システム制御最適化特論

前期前半 月 5, 6限 14:00-16:10 5号館 第16講義室

担当:平田 健太郎

6/24 第2回 内点法

講義日程（予定）

6/17　第１回　最適化問題と線形計画法（LP）
6/24　第２回　内点法
7/1　第３回　最短経路問題と動的計画法（DP）
7/8　第４回　最適制御
7/18* 第５回　二次計画法(QP)とモデル予測制御(MPC)
7/22　第６回　凸解析と線形行列不等式
7/29　第７回　線形行列不等式(LMI)による制御系解析・設計
8/5　第８回　非線形最適化

* irregular

最適化問題
Optimization Problem

$$\min f(x)$$

$$\text{subject to } x \in S.$$

$f(x), S$ の取り方によって多様な問題を含む

↓

簡単な例: $x \in \mathbb{R}^n$, $f(x)$ および制約が1次式（線形）

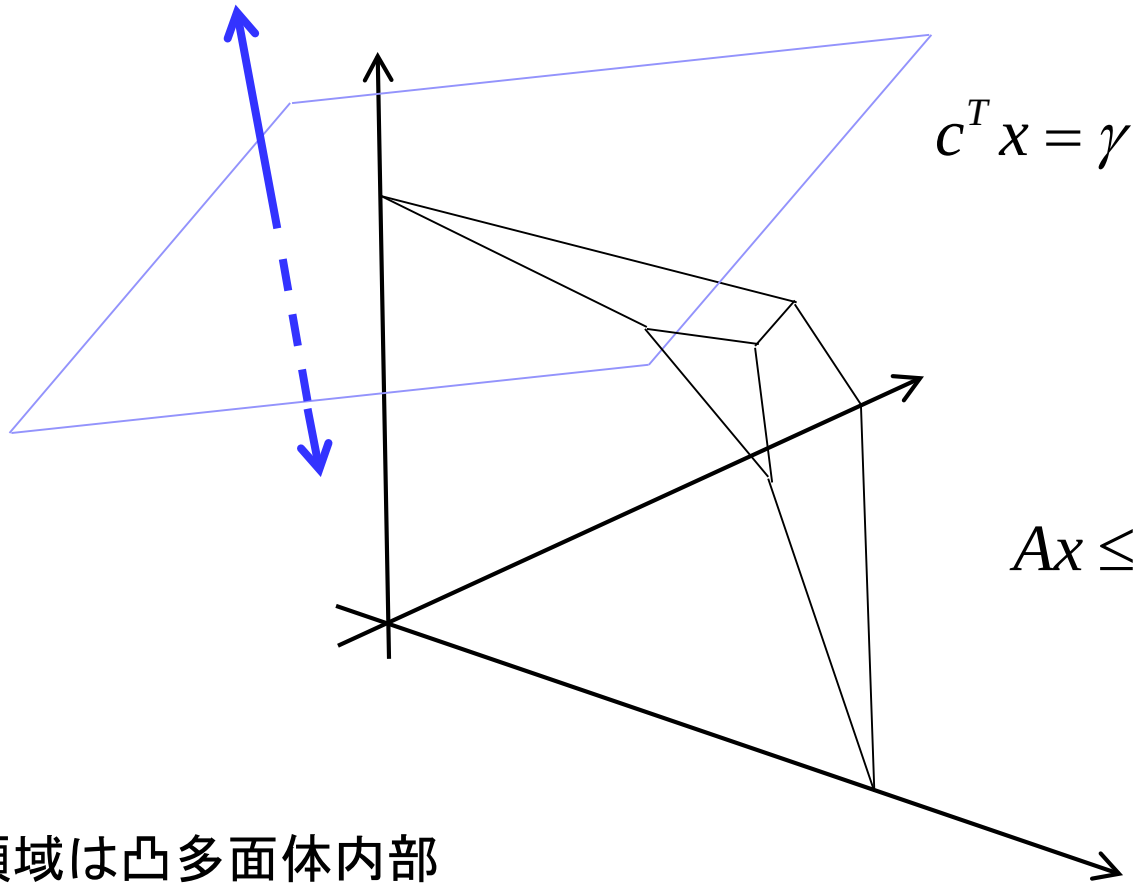
↓

線形計画問題 (Linear Programming; LP)

↓

スタンダードな解法（シンプレックス法）

Idea of Simplex Method



実行可能領域は凸多面体内部

平面の切片の最小化⇒最適解は必ず頂点

頂点は変数のどれかを 0 に固定し,線型方程式を解くことで求まる.

目的関数値が効率的に減少するように0 に固定する変数を選ぶ.

基本となる式: $Bx_B + Nx_N = b$

- 基底変数 x_B を非基底変数 x_N を用いて表現

$$x_B = B^{-1}b - B^{-1}Nx_N$$

- c も c_B, c_N に分割
- 目的関数の値

$$c^T x = c_B^T x_B + c_N^T x_N = c_B^T B^{-1} b + (c_N^T - c_B^T B^{-1} N) x_N$$

相対コスト係数

相対コスト係数の中に負の要素があれば, 対応する非基底変数 x_N の要素を 0 から正の値に変化させることによって目的関数値が減少する.

↓

相対コスト係数が全て非負ならば, ピボット操作によって目的関数値が減少することはない.

↓

現状の基底の取り方が最適

1. $x = (2,3,0,0)^T$, $x_B = (x_1, x_2)^T$, $x_N = (x_3, x_4)^T$ について **check it!**

$$\min \; -x_1 - x_2$$

$$\text{s.t.} \quad 3x_1 + 2x_2 + x_3 = 12$$
$$x_1 + 2x_2 + x_4 = 8$$
$$x_i \geq 0, \forall i.$$

$$\min c^T x$$

$$\text{s.t.} \quad Ax = b$$
$$x \geq 0.$$

$$x = [x_1, x_2, x_3, x_4]^T$$

$$A = \begin{bmatrix} 3 & 2 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 12 \\ 8 \end{bmatrix}$$

$$c^T = [-1 \quad -1 \quad 0 \quad 0]$$

$B =$ [ ], $N =$ [ ], $c_B^T =$ [ ], $c_N^T =$ [ ]

$c_N^T - c_B^T B^{-1} N =$ [ ]

- 相対コスト係数の中に負の要素が複数ある場合，絶対値の大きいものの方が，減少の割合が大きい．(有望かもしれない)

- 他の非基底変数は0のまま，対応する非基底変数x_kの値を0から増加させる．基底変数x_Bを

$$x_B = B^{-1}b - B^{-1}a_k x_k$$

と変化させると制約条件$Ax = b$は満たされる．(a_kはx_kに対応するAの列)

