電 301 数値解析
第 2 回

二分法の復習 (1)

まず $f(x)$ が単調増加の場合を考える.

- 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. この時点で, 解が存在する区間 $[a_0, b_0]$ はわかっているが, 解がその区間のどこにあるかは不明.

二分法の復習 (2)

ステップ 1, ステップ 2, … というふうに, 以下の計算を誤差が許容範囲におさまるまで繰り返す.

- ループ (ステップ k): 解を含む区間 $[a_k, b_k]$ に対し …
  - $[a_k, b_k]$ の中点 $c_{k+1} = (a_k + b_k)/2$ を求める.
  - $f(c_{k+1}) > 0$ なら $[a_{k+1}, b_{k+1}] = [a_k, c_{k+1}]$
  - $f(c_{k+1}) < 0$ なら $[a_{k+1}, b_{k+1}] = [c_{k+1}, b_{k+1}]$

二分法の復習 (3)

まず $f(x)$ が単調減少の場合を考える.

- 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. この時点で, 解が存在する区間 $[a_0, b_0]$ はわかっているが, 解がその区間のどこにあるかは不明.

二分法の復習 (4)

ステップ 1, ステップ 2, … というふうに, 以下の計算を誤差が許容範囲におさまるまで繰り返す.

- ループ (ステップ k): 解を含む区間 $[a_k, b_k]$ に対し …
  - $[a_k, b_k]$ の中点 $c_{k+1} = (a_k + b_k)/2$ を求める.
  - $f(c_{k+1}) < 0$ なら $[a_{k+1}, b_{k+1}] = [a_k, c_{k+1}]$
  - $f(c_{k+1}) > 0$ なら $[a_{k+1}, b_{k+1}] = [c_{k+1}, b_{k+1}]$

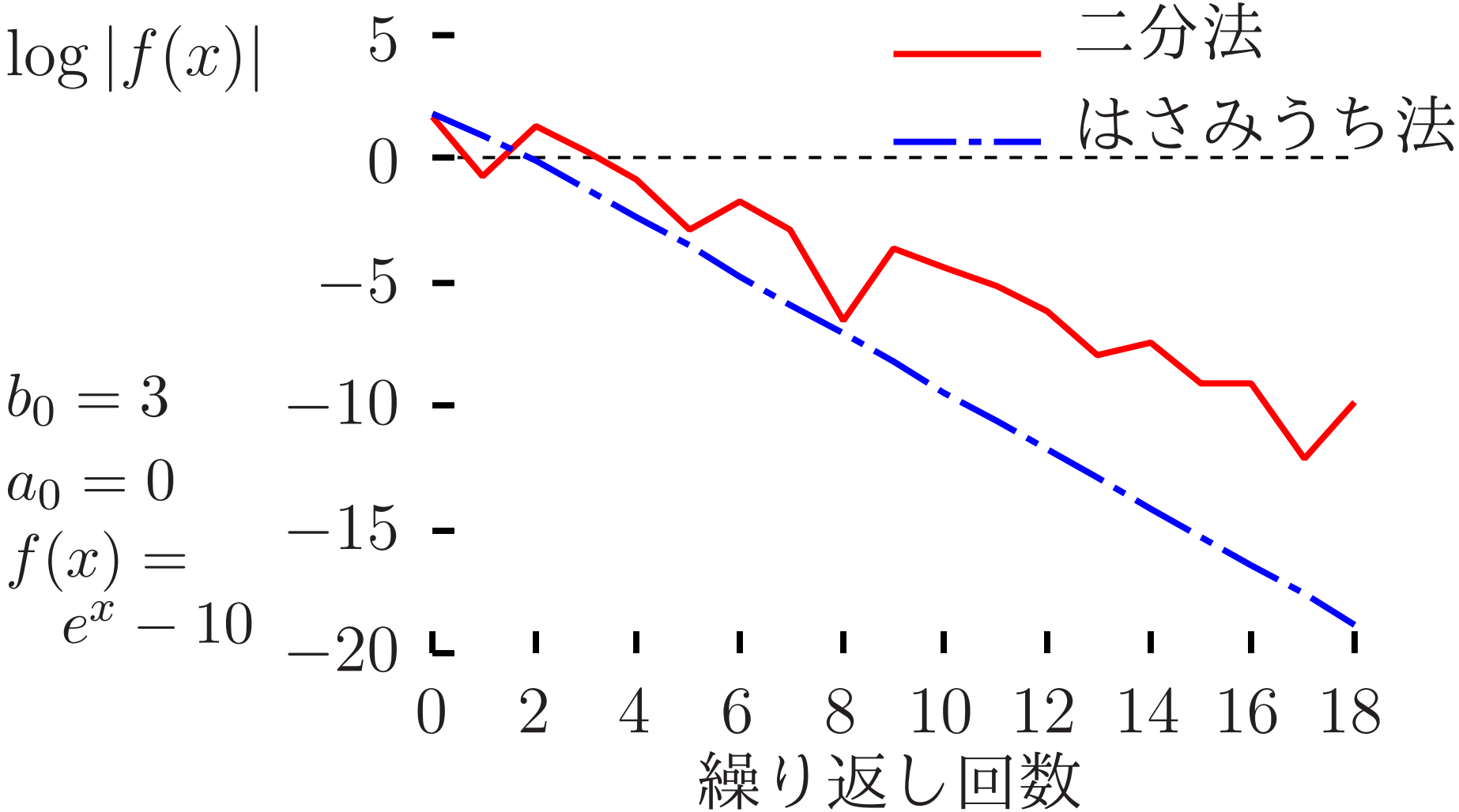
二分法 (続き)(1)

