電 301 数値解析

担当者: 半場 滋
(専門: 制御工学)

教科書

川田昌克, **Scilab で学ぶわかりやすい数値計算法**, 森北出版 (2008)

- 教科書の通りに進むわけではない
- 教科書の内容をすべて説明するわけではない
- 必要に応じて教科書にない事項も解説する

- シラバスを配付するので読んでおくこと
- 12 月 3 日は推薦入試のため休講だが, この講義は実施する
- 12 月 17 日は休講の予定

第 1 回
非線形方程式の数値解法 (1)
非線形方程式とは・2 分法

数値解析とは何か (1)

数値解析

科学分野にあらわれる数学的問題を, 数値の演算によって解く方法. コンピュータの発達に伴い, 複雑なデータ処理やシミュレ– ションに用いられる. 数値計算. 実用解析 (大辞林)

数値解析とは何か (2)

数値解析

積分や方程式の解などの値, あるいは近似値を数値的に求める手法の開発を目標とする数学の分野, もしくは, それらの手法に数学的な根拠を与えることを目標とした解析学をいう. あるいは, 数値計算などを利用して数学的な対象の研究や自然現象の解明, 工学的な設計などを行うことを総称して数値解析という. (岩波入門辞典)

数値解析とは何か (3)

数値計算

式変形による計算に対比して, 具体的な数値を計算すること数値計算という. (岩波入門辞典)

数値解析とは何か (4)

- 歴史的には, 数値解析は数学と深く関係
- 古くはアルキメデス (BC283?–212) による円周の計算法や π の近似
- アイザック・ニュートン (1643–1727) も数値解析にいろいろな貢献

数値解析とは何か (5)

- 現在の数値解析は数学 (理学) と工学の境界領域に位置する

数値解析とは何か (6)

- 歴史的には, コンピュータは数値計算と深い関係
- ENIAC の目的は弾道計算
- コンピュータの用途の多様化に伴い, 数値解析は目立たない存在になったが, 重要性が減じたわけではない

数値解析とは何か (7)

- 数学を工学に使うためには, 解を数値的に求めることが必須
- この意味で, 数値解析は工学的に重要

数値解析とは何か (8)

数値解析という分野の必要性 (1)

- 解を求めるためにはアルゴリズムが必要
- そのアルゴリズムによって正解が得られることを理論的に保証する必要がある
- アルゴリズムによって, 解を得るために必要な計算の量が大きく異なる

数値解析とは何か (9)

数値解析という分野の必要性 (2)

- コンピュータによる実数の表現は近似 (厳密な意味での実数はコンピュータでは実現不能)
- 浮動小数点数という近似的な表現が使われることが多い

数値解析とは何か (10)

数値解析という分野の必要性 (3)

- 多くの数学的問題で, 解を解析的に表現することは困難であり, 数値的に近似することが必要になる
- したがって, 「良い近似計算法は何か」ということが問題になる

数値解析とは何か (11)

数値解析という分野の必要性 (4)

- いまだに良いアルゴリズムがない問題もある
- アルゴリズムは高速, 高精度かつ計算機資源の消費が少ないほど望ましいので, より良いアルゴリズム開発の需要はつねに存在する

数値解析とは何か (12)

数値解析という分野の必要性 (5)

- 数値解析は完成した学問ではなく, 今日でも研究すべき問題を多く抱え, 発展しつつある学問の分野
- この講義では基礎的な部分のみ解説

数値解析とは何か (13)

ソフトウェアの選択 (1)

- 数値解析はコンピュータで解を求めなければ意味がない
- どのソフトウェアを使うかが問題になる

数値解析とは何か (14)

ソフトウェアの選択 (2)

選択肢は…

- 高水準言語 (C 言語, Java, Python 等)
- 数値計算ソフトウェア (MATLAB, Scilab 等)
- 数式処理ソフトウェア (Mathematica, Maple 等)
- 表計算ソフト (Excel 等)

数値解析とは何か (15)

ソフトウェアの選択 (3)

- 高水準言語を用いて零から数値計算プログラムを書くことは, ふつうはやるべきではない
- 数値計算プログラムは多くの場合極めて技巧的であり, 初学者には手に負えない.
- 高水準言語で数値計算をするときには, なるべくライブラリ関数等を使う

数値解析とは何か (16)

ソフトウェアの選択 (4)

