周率を使った答が出ます.

```
> sum(1/n^2, n=1..infinity);
```

これを小数へ変換するには, evalf という手続きを用います.

```
> evalf(Pi^2/6);
```

Maple では, 非常に正確な数値計算ができます. 例えば, $e^{\pi\sqrt{163}} - 744 - 640320^3$ を C 言語の数学関数で計

算しますと -480 という答を得ます. Maple を使うと, C 言語の計算がとんでもない誤差を含む事がわかります. 起動時では, 浮動小数点の仮数部は 10 桁に設定されており, そのままですと, この結果は真の値の約 40 倍という答になります. そのため仮数部の桁数を事前に設定します. 仮数部の桁数は, Maple のシステム変数 Digits に格納されていますから, この値を変更します. Maple では変数への代入に := を用います.

```
> Digits:=50;
> evalf(exp(Pi*sqrt(163))-744-640320^3);
```

C 言語のライブラリを用いた計算が, 真の値の 6×10^{14} 倍以上の値になっている事がわかります. (exp は exponential の略)

課題 (難): $e^{\sqrt{163}\pi}$ の値が整数に近い理由を調べよ. 菅にこれをきちんと説明できる人は, 私が担当する科目の全ての単位を A であげます. (同じ理由で, $e^{\sqrt{67}\pi}$ も整数に近い.)

文字式の展開, 因数分解も可能です.

```
> expand((x+y)^5);
> factor(a^8-b^8);
```

次の問題は, 2000 年の琉球大学入学試験問題前期日程数学甲の $\boxed{1}$ です.

1. 関数 $\dfrac{x}{\sqrt{1+x^2}}$ を微分せよ.
2. 不定積分 $\displaystyle\int x\sqrt{x^2+2}\,dx$ を求めよ.
3. 定積分 $\displaystyle\int_{\frac{1}{e}}^{2e} x^3 \log x\,dx$ を求めよ.
4. $\displaystyle\lim_{x\to 0}\frac{\sqrt{2x+1}-1-x}{x^2}$ を求めよ.

このような単純な計算は, Maple は得意です. (試験では, 答だけを書いても満点にはならない可能性があります.) Maple V では, 直前の結果を % で参照できます.

```
> diff(x/sqrt(1+x^2),x);
> simplify(%);
> int(x*sqrt(x^2+2),x);
> int(x^3*log(x), x=exp(-1)..2*exp(1));
> limit((sqrt(2*x+1)-1-x)/x^2, x=0);
```

方程式 $f(x)=0$ の解を求める様々な方法も Maple には用意されています. $f(x)$ が 4 次以下の多項式なら, この方程式には代数的な解法が存在する事が知られています. (3 年の代数学 I・II で勉強する予定です.) Maple はこれらの解法を知っており, solve という手続きになっています. 次を実行してみて下さい. これらの解には複素数が含まれますが, Maple では虚数単位は大文字の I で表示されます.

```
> solve(x^3+1,x);
> solve(x^3+3*x+1,x);
```

5 次以上の方程式には, 代数的な解法が一般には存在しない事が知られています (代数的という制限を外せば, 別な解法はあります). 次を実行してみてください.

```
> solve(x^5+x^2+1, x);
```

RootOf($\_Z^5 + \_Z^2 + 1$) という解が出て来ます. もちろん, これは単なるトートロジーに過ぎないのですが, Maple は代数的数を扱えるので, この解 (代数的数) を用いた計算が今後記号的に可能です.

代数的な解法がある場合でも, その解法が複雑な場合には, 残念ながら代数的な解を出力しません. 例えば, 1 の 7 乗根を計算させようとしても, de Moivre (ド モアブル) の公式から出てくる解が単純に出力されるだけです.

```
> solve(x^7-1, x);
```

上で述べたように Maple では, 代数的数が扱えます. これを利用すると, $x^7 - 1 = 0$ の代数的な解も求める事ができます. $x^7 - 1 = (x-1)(x^6 + x^5 + x^4 + x^3 + x^2 + x + 1)$ ですが, 積の右側の 6 次式は $\sqrt{-7}$ を使うと, 2 つの 3 次式の積に因数分解されます。この様な因数分解は, 付け加える数を factor の第 2 引数に加える事で可能です.

```
> factor(x^6+x^5+x^4+x^3+x^2+x+1, (-7)^(1/2));
```

従って, 1 の複素 7 乗根は, 次の 2 つ方程式の解全体です.

```
> solve(2*x^3+x^2-I*sqrt(7)*x^2-x-I*sqrt(7)*x-2,x);
> solve(2*x^3+x^2+I*sqrt(7)*x^2-x+I*sqrt(7)*x-2,x);
```

$f(x)$ が 5 次以上の多項式や, 多項式以外の場合には, 一般的な解法が存在しません. このような方程式の近似解を数値的に計算する方法も Maple は知っています. 上の方程式の数値解も, 次で計算してくれます.

```
> fsolve(x^5+x^2+1, x);
```

実際, 関数のグラフを描画する plot を使って,

```
> plot(x^5+x^2+1,x=-5..5);
> plot(x^5+x^2+1,x=-2..0);
```

等を実行しますと, fsolve で求めた値が $x^5 + x^2 + 1 = 0$ の唯 1 の実数解の近似値である事がわかります.

4.3 パッケージ (package) を利用する

これまでの計算では, 特別な事をせずともそれを実行しますが, 行列の計算などはそのままでは実行してくれず, それを計算するためのパッケージ (ライブラリ) を呼び出す必要があります. ここでは, 線形代数学のパッケージを呼び出して, そこで定義されている計算をしてみます.

パッケージを読み込むには with という手続きを使います. 線形代数学のパッケージ名は linalg (linear algebra の略) となっておりますので, 次のように入力します.

```
> with(linalg);
```

この時に出力されるのが, linalg のパッケージで定義されている手続き名です. 一例をあげますと, 次のようになります.

```
> B:= matrix(2,2);
> trace(B);
> det(B);
> inverse(B);
> eigenvalues(B);
```

どのようなパッケージが利用できるかは, 手続き help に引数 package をいれて利用する事でわかります.

```
> help(package);
```

4.4 グラフィックス機能

上でも述べたように, 例えば, $\sin x$ の関数のグラフを書くには, 次のように入力します.

```
> plot(sin(x), x = -Pi .. Pi);
```

2 変数関数のグラフも簡単に書けます. (基盤情報統括センターの現在の Linux 版 Maple では, 3D グラフィックスが動作しません.)

```
> plot3d(cos(2*x^3+y^2), x=-2..2, y=-2..2);（現在これは Linux では動作しない）
```

マウスの左ボタンで図形を掴んで動かせば, 立方体が動きます.

Maple では, 座標を順に与えてそれを線分で結び多角形を描く事ができます. まずは, グラフィック表示のためのパッケージ plots を読み込みます.

```
> with(plots);
```

次を実行して見て下さい.

```
> cornercoordinates := [[0,0],[1,0],[1,-1] ,[0,-1]]:
> asquare := polygonplot(cornercoordinates):
> display(asquare);
> ngon := n -> [seq([cos(2*Pi*i/n),sin(2*Pi*i/n) ], i = 1..n)]:
> display([polygonplot(ngon(8))]);
> fivestar:=[seq([cos(2*Pi*(2*i+1)/5),sin(2*Pi*(2*i+1)/5)], i = 1..5)]:
> display([polygonplot](fivestar));
```

4.5 Help

Maple 本体のウィンドウのメニューバーの右端にヘルプメニューがあります. ここからさまざまな機能を知る事ができます. Maple は非常に多機能です. ヘルプブラウザーで必要な項目にたどり着くのも大変です. ブ

ラウザー以外にもヘルプメニューの Topic Search, Full Text Search で検索することができます. さらに, 上で述べたように Maple のプロンプト行で

```
> ? キーワード
```

あるいは

```
> help(キーワード);
```

とすると, Topic Search とほぼ同じ事が実行されます. 次でも, 利用上役立つ様々な情報が得られます. 検索すべきキーワードが何であるかというのは, 他の講義から学習してください.

```
> help(help);
```

5　2 変数関数のグラフ (微分積分学の復習)

現在のこの節の内容は, Linux では, 最後の陰関数のグラフ以外は動作しません.

5.1　微分可能性と連続性

次の函数は, 原点で x, y について共に偏微分可能ですが, 原点で不連続です.

$$f(x) = \begin{cases} 0 & (x,y) = (0,0) \\ \dfrac{xy}{x^2+y^2} & (x,y) \neq (0,0) \end{cases}$$

原点の近くでグラフを書かせます. plot3d で, 3 次元のグラフが描けます.

```
> plot3d(x*y/(x^2+y^2), x=-1..1, y=-1..1);
```

次の函数は, 原点で x, y について共に偏微分可能ですが, (全) 微分可能ではありません.

$$f(x) = \begin{cases} 0 & (x,y) = (0,0) \\ \dfrac{x^2|y|}{x^2+y^2} & (x,y) \neq (0,0) \end{cases}$$

実際にグラフを書かせてみますと, 原点で接平面をひく事ができない事が視覚的にわかります.

```
> plot3d(x^2*abs(y)/(x^2+y^2), x=-1..1, y=-1..1);
```

問 1　この関数が原点で全微分可能でないことを示せ.

上の入力で abs(y) は, $|y|$ の事です. (absolute value)

```
> ?abs
```

で abs の詳しい説明が表示されます. 複素数を代入しても正しい絶対値が出力される事がわかります.

