連続系アルゴリズム レポート課題1

- 宿題(レポート)
  - 提出状況に応じて，期末試験の点に最大 50 点を加算します
    - まったく提出しなくても，期末試験の点で成績が出ます
    - ただし，試験を受けなければ単位は出ません
  - プログラムだけはメールの添付ファイルで
    reiji@is.s.u-tokyo.ac.jp
    に送付してください
    - 所属学科，学年，学籍番号，氏名をメール本文に明記すること
    - メールの題名(Subject)は「連続系レポート課題第1回 提出者氏名」とすること
  - 手計算をするものや，計算結果と考察など，プログラム以外は A4 の紙 1 枚(裏もつかってよい)にまとめてください
    - 所属学科，学年，学籍番号，氏名をレポートの最初に明記すること
    - 次回の講義の前に集めます
    - あるいは PDF で A4 サイズ 2 ページでもよい(プログラムと一緒にメールで送付してください)

> 使える PC がない場合
> 情報基盤センターの「講習会」(ほぼ毎週やっている)に参加すると，センターの Mac が使えます

- 問題1
  - 実数 x に対して，$y = \cos(x)$ を浮動小数点数で計算したときに，y に含まれる誤差を見積もれ
    - 実数 r を浮動小数点数で表現するときの誤差を $\varepsilon|r|$ 程度と見積もれ
    - x を浮動小数点数で表現するときの誤差と，$\cos(x)$ を浮動小数点数で表現するときの誤差を考えよ
  - $|x|$ が大きいとき，$\cos(x)$ の精度が下がってしまう。なぜか？
    - できれば、精度よく計算する工夫を考えよ
  - x が 0 に近いとき，$1 - \cos(x)$ の相対精度が下がってしまう。なぜか？
    - できれば、精度よく計算する工夫を考えよ
- 問題2
  - 二次方程式 $ax^2 + bx + c = 0$ の解を求めるプログラムを書け
    - 最低限，double を使って計算するプログラムを書け
    - できれば float と double の両方で計算をおこない，解の違い(差)を確認せよ
    - さらにできれば，いわゆる「解の公式」と，講義で説明した「根と係数の関係」による方法を用いて計算し，これらの比較をせよ

センターのマックでのプログラムの仕方(Unix 風)

- プログラムを書く
  - 「mi」を起動する
    - 下に並んでるアイコンの左から8つめ
  - プログラムを書く
    - とりあえず簡単なものを
  - ファイルに名前をつけて保存する
    - ウィンドウの左上にある、フロッピーディスクの印をクリックする
    - ファイル名はアルファベットにして，<u>必ず</u>「.c」をつけること
    - 例えば hello.c とか
  - Emacs を知っておくと便利かも
    - UNIX / Linux などでよく使われている
      - C 言語用の便利な機能がいろいろ付いている
    - 使い方は、チュートリアルで短時間で独習できる
      - メニューの「Help」から「Emacs Tutorial」を選択する
- 詳しくは http://www.ecc.u-tokyo.ac.jp/ を参照
- MAC では Xcode というのが標準らしい

- プログラムを実行する
  - 「ターミナル」を起動する
    - 下に並んでるアイコンの左から6つめ
    - これが UNIX の端末になっている
    - UNIX (Linux) の使い方についても，適当な書籍などを見てください
  - cm30203% とかいうのが「プロンプト」
    - このあとにコマンドを入力する
  - プロンプトに続き
    cc ファイル名 -lm
    と入力
    - 例えば cc hello.c -lm とか
      - うまくいけば，黙って終わる
      - 何か表示されたら、よく読んでプログラムを修正
  - プロンプトに続き
    ./a.out
    と入力
    - 実行結果が表示される

7.

本日のC言語

- いつでも共通の「枠組み」

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(void) {

  /* ここに計算内容を書く */

  return 0;
}
```

- #include の部分は，改行も含めこのとおりに
  - それ以外は，「改行」と「空白」は同じ
- インデント（行頭の空白）で「構造」を見やすく
  - 須田は { } 内では 2 文字のインデント
- コメントは /* と */ の間に書く
  - 改行が入ってもよいが，入れ子にはできない
- main は「関数」と呼ばれている
  - { } 内を「本体」と呼ぶことにする

- 変数と出力

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(void) {
  double a, b;

  a = 1.0;  b = 7.0;
  printf("%f¥n", a / b);

  return 0;
}
```

- 変数とは，「値を格納しておく入れ物」
  - 関数本体の冒頭で「型」と「名前」を「宣言」する
  - 「型」: double は倍精度の浮動小数点数
  - 「名前」: ここでは a と b の 2 つを一度に宣言
- a = 1.0 と b = 7.0 は「代入」という
  - 変数に値を格納する操作（「等式」ではない）
    - 何度も代入すると，最後の値のみ覚えている
  - それぞれの処理の最後にセミコロンを書く

本日のC言語

- 出力には printf を使う
  - カッコ内最初の " と " の間は「文字列」と呼ばれる
  - この部分が画面に印字される
- 一番簡単な例

```
printf("Hello, world!¥n");
```

  - Hello, world! と印字され，改行される
  - 改行は自動ではない．¥n が改行文字を表す
    - 改行しないと見にくいし，システムによってはそもそも表示されないこともあるから，忘れず入れる
    - ¥n で「1文字」扱い
- さらに別の例

```
printf("a / b = %f¥n", a / b);
```

  - 文字列のうち，%f と書いてあるところに，a/b の値が組み込まれて出力される
  - %f と ¥n 以外はそのまま印字される．つまり
    a / b = 0.142857
    のように表示される
  - %20.16f とかすると表示桁数が変えられる
- 4 は整数、4.0 は倍精度、4.0f が単精度