- 数値計算ソフトウェアが数値解析にもっとも適している
- アルゴリズムを書き下すには数式処理ソフトウェアの方が楽なこともあるが, 多くの場合数式処理ソフトウェアは数値計算ソフトウェアより低速

数値解析とは何か (17)

ソフトウェアの選択 (5)

Excel は…

- ある程度の数値計算はできる
- 処理が遅いため大規模計算には使いにくい

数値解析とは何か (18)

ソフトウェアの比較 (1)

1000 行 1000 列の乱数行列の生成

Excel	MATLAB	Scilab
約 15 秒	0.0093 秒	0.0112321

1000 行 1000 列の乱数行列の積

Excel	MATLAB	Scilab
約 25 秒	0.0152	0.0722285

数値解析とは何か (19)

ソフトウェアの比較 (2)

- Excel は試行 1 回, MATLAB と Scilab は 100 回の試行の平均,
- Excel では時計で時間測定
- MATLB では組み込み関数 tic と toc により時間測定
- Scilab では組み込み関数 timer により時間測定

数値解析とは何か (20)

ソフトウェアの比較 (3)

MATLAB, Scilab における上記の操作方法は

- 乱数行列の生成: `A=rand(1000,1000);`
- 行列の積: `A*B;`

数値解析とは何か (21)

ソフトウェアの比較 (4)

Excel では

- 乱数行列の生成にはマクロが必要

```
For i = 1 To 1000
For j = 1 To 1000
Cells(i, j) = Rnd
Next j
Next i
```

数値解析とは何か (22)

ソフトウェアの比較 (5)

Excel では

- 行列の積: 計算結果を収納するセル (この例では 1000×1000) をドラッグして

  `=MMULT(A1:ALL1000,A1001:ALL2000)`

  とタイプし, `[Ctrl]+[Shift]+[Enter]`

数値解析とは何か (23)

ソフトウェアの比較 (6) 上記の実行環境

```
Intel Core i5-4690 3.50GHz
32GB of memory
Windows7 Professional 64bit Service Pack 1
(MATLAB) R2015b for Windows 64bit
(Scilab) 5.5.2 for Windows 64bit
(Excel) Excel2013
```

数値解析とは何か (24)

アルゴリズムの比較 (1)

MATLAB R2015b で

- 余因子行列と行列式を使って逆行列を計算 (多くの数学の教科書で解説されている方法)
- 行列方程式 $AX = I$(I は単位行列) を解いて逆行列を計算

数値解析とは何か (25)

アルゴリズムの比較 (2)

先に述べたコンピュータで, 100 行 100 列の行列の逆行列を求める試行を 100 回繰り返したときの平均計算時間：

余因子行列を使ったとき　1.0356 秒
線形方程式を解いたとき　0.0002434 秒

数値解析とは何か (26)

- ソフトウェアやアルゴリズムには得手不得手がある
- 適切なソフトウェアとアルゴリズムの選択が必要
- 数値計算には誤差が含まれるので, 得られた解の正当性に関する検討が必要
- 解の正当性の検討のために複数のソフトウェアやアルゴリズムを使うことがある

数値解析とは何か (27)

- 分野によってはベンチマーク問題と呼ばれる標準的な問題が用意されていることがあり, ベンチマーク問題に対する性能でアルゴリズムの良し悪しを評価することも行われている
- アルゴリズムの比較をするときには, アルゴリズムを実行した計算機の環境を明示することが必須

乱数 (1)

- 数値計算の際には, いろいろな状況で乱数が用いられる
- コンピュータが生成する乱数は厳密には擬似乱数と呼ばる. 乱数生成器が生成する擬似乱数は, 不規則な数ではなく, 一見不規則に見えるが規則性を持った数である.

乱数 (2)

- 同一条件で初期化された乱数生成器はいつも同じ数を生成するので, 数値計算に擬似乱数を使うときには注意が必要.

- 擬似乱数の「品質」にも良し悪しがある (不規則に近いほど良い). また, 対応する確率分布にも色々なものがある.

浮動小数点数 (1)

- コンピュータは厳密な意味での実数を取り扱うことができない
- ふつうは浮動小数点数という近似的な表現が用いられる
- IEEE754 という規格が標準的, この規格は以下の通り.

浮動小数点数 (2)

出発点は次のような形の数の表現：

$-1.60217733 \times 10^{-19}$

この各部に次のような名前を付ける.

符号	仮数部		指数部
$-$	1.60217733	$\times$	10^{-19}

ただし, コンピュータで基本になるのは, 10 のべきではなく, 2 のべき.

