数値解析
第一回　数値計算と誤差

舟木　剛

平成28年10月5日2限

シラバス

- 授業の目的
  - 工学分野でよく用いられる数値計算の算法ならびにそれらの数値的な特性について理解させる.
- 授業計画
  - 数値計算と誤差（1回）
    - 数値計算の必要性ならびに誤差の種類について説明するとともに，行列式とその性質，行及び列の交換などの基本演算，逆行列の定義と求め方について説明する.
  - 代数方程式（2回）
    - 2分法，Newton-Raphson 法，Bailey 法について解説する．また，多変数に対するNewton-Raphson 法と収束に関する留意点について述べる.
  - 連立方程式（3回）
    - Gauss の消去法，Gauss-Jordan の掃き出し法，LU 分解法の手順ならびに計算複雑度について解説する．また，Jacobi 法，Gauss-Seidel 法，SOR 法などの反復法について手順並びに収束条件について解説する.
  - 行列の固有値（3回）
    - 固有値の性質，べき乗法，Householder 法，QR 法について述べる．QR 法の収束定理について解説する.
  - 補間法（2回）
    - 線形補間，Lagrange 補間，Newton 補間，スプライン補間について述べる．多項式補間については算出方法をスプライン補間については3次のスプライン関数の導出方法を説明する．また，自然なスプライン関数とその特徴について解説する.
  - 関数近似（1回）
    - 最小2乗法による関数の近似について解説する.
  - 数値積分（1回）
    - 台形公式，シンプソンの公式による数値積分について解説する.
  - 常微分方程式（1回）
    - 単区分法である Euler 法，修正Euler 法，Runge-Kutta 法ならびに複区分法である Adams-Moulton の予測子・修正子法について解説する．また，数値不安定性についても解説する.
- 教科書 佐藤・中村共著「よくわかる数値計算」日刊工業新聞社

計算機あれこれ

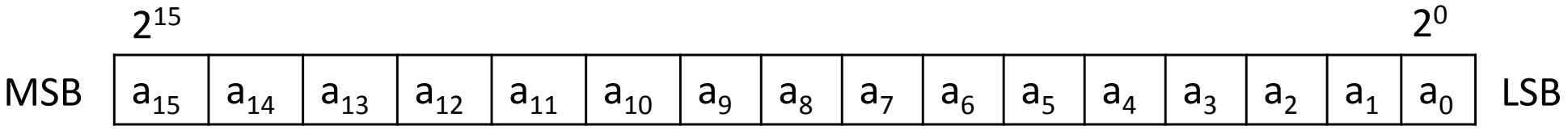
- そろばん
- 計算尺
- アナログコンピュータ
- 手回し計算機

数値解析について

- 数値解析とは
  - 計算機(コンピュータ)をもちいて計算問題(微分方程式，代数方程式)を数値的に解く　⇔解析解
- 数の表し方
  - 整数　→　金，人数
  - 実数　→　計量単位等
- 数の数え方
  - 10進法，60進法　→　生活で使用
  - 2進法　→　計算機で使用(見やすくするため8,16進表記)
    - 0，1で数を表現
    - bit　→ binary digitの略
- 今回の授業
  - 整数・実数と10,2進数の関係・表現方法

整数型の数値表現

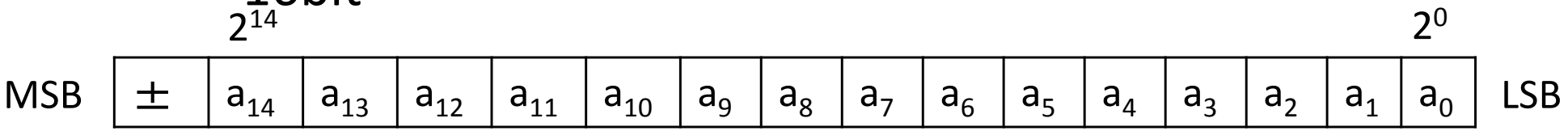
- 計算機のビット数（桁数）
  - 計算に用いる0,1のセットの数
    - 16bit



