計算機実験 I (第 1 回)

藤堂眞治
wistaria@phys.s.u-tokyo.ac.jp

2023/04/05

講義・実習の目的

- 理論・実験を問わず、学部〜大学院〜で必要となる現代的かつ普遍的な計算機の素養を身につける
- UNIX 環境に慣れる (シェル、ファイル操作、エディタ)
- ネットワークの活用 (リモートログイン、共同作業)
- プログラムの作成 (C 言語、コンパイラ、プログラム実行)
- 基本的な数値計算アルゴリズム・数値計算の常識を学ぶ
- 科学技術文書作成に慣れる (LaTeX, グラフ作成)

身に付けて欲しいこと

- ツールとしてないものは自分で作る (物理の伝統)
- すでにあるものは積極的に再利用する (車輪の再発明をしない)
- 数学公式と数値計算アルゴリズムは別物
- 刻み幅・近似度合いを変えて何度か計算を行う
- グラフ化して目で見てみる
- 計算量 (コスト) のスケーリング (次数) に気をつける
- (計算機は指示したことを指示したようにしかやってくれないということを認識する)

講義・実習内容

- UNIX 操作・ネットワーク
- プログラミング: C 言語、数値計算ライブラリの利用
- ツール: エディタ、コンパイラ、LaTeX、gnuplot
- 数値計算の基礎
- 常微分方程式の解法
- 連立一次方程式の解法
- 行列の対角化
- 線形回帰

スタッフ・進め方・評価

- スタッフ computer@exa.phys.s.u-tokyo.ac.jp
  - 講義: 藤堂
  - 実習: 高橋助教、山崎助教
  - 実習 TA: 磯貝、菅原
- 講義・実習の進め方
  - 講義 (座学) と実習を交互に実施
  - 実習の回には各自 PC を持参すること
- 評価
  - 出席 (ITC-LMS でのアンケートに回答)
    講義・実習当日 11:00-13:00 の間に回答
  - レポート (計 3 回)

講義日程 (予定)

- 全 8 回 (水曜 2 限 10:25-12:10)
  - 4 月 5 日 講義 1: 講義の概要・基本的なアルゴリズム
  - 4 月 19 日 演習 1: 環境整備・C 言語プログラミング・図のプロット
  - 4 月 26 日 講義 2: 常微分方程式
  - 5 月 10 日 演習 2 (グループ 1): 基本的なアルゴリズム・常微分方程式
  - 5 月 17 日 演習 2 (グループ 2): 基本的なアルゴリズム・常微分方程式
  - 5 月 24 日 休講
  - 5 月 31 日 休講
  - 6 月 7 日 講義 3: 連立方程式
  - 6 月 14 日 演習 3 (グループ 1): 連立方程式
  - 6 月 21 日 演習 3 (グループ 2): 連立方程式
  - 6 月 28 日 休講
  - 7 月 5 日 講義 4: 行列の対角化
  - 7 月 12 日 演習 4 (グループ 1): 行列の対角化
  - 7 月 19 日 演習 4 (グループ 2): 行列の対角化
- 2 回目以降の演習はクラスを 2 グループに分けて実施 (グループ 1: 学生証番号が奇数、グループ 2: 偶数)

レポート (予定)

- レポート
  - 各自が LaTeX で作成の上提出 (計 3 回)
  - 提出方法: ITC-LMS で PDF ファイルを提出
- レポート No.1
  - [数値誤差・アルゴリズム基礎] 2 題、[常微分方程式] うちから 1 題の合計 3 題を選択
  - 締切: 5 月 31 日
- レポート No.2
  - [連立一次方程式] から 2 題を選択
  - 締切: 7 月 5 日
- レポート No.3
  - [対角化] から 2 題を選択
  - 締切: 8 月 2 日
- 詳細は ITC-LMS に掲示予定