浮動小数点数 (3)

このページの記述の典拠は http://pc.nikkeibp.co.jp/pc21/special/gosa/eg4.shtml

- 上記のような数を有限長の 2 進数であらわすことを考える
- 単精度と呼ばれる方法では は 32 ビット, 倍精度と呼ばれる方法では は 64 ビットを使う
- 符号には 1 ビットを使い, 正の数の 0, 負の数は 1 とする

浮動小数点数 (4)

このページの記述の典拠は http://pc.nikkeibp.co.jp/pc21/special/gosa/eg4.shtml

- 指数部は整数であるが, これを 2 進数で表現する
- 単精度では 8 ビットを指数部にあて, -126 から 127 までの 256 通りの値が表現できるようにする. 指数部のデータには, 表現したい数の実際の指数に 127(バイアス値) を加えた数値を格納する (この処理は, 数 1 が 8 ビット 2 進数 10000000 と対応するようにするため)

浮動小数点数 (5)

このページの記述の典拠は http://pc.nikkeibp.co.jp/pc21/special/gosa/eg4.shtml

- 倍精度では 11 ビットを指数部にあて，−1023 から 1024 までの 2048 通りの値が表現できるようにする． 指数部のデータには，表現したい数の実際の指数に 1023(バイアス値) を加えた数値を格納する (この処理は，数 1 が 11 ビット 2 進数 10000000000 と対応するようにするため).

浮動小数点数 (6)

このページの記述の典拠は http://pc.nikkeibp.co.jp/pc21/special/gosa/eg4.shtml

- 仮数部は, 表現したい数の絶対値を 1 以上 2 以下に正規化し 2 進小数で近似したものである.
- 単精度では 23 ビット, 倍精度では 52 ビットで近似をする.
- 表現したい数が零以外なら, $d_0 = 1$ となるので, d_0 は略す.

浮動小数点数 (7)

このページの記述の典拠は http://pc.nikkeibp.co.jp/pc21/special/gosa/eg4.shtml

- 仮数部の近似のしかたは複数あるが, 最近偶数丸めという方法がデフォルト.
- 最近偶数丸めでは, 格納しきれない最初の桁が 0 なら切り捨て, 1 なら切り上げるが, 端数が最終桁のちょうど半分のときには切り捨てる.
- 表現したい数が零のときは, 指数部と仮数部をすべて零にする.

浮動小数点数 (8)

- 浮動小数点数を用いた数値計算にはつねに誤差が含まれるので, 誤差の影響に関する注意が必要.
- 表現したい数の絶対値の 2 進小数表現を求めれば, そこから指数部と仮数部の計算ができる.
- 配付資料に計算例を示す.

非線形方程式 (1)

- 工学的な問題を数学的に解くためには, 解くべき問題を数式によって表現する必要がある.

- 問題の表現にはさまざまな形があるが, 変数 (1 個でも複数でもよい) の関係が等号であらわされているものを方程式, 不等号であらわされているものを不等式という.

非線形方程式 (2)

- 式の中に変数の微分が含まれている場合には「微分」, 式が線形であるばあいには「線形」, 非線形である場合には「非線形」という修飾語が付く.

線形方程式, 線形微分方程式, 線形不等式, 線形微分不等式, 非線形方程式, 非線形微分方程式, 非線形不等式, 非線形微分不等式など

非線形方程式 (3)

- たとえば, 「体積が 1 となる球の半径を求めよ」という問題は非線形方程式で表現される. 半径が r の球の体積の公式は $4\pi r^3/3$ であるから, $r = \sqrt[3]{\frac{3}{4\pi}}$ が求める解である.
- 上記のように解が解析的に表現できる問題は稀. 大抵は数値的な近似解しか求められない.

非線形方程式 (4)

以下の議論では，数値的な近似解を求める手法を考えてゆく．

非線形方程式 (5)

- 1個の実変数xに関する単一の方程式は, $f(x) = 0$という形で書ける. なお, $g(x) = c$(ただしcは定数)という方程式は, $g(x) - c = 0$と書き直し, $f(x) = g(x) - c$とおけば, $f(x) = 0$という形になるので, はじめから$f(x) = 0$という方程式だけを考えればよい.

非線形方程式 (6)

- 非線形方程式を解くことは, 関数のグラフと x 軸の交点を求めることに対応する.
- 交点がただひとつの場合は比較的単純.
- 交点がたくさんある場合は初期値に関する注意が必要. 解をすべて求めるのは易しくない.