    - 0,1がa_iに入る
    - MSB: Most Significant Bit　最上位ビット
    - LSB:　Least Significant Bit　最下位ビット
    - 16bit符号なし(>0)整数の表す範囲
      - 0~65535(=2^{16}-1)
    - 10進数への変換
      - $a = a_0 \times 2^0 + a_1 \times 2^1 \cdots a_{15} \times 2^{15}$
    - 2進数からの10進数変換　→　割り算の余りと商

整数型の数値表現

- 符号付整数（正負の数を表す）
  - MSBを符号ビット(0:正，1:負)として使用
    - 16bit



  - 計算に利用しやすい負数の表し方
    - 負の数aを表すための補数表現(nビット)
      - 1の補数表現　$2^{n-1}-1-|a|$
        - $|a|$を2進数で表し，各0,1を反転したもの
      - 2の補数表現　$2^{n}-|a|$
        - $|a|$を2進数で表し，各0,1を反転し，LSBに1足したもの

2の補数表現

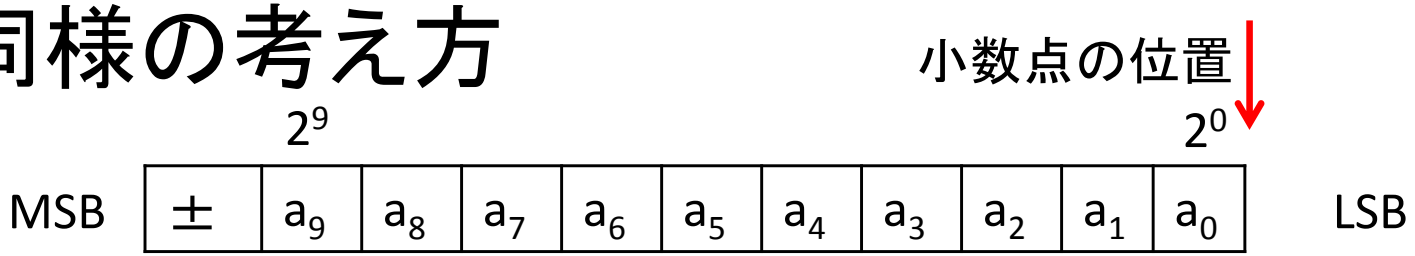
- 16bitの負の数a
  - 2の補数表現$2^{16} - |a|$ (2^{16}は17bit目の数)
  - 正の数$|a|$の和
    - $2^{16} - |a| + |a| = 2^{16} = 0$
      - 17bit目は無視。フラグとして扱う
    - 足したら0になる
    - 減算は2の補数の加算と等価になる

小数の表し方

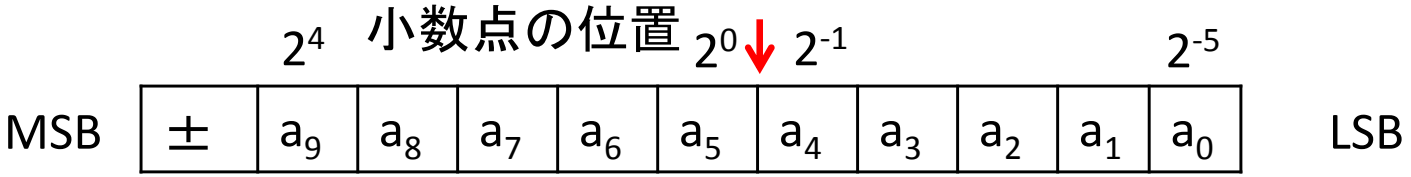
- 整数(nビット)
  - $a_{n-1}a_{n-2}\cdots a_2a_1a_0$
  - $a = a_{n-1} \times 2^{n-1} + \cdots + a_1 \times 2^1 + a_0 \times 2^0$
- 小数
  - $0.a_{n-1}a_{n-2}\cdots a_2a_1a_0$
  - $a = a_{n-1} \times 2^{-1} + \cdots + a_1 \times 2^{1-n} + a_0 \times 2^{-n}$
    - 2^{-n}の倍数しか誤差無しに表すことはできない