- c_{k+1} を $[a_k, b_k]$ の中点とするかわりに, 点 $(a_k, f(a_k$ と点 $(a_b, f(b_k))$ を結ぶ線分が x 軸と交わる点とする方法もある (はさみうち法)

- はさみうち法は, c_{k+1} を求める方法が 2 分法と違うだけで, 他の部分は同じ.

二分法 (続き)(2)

- 実は, はさみうち法はニュートン法の近似であるセカント法とよく似ている.
- プログラムは教科書にある. 著者のホームページからダウンロードできるので, 試してみるとよい.
- 次のページに数値例を示す.


$\log|f(x)|$
5
0
−5
−10
−15
−20
二分法
はさみうち法
$b_0 = 3$
$a_0 = 0$
$f(x) = e^x - 10$
0 2 4 6 8 10 12 14 16 18
繰り返し回数

二分法 (続き)(3)

- 教科書では $f(x, y) = 0$, $g(x, y) = 0$ という 2 変数非線形連立方程式を二分法で解く手法が解説されているが …

- この方法は, $f(x, y)$ と $g(x, y)$ のどちらか一方がある変数について解けることを前提にしている

二分法 (続き)(4)

- たとえば $f(x, y) = 0$ が $y = h(x)$ と書き直せるなら, 解くべき非線形方程式は $g(x, h(x)) = 0$ となり, 1 変数の二分法が適用できる.

- 数学的には, $f(x, y) = 0$ を局所的に $y = h(x)$ と書き直せるための条件はわかっている.

二分法 (続き)(4)

- 数値解析の観点から言うと, $f(x, y) = 0$ という式から $y = h(x)$ という式を解析的に求めるのは, 余程簡単な問題でなければ無理.

- というわけで 2 変数の二分法についてはこれ以上説明しない.

計算結果の表記について

- 結果を表記するときには精度に注意
- 倍精度の場合は仮数部の最小桁は $2^{-52} \simeq 2.2 \times 10^{-16}$ なので, 仮数部の有効数字は 15 桁程度.
- 計算の過程でさらに精度が落ちていることもあり得る.
- コンピュータが表示した数字の桁を全部記録することには必ずしも意味はない.

Newton 法 (1)

- Newton 法には, 非線形最小化問題を解くためのものと, 非線形方程式を解くためのものがある.

- 教科書で取り扱われているのは非線形方程式を解くためのニュートン法のみ. 教科書にある方から説明する.