講義資料

- 講義資料置き場:
  https://github.com/todo-group/ComputerExperiments/tree/2023s-computer1
  - 計算機実験ハンドブック
    - UNIX 入門
    - gnuplot 入門
    - C 言語入門
    - LaTeX 入門
  - 計算機実験のための環境整備 (https://utphys-comp.github.io)
  - 講義資料
  - 実習課題
  - サンプルコード

質問がある場合には…

- 講義・実習時間中は自由に質問してください
  - 計算機実験 Slack でこっそり質問も可
- 講義時間外も質問を受け付けます
  - 計算機実験 Slack (参加方法は ITC-LMS の掲示板参照)
  - ITC-LMS「担当教員へのメッセージ」
  - メール (computer@exa.phys.s.u-tokyo.ac.jp)
- メール・Slack などで質問するときに注意すべきこと
  - (メールの場合) Subject をきちんとつける、きちんと名乗る
  - 実行環境を明示する
  - 問題を再現する手順を明記する
  - 関連するファイル (C や LaTeX のソースコード等) を添付する
  - エラーメッセージを添付する

計算機実験に必要な環境整備

- 「計算機実験ハンドブック」に書いてあることが、自分の PC で一通り試せるような環境を準備する
  - プログラミング (オフライン・リモート利用)
    エディタ、コンパイラ (C, C++, Fortran, BLAS/LAPACK, MPI/OpenMP)
  - 計算結果のプロット (オフライン利用)
    gnuplot (python/matplotlib, MATLAB でも可)
  - 文書 (レポート、論文) の作成 (オフライン・クラウド利用)
    LaTeX(TeX Live あるいは Overleaf)
  - ネットワークの利用 (ECCS などの大学の計算機にリモートアクセスできる環境)
    ターミナル、SSH
  - インタプリタ環境 (オフライン・クラウド利用)
    MATLAB、Python
- 「計算機実験のための環境整備」(https://utphys-comp.github.io) を参考に、環境を整備する ⇒ 実習 1

数値誤差の原因

- 打ち切り誤差: テイラー展開による近似を有限項で打ち切ることによる誤差 (例: 数値微分)
- 丸め誤差: 無理数や 10 進数を有限のビットの 2 進数で表現することによる誤差 (例: 0.1 が 0.10000000000000006 になる)
- 桁落ち: 非常に近い数の引き算により生じる
- 情報落ち: 非常に大きな数に小さな数を足し込む場合に生じる (例: 数値積分や常微分方程式の初期値問題で刻み幅を小さくしすぎると生じる)
- オーバーフロー (桁あふれ): 表現できる最大値を超えてしまう

桁落ち

- 2 次方程式 $ax^2 + bx + c = 0$ の解の公式

$$x_{\pm} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

  $b^2 \gg |ac|$ の時、桁落ちが生じる

- 例) $2.718282x^2 - 684.4566x + 0.3161592 = 0$ の解を 7 桁の精度で計算してみる (伊理・藤野 1985)

$$\sqrt{D} = \sqrt{(684.4566)^2 - 4 \times 2.718282 \times 0.3161592} = 684.4541$$

$$x_+ = \frac{684.4566 + 684.4541}{2 \times 2.718282} = \frac{1368.911}{5.436564} = 251.7970$$

$$x_- = \frac{684.4566 - 684.4541}{2 \times 2.718282} = \frac{0.0025}{5.436564} = 0.0004\underline{598493}$$

MATLAB での計算

- MATLAB はシンボリック変数を含む式で小数を使うと、中で分数に直して (厳密に) 計算する

```
>> syms x
>> solve(2.718282*x^2-684.4566*x+0.3161592,x)
ans =
 2568765407259397l2/2040342300381321 -
      659850729835795709785858342301921O5^(1/2)/2040342300381321
 659850729835795709785858342301921O5^(1/2)/2040342300381321 +
      2568765407259397l2/2040342300381321
```

- 10 桁の精度で解を評価してみる

```
>> vpa(solve(2.718282*x^2-684.4566*x+0.3161592,x),10)
ans =
 0.000461913553
    251.7970337
```