$$\because Ax = Bx_B + Nx_N = b$$

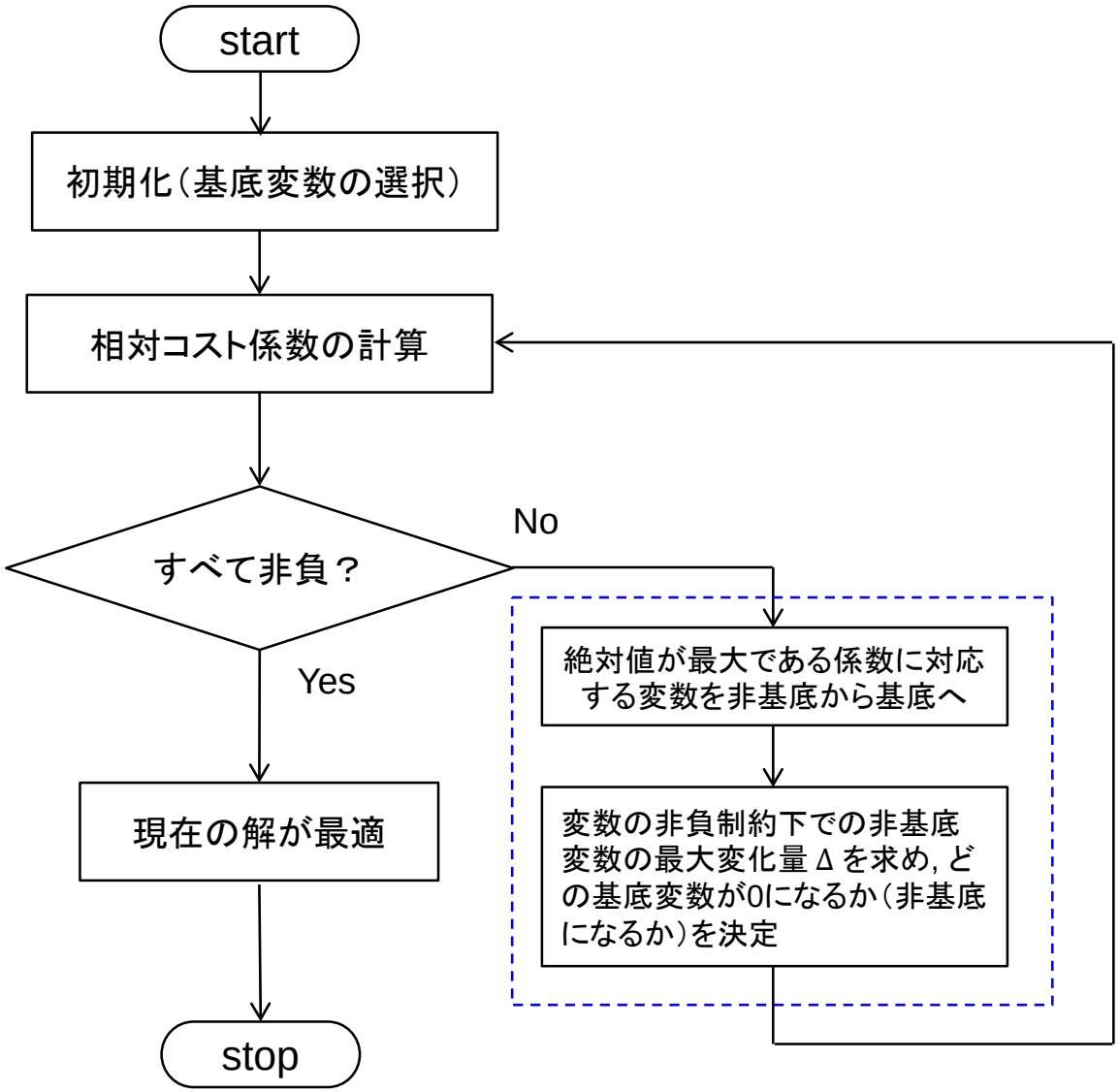
- $\bar{b} = B^{-1}b$，$y = B^{-1}a_k$とすると，非基底変数x_kを最大

$$\Delta = \min\left\{\bar{b}_i/y_i \mid y_i > 0\right\}$$

まで増加させても非負条件$x \geq 0$は満たされる．

- x_kをΔまで増加させ，新たに0になった変数を非基底変数とする．

- 最適条件が満たされるまで繰り返す


start
初期化（基底変数の選択）
相対コスト係数の計算
すべて非負？
No
Yes
絶対値が最大である係数に対応する変数を非基底から基底へ
変数の非負制約下での非基底変数の最大変化量 Δ を求め，どの基底変数が0になるか（非基底になるか）を決定
現在の解が最適
stop
基底/非基底変数の入れ替え（ピボット操作）

- 例題:

最適解を得るまでの計算を自分でやってみる.

例題

$$\begin{aligned} \min \quad & -x_1 - x_2 \\ \text{s.t.} \quad & 3x_1 + 2x_2 + x_3 = 12 \\ & x_1 + 2x_2 + x_4 = 8 \\ & x_i \geq 0, \forall i. \end{aligned}$$

まず $\boldsymbol{x}$, $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$ を用いた表現に直す

[第1反復]

頂点6: $x = (0, 0, 12, 8)^T$ から始めよ

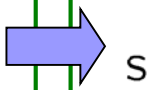
a) B, N, c_B, c_N の計算 $\rightarrow$ 相対コスト係数

b) 新たな基底変数候補の選択

c) △の計算，新たな実行可能基底解の設定

$$\min \ -x_1 - x_2$$

$$\text{s.t.} \quad 3x_1 + 2x_2 + x_3 = 12$$
$$x_1 + 2x_2 + x_4 = 8$$
$$x_i \geq 0, \forall i.$$



$$\min c^T x$$

$$\text{s.t.} \quad Ax = b$$
$$x \geq 0.$$

$$x = [x_1, x_2, x_3, x_4]^T$$

$$A = \begin{bmatrix} 3 & 2 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 12 \\ 8 \end{bmatrix}$$

$$c^T = [-1 \ \ -1 \ \ 0 \ \ 0]$$

[第1反復] $x = (0, 0, 12, 8)^T$ から始める

$$x_B = (x_3, x_4)^T, \ x_N = (x_1, x_2)^T$$

$$A = \left[\begin{array}{cc|cc} 3 & 2 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{array}\right] \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ N = \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix}$$

$$c^T = [-1 \ \ -1 \ \ 0 \ \ 0] \qquad c_B^T = [0 \ \ 0], \ c_N^T = [-1 \ \ -1]$$

相対コスト係数: $c_N^T - c_B^T B^{-1} N = [-1 \ \ -1]$

いずれも負なのでまだコストは小さくできる

相対コスト係数: $c_N^T - c_B^T B^{-1} N = [-1 \quad -1]$

絶対値が大きい＝コスト改善効果大

こちらを選ぼう

相対コスト係数はx_Nに対する係数であることに注意

現在の分割ではx_Nの１番目はx_1に対応

x_1を基底変数へ．非基底変数になるのはどれか？

$$\bar{b} = B^{-1}b = \begin{bmatrix} 12 \\ 8 \end{bmatrix}, \quad y = B^{-1}a_k = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

x_1に対応するA(またはN)の列

(x_1を０から増加させるときの変化分を調べるので)

$$\bar{b} = \begin{bmatrix} 12 \\ 8 \end{bmatrix}, \; y = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

「$y_i > 0$であるような行iに関して，$\bar{b}_i/y_i$の最小値をとり，それをΔとおく」という意味

$$\Delta = \min \left\{ \bar{b}_i / y_i \mid y_i > 0 \right\}$$

今各行ともy_iは正なので，4と8の最小値をとって$\Delta = 4$

x_1を4まで増加させるとき，x_Bの1番目が先に0になる

現在の分割ではx_Bの1番目はx_3に対応

x_3が非基底変数に．次反復の基底変数は$(x_1, x_4)^T$

[第2反復]

$$x_B = (x_1, x_4)^T,\ x_N = (x_2, x_3)^T$$

$$A = \begin{bmatrix} 3 & 2 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix},\ B = \begin{bmatrix} 3 & 0 \\ 1 & 1 \end{bmatrix},\ N = \begin{bmatrix} 2 & 1 \\ 2 & 0 \end{bmatrix}$$

$$c^T = [-1 \quad -1 \quad 0 \quad 0],\ c_B^T = [-1 \quad 0],\ c_N^T = [-1 \quad 0]$$

$$c_N^T - c_B^T B^{-1} N = [-1 \quad 0] - [-1 \quad 0]\frac{1}{3}\begin{bmatrix} 1 & 0 \\ -1 & 3 \end{bmatrix}\begin{bmatrix} 2 & 1 \\ 2 & 0 \end{bmatrix}$$

$$= [-1 \quad 0] - \frac{1}{3}[-2 \quad -1] = \left[-\frac{1}{3} \quad \frac{1}{3}\right]$$