Newton 法 (2)

- Newton 法は, 非線形方程式を線形近似し, 近似した線形方程式を解く手順を繰り返す方法.
- 単純なニュートン法は, 初期値が真の解に十分近くないと発散することがある.
- 今日では, 発散を防ぐ方法が色々と知られている.

Newton 法 (3)

- 微分可能な 1 変数実数値関数関数 f に関する方程式 $f(x) = 0$ を解きたい.
- Newton 法が適用できるためには, f の導関数が零にならないことが必要.
- f の導関数が零になる場合には, 他の解法 (二分法など) を使う必要がある.

Newton 法 (4)

- 初期値 x_0 が与えられているものとする. $f(x_0) = 0$ なら x_0 が解であり, これ以上計算する必要はない. そこで, $f(x_0) \neq 0$ の場合についてのみ考える.

Newton 法 (5)

- 数値解を x_0 から $x_0 + d$ に変更したとき, テイラー展開によって関数値の変化を近似すると, $f(x_0 + d) \simeq f(x_0) + f'(x_0)d$ となる. 線形近似の精度が良ければ, $d = -\dfrac{f(x_0)}{f'(x_0)}$ とすることにより, 関数値は零に近付く筈である. この手順を繰り返すのが Newton 法.

Newton 法 (6)

アルゴリズムをきちんと書き下すと次のようになる.

- 初期化: 初期値 x_0 と誤差の許容値 $\varepsilon > 0$ を定める. $k = 0$ とする.
- ループ: $|f(x_k)| < \varepsilon$ であれば終了. そうでない場合には,

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

とし, k に 1 を加えてループの先頭に戻る.

Newton 法 (7)

- 教科書には, x_{k+1} と x_k の差が大きすぎるときには発散と見做して初期値を選び直す部分と, 繰り返し回数が一定を超えたら発散と見做して初期値を選び直す部分が含まれる.

- 単純なニュートン法を使う場合には, 上記のような工夫が必要.

Newton 法 (8)

- 単純な Newton 法には, 解が高速に得られる (2 次収束) という長所がある一方で, 初期値の取り方しだいで発散することがあるという問題がある.

- 二分法と異なり, Newton 法は素直に多変数の問題に拡張できる.

Newton 法 (9)

- Newton 法の収束に関する数学の定理は繁雑なので, この講義では取り扱わない.

- 今日では, 1 変数, 多変数の場合の双方について, 関数 f が一定の条件を満たすとき, 初期値によらず解に収束するニュートン法が知られている (1990 年前後に確立).

Newton 法 (10)

- 大域的に収束する Newton 法を使えば, 初期値を選び直す作業は不要.
- この手法は直線探索法と信頼領域法に大別される. いずれも複雑なのでこの講義では名前を紹介するだけだが, 数値計算ソフトには実装されていることが多い.

セカント法 (1)

- Newton 法において, $f'(x_k)$ を $\dfrac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$ で近似したものが, セカント法. 微分の定義に戻って考えれば, 上記の式が $f'(x_k)$ の近似になっていることがわかる.

- 関数の微分の評価が難しいときにはセカント法を使う.

セカント法 (2)

- 式をよく見るとわかるが, セカント法は, はさみうち法と似ている.
- セカント法の計算効率は Newton 法に劣る. これは微分の計算が困難なときに使う方法.

セカント法 (3)

- このタイプの解法は, Inexact Newton Method あるいは pseudo-Newton Method という形で多変数に拡張される. 今日では大域的に収束する解法が知られていることも Newton 法と同じ.

高次解法

- Newton 法はテイラー展開の 1 次の項までを使う.
- 高次の項を使えばより高速な解法が得られる「かも」. 実際にそういう解法はあるが, あまり使われないので, 説明を省略する. 興味がある者は教科書を参照.

複素解

- これまでの議論では実数解のみを対象にしてきた.
- 複素解を求めることも非線形方程式を解くことの一種なのであるが, 慣例的に, 非線形方程式の複素解を求める方法は, 代数方程式の解法と呼ばれることがある.
- 代数方程式の解法については次回.