桁落ちを防ぐ方法

- b の符号に応じて、一方を求める (この例では x_+)
- 他方は解と係数の関係を使って求める

$$x_- = \frac{c/a}{x_+} = \frac{0.3161592/2.718282}{251.7970} = 0.0004619138$$

- 回避できない例: 重解に近い場合
  $2.718282x^2 - 1.854089x + 0.3161592 = 0$

$$\begin{aligned}\sqrt{D} &= \sqrt{(1.854089)^2 - 4 \times 2.718282 \times 0.3161592} \\ &= 0.00264575 \\ x_\pm &= 1.854089 \pm 0.00264575 = 1.856737, 1.851445\end{aligned}$$

刻み幅を変えた計算

- 刻み幅を変えて何度か計算を行い、収束の様子をみる
- グラフ化して目で見てみる
- 理論式と比較
  - 計算式の正しさの確認
  - 近似の改良 (収束の加速・補外)
- 桁落ち・情報落ちの影響の有無

数値微分 (差分)

- 関数のテイラー展開

$$f(x+h) = f(x) + hf'(x) + h^2 f''(x)/2 + h^3 f'''(x)/6 + \cdots$$

- 数値微分の最低次近似 (前進差分)

$$f_1(x,h) \equiv \frac{f(x+h) - f(x)}{h} = f'(x) + hf''(x)/2 + O(h^2)$$

- より高次の近似 (中心差分)

$$f_2(x,h) \equiv \frac{f(x+h) - f(x-h)}{2h} = f'(x) + h^2 f'''(x)/6 + O(h^3)$$

- 刻み h を小さくすると打ち切り誤差は減少するが、小さすぎると今度は桁落ちが大きくなる

差分の一般式

- 関数のテイラー展開: $f(x+h) = \sum_k \frac{h^k}{k!} f^{(k)}(x)$
- $f^{(m)}(x)$ を n 個の $f(x+h_j)$ の線形結合で表す $(n \geq m+1)$

$$f^{(m)}(x) \approx \sum_j a_j f(x+h_j) = \sum_j a_j \sum_k \frac{h_j^k}{k!} f^{(k)}(x)$$

$$= \sum_k C_k f^{(k)}(x) \qquad (C_k \equiv \sum_j a_j \frac{h_j^k}{k!})$$

- $C_k = \delta_{k,m}$ $(k = 0 \cdots n-1)$ となるように $a_0 \cdots a_{n-1}$ を決める
- 行列 $G_{kj} = \frac{h_j^k}{k!}$ と列ベクトル a_j と $b_k = \delta_{k,m}$ を導入すると、条件式は $Ga = b$ と書ける (連立一次方程式)

ニュートン法

- 反復法により方程式 $f(x) = 0$ の解を求める
- 真の解を x_0、現在の解の候補を $x_n = x_0 + \epsilon$ とすると

$$0 = f(x_0) = f(x_0 + \epsilon - \epsilon) = f(x_n) - f'(x_n)\epsilon + O(\epsilon^2)$$

- 次の解の候補 (反復法、逐次近似法)

$$\epsilon \approx \frac{f(x_n)}{f'(x_n)} \qquad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

- 複素変数の複素関数や多変数の場合にも自然に拡張可

ニュートン法の収束

- x_n が x_0 に十分近い時

$$f(x_n) \approx f'(x_0)(x_n - x_0) + f''(x_0)\frac{(x_n - x_0)^2}{2}$$
$$f'(x_n) \approx f'(x_0) + f''(x_0)(x_n - x_0)$$

- ニュートン法で一回反復すると

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \approx x_n - (1 - \frac{f''(x_0)}{f'(x_0)}\frac{(x_n - x_0)}{2})(x_n - x_0)$$
$$(x_{n+1} - x_0) \approx \frac{f''(x_0)}{2f'(x_0)}(x_n - x_0)^2$$