固定小数点計算

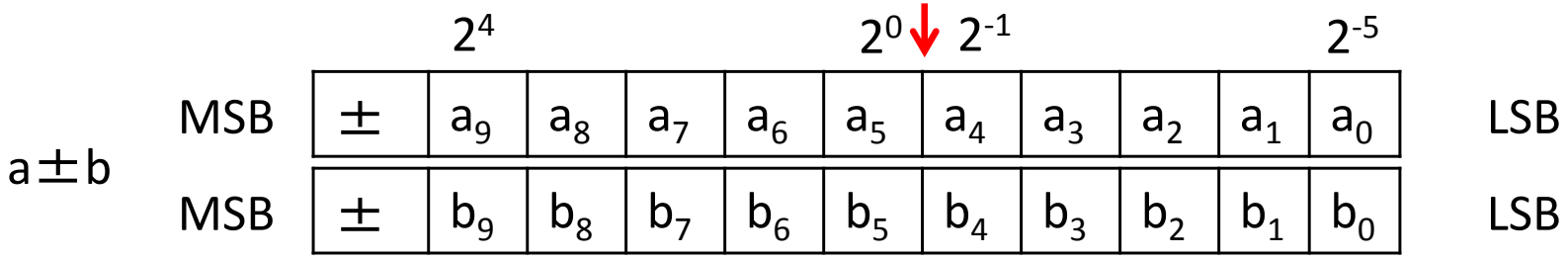
- 整数型と同様の考え方
  - 整数型



  - 固定小数点型
    - 整数型と同様だが，小数点の位置を考える



    - 加減算には小数点の位置を合わせる必要有



整数・固定小数点計算

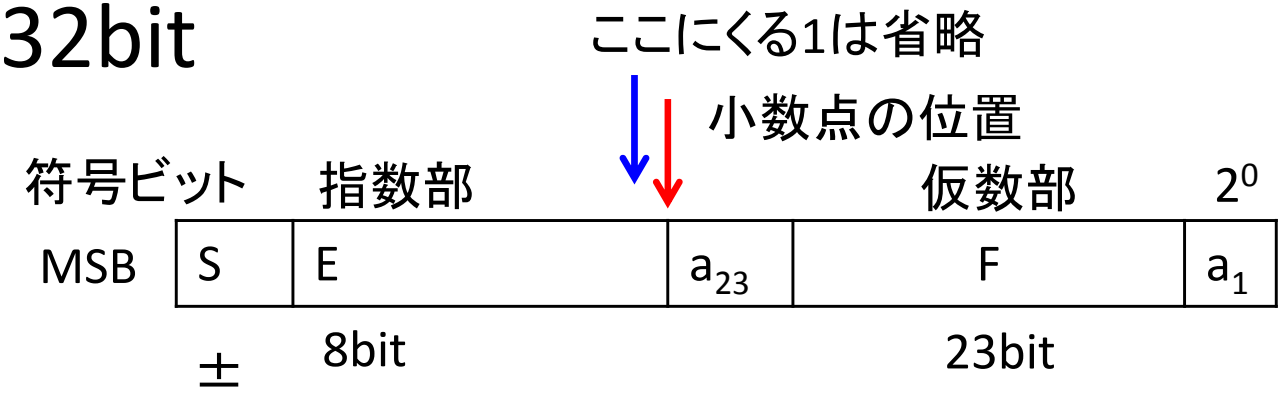
- 加減算
  - 計算の前後で同じビット数
    - キャリー・ボロー等有
  - 長さが異なる場合はどちらかに合わす必要あり
- 乗算
  - Nbit x mbit = n+m bit
- 割り算
  - 割り切れない場合がある

浮動小数点計算

- 実数型
  - より広い範囲の数を実用的に表す
  - 数値の表現　$x = F \times b^E$
    ただし，F:仮数部，E:指数部，b:基数
    - 学生実験での有効桁数を思いだそう
    - 10進数$(b = 10)$　　$123.456 = 1.23456 \times 10^2$
      $x = (-1)^s (f_1 . f_2 f_3 \cdots f_i \cdots)_{10} \times 10^E$
      ただし，最上位:$f_1 \neq 0$（桁下がりするのを防ぐ），
      - 正規化:仮数部と指数部を上の条件に合うようにする
    - 2進数$(b = 2)$，最上位:$f_1 \neq 0$ は$f_1 = 1$しかない
      - IEEE形式では省略してビット数を節約