多変数の Newton 法 (1)

- $\boldsymbol{x}$ を n 次のベクトル, $\boldsymbol{f}(\boldsymbol{x})$ を n 次のベクトル値関数とする.
- 多変数の非線形方程式を解くということは, $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ という非線形連立方程式を解くということである ($\boldsymbol{0}$ は零ベクトル).

多変数の Newton 法 (2)

多変数の Newton 法は以下の通り.

- 初期化: 初期値 $\boldsymbol{x}_0$ と誤差の許容値 $\varepsilon > 0$ を定める. $k = 0$ とする.
- ループ: $|\boldsymbol{f}(\boldsymbol{x}_k)| < \varepsilon$ であれば終了. そうでない場合には, $\boldsymbol{J}(x_k)\boldsymbol{d} = -\boldsymbol{f}(\boldsymbol{x}_k)$ という線形方程式を解いて d を求め, $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + d$ とし, k に 1 を加えてループの先頭に戻る. ただし, $\boldsymbol{J}(x_k) = \left.\dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_k}$.

多変数の Newton 法 (3)

- 1 変数の場合は f の導関数 $f'(x_k)$, 多変数の場合は $\boldsymbol{f}$ の Jacobi 行列 $\boldsymbol{J}(x_k)$
- 1 変数の $1/f'$ には $(\boldsymbol{J}(x_k))^{-1}$(逆行列) が対応するが …
- 逆行列を直接求めるのではなく線形方程式 $\boldsymbol{J}(x_k)\boldsymbol{d} = -\boldsymbol{f}(\boldsymbol{x}_k)$ を解く.

多変数の Newton 法 (4)

- 教科書には 2 変数の場合のみ書かれているが, 変数がいくつあってもやることは同じ.
- Jacobi 行列 $\boldsymbol{J}(x_k)$ が正則でない場合には使えない.

多変数の Newton 法 (5)

- 単純な Newton 法が収束するか否かは初期値に依存する．直線探索法や信頼領域法を使えば初期値によらず収束するアルゴリズムが作れる．

多変数の Newton 法 (6)

- 多変数の問題では, Jacobi 行列を求めるだけでも CPU の負荷が高すぎる場合がある．このような場合には, Jacobi 行列の近似が用いられる (1 変数のセカント法に対応).

- 近似の方法には様々なものがあり, 今日では大域的に収束する手法が知られている.

Newton 法で関数の最小値を求める (1)

- 関数 $f(\boldsymbol{x})$ が最小になる点 x を求める問題を考える.
- 最大値を求める問題も同様に取り扱える.
- $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ は n 次のベクトルで, f は実数値関数とする.

Newton 法で関数の最小値を求める (2)

- f は 2 階連続微分可能と仮定する.
- $\boldsymbol{p}(\boldsymbol{x}) = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})$ とする.
- $\boldsymbol{H}(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots \frac{\partial^2 f}{\partial x_1 x_n} \\ \vdots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$ とする.

Newton 法で関数の最小値を求める (3)

- 行列 $\boldsymbol{H}$ が x_* で正値なら f は x_* で最小値となり, $\boldsymbol{H}$ が x_* で負値なら f は x_* で最大値となることが示せる.

- 以下の解説では f が最小値を持つことが前提であるので, $\boldsymbol{H}(\boldsymbol{x})$ は $\boldsymbol{x}$ を固定すると正定となることを仮定する.

Newton 法で関数の最小値を求める (4)

- 初期値を $\boldsymbol{x}_0$ とし, $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{d}$ とした場合 (すなわち解をベクトル $\boldsymbol{d}$ の分だけ動かした場合) の関数 f の値の変動を調べる.

- 記法の簡単のため, $\boldsymbol{p}_0 = \boldsymbol{p}(\boldsymbol{x}_0)$, $\boldsymbol{H}_0 = \boldsymbol{H}(\boldsymbol{x}_0)$ と書く.