次の函数は, $\dfrac{\partial^2 f}{\partial x \partial y}(0,0) \neq \dfrac{\partial^2 f}{\partial y \partial x}(0,0)$ となる函数です.

$$f(x) = \begin{cases} 0 & (x,y) = (0,0) \\ \dfrac{xy(x^2-y^2)}{x^2+y^2} & (x,y) \neq (0,0) \end{cases}$$

$f(x,y)$ のグラフからは, 上の事を想像するのは難しいですが, $\dfrac{\partial^2 f}{\partial x \partial y}$ のグラフを書かせると, 原点の近傍での異常さが良く見えます (原点以外では, $\dfrac{\partial^2 f}{\partial x \partial y}(x,y) = \dfrac{\partial^2 f}{\partial y \partial x}(x,y)$). 関数の微分は, diff で計算できます.

問 2　この函数について $\dfrac{\partial^2 f}{\partial x \partial y}(0,0)$, $\dfrac{\partial^2 f}{\partial y \partial x}(0,0)$ を計算せよ.

```
> f:=x*y*(x^2-y^2)/(x^2+y^2);
> plot3d(f, x=-1..1, y=-1..1);
> diff(f, x, y);
> g:=simplify(%);
> plot3d(g, x=-1..1, y=-1..1);
```

ここで % は直前の結果を参照するのに用います.

5.2　停留点 (臨界点) と極大極小

滑らかな函数 $f(x_1, \cdots, x_n)$ に対して $\dfrac{\partial f}{\partial x_1}(p_1, \cdots, p_n) = \cdots = \dfrac{\partial f}{\partial x_n}(p_1, \cdots, p_n) = 0$ を満たす点 $P = (p_1, \cdots, p_n)$ を f の停留点 (stational point) あるいは臨界点 (critical point) といいます. 停留点は f の極大点極小点の候補ですが, 極大あるいは極小になることは保証しません.

f が P で停留点である時, Hessian

$$\mathrm{Hess}(f)(P) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1}(P) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(P) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(P) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n}(P) \end{pmatrix}$$

を考えます. $\mathrm{Hess}(f)(P)$ が非退化な時, 非退化な臨界点, 退化する時, 退化する臨界点といいます [8].

非退化な臨界点の近傍では, 上手に座標変換をする事により, $\mathrm{Hess}(f)(P)$ の符号数を $(p,q), (n = p+q)$ とすると 2 次形式

$$f(x_1, \cdots, x_n) = {x_1}^2 + \cdots + {x_p}^2 - {x_{p+1}}^2 - \cdots - {x_n}^2$$

の形で近似されます. (Morse の補題 [8]). また, このことが根拠で, 非退化な臨界点での f の極大性, 極小性が判定できます. 2 変数函数で典型的な場合のグラフを書いてみます.

1. 符号数 $(2,0)$, 極小点.

   ```
   > plot3d( x^2+y^2, x=-1..1, y=-1..1);
   ```

2. 符号数 $(1,1)$, 鞍点 (saddle point).

   ```
   > plot3d(x^2-y^2, x=-1..1, y=-1..1);
   ```

3. 符号数 $(0,2)$, 極大点.

```
> plot3d(-x^2-y^2, x=-1..1, y=-1..1);
```

退化する臨界点では, Hessian からは極大極小等は判断できません. 次の例をみて下さい.

1. 原点で退化かつ極小.

```
> plot3d(x^2*y^2, x=-1..1, y=-1..1);
```

2. 原点で退化かつ極大でも極小でもない.

```
> plot3d(x^3 - 3*x*y^2, x=-1..1, y=-1..1);
```

注意 1 一般に滑らかな多様体 M に対して, 退化しない臨界点しかもたない滑らかな函数が存在します. (Morse 函数). この Morse 函数の臨界点で, その Hessian の正符号の次元の胞体を張り合わせる事により, もとの多様体の胞体分割が得られます. これを Morse 理論といいます. 詳しくは, [8] を読んで下さい.

5.3 陰函数のグラフ

$f(x,y)=0$, $f(x,y,z)=0$ の形のグラフは, implicitplot, implicitplot3d で書く事ができます. これらは, plots package を読み込まないと使えません.

```
> with(plots);
```

1. 楕円曲線 $y^2=x(x-1)(x-2)$

```
> implicitplot(y^2-x*(x-1)*(x-2), x=-1..3, y=-3..3);
```

2. 球面 $x^2+y^2+z^2=1$

```
> implicitplot3d(x^2+y^2+z^2-1, x=-1..1, y=-1..1, z=-1..1);
```

注意 2 楕円曲線の場合の Morse 函数は $(0,0)$ からの距離, 球面の場合は $(0,0,1)$ からの距離ととれます. 楕円曲線は実数でなく, 複素数 (の射影空間) で考えるほうが, よりその性質が分かります. 実際, 複素射影平面 (2 次元複素空間に無限遠点を付け加えた空間) なかでの曲面だと思うと, トーラス (ドーナツの表面) と同じ形をしていることが示されます.

6 Maple での procedure 入門と高木の関数

ここでは例として, 不思議な性質を持つ連続函数を扱います. 次の函数を定義する事を例にプログラムの書き方の第一歩を述べます.

$$t(x)=\begin{cases} 0, & x<0 \\ 2x, & 0\leqq x<\frac{1}{2} \\ 2(1-x), & \frac{1}{2}\leqq x<1 \\ 0, & 1\leqq x \end{cases}$$

この函数は, Maple の中で定義されていませんから, プログラムを作って定義します. 次を入力してみて下さい. Warning(警告) が表示されますが, end; を入力した時に消えます.

```
> t:=proc(x)
>   if x < 0 then 0;
>   elif x < 0.5 then 2*x;
>   elif x < 1 then 2*(1-x);
>   else 0;
>   fi;
> end;
> t(1/3);
> t(-1);
> t(0.2);
> plot(t, -1..2);
```

上では, procedure を定義しています. Maple でのプログラミングは, この procedure を書くです. Maple では procedure とは何らかの 1 連の仕事を定義する事になります.

一般的な話として, procedure(日本語で「手続き」) とは, プログラムを意味のあるまとまりに分解するための, プログラミング言語が持つ仕様であると考えて下さい. プログラムを仕事単位に分割する事は, 大きなプログラムを書くには絶対に必要になります. 従って, ほとんどの手続き型プログラミング言語では, procedure に似た概念が必ずあります. (C 言語だと function(函数), PASCAL だと function と procedure.) 上で, 手続き型プログラミング言語と書きましたが, そうでないプログラミング言語もたくさんあります. 例えば, Objective C, Swift, Java は, オブジェクト指向プログラミング言語といわれます. このような言語では, プログラムの分割には別の考え方をしますが, ここではこれ以上の事は述べません.

最初の行が, 手続き t の宣言部分で, t は 1 つの変数 (プログラミングでは, 引数と言う事が多い. 英語で argument) を持つ事を宣言しています. Maple では, := は代入を意味します (awk では単なる = が代入ですから, 処理系によって記号が違うことを理解してください. また, 文の終わりには, 必ずセミコロン ; が必要です.). 次の行からが t の定義で, if 文によって条件分岐をさせています. 条件分岐は, awk でもでましたのでやらせたい事はわかると思います. 記述の仕方が awk とは異なります.

1. Maple では else if **ではなく** elif
2. Maple では, さいごに fi (if をひっくり返したもの) で閉じる.

そして, 最後の `end;` で手続きを閉じます. これを入力した時点で, Maple はその手続きを評価して現在の状態にその手続きを付け加えます. 従って, その直後に `t(1/2)` の値が計算できるようになります.

6.1 高木の函数

微分積分学等では, 通常, 微分できる滑らかな函数の性質を調べます. 1 変数函数では, 微分可能なら連続です. では, 連続と微分可能性には, どの程度の差があるのでしょう.

1 つの答が, Weierstrass(ワイヤストラス) によって与えられました. Weierstrass は閉区間 $[a, b]$ で連続で, かつ全ての点で微分不可能な函数の存在を具体的な例で示しました. ここでは Weierstrass の例は取り上げず,

高木貞治によって見出された, このような函数の例 (の近似) のグラフを書いてみます.

高木 (高木貞治) の函数は, 次のように定義されます. $t(x)$ を前節で定義した函数とします. $t(x)$ を n 回合成してできる函数を $t_n(x)$ とする時, 高木の函数 $T(x)$ は次で定義されます.

$$T(x) = \sum_{n=1}^{\infty} \frac{t_n(x)}{2^n}$$

問 3 1. 高木の函数が連続函数である事を示せ.
2. 高木の函数が区間 $[0,1]$ で至るところ微分不可能である事の証明を考えよ.

T(の近似) のグラフを書くプログラムを書きます. $t(x)$ は上で定義しました. maple では, `(t@@n)` で t の n 回の合成函数となります.

```
> plot((t@@5), 0..1);
```

高木の函数の定義において, 第 10 項までの部分和を T とすると, Maple では次のように書けます. ここで, `->`も函数 (写像) の定義として使う事ができます. `sum` は和の意味です.

```
> T:= x-> sum((t@@k)(x)/2^k, k=1..10);
```

T を $[0,1]$ でプロットしてみます.

```
> plot(T, 0..1);
```

いたるところ微分できない函数については, 次の事が証明できます.

定理 1 $C[0,1]$ を区間 $[0,1]$ で連続な函数全体がなすベクトル空間とする. $f \in C[0,1]$ に対して

$$\| f \| = \max_{0 \le x \le 1} |f(x)|$$

とする.
1. $C[0,1]$ は上の $\| \cdot \|$ に関して完備なノルム空間になる (従って, 完備な距離空間になる).
2. $C'[0,1] = \{ f \in C[0,1] \mid f$ は全ての点で微分できない $\}$ は, $C[0,1]$ で稠密な部分集合になる.