現在の分割ではx_Nの１番目はx_2

[第2反復]（続）

$$x_B = (x_1, x_4)^T,\ x_N = (x_2, x_3)^T$$

x_2 が基底変数に入る

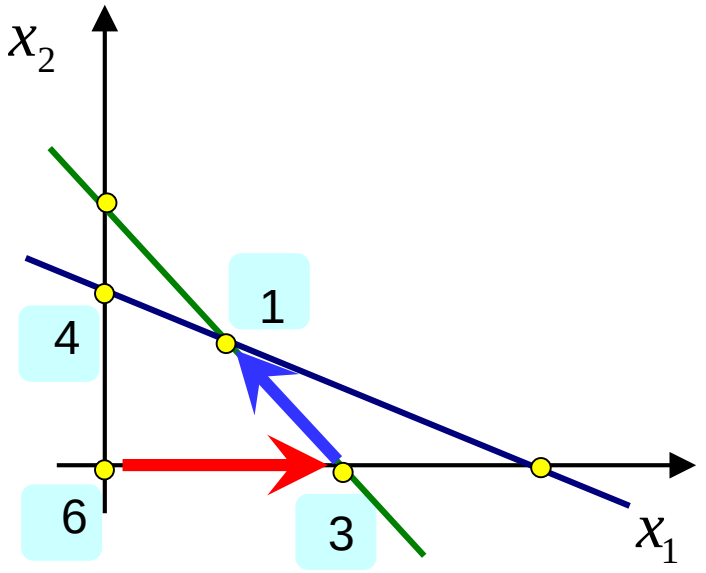
$$\bar{b} = B^{-1}b = \frac{1}{3}\begin{bmatrix} 1 & 0 \\ -1 & 3 \end{bmatrix}\begin{bmatrix} 12 \\ 8 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}, y = B^{-1}a_2 = \frac{1}{3}\begin{bmatrix} 1 & 0 \\ -1 & 3 \end{bmatrix}\begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 2/3 \\ 4/3 \end{bmatrix}$$

$$Ax = Bx_B + Nx_N = b$$

$$x_B = B^{-1}b - B^{-1}a_k x_k = \begin{bmatrix} 4 \\ 4 \end{bmatrix} - \begin{bmatrix} 2/3 \\ 4/3 \end{bmatrix} x_2$$

$$\Delta = \min\{\bar{b}_i / y_i \mid y_i > 0\} = 3$$

x_2を3まで増加させるとx_Bの下側が先に0になる.



⇒ x_4 が非基底変数に入る

次反復では $x_B = (x_1, x_2), x_N = (x_3, x_4)$

このとき，最適条件を満たすのは前回確認済み⇒次反復で終了.

線形計画問題の標準形

系統的な解法を考えるにあたって, 問題毎の定式化に差異があると不都合
(例: 最大化 or 最小化, 制約条件の与え方)

⇒ 標準形を定めて, その解法を与える

標準形

$$\min c^T x$$

$$\text{subject to } Ax = b, x \geq 0.$$

(1) 最小化

(3) 全ての変数に非負条件

(2) 等号制約

標準形でない例

$$\begin{aligned} \max \quad & -2x_1 + 5x_2 \\ \text{subject to} \quad & 4x_1 - 6x_2 = 30 \\ & 2x_1 + 8x_2 \leq 50 \\ & 7x_1 + 5x_2 \geq 10 \\ & x_1 \geq 0, \end{aligned}$$

$$\min c^T x$$

$$\text{subject to } Ax = b, x \geq 0.$$

- 最大化 ⇒ 係数cの符号反転
- x_2 に符号制約なし ⇒ 2つの正数の差で表現
- 2,3番目が不等号制約 ⇒ 両辺のギャップをある正数を導入して表現（スラック変数）

⇒ 標準形に変換できる

$$\max \quad -2x_1 + 5x_2$$

$$\text{subject to} \quad 4x_1 - 6x_2 = 30$$
$$2x_1 + 8x_2 \leq 50$$
$$7x_1 + 5x_2 \geq 10$$
$$x_1 \geq 0, x_2 : \text{free}$$

➡ 最小化でない

➡ 等式制約でない

➡ 等式制約でない

➡ 非負条件がない

非標準形の例

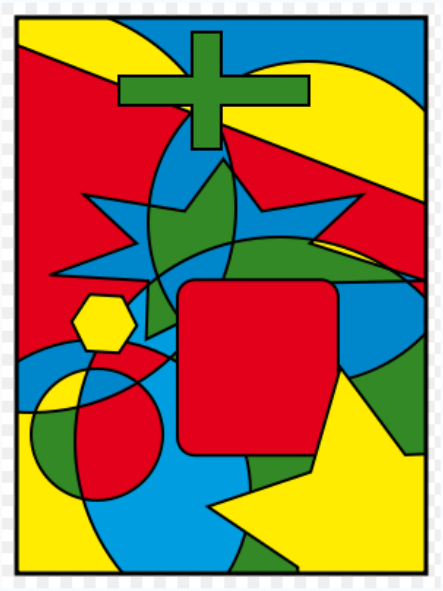
➡

Today's minutes paper

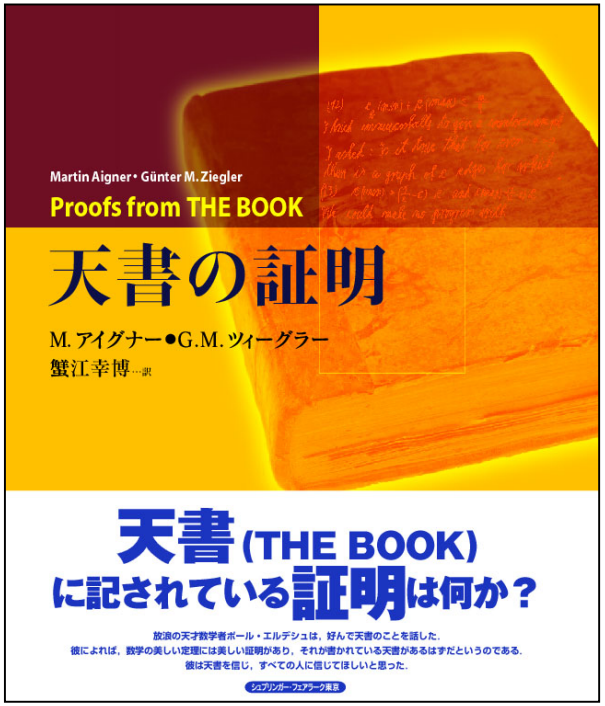
$$\min \quad c^T x$$
$$\text{subject to}$$
$$Ax = b, x \geq 0.$$

⇒標準形

Elegant versus elephant



四色定理



ポール・エルデシュ (1913-1996)

天書の証明　第1章

定理:　素数の数は無限個である.

証明:
（ユークリッド「原論」）

素数の有限集合$\{p_1, \ldots, p_r\}$に対して$n = p_1 p_2 \cdots p_r + 1$という数を考える.　この$n$は素因数$p$を持つ.　しかし，$p$は$p_i$のどれでもない.　pがp_iのどれかならば，pはnの約数でも$p_1 p_2 \cdots p_r$の約数でもあることになり，それゆえその差$n - p_1 p_2 \cdots p_r = 1$の約数でもあるあるが，これはあり得ない.だから，有限集合$\{p_1, \ldots, p_r\}$はすべての素数の集合にはなり得ないのである.

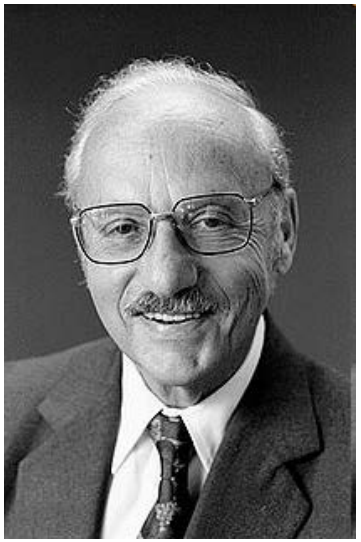
Paul Erdös （ポール・エルデシュ）

稀代の数学者。住む所を持たず、荷物はブリーフケース一つだけ。25か国以上を飛び回り、数学を最も愛した人。発表した論文は1500以上。ライバルは美しい証明を独り占めしている「SF　(Supreme Fascist = 神様)」だけ。子供をε（イプシロン）と呼び、女は「ボス」、男は「奴隷」と独特のエルデシュ語を使う。

これ以上に多産な数学者はレオンハルト・オイラーのみである。

エルデシュ数
Erdös Number

George Bernard Dantzig (Nov 8, 1914 – May 13, 2005) was an American mathematician, and Professor of Operations Research and of Computer Science at Stanford. He is known as the father of linear programming and as the inventor of the "simplex method“.