非線形方程式 (7)

- 1 変数であれば, 関数のグラフを描画すれば非線形方程式の近似が得られる.
- とはいっても, グラフを描画する方法にはいろいろ欠点がある.
- しかし 2 変数関数のグラフは見にくいし, 3 変数関数以上ではそもそもグラフが描けない.
- グラフから読み取った交点の精度は高くない.

非線形方程式 (8)

- コンピュータで描画した関数のグラフは, x 軸の多数の点で関数値を評価し, それを繋げたもの.
- 関数値の評価自体の計算量が多い関数では, 描画に極めて手間がかかる.
- より少ない関数値の評価回数で高精度の近似解が求められることが望ましい.

非線形方程式 (9)

- 以下の議論では, 解くべき方程式の解を「真の解」と呼び, 数値計算によって求められた近似解と区別する.

- 非線形方程式を解くための解法はふつうは反復解法である. 反復解法とは, 適切な初期値から出発し, 何度も近似解を改善することによって, 近似解を真の解に収束させる解法である.

非線形方程式 (10)

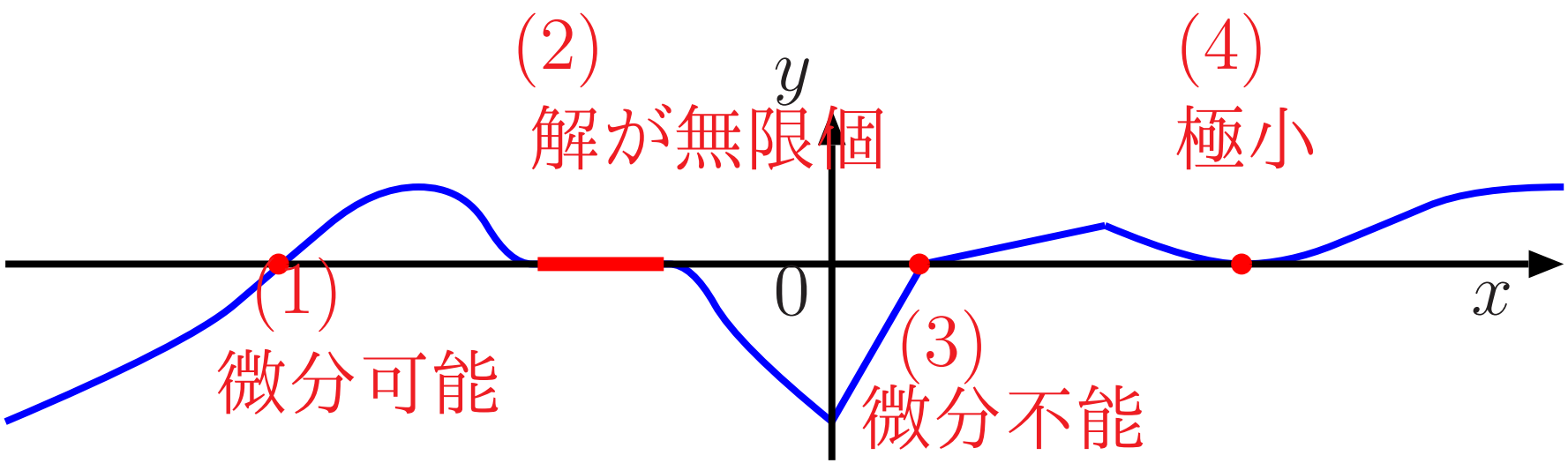
- 線形問題に対する解法には，万能に近いものが多い.
- 非線形問題に対する解法には得手不得手がある
  - ▷ 応用範囲が広い解法は効率が悪く，応用範囲が狭い解法は効率が良いという傾向がある.
  - ▷ 問題に適した解法を選ぶ必要がある..

非線形方程式 (11)

- 解くべき非線形方程式を $f(x) = 0$ とする．f は実変数の実数値関数である．
- 非線形方程式の解はたくさんあることもある．
- 以下では，ある解 x_* の近傍における関数の振舞いについて考える．

非線形方程式 (12)

解には色々なパターンがある



非線形方程式 (13)

- (1) から (3) までの場合を一応カバーできるのが 2 分法と呼ばれる解法.
- (4) は最小化問題として解く必要がある.
- 2 分法は汎用性が高く, $f(x)$ が単調関数ならつねに収束するが, 収束が遅い.
- 高速で実用的なのは次回解説するニュートン法だが, (1) の場合にしか対応できない.