問 4 上の定理を証明せよ.

7 Cantor(カントール) 集合と Cantor の階段 (再帰入門)

Cantor の階段というのは [0, 1] で定義された関数 $C(x)$ のグラフで, 次の性質を持ちます.

1. $C(0) = 0,\ C(1) = 1.$
2. $C(x)$ は区間 $[0,1]$ で連続かつ広義単調増加.
3. $C(x)$ は, 殆んどいたるところ (Lebesgue(ルベーグ) 測度 0 の集合の補集合) で微分できかつそこで $C'(x) = 0.$
4. $C(x)$ が増加している場所は, 上のことから Lebesgue 測度 0 であるが, その集合の濃度は非可算個.

ここでは, Cantor の階段が持つ自己相似性を利用して, 再帰的に Cantor の函数の近似を定義します.

7.1 再帰

再帰と言うのは, 手続きを実行する際にその手続き自身を呼び出す操作の事をいいます. しかし, 数理科学科の皆さんには, 漸化式をプログラムで書くと言った方がわかりやすいでしょう. ただし, 漸化式は高校では帰納的定義と呼ばれ, 1 から順に数列を定義して行く方法ですが, 再帰はこれとは逆に n を与えて, ここから小さい方に向かって計算し, 最後に 1 まで到達して手続きが終了するという形になります.

例えば, 1 から n までの和を求める手続きを summation と言う名前でつくるとします. summation は漸化式, summation(1)=1, summation(n) = summation(n-1)+n を満たします. ほとんどのプログラミング言語では, この漸化式をそのままプログラムすれば, 手続きが完成します. 実際, Maple では summation は次のようになります.

```
> summation:=proc(n)
>   if (n=1) then 1;
>   else summation(n-1)+n;
>   fi;
> end;
> summation(10);
> summation(10000);
```

上でわかるように, summation(10) は summation(9) + 10 と計算されます. すなわち, summation(10) の計算に, 手続き summation が引数を 9 として呼ばれているわけです. もちろん summation(9) はまだ確定していませんからさらに summation(8) + 9 と計算されます. これをどんどん繰り返して summation(1) = 1 まで実行する事により, summation(10) の計算が終ります.

このような函数の呼び出し方法を, 再帰呼び出しといいます. 上の例でもわかるように, プログラムの記述がとても単純になるところが再帰の利点であり, これを処理する計算機の方にすると, スタックと呼ばれるメモリ領域を消費するので (初期値に戻るまでの計算部分を全て保存しておかなければいけない), システムによってはメモリエラー (メモリ不足) の原因になり得るという側面を持っています.

注意 3 上の summation は Maple 自身で既に定義されている sum という手続きを使って, 計算できます.

```
> sum('k', k=1..10);
> sum('k', k=1..n);
> sum('k^2', k=1..n);
```

問 5 1. Fibonacci(フィボナッチ) の数列, fib(1)= a, fib(2)= b, fib(n) = fib($n-1$)+ fib($n-2$) の第 n 項を出力する手続き fib を再帰を使って書け.
2. 上の Fibonacci 数列の一般項を (数式を用いた計算で) 求めよ.

7.2 Cantor の集合

Cantor の集合は, $[0,1]$ の部分集合で, 次のように定義されます.

$E_0 = [0,1]$
$E_1 = E_0 \setminus \left[\frac{1}{3}, \frac{2}{3}\right)$
$E_2 = E_1 \setminus \left[\frac{1}{3^2}, \frac{2}{3^2}\right) \setminus \left[\frac{7}{3^2}, \frac{8}{3^2}\right)$
$\cdots$

としたとき, $E = \bigcap_{i=1}^{\infty} E_i$ を Cantor 集合といいます.

問 6 1. E は非可算の濃度を持つ事を示せ (E と $[0,1]$ の間の全単射を作れ).
2. E の Lebesgue 測度は 0, 言い換えると E の補集合の Lebesgue 測度が 1 である事を示せ.

7.3 Cantor の函数

ここでは, Cantor の函数の近似は, 次のように定義します. 実際の Cantor の函数とはすこしずれます. 下の定義では, 函数が微分できない点で 0 としてあります. それらの点は極限では測度 0 なので, 近似のグラフとしては, 十分だと思います.

まず, $C_1(x)$ を次で定義します.

$$C_1(x) = \begin{cases} 0 & 0 \leqq x < \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} \leqq x < \frac{2}{3} \\ 0 & \frac{2}{3} \leqq x < 1 \\ 1 & x = 1 \end{cases}$$

次に $C_n(x)$ を次で定義します.

$$C_n(x) = \begin{cases} \frac{1}{2} C_{n-1}(3x) & 0 \leq x < \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} \leq x < \frac{2}{3} \\ \frac{1}{2}\left\{C_{n-1}(3(x-\frac{2}{3})) + 1\right\} & \frac{2}{3} \leq x \leq 1,\ C_{n-1}(3(x-\frac{2}{3})) \neq 0 \\ 0 & \frac{2}{3} \leq x \leq 1,\ C_{n-1}(3(x-\frac{2}{3}) = 0 \end{cases}$$

この時, $C(x) = \lim_{n\to\infty} C_n(x)$ を Cantor の函数の近似とします. (実際の Cantor 関数は, Cantor 集合を除いたところで定義されるものを連続に拡張した関数.)

問 7 Cantor 集合 E の補集合で $C(x)$ は微分できて, そこで $C'(x) = 0$ となることをしめせ.

これの第 n 項を出力する手続きは, 次のようになります. n に関して再帰を用いています.

```
> cantor:=proc(n,x)
>  if (x<0) then 0 fi;
>  if (x>1) then 0 fi;
>  if (n=1) then
>  if (x<1/3) then 0;
>   elif (x<2/3) then 1/2;
>   elif (x=1) then 1;
>   else 0;
>   fi;
>  else
>    if (x<1/3) then cantor(n-1, 3*x)/2;
>    elif (x<2/3) then 1/2;
>    else
>      if (cantor(n-1, 3*(x-2/3))=0) then 0;
>      else cantor(n-1, 3*(x-2/3))/2 +1/2;
>      fi;
>    fi;
>  fi;
> end;
> C:=x->cantor(20,x);
> plot(C, 0..1);
```

8 更にフラクタル

高木の関数や Cantor 関数のグラフはフラクタル (Fractal) と呼ばれる図形で, 自然界ではリアス式海岸の海岸線や, 人体の肺の構造等に似たような物が現れます. ここでは, 複素力学系と呼ばれるもので現れるフラクタルを 2 つ紹介します.

複素力学系とは, 複素変数の有理型関数 f を用いて漸化式 $a_n = f(a_{n-1})$ で与えられる数列の $n \to \infty$ の時の漸近的挙動を調べるものです. f が簡単な関数であっても, 複素数値までこめて考えるとその漸近挙動を予想する事が, ほとんど不可能であると言う事が分かります. ([5]).

8.1 Julia(ジュリア) 集合

Julia 集合は f が多項式の時に上で決る数列の漸近挙動を問題にします. ここでは, f として非自明な結果を与える最も単純な $f(z) = z^2 + c$ を扱います. $|c|$ が小さい時, 初期値 $|a_0|$ が小さければ, 数列は無限大に発散せず収束または準周期的な運動をしますが, 大きくなると無限大に発散します. Julia 集合は, この発散する点とそうでない点の境界の事です. 下の入力で現れる絵では, 緑系統の色と赤系統の色の境界部分にがそれにあたります.

グラフィックスパッケージを使うので, まずそれを利用する事を Maple に知らせます. また, できるだけ正確な絵を描かせるために, 計算の精度を上げます.

```
> with(plots);
> Digits:=20;
```

julia と言う名前の procedure を定義し, 発散がはっきりするまでの繰り返しの回数を出力するようにし, それを densityplot という密度表示グラフィックス関数で表示します.

```
> julia := proc(x,y)
>   global c;
>   local z, iter;
>   iter:=0;
>   z:=evalf(x+y*I);
>   to 32 while evalf(abs(z)) <2.0 do
>     z:=evalf(z^2+c);
>     iter:=iter+1;
>   od;
>     iter;
>  end;
> c:=0.3;
> densityplot(julia, -1.5..1.5, -1.5..1.5,colorstyle=HUE,
   style=patchnogrid,grid=[256,256]);
```

絵が現れるまで数分かかると思います.

手続きの中で, 変数が宣言されています. 通常のプログラミングでは, 「変数の宣言」という行為が必要で, 面倒と思うかも知れませんが, これがある事でプログラムの誤りの発見に繋がる事もあります. `global c;` の部分は大域変数 (グローバル変数) の宣言ですが, maple では, これは手続きの外からも変更可能な変数を意味します. `local z, iter;` は局所変数で, 手続きの中だけで使われ, この値は 返り値にしない限り, 手続きの外からは参照も変更もできません. また手続きの中の I と言う文字は虚数単位です. evalf は浮動小数点数としての値の計算です. こうしておかないと Maple は完全に誤差の無い計算を行いますが, それでは計算量が膨大になって, 計算終了までの時間がかかりすぎるのです.