2進数での浮動小数点表現 (IEEE形式)

- 32bit



$2^{-127} \leq 2^E \leq 2^{128}$
として表現できる
範囲を広げる

  - 仮数部　$-1 \leq F < 1$
  - $x = (-1)^s \times \left(1 + \sum_{i=1}^{23} \frac{a_i}{2^i}\right) \times 2^{E-127}$
- 指数の範囲外になると指数部オーバーフロー，アンダーフローとなる　→　正しい結果とならない

浮動小数点形式

- 単精度(32bit, float)
  - 符号部1ビット，指数部8ビット，仮数部23ビット
  - 表現範囲(10進数)
    - 1.2E-38～3.4E+38
    - 0
    - -1.2E-38～-3.4E+38
- 倍精度(64bit, double)
  - 符号部1ビット，指数部11ビット，仮数部52ビット
  - 表現範囲(10進数)
    - 2.2E-308～1.8E+308
    - 0
    - -2.2E-308～-1.8E+308

実数表現の精度

- 仮数部に依存
  - 仮数部23bit(単精度) $\log_{10} 2^{23} \cong 7.22$→10進数で7桁の精度
- 10進数の実数が必ずしも有限ビット数の浮動小数点で表されるわけではない
  - 小数点以下nビットの仮数部 → $1/2^n$毎の値しか表現できない
    - 10進数0.1の2進数変換
      $0.1_{10} = (0.0001100110011001100\cdots)_2$→循環小数
      - 10進数の有理数は2進数において無理数となることがある

数値表現における誤差

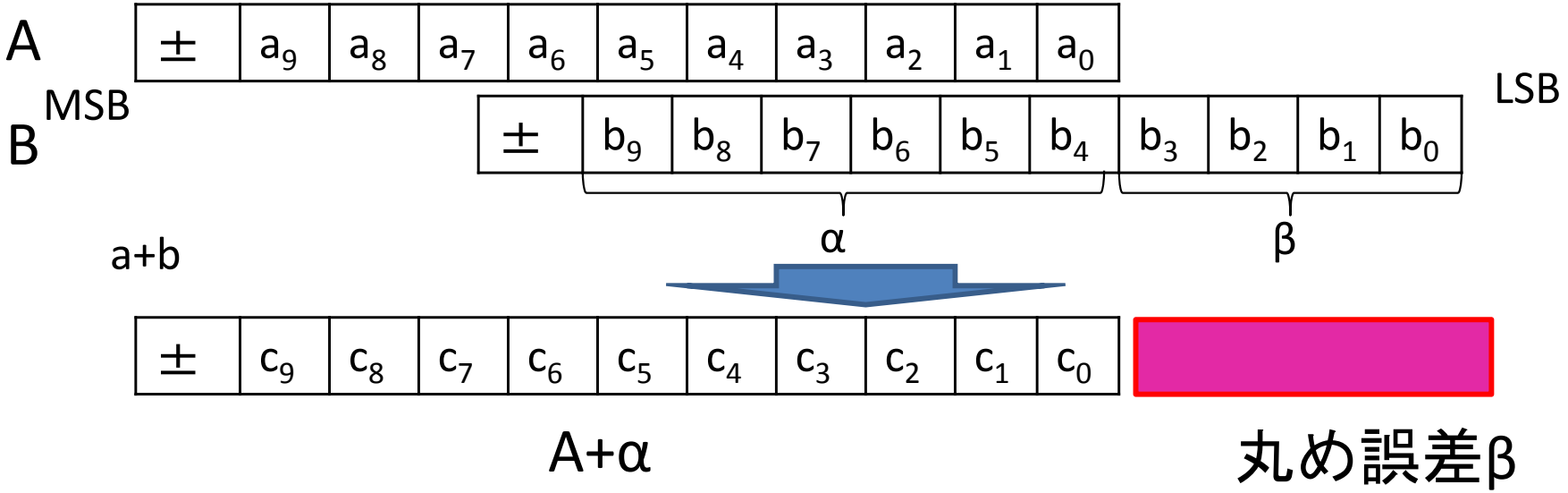
- 真値: a, 計算機での数値表現(計算結果):x
  - 誤差:$E = x - a$
  - 絶対誤差: $E_A = |E| = |x - a|$
  - 相対誤差: $E_R = \frac{E}{a} = \frac{x-a}{a}$
  - 誤差の限界:$\varepsilon(> 0) \rightarrow x - \varepsilon < a < x + \varepsilon$
    - εが小さいほど精度が良い
- 誤差解析
  - 真値が不明の場合，誤差も不明
  - 計算した値の誤差の範囲(限界)→有効桁数
  - 計算の順序で変わる