1975年にカントロビッチ（L.V. Kantorovich）とクープマンス（T.C. Koopmans）は線形計画に関する研究でノーベル経済学賞を得た. これに対して, Dantzigが受賞しなかったのは，公平を欠いているという見解もある.

（ちなみに数学に対してノーベル賞は与えられない. ）

【LPが解くべき現実的な問題の例】

デルタ航空: 全米166都市間を運行する400機の飛行機と7,000名の乗務員に関するスケジューリング問題 (できれば年間10M$の経費節減)

シンプレックス法は実用上十分な速度を有していたが
厳密には「速い」アルゴリズムではなかった. 問題の規模
によってはさらに速いアルゴリズムが望まれていた.

多項式時間アルゴリズム

その計算量が変数の数，制約条件の数などの問題の規模を表すパラメータの多項式関数で表されるもの．一般的に“効率的な”アルゴリズムとみなされる

- シンプレックス法　(1947 Danzig)

  反復回数は経験的に制約条件数の1.5倍から3倍. 実用上問題なし
  人工的なある種の問題では問題のサイズにつれて計算量が爆発的に増加. 多項式時間アルゴリズムでない.

- 楕円体法 (1979 Khachiyan)

  線形計画問題に対する初めての多項式時間アルゴリズム．実際の計算効率ではシンプレックス法に劣る.

Breakthrough in Problem Solving

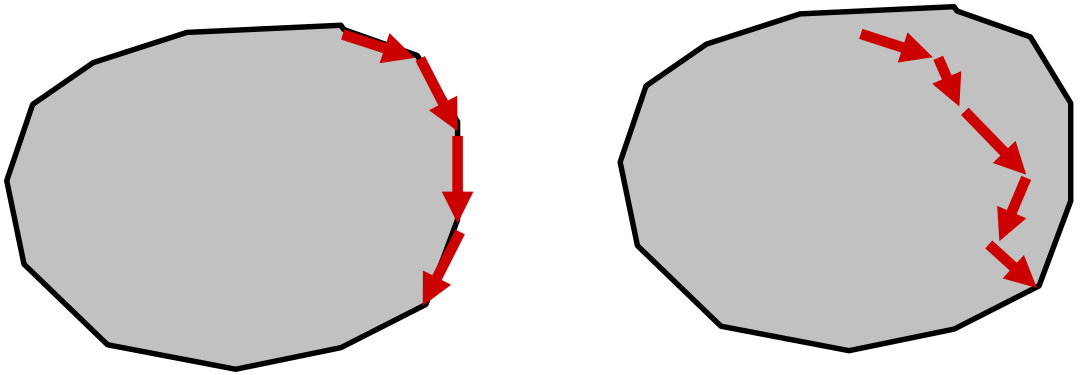
New York Times, November 19, 1984, Monday

A 28-year-old mathematician at A.T.&T. Bell Laboratories has made a startling theoretical breakthrough in the solving of systems of equations that often grow too vast and complex for the most powerful computers. The discovery, which is to be formally published next month, is already circulating rapidly through the mathematical world. It has also set off a deluge of inquiries from brokerage houses, oil companies and airlines, industries with millions of dollars at stake in problems known as linear programming. Faster Solutions Seen These problems are fiendishly complicated systems, often with thousands of variables. They arise in a variety of commercial and government applications, ranging from allocating time on a communications satellite to routing millions of telephone calls over long distances or whenever a limited, expensive resource must be spread most efficiently among competing users. And investment companies use them in creating portfolios with the best mix of stocks and bonds. The Bell Labs mathematician, Dr. Narendra **Karmarkar**, has devised a radically new procedure that may speed the routine handling of such problems by businesses and Government agencies and also make it possible to tackle problems that are now far out of reach.

- 内点法 (1984 Karmarkar)

  新しい多項式時間アルゴリズム．大規模な問題に対してはシンプレックス法をしのぐ方法として評価が定着している．

  ⇒ システム制御工学における LMI approach の普及にも関連



- アイデア

線形計画問題を考えるにあたって,「線形方程式を解く」という立場から離れて, 高速な「非線形方程式の求解法」からの接近を図った.

線形方程式の求解に, 高速な非線形方程式の求解法を用いるという例そのものではないが,非線形方程式の求解アルゴリズムがいかに速くなりうるかという例を見てみよう.

■ 非線形方程式の求解

例) $\sqrt{2}$の数値$(1.41421356\ldots)$を求めよ

⟺ $f(x) = x^2 - 2$ を満たす$x > 0$ を求めよ

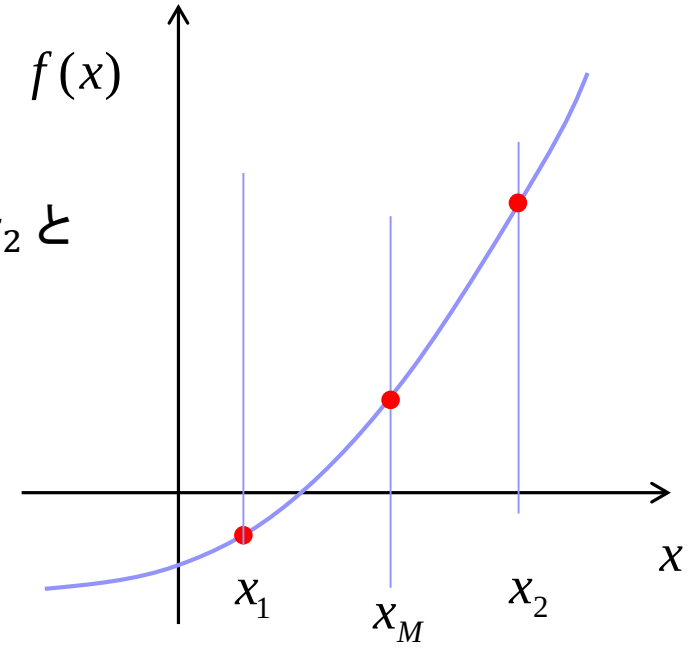
非線形方程式 $f(x) = 0$ の求解

Algorithm 1:

$f(x_1)f(x_2) < 0$ であるような2点 x_1, x_2 をとる.

$x_M = \dfrac{x_1 + x_2}{2}$ とし, $f(x_M)$ と同符号のx_1 または x_2 と x_M を入れ替える.

所望の近似値が得られるまで繰り返す.



二分法 (Bisection Method)

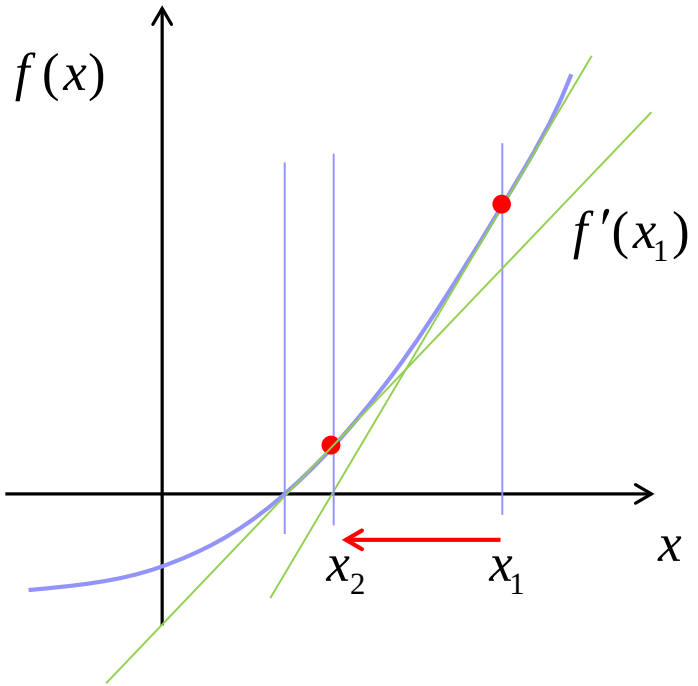
⇒ 収束は遅い

Algorithm 2:

初期値 x_1 に対して x_1 における $f(x)$ の接線を引く.