8.2 Mandelbrot(マンデルブロ) 集合

Mandelbrot 集合は, 上と同じ関数 $f(z) = z^2 + c$ を扱います. ただし, 初期値は $a_0 = 0$ に固定し, 定数項 c の値によって $n \to \infty$ 数列 a_n で発散するかどうかを見てその境界を問題にします.

```
> mandelbrot := proc(x,y)
>  local z, iter;
>  iter:=0;
>  z:=0;
>  to 64 while evalf(abs(z)) <4.0 do
>    z:=evalf(z^2+x+I*y);
>    iter:=iter+1;
>  od;
>  iter;
> end;
> densityplot(mandelbrot,-2.0..1.0,-1.5..1.5,colorstyle=HUE,
  style=patchnogrid,grid=[256,256]);
```

Cantor 集合や Cantor 関数は, Lebesgue 積分の教科書には, だいたい書いてあると思います. Lebesgue 積分は 3 年次の関数解析学で習うはずです.

8.3 Newton 法から出てくるフラクタル

次は, 複素数で Newton 法を行ってできるフラクタルの絵です. 詳しくは, 下に挙げた参考書を見てください. 時間の都合で解説はできませんが, 時間に余裕のある方はお楽しみください

```
> cnewton:=proc(x,y)
>  local z, nxt, iter, err;
>  z:=evalf(x+I*y);
>  iter:=0;
>  nxt:=0;
>  err:=evalf(abs(nxt-z));
>  to 20 while(abs(err)> 0.000001) do
>    nxt:=evalf(z-(z^3-1)/(3*z^2));
>    err:=z-nxt;
>    z:=nxt;
>    iter:=iter+1;
>  od;
>  iter;
> end;
> densityplot(cnewton,-1..1,-1..1,colorstyle=HUE, style=patchnogrid, grid=[256,256]);
```

9 Taylor 級数 (Maclaurin 級数)

微分積分学で習った Taylor 級数も Maple で計算出来ます. まずは復習です. 函数 $f(x)$ が $x=a$ の付近で n 階微分可能なとき, $x=a$ における Taylor の定理は次のようになりました.

$$f(x) = f(a) + (x-a)\frac{f'(a)}{1!} + (x-a)^2\frac{f''(a)}{2!} + \cdots + (x-a)^{n-1}\frac{f^{(n-1)}(a)}{(n-1)!} + R_n.$$

ここで

$$R_n = (x-a)^n\frac{f^{(n)}(\xi)}{n!}$$

は剰余項で ξ は x と a の間に存在します.

さて, 函数 $f(x)$ が $x=a$ の付近で何回でも微分可能で, さらに剰余項が $\lim_{n\to\infty} R_n = 0$ なら, $f(x)$ は a の近傍で無限級数の形に書けます. すなわち,

$$f(x) = f(a) + (x-a)\frac{f'(a)}{1!} + (x-a)^2\frac{f''(a)}{2!} + \cdots + (x-a)^n\frac{f^{(n)}(a)}{n!} + \cdots .$$

となります. この時, 函数 $f(x)$ は, $x=a$ で実解析的といい, これを $f(x)$ の $x=a$ での Taylor 展開と呼びます. また, 複素函数 $f(z)$ は, $z=a$ で (複素函数として) 微分可能であれば, Taylor 展開が可能です.

特に $a=0$ のときを Maclaurin 展開と呼びます. 上の式から明らかですが, 復習のため $f(x)$ の Maclaurin 級数展開を, 各自書いてみて下さい.

9.1 Maple での級数展開

しばらくは, Maple に級数の部分和のグラフを書かせることで, Taylor 展開の意味を再確認します.

Maple は函数を有限項の Taylor 級数 (より一般には Laurent 級数) に展開することが出来ます. この展開には series という手続きを使います. e^x の $x=0$ における Taylor 展開を求めるには次の様に入力します.

```
> series(exp(x), x=0);
```

ここで最後の項に現われる $O(x^6)$ は 6 次以上の項という意味です. Maple のデフォルトの設定では, 最初の 5 項まで (0 の項がある場合, それは表示されないが, 項数としてはカウントされる.) が出力され, それ以上の項はまとめられます. (3 番目のパラメータに数字を入れることによって, この設定を変えることが出来ます.)

次に $\log x$ の $x=1$ での展開を 3 次の項までを求めてみましょう. まず, Maple に計算させる前に上の Taylor 展開の式に従って手計算で 3 次の項まで求めてみて下さい.

次に Maple に計算させて結果を確かめます. Maple では自然対数は ln です.

```
> series(ln(x), x=1, 4);
```

3 番目のパラメータの 4 は 4 次以上の項はまとめるという意味です. $\sin x$ の $x=0$ での展開を 7 次の項まで求めてみます.

```
> series(sin(x), x=0, 8);
```

9.2 Laurent(ローラン) 展開 (3 年次で学習する)

複素函数 $f(z)$ が $z=a$ で極を持つとは, $a(z\neq a)$ の近傍で z について微分可能で, $\lim_{z\to a}|f(z)|=\infty$ となることを言います. このとき, 自然数 n が定まって, $z=a$ の近傍で, 次の式が成立します.(左辺の $f(z)$ と右辺の無限和が, 適当な正の数 δ が存在して, 領域 $\{z\in\mathbb{C}:|z-a|<\delta,\ z\neq a\}$ で一致する.)

$$f(z)=\frac{a_{-n}}{(z-a)^n}+\frac{a_{-n+1}}{(z-a)^{n-1}}+\cdots+a_0+a_1(z-a)+\cdots+a_k(z-a)^k+\cdots\qquad(a_{-n}\neq 0)$$

このとき, 上の式の右辺を f の $z=a$ での Laurent 展開といい, n を f の $z=a$ での極の位数, a_{-1} をその極での留数と言います. Laurent 展開も series で計算できます. Laurent 展開は, 解析学 I, II で講義されますので, 3 年生の時にしっかり勉強して下さい. 留数は, ある種の難しい定積分の計算に威力を発揮します.

```
> series(1/sin(x), x=0);
> series(Zeta(s), s=1);
> series(tan(x), x=Pi/2);
```

ここで, Zeta(s) は, Riemann のゼータ函数で, $\Re(s)>1$ で $\zeta(s)=\sum_{n=1}^{\infty}\frac{1}{n^s}$ で定義したものを全平面に有理型函数として解析接続したものです. ゼータ函数の Laurent 展開の定数項は Euler の定数で, $\lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k}-\log n\right)=0.5772156649\cdots$ です.

極での留数は residue ライブラリを読み込ませれば計算する事ができます.

```
> readlib(residue);
> residue(Zeta(s), s=1);
> residue((1/sin(x))^3, x=0);
```

問 8（3 年生向き） $\dfrac{1}{\sin^3 z}$ の $z=0$ での留数を求める計算法は?(計算の手順を定義にしたがって Maple で順に計算する事を考えても良い.)

9.3 Taylor 級数の部分和とグラフ

Taylor 級数の部分和は多項式になりますが, Maple では関数 convert(xxxx,polynom) を使うと, まとめられた項 $O(x^n)$ を取り除いて多項式にすることが出来ます. (関数 convert には他にも機能がいろいろあります. 興味のある人は?convert を見ると良いでしょう)

$\sin x$ の 3 次までの部分和は, 次になります

```
> s3:= convert(series(sin(x), x=0, 4), polynom);
```

練習: $\sin x$ の 5 次までの部分和 $s5$ と 9 次までの部分和 $s9$ を Maple で求めて下さい.

部分和の次数が高くなるに従って, Taylor 展開がもとの関数に近づいていく様子をグラフにして確かめてみましょう. グラフを描くための関数は plot でした.

```
> plot(s3, x=-Pi..Pi);
```

となります. 上記の s3 を, s9 とかえてグラフの様子を確かめてみて下さい.

```
> plot(s9, x=-Pi..Pi);
```

s9 では $-\pi \sim \pi$ の範囲では sin のグラフに近づきますが, 少し範囲を広げるとどんどん遠ざかります.

```
> plot(s9, x=-2*Pi..2*Pi);
```

もう少し高い冪まで計算するとかなり広い範囲まで sin 曲線を近似しているのがわかります. $\sin x$ のグラフと同時に表示させてみましょう.

```
> with(plots);
> A:=plot(convert(series(sin(x),x=0,22),polynom), x=-10..10, color=BLUE);
> B:=plot(sin(x), x=-10..10, color=RED);
> display({A, B});
```

10 常微分方程式

未知関数をそれらの導関数の間の関係式から, もとの未知関数を求める事を問題にするのが, 微分方程式論です. とくに, 未知関数が 1 変数関数で関係式がその関数の導関数を使って表示される時, 常微分方程式といいます. これに対して, 熱方程式のように偏微分を含むものを, 偏微分方程式といいます. 熱方程式以外に Laplace(ラプラス) 方程式 (ポテンシャル論) と波動方程式が, 基本的な偏微分方程式として重要です.

例えば, 次のような微分方程式は, どの常微分方程式の本にも書いてあるものです.

$$y' = ky$$

$$y' = y^{\frac{1}{3}}$$

$$y'' + a(x)y' + b(x)y = 0 \quad (2 \text{ 階線形微分方程式})$$

$$x(1-x)y'' + \{\gamma - (\alpha+\beta+1)x\}y' - \alpha\beta y = 0 \quad (\text{Gauss(ガウス) の超幾何微分方程式})$$

最も一般的には, 常微分方程式とは適当な領域で定義された適当な函数 F があって, 式

$$F(x, y, y', \cdots, y^{(n)}) = 0$$

の事を常微分方程式と言います. 常微分方程式を解くとは, 上の式を満たす函数を見つける事に他なりません. そのためには, 解の存在と一意性を調べる必要がありますが, これに関しては次の定理があります

定理 2 $F(x, y)$ が平面内の領域 D で連続で次のリプシッツ (Lipshitz) 条件を満たしているとする.

$$|F(x, y) - F(x, y_1)| \leqq L|y - y_1|$$

ここで, L は正の定数である. このとき, $(a, b) \in D$ に対して,

$$y' = F(x, y), \quad y(a) = b$$

となる函数 $y(x)$ が (a, b) の近くでただ 1 つ存在する.