非線形方程式 (14)

- 素朴な形のニュートン法は高速であるが, 初期値が真の解の近くにないと発散する可能性がある.
- 今日では, とくに多変数の場合には, 可変ステップ幅のニュートン法と呼ばれる方法が使われることが普通. ステップ幅の決定には, 直線探索あるいは信頼領域法と呼ばれる手法が使われる. こちらは, 一定の条件のもとで, 初期値によらず真 010 の解に収束する (この講義では深入りしない).

二分法 (1)

- (1) から (3) までに共通するのは, $f(x)$ が真の解の近傍で単調関数であるということ.
- 議論が繁雑になるのを防ぐため, $f(x)$ は単調増加関数と仮定する (単調減少関数についても考え方は同じ).
- 二分法のアルゴリズムは以下に述べるようなものである.

二分法 (2)

- 初期化: $f(a_0) < 0$, $f(b_0) > 0$ を見たす a_0, b_0 を何らかの方法で見付ける. $k = 0$ とする. 誤差の許容値 $\varepsilon > 0$ を定める.

二分法 (3)

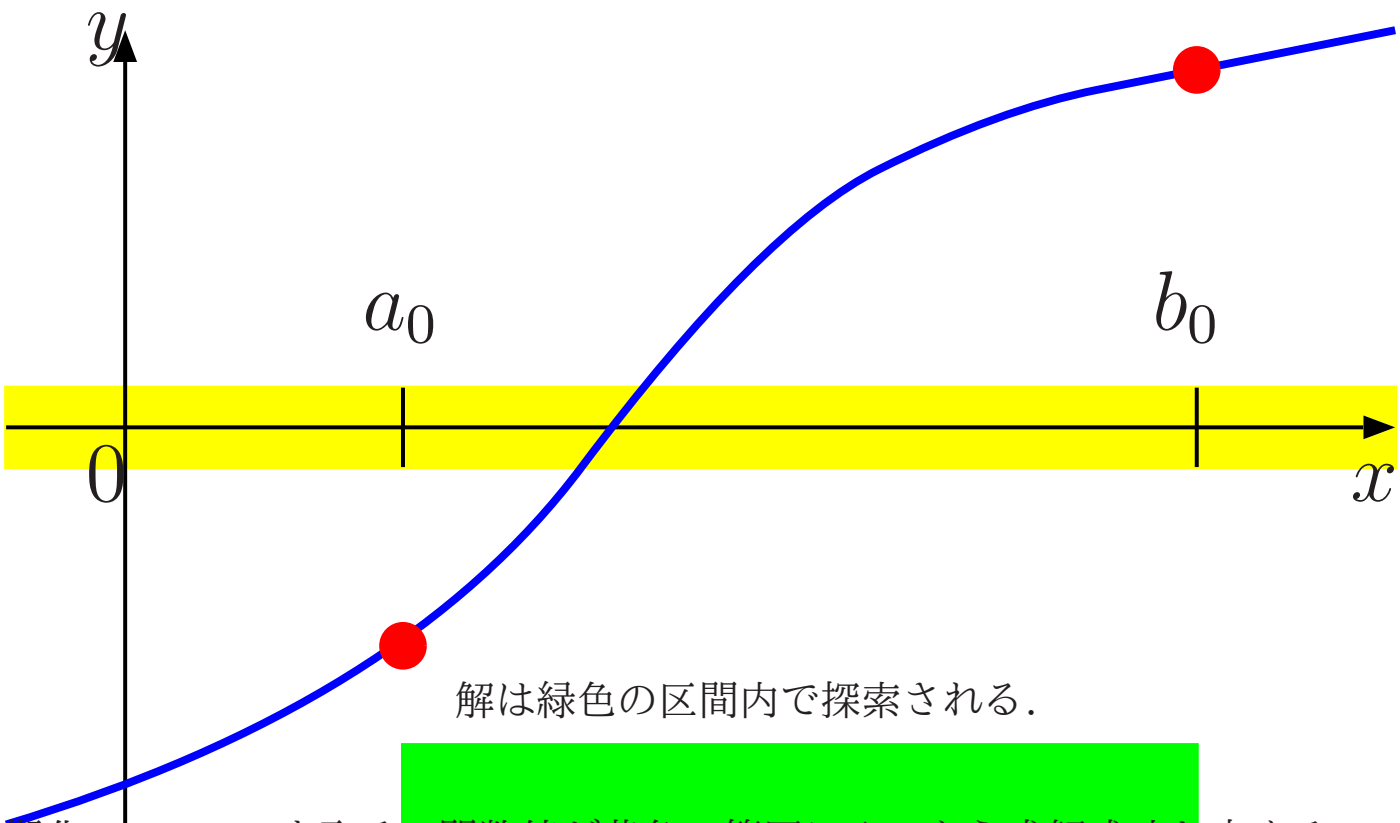
- ループ: k に 1 を加える. $c_k = \dfrac{a_k + b_k}{2}$ とし, $f(c_k)$ を評価し, 続いて以下を順に実行する.