接線と x 軸の交点を x_2 とする.

所望の近似値が得られるまで繰り返す.



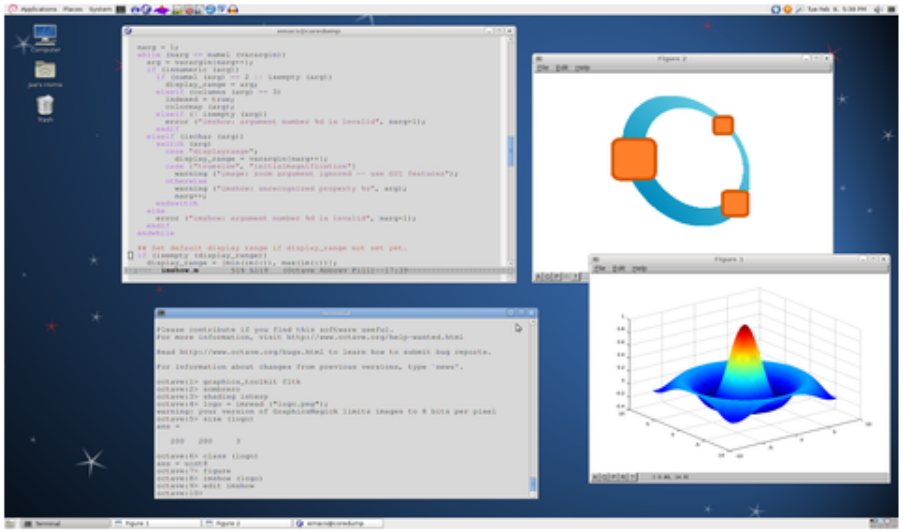
ニュートン法 (Newton's Method)

⇒ 収束は速い

非線形方程式の求解において, 線形近似した1次方程式を逐次解いて解に収束する点列を生成する反復法をNewton法という.

フリーな数値解析ソフトウェア Octave を使って数値計算してみる
（利用できる環境があればMatlabでもよい）

GNU Octave



GNU Octave is a high-level interpreted language, primarily intended for numerical computations. It provides capabilities for the numerical solution of linear and nonlinear problems, and for performing other numerical experiments. It also provides extensive graphics capabilities for data visualization and manipulation. Octave is normally used through its interactive command line interface, but it can also be used to write non-interactive programs. The Octave language is quite similar to Matlab so that most programs are easily portable.

Octave is distributed under the terms of the GNU General Public License.

Donate

Your donations help to fund continuing maintenance tasks, development of new features and the organization of Octave conferences.

Continue

Following the Continue link will take you to a Free Software Foundation page for payment processing.

Donate Bitcoins

Bitcoin donations also accepted at this address.

1E6HchBMX1EfiJQh$UanuF4VYKk552tEHF

Download GNU Octave

GNU Octave 3.8.2 was released August 6, 2014. Please read the announcement on the front page of the Octave web site.

GNU/Linux systems

Executable versions of Octave for GNU/Linux systems are provided by the individual distributions. Distributions known to package Octave include: Debian, Fedora, Gentoo, and SuSE. These packages are created by volunteers. The delay between an Octave source release and the availability of a package for a particular GNU/Linux distribution varies. The Octave project has no control over that process.

BSD systems

Executable versions of Octave for BSD systems are provided by the individual distributions. Both FreeBSD and OpenBSD have Octave packages. These packages are created by volunteers. The delay between an Octave source release and the availability of a package for a particular GNU/Linux distribution varies. The Octave project has no control over that process.

OS X

The Wiki has some instructions for installing Octave on OS X systems.

Windows

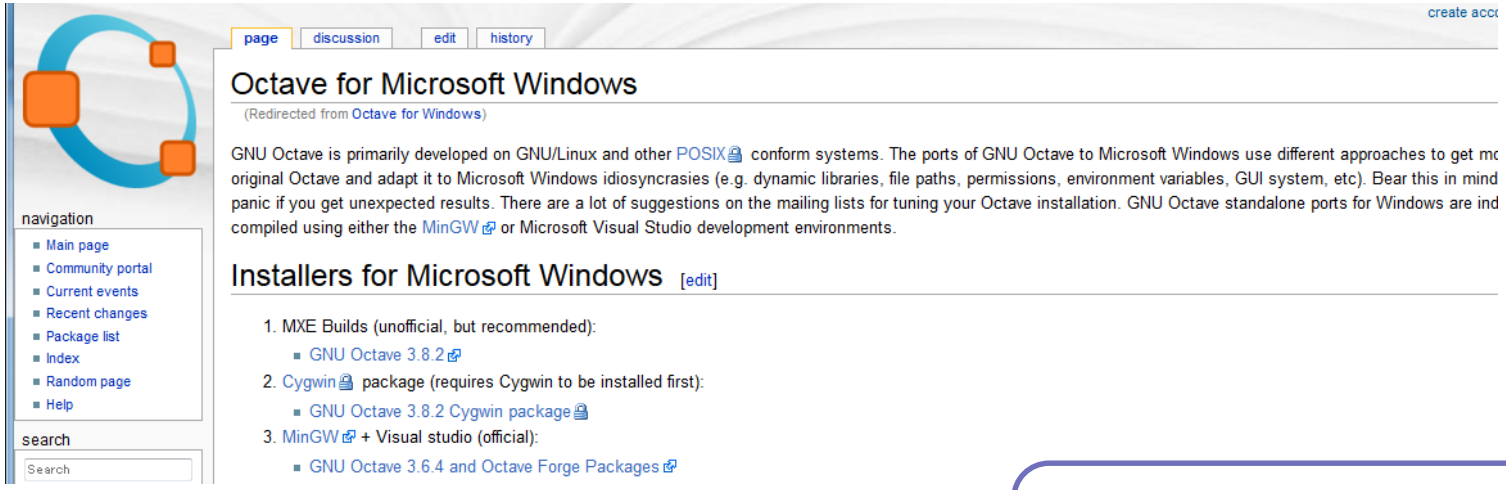
Cygwin

There is an Octave package for Cygwin.

MinGW

The Wiki has some instructions for installing Octave on Windows systems.

Sources



page | discussion | edit | history

Octave for Microsoft Windows

(Redirected from Octave for Windows)

GNU Octave is primarily developed on GNU/Linux and other POSIX conform systems. The ports of GNU Octave to Microsoft Windows use different approaches to get mo original Octave and adapt it to Microsoft Windows idiosyncrasies (e.g. dynamic libraries, file paths, permissions, environment variables, GUI system, etc). Bear this in mind panic if you get unexpected results. There are a lot of suggestions on the mailing lists for tuning your Octave installation. GNU Octave standalone ports for Windows are ind compiled using either the MinGW or Microsoft Visual Studio development environments.

navigation

- Main page
- Community portal
- Current events
- Recent changes
- Package list
- Index
- Random page
- Help

search

Search

Installers for Microsoft Windows [edit]

1. MXE Builds (unofficial, but recommended):
   - GNU Octave 3.8.2
2. Cygwin package (requires Cygwin to be installed first):
   - GNU Octave 3.8.2 Cygwin package
3. MinGW + Visual studio (official):
   - GNU Octave 3.6.4 and Octave Forge Packages

あとは，指示に従って，
適切にインストールする

2.4.6.2 Content
2.4.7 Alternative

Detailed instructions [edit]

MXE Builds [edit]

Untouched MXE builds for the current 3.8.x release can be found here.