- 一回の反復で誤差が 2 乗で減る (正しい桁数が倍に増える) ⇒ 二次収束

多次元の場合

- $f(x) = 0$: d 次元 (非線形) 連立方程式
- x は d 次元のベクトル: $x = {}^t(x_1, x_2, \cdots, x_d)$
- $f(x)$ も d 次元のベクトル: $f(x) = {}^t(f_1(x), f_2(x), \cdots, f_d(x))$
- 真の解のまわりでの展開 ($x_n = x_0 + \epsilon$)

$$0 = f(x_0) = f(x_0 + \epsilon - \epsilon) = f(x_n) - \frac{\partial f(x_n)}{\partial x} \cdot \epsilon + O(|\epsilon|^2)$$

- ヤコビ行列 ($d \times d$): $\left[\frac{\partial f(x)}{\partial x}\right]_{ij} = \frac{\partial f_i(x)}{\partial x_j}$
- 次の解の候補: $x_{n+1} = x_n - \left[\frac{\partial f(x_n)}{\partial x}\right]^{-1} f(x_n)$

ニュートン法による最適化

- x は d 次元のベクトル: $x = {}^t(x_1, x_2, \cdots, x_d)$、目的関数 $f(x)$ はスカラー
- 勾配ベクトル: $[\nabla f(x)]_i = \dfrac{\partial f(x)}{\partial x_i}$
- 極小値 (最小値) となる条件: $\nabla f(x) = 0$
- ニュートン法で $f(x)$ を $\nabla f(x)$ で置き換えればよい
- 次の解の候補: $x_{n+1} = x_n - H^{-1}(x_n)\nabla f(x_n)$
- ヘッセ行列 (Hessian): $H_{ij}(x) = \dfrac{\partial^2 f}{\partial x_i \partial x_j}(x)$

準ニュートン法

- 準ニュートン法: それまでの反復で計算した勾配ベクトルから、ヘッセ行列の逆行列を近似 ($C_n \approx H^{-1}$)
  - 極小点の近傍では

$$x - x_n = -H^{-1} \cdot \nabla f(x_n)$$
$$x - x_{n+1} = -H^{-1} \cdot \nabla f(x_{n+1})$$

  - 差をとると ($s_n = x_{n+1} - x_n$, $y_n = \nabla f(x_{n+1}) - \nabla f(x_n)$)

$$s_n = H^{-1} \cdot y_n$$

  - この式が満たされるように、C_{n+1} を C_n + (補正) の形で構成 (DFP 法: Davidon-Fletcher-Powell)

$$C_{n+1} = C_n + \frac{s_n s_n^T}{s_n^T y_n} - \frac{C_n y_n (C_n y_n)^T}{y_n^T C_n y_n}$$

  - 他にも、BFGS 法など

最急降下法 (steepest descent)

- 関数の微分の情報を使う
- 現在の点 x における勾配を計算

$$-\nabla f|_i = -\frac{\partial f}{\partial x_i}$$

- 坂を下る方向にそって、一次元最適化 (ニュートン法、囲い込み法)
- 動いた先の勾配の方向でさらに最適化を繰り返す
- 関数値は単調減少 $\Rightarrow$ 極小値に収束

勾配降下法 (gradient descent)

- 勾配方向に一次元最適化を行うかわりに、あらかじめ決めた一定量 (ϵ) だけ坂を下る

$$x_{n+1} = x_n - \epsilon \nabla f$$

- あらかじめ最適な ϵ を知るのは困難
- 機械学習の分野では、(なぜか) $\epsilon = 0.1$ が良いとされている
- この方法を「最急降下法」、一次元最適化を行う勾配法を「最適降下法 (optimum descent)」と呼ぶ場合も
- ϵ を自動的に調整する手法も提案されている: ADAM, Adagrad, etc