問 9　$y' = y^{\frac{1}{3}}$ がリプシッツ条件を満たすための領域 D の満たす必要十分条件は何か

これは, 1 階の特別な形の方程式の解の存在と一意性の定理ではありますが, 本質的にこの定理だけで, 一般の (高階の) 解の存在と一意性の証明がなされます. また, 後で見るように, リプシッツ条件は一意性を保証するためには, 本質的です[*3]. 詳しくは常微分方程式の本を読んで下さい.

10.1　1 階の常微分方程式とベクトル場

1 階の常微分方程式

$$y' = F(x, y)$$

を考えます. y' というのは, 函数 $y = y(x)$ の接線の傾き, すなわち接線ベクトルの向きですから, 上の式は,

$$(x, y) \longmapsto (1, F(x, y))$$

とみると, 平面内のベクトルの分布を与えている事になります.

このようなベクトルの分布の事を, ベクトル場 (vector field) と呼びます. 本当はこれも多様体の上で考えるのが適切なのですが, ここではそこまで立ち入りません. 1 階の常微分方程式を解くというのは, このベクトル場に沿ったグラフを持つような函数 (これを積分曲線と言う) を, うまく見つけるという事に他なりません.

10.2　Maple によるベクトル場の視覚化

ベクトル場を表示するには, fieldplot(平面) と fieldplot3d(空間) を使います. 次を実行して見て下さい.

```
> fieldplot([1, y], x=-1..1, y=-1..1);
> fieldplot([y, -x], x=-1..1, y=-1..1);
> fieldplot([1, y^(1/3)], x=-1..1, y=0..1.5);
```

Maple には, もう 1 つ gradplot という手続きも用意されています. 函数 $f(x, y)$ に対して,

$$\mathrm{grad} f = \left(\frac{\partial f}{\partial x}(x, y),\ \frac{\partial f}{\partial y}(x, y) \right)$$

というベクトル場を f の勾配 (gradient) といいます. これを表示するための手続きです.

```
> gradplot(x^2+y^2, x=-1..1, y=-1..1);
```

問 10　1. $(g(x, y), h(x, y))$ が適当な函数 f に対して, $\mathrm{grad} f = (g(x, y), h(x, y))$ となるためには, $\dfrac{\partial g}{\partial y} = \dfrac{\partial h}{\partial x}$ が必要である事を示せ.(函数は全て滑らかとする)
2. 上の条件は十分条件であるか

次のベクトル場は, 原点に正の単位電荷, 点 (0.5, 0, 0) に負の単位電荷をおいた時の電場になります. gradplot3d は, $f(x, y, z)$ に対する勾配ベクトル場を表示する手続きです.

$$\mathrm{grad} f = \left(\frac{\partial f}{\partial x}(x, y, z),\ \frac{\partial f}{\partial y}(x, y, z),\ \frac{\partial f}{\partial z}(x, y, z) \right)$$

[*3] 存在だけだと, F の連続性だけで証明できる

```
> gradplot3d((x^2+y^2+z^2)^(-1/2)-((x-0.5)^2+y^2+z^2)^(-1/2),
  x=0..0.5,y=-0.25..0.25,z=-0.25..0.25);（現在動作しない）
```

10.3 Maple で常微分方程式を解く

Maple で微分方程式を解くには, dsolve を使います.

```
> de1:= diff(y(x),x)-y(x)=0;
> dsolve(de1, y(x));
> de2:= diff(y(x),x)-y(x)^(1/3)=0;
> dsolve(de2, y(x));
> de3:=x*(1-x)*diff(y(x),x$2)+(gamma-(alpha+beta+1)*x)*diff(y(x),x)-alpha*beta*y(x) = 0;
> dsolve(de3,y(x));
```

2 番目は, 解の一意性が成り立たない例ですが, $y \equiv 0$ という自明な解は出力されません. 3 番目の解の hypergeom というのはガウスの超幾何関数と呼ばれるものです. diff の中にある $2 と言う記号は, 2 階微分の意味です. これは, 微分方程式を冪級数解で解くとどういう物かがわかります.

```
> dsolve(de3, y(x), type=series);
```

de3 は 2 階の線形常微分方程式ですから, 1 次独立な解が 2 つあり, それらの線形結合が一般解です. 上の結果では後半の方 (係数が $\_C2$) の部分が Gauss の超幾何級数と呼ばれる有名な級数です.

10.4 解とベクトル場を同時にグラフにする

さて, 上で求めた解と元の方程式が定めるベクトル場を同時に表示する方法を述べます. これは, plot の結果を最初は表示せず, 最後にまとめて表示するとできます. 表示しないというのは, これまでにも用いて来ましたが, 最後をセミコロンではなく, コロン : を用いるとできます. 最後をコロンで終りますと, Maple では, その式の評価はしますが, 結果の表示をしません. グラフの表示には, display という手続きで一気に表示します.

次は, 解の一意性が成り立たないのを目で見ようとしたものです. うまく行くでしょうか ?

```
> F:=plot(((2/3)*x)^(3/2),x=0..2):
> G:=fieldplot([1, y^(1/3)], x=0..2,y=0..1.5):
> H:=plot(((2/3)*(x-0.5))^(3/2), x=0.5..2):
> display({F,G,H});
```

コロンで終らないとどうなるかを実験して見ましょう.

```
> K:=plot((2/3)*(x^(3/2)),x=0..2);
```

11 Legendre(ルジャンドル) の多項式

Report No. 2(LaTeX で証明を書くというやつ) の問題で, 年度によっては, 直交多項式系を問題にします. また, 年度によっては, Hermite(エルミート) の多項式を問題にしてあります. ここでは, 今年の問題になっている Legendre の多項式を取り上げます. Legendre の多項式は, 非負整数 n に対して, $P_n(x) = \dfrac{1}{2^n n!}\dfrac{d^n}{dx^n}(x^2-1)^n$ で定義される n 次の多項式で, 次の性質を持ちます.

$$\int_{-1}^{1} P_m(x)P_n(x)dx = \frac{1}{2n+1}\delta_{m,n} \quad \text{(直交関係)}$$

$$(n+1)P_{n+1}(x) - (2n+1)xP_n(x) + nP_{n-1}(x) = 0 \quad \text{(漸化式)}$$

$$(x^2-1)P_n''(x) + 2xP_n'(x) - n(n+1)P_n(x) = 0, \quad \text{(微分方程式)}$$

これらのうち最初の式は, 積分で定義された内積に関する直交関係で, ある種の内積に対して直交関係を持つ関数族を「直交関数系」といいます[*4]. 最も良く使われる直交関数系は $\left\{\dfrac{1}{2}, \cos nx, \sin nx\right\}_{n=1,2,\cdots}$ で, Fourier 級数に用いられます. Legendre の多項式は, 直交する多項式の族のひとつです.

さて, 3 番目の関係式は, Legendre の多項式が次の 2 階の線形微分方程式

$$(x^2-1)y''(x) + 2xy'(x) - n(n+1)y(x) = 0$$

の解である事を示しています. これを Maple で解くとどうなるでしょうか.

```
> de4:=(x^2-1)*diff(y(x),x$2)+2*x*diff(y(x),x)-n*(n+1)*y(x)=0;
> dsolve(de4);
```

出てくる答は, LegendreP と LegendreQ という 2 つの関数の和というものです. このうち, LegendreP の方が Legendre 多項式です. LegendreQ については, Help で調べてください.

2 番目の式は, $P_n(x)$ の漸化式です. 従って, 今回の最初の方でやった再帰で計算させる事もできます. legendre という手続き名でそれを書いてみます.

```
> legendre :=proc(n)
>  if (n=0) then 1;
>  elif (n=1) then x;
>  else ((2*n-1)*x*legendre(n-1)-(n-1)*legendre(n-2))/n;
>  fi;
> end;
> legendre(4);
> simplify(%);
```

上で `n = 0` の部分は n が 0 と等しいか否かを判断している比較文です. Awk では `n = 0` は代入文で, `n == 0` が比較文ですが, Maple では, `n := 0` が代入文で, `n = 0` は比較文です. 処理系による記号の意味

[*4] 実際には, さらにその内積に対しての「完全性」, すなわち, ノルム (長さ) が有限な元は, これらの関数の線形結合で, その内積が定める距離に対して幾らでも近似できることを要求します.

の違いには, 柔軟に対処できるようになってください. Maple では文字式が数学と同じように扱えますので, 上のように漸化式で定義された関数を計算して出力する事のもできます.

Maple では微分が計算できますから, Legendre の多項式を定義に従って計算させる事も出来ます.

```
> legen:= n-> diff((x^2-1)^n,x$n)/(n!*2^n);
> legen(4);
> simplify(%);
```

このように, 1 つの事を実現するのに何通りかのやり方があるというのは良い事です. それらの中でどの方法をとるかは, 問題によって適切な方法を選ぶという態度が必要です.

直交多項式系には, これ以外にも Laguerre(ラゲール) の多項式等があり, それぞれ数学的にも応用上も重要です. それらはやはり, ある種の微分方程式の解で, 漸化式を満たし, (当然ですが) なんらかの内積に対して直交します. Maple で定義されている直交多項式系は

```
> help(orthopoly);
```

で知る事が出来ます (直交=orthogonal, 多項式=polynomial).