However, these are **unofficial** builds! Remember to keep this in mind if you refer to it on bug reports.

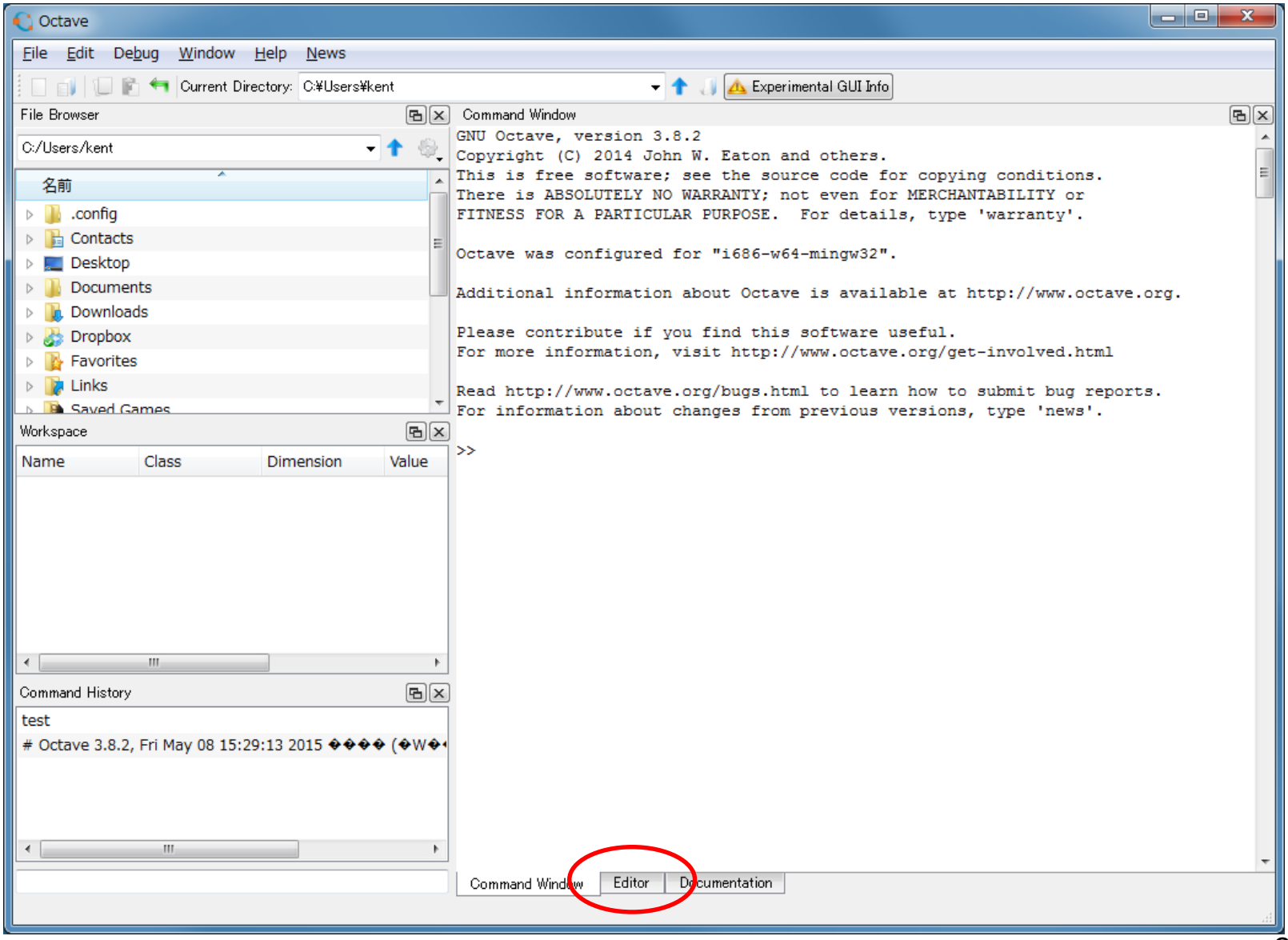
If you got any problems while running Windows 8 or libstdc++-6.dll errors, try this octave-gui.bat file and place it into your octave-3.8.0 folder (e.g. `C:/octave-3.8.0/`).

```
@echo off
set PATH=%CD%\bin\
start octave --force-gui -i --line-editing
exit
```

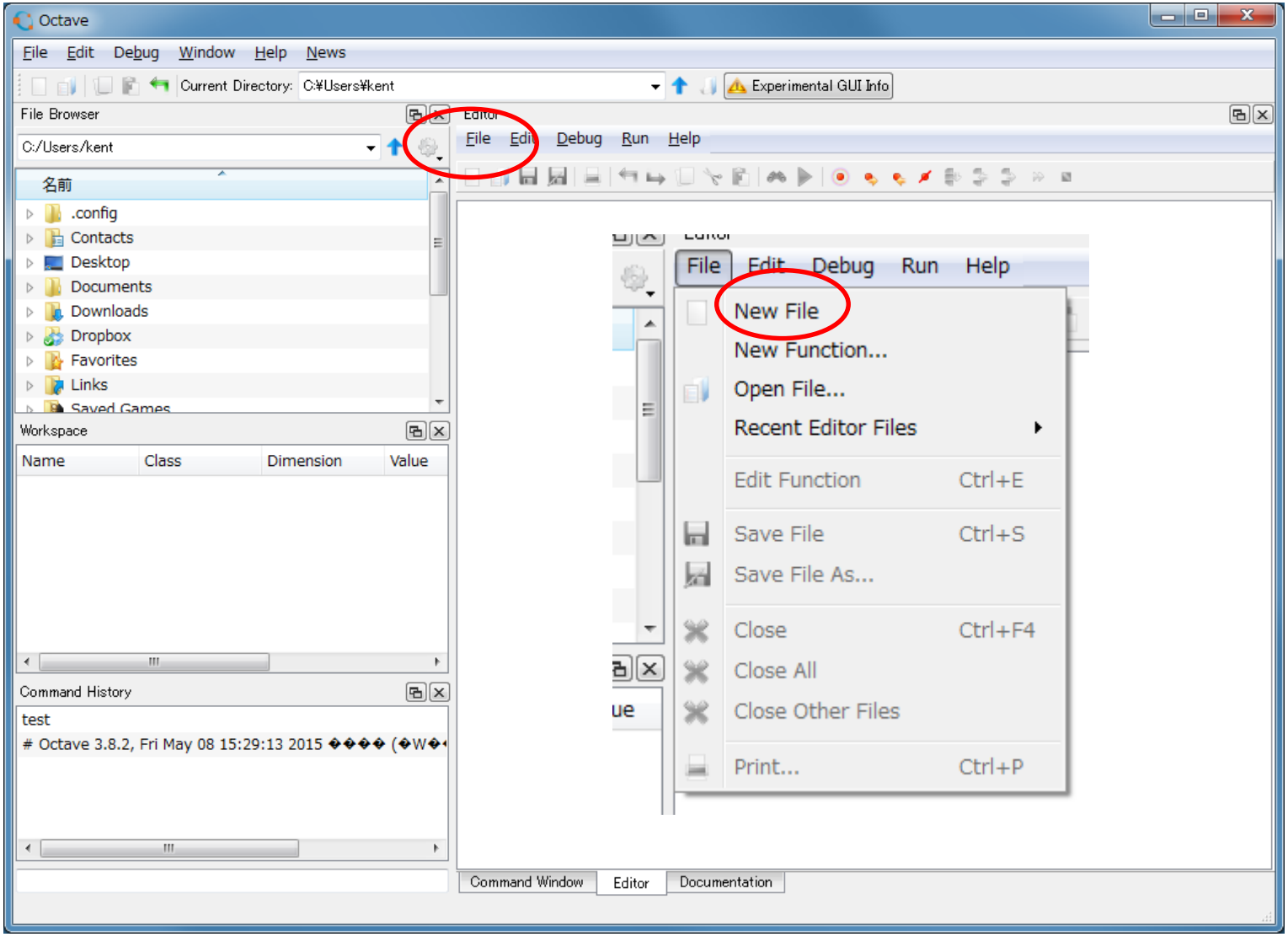
The plot command also doesn't work well in case of Windows 8 system.