  1. $|f(c_k)| < \varepsilon$ なら c_k が近似解 (終了).
  2. $f(c_k) > 0$ なら解は区間 $[a_k, c_{k+1}]$ にあるので, $a_{k+1} = a_k, b_{k+1} = c_{k+1}$ とする.
  3. $f(c_k) < 0$ なら解は区間 $[c_{k+1}, b_k]$ にあるので, $a_{k+1} = c_{k+1}, b_{k+1} = b_k$ とする.

  上記がすべて終わったらループの先頭に戻る.

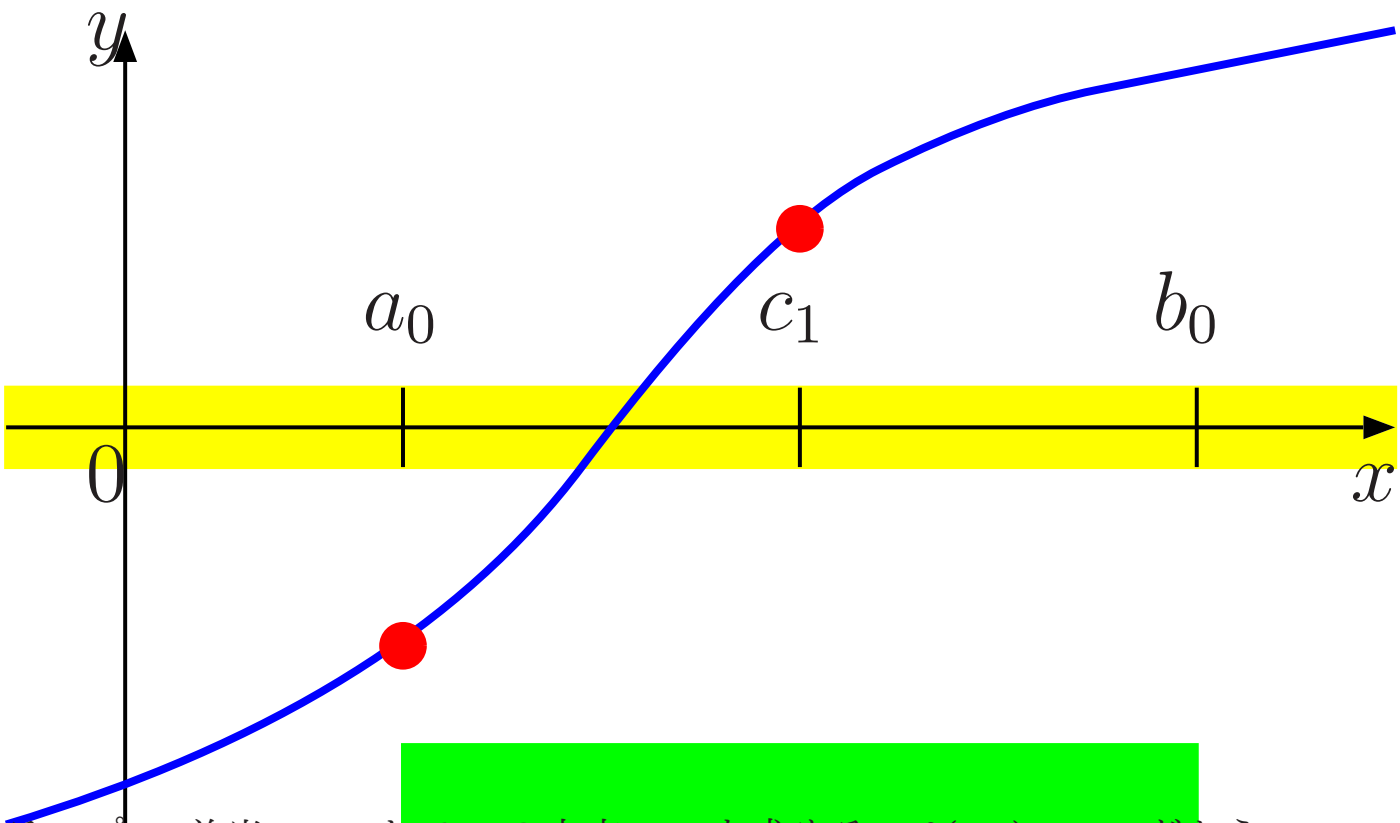
二分法 (4)