12 Fourier 級数

12.1 $f(x) = x$ の Fourier 級数

Fourier 級数や Fourier 変換は解析学 IV(III) で講義される予定のものです. 今の電話での通話では, Fourier 級数の理論を応用したものが実装され, それで通信しています.

前回, 正弦曲線を多項式近似しましたが, 逆に直線 $y = x$ を sin, cos の級数で表わすことを考えます. どちらも周期関数なので, その和も周期関数になりますが, とりあえず, $[-\pi, \pi]$ の範囲で近似してみます.

関数 $f(x)$ が次のように三角関数の級数に展開されたと仮定します.

$$f(x) = \frac{a_0}{2} + a_1 \cos x + b_1 \sin x + \cdots + a_n \cos nx + b_n \sin nx + \cdots .$$

この式の両辺を $[-\pi, \pi]$ で積分すると下のことが分かります.

$$\int_{-\pi}^{\pi} f(x)dx = \pi a_0$$

次に (四角の中は, 各自計算すること),

$$\int_{-\pi}^{\pi} \cos^2 nx\, dx = \int_{-\pi}^{\pi} \sin^2 nx\, dx = \boxed{}$$

$$\int_{-\pi}^{\pi} \cos mx \cos nx\, dx = \int_{-\pi}^{\pi} \sin mx \sin nx\, dx = \boxed{} \quad (m \neq n)$$

$$\int_{-\pi}^{\pi} \cos mx \sin nx\, dx = \boxed{}$$

を利用すると

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos nx\, dx \quad (n = 0, 1, 2, \ldots), \qquad b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx\, dx \quad (n = 1, 2, \ldots)$$

であることが分かります. したがって, 上記で $f(x) = x$ とおいて計算すると

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} x\cos nx\,dx = 0, \quad b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} x\sin nx\,dx = (-1)^{n-1}\frac{2}{n}$$

となります ($f(x) = x$ は奇関数だから cos の項が消える.). したがって, 次の式が得られます.

$$x = 2\left(\sin x - \frac{\sin 2x}{2} + \frac{\sin 3x}{3} - \cdots\right). \tag{1}$$

右辺は, $(-\pi, \pi)$ では収束して, 左辺と等しいことが証明できます. 上の式の両辺に $\pi/2$ を代入すると,

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots. \tag{2}$$

が得られます. これは正しく, 無理数 π を有理数の無限和で表す最も簡単な式です. ただし, 右辺の収束は条件収束でとても遅いので, これで π の近似値を求めようとしても, あまり良い結果は得られません.

問 11 (2) の右辺は絶対収束しないことを示せ.

12.2 Fourier 級数の部分和とグラフ

Taylor 展開のときと同様に Fourier 級数の部分和の項を増やすに従い, もとの関数に近づいていくことを確かめましょう. まず $f(x) = x$ の Fourier 展開を

$$x = \sum_{i=1}^{\infty} b_i \sin ix = \sum_{i=1}^{\infty} (-1)^{(i-1)}\frac{2}{i}\sin ix = \sum_{i=1}^{\infty} f(i)$$

とおいて, 級数 $\sum f(i)$ の i 番目の項を i の関数 $f(i)$ として定義します.

```
> f := i -> (-1)^(i-1)*2*sin(i*x)/i;
```

次に n 番目までの部分和も n の関数 `sum_f(n)` として定義します.

```
> sum_f := n -> sum(f(i), i=1..n);
```

部分和を大きくしていくに従ってグラフが直線に近づいて行くのを確かめてみて下さい.

```
> plot(sum_f(1), x=-Pi..Pi);
> plot(sum_f(2), x=-Pi..Pi);
> plot(sum_f(3), x=-Pi..Pi);
> plot(sum_f(5), x=-Pi..Pi);
> plot(sum_f(10), x=-Pi..Pi);
> plot(sum_f(20), x=-Pi..Pi);
```

では, 区間 $[-\pi, \pi]$ の外側では関数の様子はどのようになっているでしょうか?

```
> plot(sum_f(20), x=-10..10);
```

12.3 Weierstrass の定理

Weierstrass は次の定理を証明しました. この級数は Fourier 級数と少し違いますが, 本質的には同じです.

定理 3 $f(x)=\sum_{n=0}^{\infty}a^n\cos(b^n\pi x)$ $(0<a<1,\ b$ は正の奇数, $ab>1+\frac{3}{2}\pi)$ とする. この時, $f(x)$ は, 実数全体で連続な函数になるが, すべての点で微分することができない.

問 12 Weierstrass の定理の右辺の Fourier 級数は $(-\infty,\infty)$ で広義一様に絶対収束し, 実数全体の上での連続関数を定義することを示せ.

Weierstrass の函数の定義に現れる級数の有限項の和はもちろん C^∞ 級 (もっと強く実解析的) ですが, 無限和の極限が, 各点で微分できなくなる感じを見ることはできます. 次を maple で実行してみませう.

```
> f:=i->(0.5)^i*cos(13^i*Pi*x);
> f(2);
> g:=n->sum(f(i),i=0..n);
> plot(g(30),x=-10..10);
```

13 熱方程式 (Fourier 級数の源)

13.1 Fourier のアイデア (数学のお話)

均質な物体内の温度分布は, (近似的に) 次の方程式に従う事が知られています. [*5]

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}-\frac{\partial}{\partial t}\right)f(x,y,z,t)=0$$

ここで, x,y,z は空間座標で, t は時刻です. 上の方程式を熱方程式といいます. たとえば, 3 次元の Gauss 分布 (の定数倍) $g(x,t)=\dfrac{1}{\left(\sqrt{t}\right)^3}\exp\left(-\dfrac{(x-a)^2+(y-b)^2+(z-c)^2}{4t}\right)$ は, 任意の $(a,b,c)\in\mathbb{R}^3$ に対して, 熱方程式の解です.

問 13 1. このことを確かめよ. 2. $\iiint_{\mathbb{R}^3}g(x,t)dxdydz$ を計算せよ.

Fourier 級数やその変種は, 現在では信号解析等の分野に広く応用されていますが, もとは, Fourier は熱方程式を解く方法として, Fourier 級数を見出しました. ここでは, Fourier のアイデアを最も簡単な場合に述べ, 温度の変化の様子を視覚化します. 簡単のため, 空間は 1 次元にして, 次の偏微分方程式を考えます.

$$\left(\frac{\partial^2}{\partial x^2}-\frac{\partial}{\partial t}\right)f(x,t)=0$$

微分方程式を考える領域は, 簡単のために $(x,t)\in[0,\pi]\times[0,\infty)$ とします.

上の偏微分方程式で, $f(x,t)=h(x)g(t)$ の形に変数分離できると仮定します. このとき上の微分方程式は, $h''(x)g(t)-h(x)g'(t)=0$ となります. 従って, $\dfrac{h''(x)}{h(x)}=\dfrac{g'(t)}{g(t)}$ となり, 左辺は x の函数, 右辺は t の函数と

[*5] 近似的というのは, 本当に上の方程式を満たせば, 熱は無限大の速さで伝播する事になり, 相対論に矛盾します.

なるので, この共通の値は定数 λ となります. 従って, 次の 2 つの常微分方程式を考える事になります.

$$\text{(A)} \left\{ \begin{array}{l} h''(x) = \lambda h(x) \\ g'(t) = \lambda g(t) \end{array} \right.$$

さらに, 元の熱方程式に次の初期条件と境界条件を課します.

$$\text{(B)} \left\{ \begin{array}{ll} f(x,0) = h_0(x) & \text{初期条件} \\ f(0,t) = f(\pi,t) = 0 & \text{境界条件} \end{array} \right.$$

つまり, 時刻 0 で $h_0(x)$ の温度分布をしていた物体 (線分) の両端を氷水 (0 度の水) につけた時の内部の温度変化を見る事になります.

さて, (B) の境界条件より, 常微分方程式 (A) で $\lambda < 0$ が示されます. 実際, 次の式が成立し,

$$\lambda \int_0^\pi h(x)^2 dx = \int_0^\pi h(x)h''(x)dx = [h(x)h'(x)]_0^\pi - \int_0^\pi (h'(x))^2 dx = -\int_0^\pi (h'(x))^2 dx \leqq 0$$

等号は $h(x) \equiv 0$ という自明な解になるからです. 従って, $h(x) = a\sin\sqrt{-\lambda}(x - x_0)$ です. もう一度 (B) の境界条件を用いると $h(x) = a\sin nx, (n = 1, 2, \ldots)$ となります. この時, $g(t) = be^{-n^2 t}$ です. 従って, 熱方程式の解として $Ae^{-n^2 t}\sin nx, (n = 1, 2, \ldots)$ が得られます. 熱方程式は線形 (すなわち, 解の和や定数倍も解になる) ですので, 次の線形結合も (右辺が収束して項別微分が可能であれば) 熱方程式の解となります.

$$f(x,t) = \sum_{n=1}^{\infty} c_n e^{-n^2 t}\sin nx$$

さて, ここで $t = 0$ の初期条件を考えますと

$$f(x,0) = h_0(x) = \sum_{n=1}^{\infty} c_n \sin nx$$

となり, これは $h_0(x)$ の Fourier 級数に他なりません. すなわち, 上の線形結合の係数 c_n は, $h_0(x)$ の Fourier 係数です. したがって, 函数を Fourier 級数に表す事さえ保証されれば, 熱方程式の上の形の初期値, 境界値問題の解が得られるわけです.

ただし, 函数を Fourier 級数に表す事を保証したり, 上以外に解が無い事 (解の一意性) を示すのは, 自明な事ではありません.