計算をいつやめるか？

- 残差による判定
$$|f(x)| < \delta$$
- 誤差による判定
$$|x_{n+1} - x_n| < \epsilon$$
- 解 $x = x_0$ が m 重解の場合、$x = x_0$ のまわりで展開すると
$$f(x) \simeq \alpha(x - x_0)^m$$
残差が δ 程度になったときの誤差は、$\delta^{1/m}$ 程度
逆に $|x - x_0|$ が $\delta^{1/m}$ 以下になると、$f(x)$ の値がそれ以上変化しない $\Rightarrow$ m 重解の精度は計算精度の $1/m$ 桁程度しかない
- 残差による判定と誤差による判定を併用するのがよい

反復計算

- while による反復 (ハンドブック 2.2.5 節)

```
double residual = 1;      /* 残差: 適当に大きな値 */
double error = 1;         /* 誤差: 適当に大きな値 */
double delta = 1.0e-12; /* 欲しい精度 */
while (residual > delta && error > delta) {
  /* ニュートン法の漸化式 */
  /* residual と error を計算 */
}
```

残差と誤差のどちらかが欲しい精度に達したら計算を終了

- break を使う例 (ハンドブック 2.2.6 節)

```
for (;;) {
  /* ニュートン法の漸化式 */
  /* residual と error を計算 */
  if (residual < delta || error < delta) break;
}
```

初期段階における収束の改善

- Newton 法は初期値によっては収束しない
- 発散や振動を抑える方法として「減速」が有効な場合も
- 減速
  - 反復式を少し修正する

$$x_{n+1} = x_n - \mu_n \frac{f(x_n)}{f'(x_n)}$$

  - まずは $\mu_n = 1$ として計算
  - $|f(x_{n+1})| < |f(x_n)|$ が成り立たないようであれば、μ_n を半分にして再計算
  - μ_n が十分に小さくなれば、$|f(x)|$ は必ず減少する

二分法

- 反復法により一次元の方程式 $f(x) = 0$ の解を求める
- 導関数を使わず関数値のみを利用 (c.f. ニュートン法)
- 初期条件として、$f(a) \times f(b) < 0$ を満たす 2 点の組 $(a < b)$ で解をはさみ込み、領域を狭めていく
- a と b の中点 $x = (a + b)/2$ を考える
  - $|f(x)|$ が十分小さい場合: x が解
  - $f(a) \times f(x) < 0$ の場合: $[a, x]$ を新しい領域にとる
  - $f(x) \times f(b) < 0$ の場合: $[x, b]$ を新しい領域にとる
- 領域 $[a, b]$ の幅が十分小さくなったら終了
- 反復のたびに領域の幅は半分になる
- 全ての解を得られる保証はない
- 二分法の例: bisection.c

囲い込み法 (一次元の最適化)

- 関数 $f(x)$ の極小点を求める
- $f(a) > f(b) < f(c)$ を満たす 3 点の組 $a < b < c$ の領域を狭めていく
- $[a, b]$、$[b, c]$ の広い方 (例えば後者) を b から見て、黄金比 $[1 : (1 + \sqrt{5})/2 \approx 0.382 : 0.618]$ に内分する点を x とする
  - $f(b) > f(x)$ の場合: $[b, c]$ を新しい領域にとる
  - $f(b) < f(x)$ の場合: $[a, x]$ を新しい領域にとる
- もともとの b が $[a, c]$ を $0.382 : 0.618$ に内分する点だった場合、新しい領域の幅は、どちらの場合も 0.618
- 最初の比率が黄金比からずれていたとしても、黄金比に収束
- 黄金分割法 (golden section) とも呼ばれる

最初の囲い込み

- 1 点を選び、適当な Δx を取る
- 左右に Δx 動かしてみて、関数値が小さくなる方へ動く
- どちらに進んでも関数値が大きくなる場合には、囲い込み完了
- 小さくなった場合、その方向へ再び増えるまで Δx を倍々に増やしながら進む
- 最後の 3 点で極小値を囲い込むことができる
- 囲い込み法のプログラムの例: golden_section.c