2 分法の続きは次回．今回は図 (配付配付資料) で計算の過程を見てゆく．

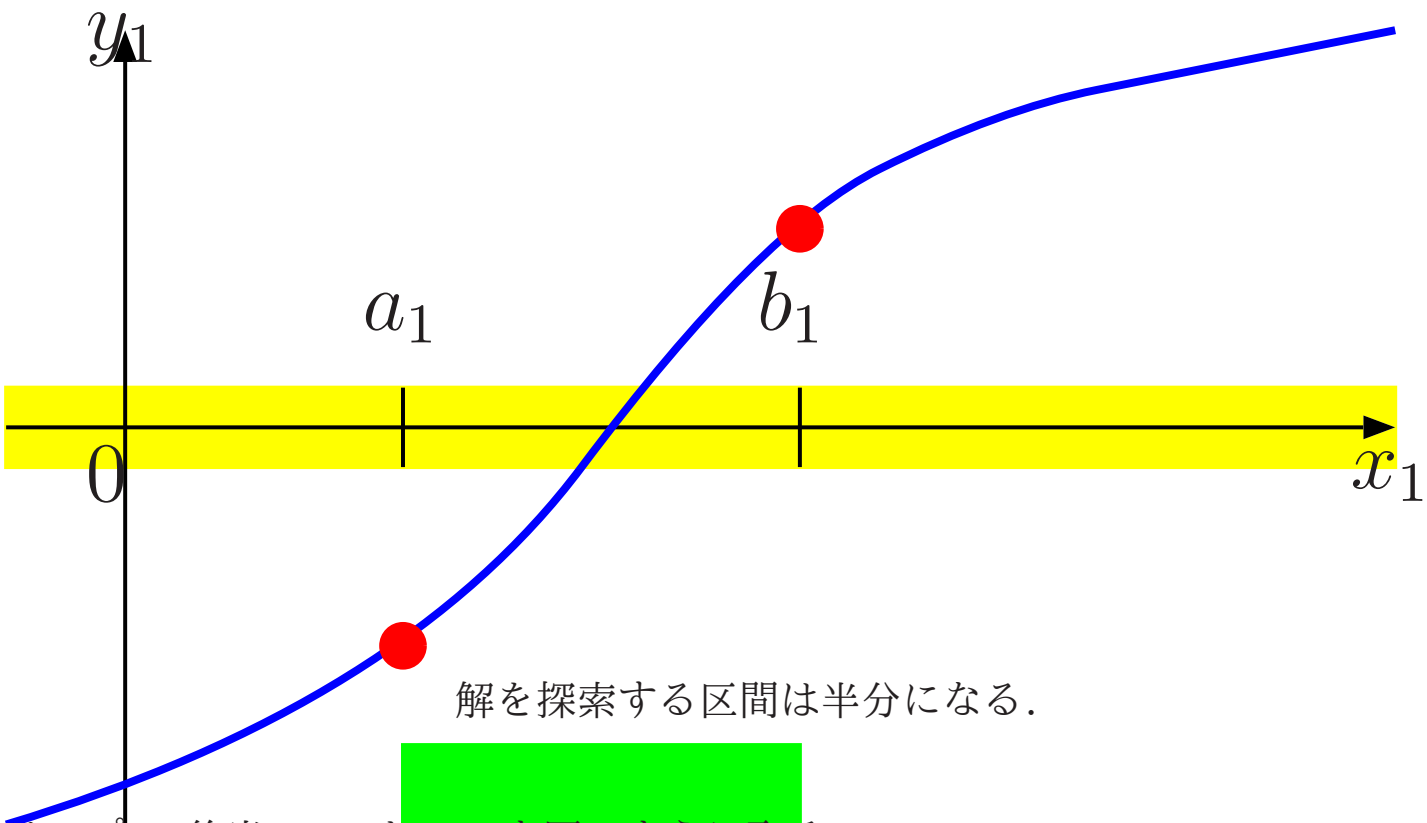


解は緑色の区間内で探索される．

初期化．a_0, b_0 を取る．関数値が黄色の範囲に入ったら求解成功と定める．

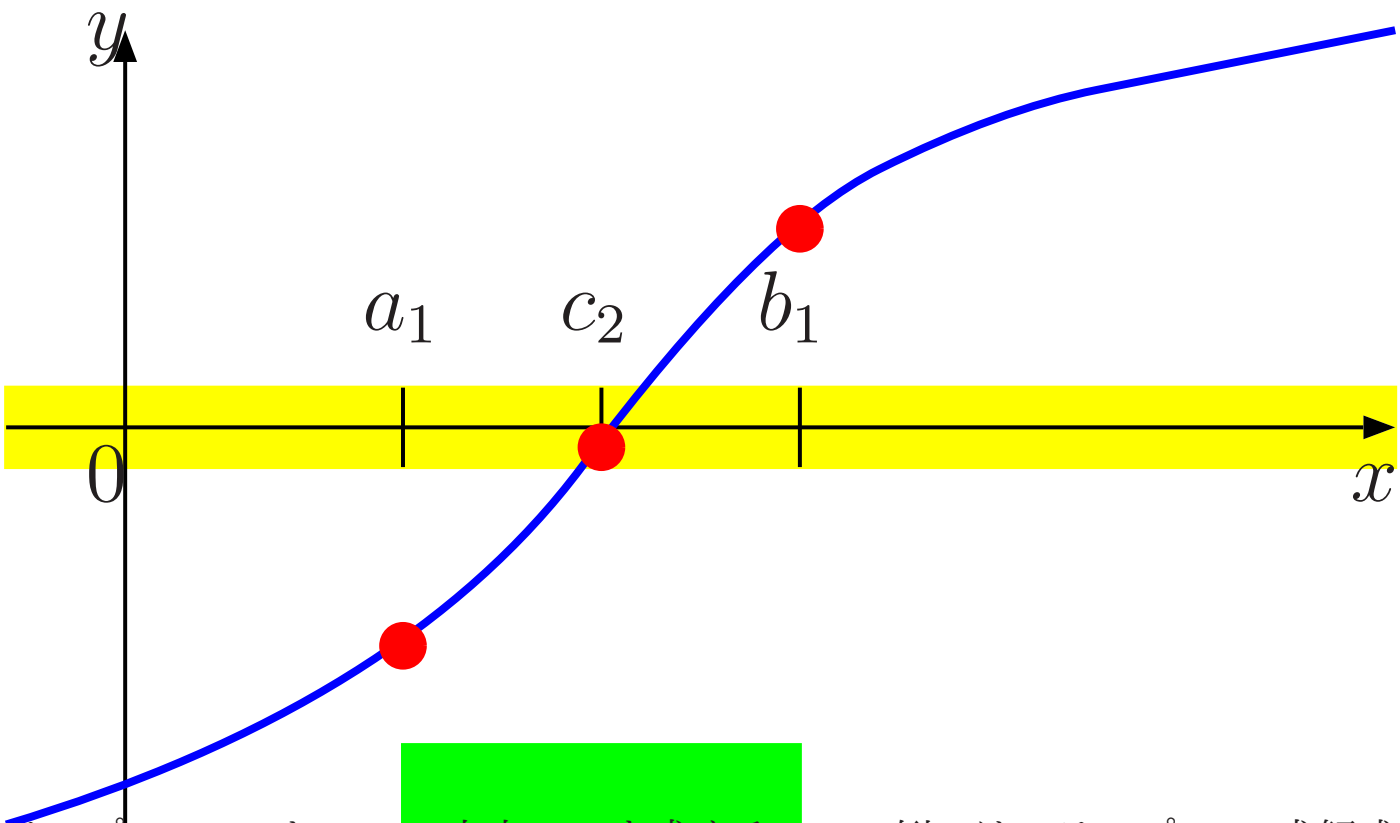


ステップ 1 前半. a_0 と b_0 の中点 c_1 を求める. $f(c_1) > 0$ だから $\cdots$



解を探索する区間は半分になる.

ステップ 1 後半. a_1 と b_1 を図のように取る.



ステップ 2. a_1 と b_1 の中点 c_2 を求める. この例ではステップ 2 で求解成功.

二分法 (5)

- 一般には, もっとずっと多くの繰り返しが必要だが, 考え方は同じ.
- 二分法に関する残った議論は次回に回す.