13.2 熱分布の視覚化

上で述べたように, 初期値の Fourier 級数が求まれば, (B) の境界条件を満たす熱方程式の解が求まります. ここでは, 初期値として $h_0(x) = x(\pi - x)$ を与えた時に, この境界条件のもとで, どのように温度が変化して行くかを視覚的に表示する方法を述べます.

今回は, animate という手続きを使ってみます. そのためにまず plots パッケージを読み込みます.

```
> with(plots);
```

$a(i)$ を $h_0(x)$ の第 i 番目の Fourier 係数とします.

```
> a:=i->int(2*x*(Pi-x)*sin(i*x)/Pi,x=0..Pi);
```

$f(x,t)$ を第 20 項までの解の部分和とし, $0 \leqq t \leqq 6$ で毎秒 10 フレイムで, アニメイションを作ります.

```
> f:=(x,t)->sum(a(i)*exp(-i^2*t)*sin(i*x), i=1..20);
> animate(f(x,t),x=0..Pi, t=0..6,frames=60);
```

メニューバーの Plot → Animate → Play と言う項目が選びますと, アニメイションがはじまります. 全体の温度が 0 度に近付く様子が見えます.

$a(i)$ を求めるときの被積分函数を変化させると, さまざまな初期値に関する温度変化の様子を見る事ができます. いろいろと試してみて下さい.

注意: 空間 1 次元では, Fourier の方法で上のように解けるのですが, 我々が住む 3 次元では, 複雑になります. 例えば, 現実にある缶ビールのような円筒形のものを解く際には, 3 角関数ではなく Bessel 関数が必要になります.

14 RSA 公開鍵暗号

今回は RSA 公開鍵暗号を Maple で組んで見ます. 今回の例は Maple V Release 3 についてきたサンプルプログラムを参考にしました.

14.1 素因数分解

自然数 n に対して次の定理が成立する事はよく知られています.

定理 4 $n \in \mathbb{N}$ は素数のベキの積として一意的に書ける. 即ち, 素数 $p_1, \ldots, p_k$ と自然数 $e_1, \ldots, e_k$ が一意的に存在して, $n = p_1^{e_1} \cdots p_k^{e_k}$ となる.

問 14 上の定理を証明せよ (きちんと一意性を示すのは, 易しくはない).

上の定理は, (定理としては書かれていないかも知れませんが) 小学校以来おなじみのものです. しかし, 現実の計算は (現時点では) 易しくないのです. 次の計算を maple にさせてみて下さい. nextprime はそれより大きい最小の素数を出力する手続きで, ifactor は整数の素因数分解をする手続きです.

```
> p:=nextprime(12345678901234567890123456789012345);
> q:=nextprime(p);
> p*q;
> ifactor(%);
```

p*q の計算は一瞬で終わりますが, それを素因数分解して元に戻すには, 少し計算時間がかかります. このことを利用して公開鍵暗号系が開発されており, それが今回のタイトルである, RSA 公開鍵暗号です.

14.2 暗号化通信

暗号化通信とは, 元の情報 (平文) を変形 (暗号化) して, 通信経路にいる盗聴者に対して情報を漏らさないようにする事です. 暗号を受信した人だけが, 暗号を元の情報に戻せる (復号化できる) ようにします.

推理小説やスパイ映画であるように, 優秀な人間に暗号の解読方法がばれるようでは, 暗号通信をする意味がありません. そこで, 現代の暗号では, 解読方法がわからない事を保証するため, 暗号化方法を公開して開発します. 従って, 現実には全ての暗号は解読可能ですが, たとえば, 「解読に必要な時間は 100 年である」のよ

うな事が証明できると, その暗号は安全であると判断します. 100 年後はその情報には価値がないからです.

現代の暗号では, 暗号化の方法は, ある特定のアルゴリズムと, それを動かすための鍵にわけて考えます. 安全性は, 鍵を知らない人が鍵を知るために必要な仕事量, あるいは, 鍵無しで復号化が可能になるための仕事量で定式化します. このような暗号系で問題になるのが, 鍵の受け渡し方法です. 鍵を受け渡すのに別の (盗聴者がいるであろう) 通信をする事はできません. そこで考えだされたのが, 公開鍵暗号です. 公開鍵暗号では, 鍵を公開鍵と秘密鍵に分けます. 公開鍵は全員に公開されますが, 秘密鍵は, 復合化する人だけが持っています. 安全性は, 公開鍵から秘密鍵を知るために必要な仕事量と, 秘密鍵を知らずに復号できるようになるまでに必要な仕事量の小さい方の大きさで計ります.

公開鍵暗号は現在至る所に使われており, その代表例が Web ブラウザです. 例えば, 皆さんは授業登録を Web で暗号化通信を利用して行っています. その登録において, 個人がどの授業を登録するかは, 盗聴している人 (例えば, 情報処理センターの人は, 皆さんのコンピュータのネットワーク通信を, あまり苦労する事無く盗聴できます) がいたとしても, その人にはわからないようになっています.

なお, 実際の暗号化通信では, 全てを公開鍵暗号で行なってはいません. 公開鍵暗号は,「計算の複雑さ」を利用するため,「暗号化–復号化」の速度が遅くなります. 暗号化通信のほとんどの部分は, 秘密鍵暗号を利用しています. 公開鍵暗号は, 秘密鍵暗号の通信をする前に, お互いの秘密鍵を交換するときに利用しています.

14.3 RSA 公開鍵暗号

RSA 公開鍵暗号は Ron Rivest, Adi Shamir, Len Adleman の 3 人によって開発されました. RSA は彼らの姓の頭文字を並べた物です. 発表されたのは 1977 年で, 公開鍵暗号としてはおそらく最初の物です. 発表に際してアメリカ国防省の横槍が入った事で話題になりました. しかし, その原理があまりにも単純だったため結局公表されました. その際, 1 つの暗号の解読問題が Rivest によって 1978 年に提出されました. かなりのヒントがあったにもかかわらず, この暗号が解読されたのは 16 年後の 1994 年で, ネットワークを用いて世界の数千台の計算機による分散処理の結果でした. 今でもこのような暗号解読競争がいくつか行われているようです. なお, RSA の発表以後さまざまな公開鍵暗号系が提案されています. 楕円曲線 $y^2 = x^3 + ax + b$ を有限体上で考えると有限可換群になりますが, この群構造を利用した楕円 ElGamal 暗号は RSA より鍵長が短くなるという特性があります. RSA 暗号では, 次のいくつかの初等整数論の定理を使います.

定理 5 p, q $(p \neq q)$ を 2 より大きい異なる素数, $n = pq$, $\lambda = \mathrm{lcm}(p-1, q-1) =$「$p-1, q-1$ の最小公倍数」とする. この時 $(a, n) = 1$ となる全ての a に対して, 次が成立する.

$$a^{\lambda} \equiv 1 \pmod{n}$$

問 15 上の定理を証明せよ.

系 1 定理と同じ条件のもとで, e, d を $ed \equiv 1 (\mathrm{mod}\ \lambda)$ となるふたつの整数とすると, $(a, n) = 1$ となる全ての a に対して,

$$(a^e)^d \equiv a^{ed} \equiv a \pmod{n}$$

上の系から, $a^{ed} \equiv a \pmod{n}$ が成立する事に注意すると, n を法として, a の e 乗を a の暗号とすると, それを d 乗して n を法として考えれば, 再び a が復元される事が判ります.

では, e と λ から d を求める方法はあるのでしょうか ?

定理 6　α, β を 2 つの整数として, $\delta = \gcd(\alpha, \beta) =$ 「α, β の最大公約数」とする. このとき

$$\alpha x + \beta y = \delta$$

となる整数 x, y が存在する. さらに x, y はユークリッドの互除法を使う事によって, 求める事ができる.

問 16　1. 上の定理の巡回群の部分群の性質を用いた群論的な証明を与えよ.
2. 上の定理をユークリッドの互除法を用いて証明すると同時に x, y の求め方を与えよ.

さて, 上の定理から整数 e が $(e, \lambda) = 1$ を満たすと, $ex + \lambda y = 1$ を満たす整数 x, y が計算されます. このとき $0 < x < \lambda$ となるように, x を取る事ができますから, それを d とおくと, $ed \equiv 1 \pmod{\lambda}$ となります. つまり, e から d を計算する良い方法があるのです. なお, ユークリッドの互除法はとても効率の良いアルゴリズムとして知られています.

上で, n, e が公開鍵, d(あるいは, p, q, λ) が秘密鍵です. A さんが B さんに暗号で情報を送る手順は次です.

1. B さんは, 大きい 2 つの素数 p, q を探し、$n = pq$, $\lambda = p - 1, q - 1$ の最小公倍数, λ と互いに素な数 e, $ed \equiv 1 \pmod{\lambda}$ となる d を求め, n, e を A さんに知らせる. (n, e が公開鍵で d が秘密鍵.)
2. A さんは, B さんに送る情報を数値化し, 得られた数を a とするとき, $a^e \equiv b \pmod{n}$ $0 < b < n$ となる b を計算し, b を A さんに伝える. (b が a の暗号化, 暗号文が長い時には文を分割するなどの操作が必要であるが, それは本質的な事で無いのでここでは触れない.)
3. A さんは, B さんからの情報 b を d 乗して n で割った余りを計算すると, 元の数値 a を得るから, 数値化の逆操作で元の情報を復元する.(この手順が復号化)

RSA 公開鍵暗号系が有効である根拠は, 次の経験則によっています.