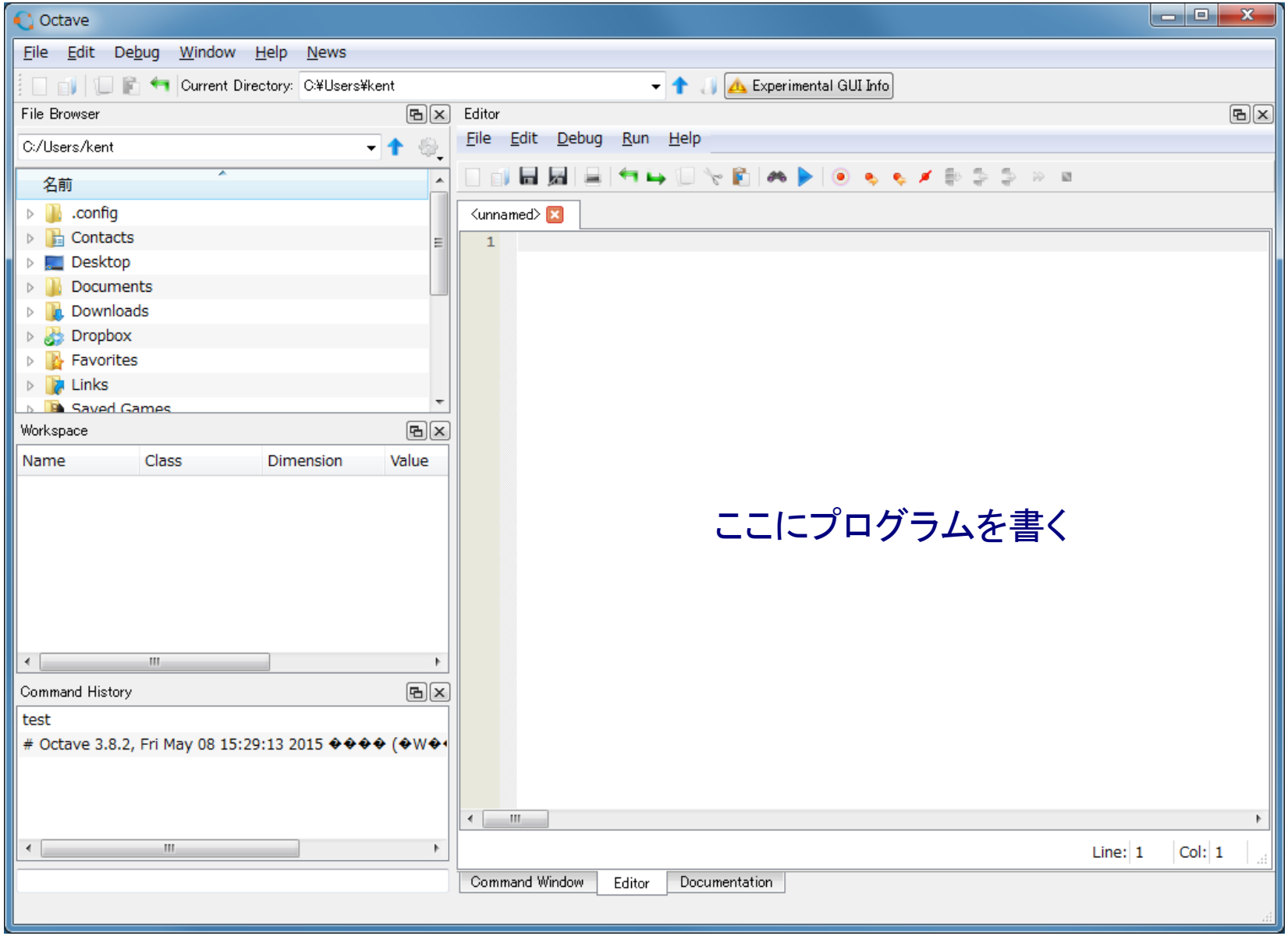
See Windows_Installer for building instructions.

標準でGUIが起動する



エディタで新規ファイル作成 (Script)




Octave
File Edit Debug Window Help News
Current Directory: C:¥Users¥kent
Experimental GUI Info
File Browser
C:/Users/kent
名前
.config
Contacts
Desktop
Documents
Downloads
Dropbox
Favorites
Links
Saved Games
Workspace
Name Class Dimension Value
Command History
test
Octave 3.8.2, Fri May 08 15:29:13 2015 ◆◆◆◆ (◆W◆
Editor
File Edit Debug Run Help
<unnamed>
1
ここにプログラムを書く
Line: 1 Col: 1
Command Window
Editor
Documentation

これをやってみよう

$\sqrt{2}$の数値$(1.41421356\ldots)$を求める

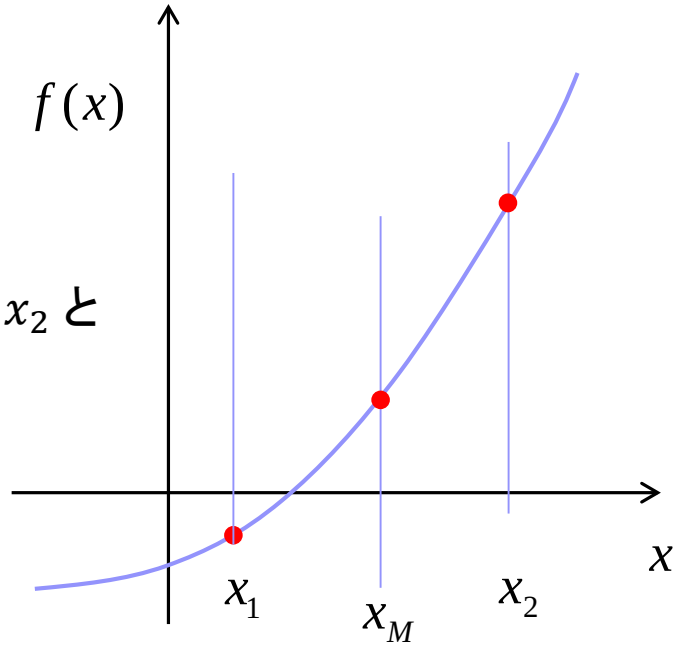
$\Leftrightarrow$ $f(x) = x^2 - 2$ を満たす$x > 0$ を求める