- 条件分岐

```
if (a < b) {
  min = a;
} else {
  min = b;
}
```

  - min は変数とする
    - 変数名は 1 文字でなくてもよい
  - この例では，a < b なら min に a の値を代入，それ以外の場合には min に b の値を代入する
    - 代入するものは式でもよい．たとえば
      sum = a + b;
- 平方根関数 sqrt

```
x = - b + sqrt(d);
```

  - これは倍精度で，単精度は sqrtf
- 型変換

```
x = (float) a + (float) b;
```

  - a と b を単精度にしてから計算
    - 単精度と倍精度の演算は倍精度にそろえて計算される
    - 異なる精度の値の代入は，代入時に型変換される

連続系アルゴリズム 10/3

浮動小数点を扱う。

近似が避けられない

実数の近似, 関数の近似, 微積分の近似

丸め誤差　　打ち切り誤差

アルゴリズム

微分積分. 方程式

正確な値 x, 近似値 $\hat{x}$

絶対誤差 $|\hat{x} - x|$

相対誤差 $\dfrac{|\hat{x} - x|}{|x|}$

浮動小数点数

プリント p1

$\underline{1}.\ \ldots 0.1 \ldots \quad 1.\underline{0}$

1は略.　　0が1ハ.

$fl(x)$. xを浮動小数点に丸めた数.

マシンイプシロン, 表現できる最小の数.

$$fl(1.0 + \varepsilon) = 1.0 + \varepsilon.$$

丸め単位. $u := \dfrac{\varepsilon}{2}$

$$fl(x \ \mathrm{op}\ y) = (x \ \mathrm{op}\ y)\underbrace{(1+\delta)}_{\text{丸め誤差}}, \quad |\delta| < u$$

$\varepsilon = 2^{-52}$, $u = 2^{-53} = 1.1 \times 10^{-16}$

絶対値小で underflow : 小さすぎて表現できない

大　over 〃　大き　〃

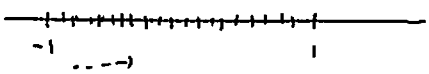
$0 \leftarrow \cdots \quad 10^{-308} \sim 10^{308} \quad \cdots \rightarrow \mathrm{inf}.$

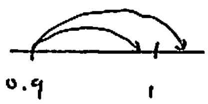
underflow　　　　overflow

丸め誤差 ex.

0.1 を 10回足しても 1になるとは限らない.

for ($x = -1$; $x \leq 1$; x += 0.1)





0.1の丸め誤差がたまっていく.

上か下か分からない.

ではどうするか → 整数でやる

for (i=-10; i ≤ 10; i++) { … x= 0.1×i; … }

もしくは

for (x=-1; x < $\underbrace{1.00\cdots01}_{\text{ちょっと大きく.}}$; x += 0.1)

2次方程式の解の公式.

倍精度, 単精度を比較してみると… プリントp3

b	8.02
√D	8.0149961
√D - b	0.0000039

2桁しか表現できていない.

→ $\boxed{\text{桁落ち}}$.

↗「たち」方.

2つの根のうちどちらかは桁落ちしない.

桁落ちしない根は公式で求める.

もう1つの根は解と係数の関係で求める.

$$x_1 x_2 = \frac{c}{a}$$

→ これなら.

単精度 -1.259351e-3

無限和の近似. ex

$$\sum_{k=1}^{\infty}\frac{1}{k^2}=\frac{\pi^2}{6}\approx 1.64493407$$

単精度で計算すると、 1.64472532

$$1.64257\boxed{1}$$ → 丸めで足せない.

$$+)\ 0.00001\boxed{2345123}$$

=

ここしか足せない。 小さい方の精度が落ちてしまう.

→ $\boxed{\text{情報落ち}}$

絶対値の大きく異なる2数の加減算で生じる. $\Sigma .. = (1+(..$ 1が大きすぎる.

対策。小さい方から足す

数値微分

微分の定義に戻って近似

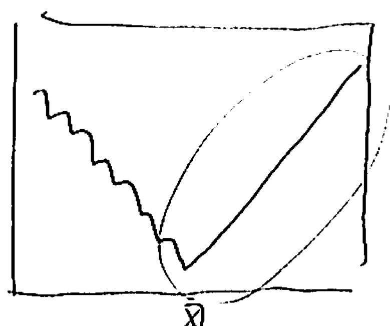
$$f'(x)\approx\frac{f(x+h)-f(x)}{h}.$$

打ち切り誤差.

$$\left|\frac{f(x+h)-f(x)}{h}-f'(x)\right| \approx \frac{h}{2}f''(x)$$

→ $O(h)$.

: 収束に影響する.



丸め誤差.

$f(x)$ とか $f(x+h)$ が誤差をもつ

$\underline{\hat{f}_x}$ $\underline{\hat{f}_{x+h}}$

ちょいもの.

$$|\hat{f}_x - f(x)| \approx u\,|f(x)|$$

$u = 10^{-16}$

表現としての誤差.

$$|\tilde{f}_{x+h} - f_{..}| \approx u\,|\ \ |$$

= これも u くらい

$$\left| (\hat{f}_{x+h} - f_x) - (f(x+h) - f(x)) \right|$$

$$\overset{\text{三角不}}{\le} \left| \hat{f}_{x+h} - f(x+h) \right| + \left| \hat{f}_x - f(x) \right|$$

$$\approx 2u \left| f(x) \right|$$

$$\rightsquigarrow \frac{2u\left| f(x) \right|}{h}.$$

hに反比例.