Newton 法で関数の最小値を求める (5)

- Taylor 展開の 2 次の項まで取って近似すると,

$$
\begin{aligned}
f(\boldsymbol{x}) &\simeq f(\boldsymbol{x}_0) + \boldsymbol{p}_0\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T\boldsymbol{H}_0\boldsymbol{d} \\
&= f(\boldsymbol{x}_0) - \frac{1}{2}\boldsymbol{p}_0^T\boldsymbol{H}_0^{-1}\boldsymbol{p}_0 \\
&\quad + \frac{1}{2}(\boldsymbol{d} + \boldsymbol{H}_0^{-1}\boldsymbol{p}_0)^T\boldsymbol{H}_0(\boldsymbol{d} + \boldsymbol{H}_0^{-1}\boldsymbol{p}_0)
\end{aligned}
$$

Newton 法で関数の最小値を求める (6)

- $\boldsymbol{H}_0$ は正定であると仮定したから, $f(\boldsymbol{x})$ の近似値は $\boldsymbol{d} = -\boldsymbol{H}_0^{-1}\boldsymbol{p}_0$ としたとき最小になる.
- Newton 法で関数の最小値を求めるアルゴリズムは, 上記を繰り返す.
- 記法の簡単のため, $\boldsymbol{p}_k = \boldsymbol{p}(\boldsymbol{x}_k)$, $\boldsymbol{H}_k = \boldsymbol{H}(\boldsymbol{x}_k)$ と書く.

Newton 法で関数の最小値を求める (7)

多変数の Newton 法は以下の通り.

- 初期化: 初期値 $\boldsymbol{x}_0$ と誤差の許容値 $\varepsilon > 0$ を定める. $k = 0$ とする.
- ループ: $|\boldsymbol{p}(\boldsymbol{x}_k)| < \varepsilon$ であれば終了. そうでない場合には, $\boldsymbol{H}_0\boldsymbol{d} = -\boldsymbol{p}_0$ という線形方程式を解いて $\boldsymbol{d}$ を求め, $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + d$ とし, k に 1 を加えてループの先頭に戻る.

Newton 法で関数の最小値を求める (8)

- 終了条件には色々な取り方があるが, 上記のように $\boldsymbol{p}(\boldsymbol{x}_k)$ のノルムが一定以下になったら終了というのはひとつの考え方

- これは, 最小値に近付くほど接線の傾きが水平に近付くという性質を使っている.

Newton 法で関数の最小値を求める (9)

- 収束が局所的であることは非線形方程式に対する Newton 法と同じ.
- 最小化したい関数が 2 次関数なら, 1 回の計算で最小値が得られる.
- 直線探索や信頼領域法を用いることで大域的な収束解法が得られる.

Newton 法で関数の最小値を求める (10)

- 非線形方程式 $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ を解く問題を $\boldsymbol{f}^T(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x})$ の最小値を求める問題に書き直してから解くことがある.

- 数値計算の誤差に対応するためにはこの方が有利なこともある.

Scilab で非線形方程式を解く (1)

- fsolve という関数を使う.
- アルゴリズムは修正 Powell 混合法 (典拠はオンラインマニュアル).
- fsolve の使い方は教科書 6 ページ．初期値依存性があるので注意．特に解が複数ある場合は要注意.

Scilab で非線形方程式を解く (2)

- fsolve は解きたい関数の Jacobian を与えても与えなくても解けるが, 一般に Jacobian を与えた方が高速.

Scilab で非線形方程式を解く (3)

- 新しいアルゴリズムを開発する際には自分でプログラムを書く必要があり, かつ既存のアルゴリズムとの比較のためにそちらもプログラムを書く必要がある.
- 上記の用途では計算過程の情報が必要になるので fsolve は不向き.

Scilab で非線形方程式を解く (4)

- 新しいアルゴリズムの開発等の特別な理由がない限り, fsolve を使うべき.
- 自分でプログラムを書くと, 繰り返しの制御構造 (for 文や while 文など) の実行自体で時間がかかったりする. スクリプト型のプログラムの処理は一般に遅い.

Scilab で非線形最適化問題を解く (1)

- 組み込み関数 optim か fminsearch を使う.
- optim および fminsearch 解くのは制約条件なしの最小化問題である. 関数 f を最大化したいときには, かわりに関数 $-f$ を最小化すればよい.