1. 公開鍵, 秘密鍵は, 素因数分解 $n = pq$ を知っている人には簡単に計算する方法がある.(ユークリッドの互除法)
2. 大きな素数を作り出すそれなりに有効な方法がある.
3. 暗号化, 復号化の計算手順 (冪乗) は有効な方法が知られている.
4. それに対して, 公開鍵 n, e から秘密鍵 d を求める方法は, 素因数分解 $n = pq$ を知る必要があり, これを知る有効な方法は知られていない. つまり, 大きな 2 つの素数の積を素因数分解する効率の良い計算方法は, 今のところ知られていないし, おそらく無いであろうと予想されている. すなわち d を求める良い方法がない. また, 素因数分解を知らずに復号する方法は無いと考えられている.

上で述べた「有効な方法」の意味は, 計算量という概念で定量化されます (単なる定性的な議論ではない) が, ここではそれには詳しく触れません. 計算量が桁数の函数として指数函数的なのか, 多項式的なのかという定式化です. また, この根拠は現在のところ経験則で, 多くの数学者は素因数分解は難しい問題であろうと信じていますが, 厳密に証明されたわけではありません.

15　実習: RSA 公開暗号系を Maple でプログラムする.

さて, 上の手順を Maple でプログラムします. Maple では大きな整数は何も考えずに扱えますし, ユークリッドの互除法なども入っていますので, これを利用すれば簡単に RSA 暗号系のプログラムが組めます. 今回, プログラムとして組むのは, 上記の手順でいうと 1 番目と 3 番目の情報の数値化とその復元部分です. こ

の部分は, アルゴリズムと共に, プログラムの組み方を勉強して下さい.

15.1 情報の数値化

簡単のため情報はアルファベットと空白の 27 文字だけを使うとし, 数値化規則は次の表で与えるとします.

文字	a	b	c	d	e	f	g	h	i	j	k	l	m
数字	1	2	3	4	5	6	7	8	9	10	11	12	13
n	o	p	q	r	s	t	u	v	w	x	y	z	空白
14	15	16	17	18	19	20	21	22	23	24	25	26	27

ここで, 1 桁の数の部分はその文字が先頭ならそのまま適用し, 途中にある時は前に 0 をつけて 2 桁に変換して用いるとします. 例えば i love you は次の数字で表示されます.

i		l	o	v	e		y	o	u
9	27	12	15	22	05	27	25	15	21

すなわち, i love you は数字 9271215220527251521 で表示されます. この操作を, Maple の procedure で書きます. Procedure の名前は tonumber とします. プログラム中 ERROR 文の中のクォーテイションは, バッククォート ‘ (標準的な日本語キーボードでは, Shift+@で入力) であることに注意してください.

```
> tonumber:=proc(st)
>    local len,n,s,i,num;
>    num:=table(["a"=1,"b"=2,"c"=3,"d"=4,"e"=5,"f"=6,"g"=7,"h"=8,
>                "i"=9,"j"=10,"k"=11,"l"=12,"m"=13,"n"=14,"o"=15,
>                "p"=16,"q"=17,"r"=18,"s"=19,"t"=20,"u"=21,"v"=22,
>                "w"=23,"x"=24,"y"=25,"z"=26," "=27]):
>    if not type(st,string) then
>       ERROR(‘wrong number (or type) of arguments‘);
>    fi;
>    len:=length(st);
>    if len=0 then
>       RETURN(0);
>    fi;
>    n:=0;
>    for i from 1 to len do
>       s:=num[substring(st,i..i)];
>       if not type(s, numeric) then
>          ERROR(‘wrong number (or type) of arguments‘);
>       fi;
>       n:=100*n+s;
>    od;
>    RETURN(n);
> end;
```

上のプログラムを入力した後, 次を入力してプログラムがきちんと動く事を確かめて下さい.

```
> a:=tonumber("i love you");
```

15.2 Maple での文字と文字定数

上のプログラムの解説を少しだけします. Maple での文字 (文字列) の扱いについてです.

Maple のワークシートでは, A という文字はオブジェクト (object) ですこれは前後関係によって, 数であった式であったり行列であったりです (変数といったほうがよいかも知れません). A に「A という文字」の意味を持たせるには, ダブルクォート " で囲みます. この講義で出て来たメタキャラクタのエスケイピングと同じです. このような事は, ほとんどのプログラミング言語で現れます. 上のプログラムでも,「A という文字」を取り扱う必要がありますから, 「ダブルクォート " で囲む」が実行されています. 配列の扱いは, 昔のテキスト (http://ftp.math.u-ryukyu.ac.jp/pub/gengo/2002/7.tex) を参考にして下さい.

15.3 数字から文字列の復元

今度は上の逆変換, すなわち数字から文字列を得るプログラムを procedure で書きます. Procedure の名前は fromnumber とします.

```
> fromnumber:=proc(n)
   local s,m,len,i,ans,alpha;
   alpha:=table(["a","b","c","d","e","f","g","h","i","j",
                 "k","l","m","n","o","p","q","r","s","t",
                 "u","v","w","x","y","z"," "]);
   m:=n;
   if not type(n,integer) then
      ERROR(`wrong number (or type) of arguments`);
   fi;
   len:=floor(trunc(evalf(log10(m)))/2)+1;
   ans:="";
   for i from 1 to len do
     m:=m/100;
     s:=alpha[frac(m)*100];
     if not type(s,string) then
        ERROR(`wrong number (or type) of arguments`);
     fi;
     ans:=cat(s, ans);
     m:=trunc(m)
   od;
   RETURN(ans);
  end;
```

プログラムを入力し終えたら, 次を実行して元の文が復元される事を確かめて下さい.

```
> fromnumber(a);
```

15.4 公開鍵, 秘密鍵を作って実験する

公開鍵を作るには, 2 つの素数 p, q と $\mathrm{lcm}(p-1, q-1)$ と素な 1 つの正整数が必要です. 実際に本格的な使用に使う様なプログラムを書く際には, この部分は暗号の安全性に本質的にかかわる部分ですので細心の注意が必要ですが, ここでは簡単のために Maple の素数生成法を利用します.

```
> p:=nextprime(12345678901234567890123456789012345678901234567890);
> q:=nextprime(98765432109876543210987654321098765432109876543210);
```

この時, 次の数が公開鍵の 1 つです.

```
> n:=p*q;
```

さて, $\lambda = \mathrm{lcm}(p-1, q-1)$ は次で計算されます.

```
> lambda:=lcm(p-1, q-1);
```

もう一方の公開鍵は, λ と互いに素な整数であれば, 何でも良いのですが, ここでは再び素数を適当に取る事にします (本来は, λ と互いに素であることを確かめる必要があります).

```
> e:=nextprime(1234543210);
```

e が決れば, 秘密キーの計算をします. 上で述べたようにこれはユークリッドの互除法で計算できます. maple にはこの関数もあります. 下の igcdex は不定方程式 $ed + \lambda k = 1$ の整数解を変数 d, k に代入します.

```
> igcdex(e,lambda,'d','k');
> d:=d mod lambda;
```

2 番目は, $0 < d < \lambda$ とするためであり, 計算量を少なくする以外に意味はありません.

さて, 暗号化を実験して見ましょう. maple には冪乗を計算する関数も剰余を求める関数も備わっており, 大きな整数計算をそのままやってくれますので, 暗号化は実に簡単です (ほとんどのプログラミング言語では, こんな簡単には行かない).

```
> M:=tonumber("mathematics");
> C:=Power(M,e) mod n;
```

上の数字が, mathematics という文字列の暗号化です. さて復号化してみましょう. うまく行くはずです.

```
> Power(C,d) mod n;
> fromnumber(%);
```

最初にでて来た文字列 i love you を暗号化して見ます

```
> b:=Power(a,e) mod n;
```

良く似た文字列 i like you を暗号化して比較して見ます

```
> x:=tonumber("i like you");
> y:=Power(x,e) mod n;
```

本文の少しの違いが暗号化で大きく違って来る事が, 判ります. これも RSA 暗号系の良い所です. 即ち, RSA 暗号は部分解読ができないと言う事です (正確な証明があるのかどうかは, 私は知らない).

補足

- 自然数が一意的に素因数分解できることの証明で, 難しい部分は一意性です. どこが難しいかというと,
  a, b を自然数, p を素数とし, p が ab の約数であるとする.
  このとき, p は a の約数であるかまたは b の約数である.
  を示さなければならないところです.
- 公開鍵暗号の可能性は, アメリカ国防省の研究機関も, 一般公開以前に気づいていた節があるようです. しかし, 研究機関の性質上それを公開することはありませんでした.

参考文献

[1] B. W. Char 他, サイバネットシステム訳, はじめての Maple, 1998, シュプリンガーフェアラーク東京

[2] B. W. Char 他, サイバネットシステム訳, よくわかる Maple, 1998, シュプリンガーフェアラーク東京

[3] K. M. Heal, M. Hansen, K. Rickard 著, 示野信一他訳, Maple V Release 5 ラーニングガイド, 1999, シュプリンガーフェアラーク東京

[4] 示野信一著, Maple V で見る数学ワールド, 1999, シュプリンガーフェアラーク東京

[5] 上田哲生, 谷口雅彦, 諸澤俊介著 複素力学系序説 倍風館

[6] 西沢清子, 関口晃司, 吉野邦生著 フラクタルと数の世界 海文堂

[7] 岡安隆照他, 微分積分学入門, 裳華房

[8] J. Milnor, Morse Theory, Annals of Math. Studies, Study 51 Princeton Univ. Press, 1969(日本語訳, 吉岡書店)

[9] 岡本龍明・太田和夫編, 暗号・ゼロ知識証明・数論, 共立出版

[10] N Koblitz, A Course in Number Theory and Cryptography, Springer Verlag 日本語訳もあり.