数値表現における誤差

- 誤差の種類
  - 入力誤差
    - 人為ミス
    - 変換誤差 （10進数→2進数）
  - 打切り誤差
    - 無理数の有限桁数での近似
    - 関数のテイラー展開
  - 丸め誤差（桁落ち）
    - 計算機の有限のビット数に対する有効桁数の調整
      - 四捨五入で発生
    - 打切り誤差と類似

数値表現における誤差

- 丸め誤差の例
  - 浮動小数点数AとBの和（A>B）
    - A,Bの桁数は同じ。
    - 指数部が異なる→対応する小数点の位置が異なる



誤差の伝搬と累積

- 真値x_a, y_a, 近似値x, y, 誤差$E_x = x - x_a$, $E_y = y - y_a$
  - 加減算 $x \pm y = (x_a + E_x) \pm (y_a + E_y)$
    $= (x_a \pm y_a) + (E_x \pm E_y)$
    - 誤差は和または差 $E_x \pm E_y$
  - 乗算 $xy = (x_a + E_x)(y_a + E_y) = x_a y_a + x_a E_y + y_a E_x + E_x E_y \cong x_a y_a + x_a E_y + y_a E_x$
    - 相対誤差はxとyの相対誤差の和 $\frac{x_a E_y + y_a E_x}{x_a y_a} = \frac{E_x}{x_a} + \frac{E_y}{y_a}$

誤差の伝搬と累積

- 除算 $\frac{x}{y} = \frac{x_a+E_x}{y_a+E_y} = \frac{x_a}{y_a} \times \frac{y_a}{x_a} \times \frac{x_a+E_x}{y_a+E_y}$

$$= \frac{x_a}{y_a}\left(1+\frac{E_x}{x_a}\right)\frac{y_a}{y_a+E_y} = \frac{x_a}{y_a}\left(1+\frac{E_x}{x_a}\right)\left(1-\frac{E_y}{y_a+E_y}\right)$$

$$= \frac{x_a}{y_a}\left(1+\frac{E_x}{x_a}\right)\left\{1-\frac{E_y}{y_a}+\left(\frac{E_y}{y_a}\right)^2-\cdots\right\}$$

$$\cong \frac{x_a}{y_a}\left(1+\frac{E_x}{x_a}-\frac{E_y}{y_a}\right)$$

  - 相対誤差はxとyの相対誤差の差 $\frac{x_aE_y+y_aE_x}{x_ay_a} = \frac{E_x}{x_a} - \frac{E_y}{y_a}$
  - ただし $\frac{x}{1+x} = x - x^2 + x^3 \cdots$
    $\Leftrightarrow (1+x)(x - x^2 + x^3 \cdots) = (x - x^2 + x^3 \cdots) + x(x - x^2 + x^3 \cdots)$
    $= (x - x^2 + x^3 \cdots) + (x^2 - x^3 + x^4 \cdots) = x$

高精度の数値計算

- 一般的にハードで実装された精度で計算
  - C言語のintは処理系に依存
- GMP(GNU Multiple Precision library)
  - 多倍長精度
    - 符号付整数(signed integer)
    - 有理数(rational number)
    - 浮動小数点数(floating point number)
  - 有効桁数を拡張
    - ソフトウェアで実装
    - ハードウェアのメモリ容量以外に精度は制限されない
- ARMv8-AではSVE(scalable vector extension)で2048bitまで対応