Scilab で非線形最適化問題を解く (2)

- optim を使う場合には, 最小化したい関数に加えて, その勾配ベクトルと終了フラグを与える必要がある.

- 次ページに使用例を示す.

Scilab で非線形最適化問題を解く (3)

```
function [f,g,ind]=costFn(x,ind)
f=x^2+x+1; //最小化したい関数
g=2*x+1; //その勾配ベクトル
endfunction
// 変数 ind は宣言するだけでよい.

x0=0; //初期値
[fopt,xopt]=optim(costFn,x0);
```

Scilab で非線形最適化問題を解く (4)

- 先の例のようにすると, fopt に関数の最適値が, xopt に最適値を与える変数ベクトルの値が格納される.
- optim の変数はスカラーでもベクトルでもよい.
- Scilab では//以下の部分は注釈と解釈される.

Scilab で非線形最適化問題を解く (5)

- fminsearch を使う場合には, 最小化したい関数のみを指定すればよい.
- 次ページに使用例を示す.

Scilab で非線形最適化問題を解く (6)

```
function y=costFn(x)
y=x^2+x+1;
endfunction

x0=0;
[xopt,fopt,exitflag]=fminsearch(costFn,x0);
```

Scilab で非線形最適化問題を解く (7)

- Scilab の多くの関数では, 引数と返却値の数は可変である.
- 引数の与え方を変えることで挙動が変わる.
- 返却値を受け取る変数の型を変えると受け取る情報が変わる.
- 詳細についてはオンラインマニュアルを参照.

計算速度の比較について

- CPU 依存の部分もあるため公平な比較は難しい
- 計算時間以外に, 収束までの繰り返し回数, 関数評価の回数などが用いられる

Test Problems

検索をかければテスト問題を公開しているサイトが見付かる. たとえば以下のようなサイト (2015.10.13 に確認, リンク切れに注意).

http://www.sfu.ca/~ssurjano/optimization.html
http://www.mat.univie.ac.at/~neum/glopt/test.html

使用例

- 配付資料に説明および例を示す.
- Scilab の基本的な使い方については, 各自で教科書付録 A(192–213 ページ) を読んでおくこと. 自分で Scilab をインストールし試してみることが望ましい. 次回以降の講義では付録 A の内容は既知であると仮定する.

その他注意事項

- Scilab は線形計画問題, 2 次計画問題, 非線形最小 2 乗問題, 非線形方程式, 制約条件なしの最適化問題を解くための手法は用意されているが, たとえば制約条件付き非線形最適化問題を解く手法は用意されていない.

- Scilab に解法が用意されていない問題を解きたいときには, 自分でプログラムを組むか, 別途適切なソフトを探す必要がある.

速度比較 (1)

非線形方程式$x+\sin(x)-1=0$を2分探索, Newton法のプログラムを使って解いた場合とfsolveを使って解いた場合の速度の比較を示す．実行環境は以下の通り．

```
Intel Core i5-4690 3.50GHz
32GB of memory
Windows7 Professional 64bit Service Pack 1
Scilab 5.5.2 for Windows 64bit
```

速度比較 (2)

- 二分法の初期値は $[0, 2]$, それ以外はすべて 1
- 自作プログラムでは $x + \sin(x) - 1$ の絶対値が 10^{-9} 未満になった時点で終了.
- 実行時間は Scilab の timer 関数で計測.
- プログラム実行時に画面に表示される `1.22D-10` などといった文字列は, 1.22×10^{-10} などといった意味になる. `1.22E-10` なども同様.

速度比較 (3)

1000 回問題を解いた場合の平均所要時間:

二分探索 自作プログラム	4.68×10^{-4} 秒
Newton 法 自作プログラム	1.87×10^{-4} 秒
fsolve, Jacobian なし	6.24×10^{-5} 秒
fsolve, Jacobian あり	4.68×10^{-5} 秒

速度比較 (4)

配付資料例 3-5 と例 3-6 を 1000 回実行したときの平均時間の比較

optim	7.80×10^{-5} 秒
fminsearch	0.108 秒