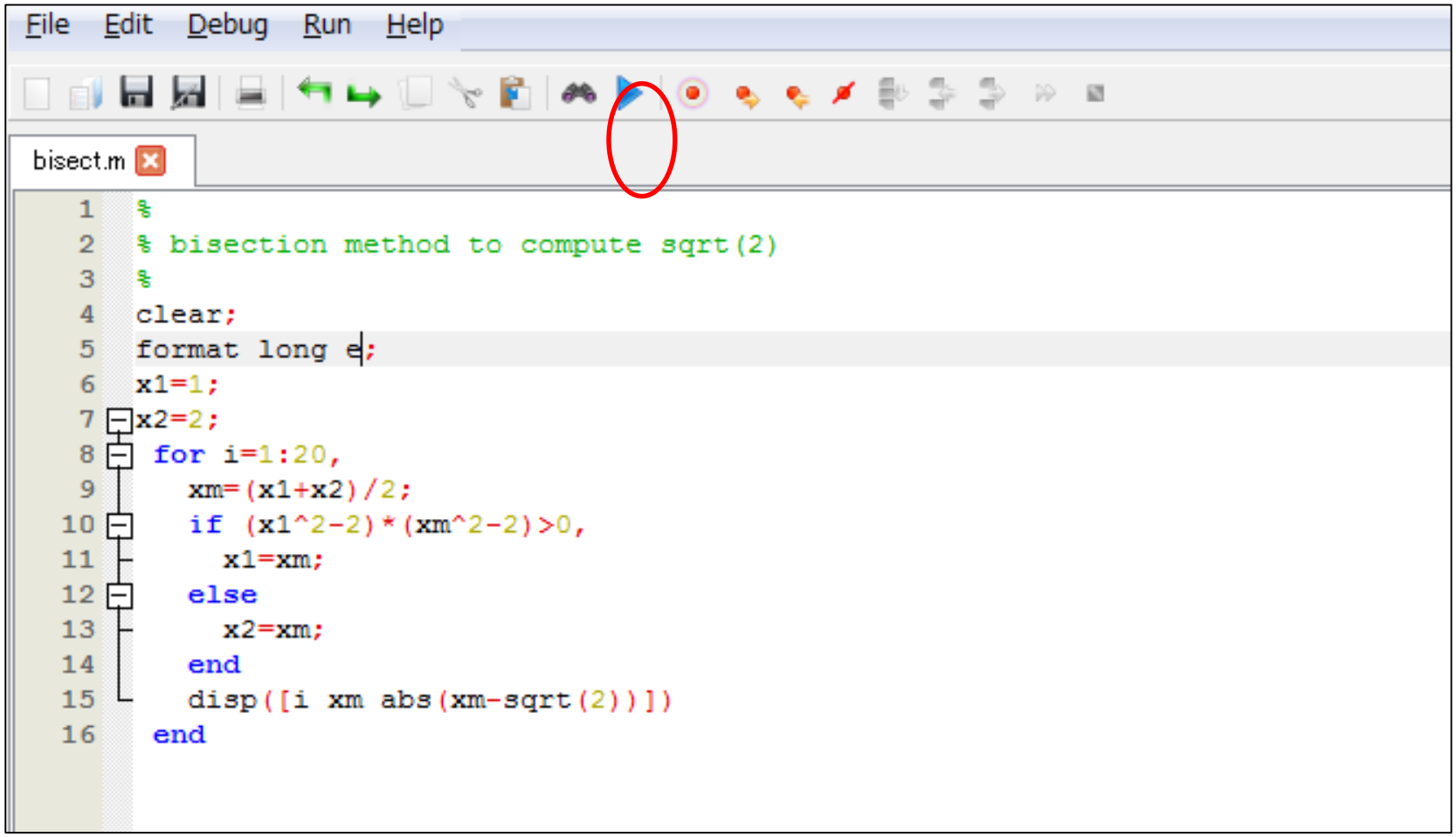
二分法 (Bisection Method)

$f(x_1)f(x_2) < 0$ であるような2点 x_1, x_2 をとる.

$x_M = \dfrac{x_1 + x_2}{2}$ とし, $f(x_M)$ と同符号のx_1 または x_2 と x_M を入れ替える.

所望の近似値が得られるまで繰り返す.

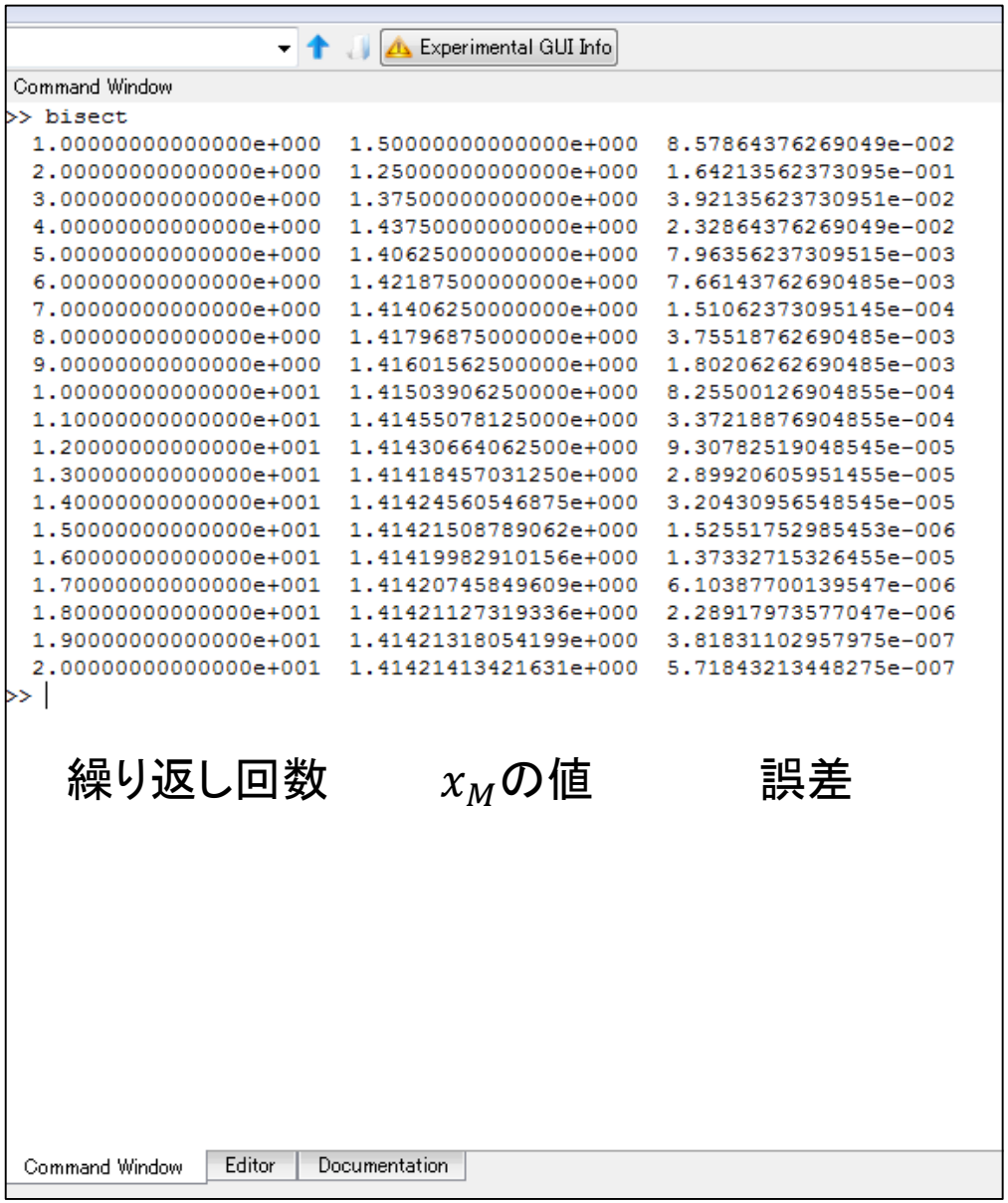




編集後, bisect.m として適当な場所に保存.

で実行. （初回はパスを通す設定ダイアログあり）

結果は Command Window に表示される（画面下タブ切り替え）.


Experimental GUI Info
Command Window
>> bisect
1.00000000000000e+000 1.50000000000000e+000 8.57864376269049e-002
2.00000000000000e+000 1.25000000000000e+000 1.64213562373095e-001
3.00000000000000e+000 1.37500000000000e+000 3.92135623730951e-002
4.00000000000000e+000 1.43750000000000e+000 2.32864376269049e-002
5.00000000000000e+000 1.40625000000000e+000 7.96356237309515e-003
6.00000000000000e+000 1.42187500000000e+000 7.66143762690485e-003
7.00000000000000e+000 1.41406250000000e+000 1.51062373095145e-004
8.00000000000000e+000 1.41796875000000e+000 3.75518762690485e-003
9.00000000000000e+000 1.41601562500000e+000 1.80206262690485e-003
1.00000000000000e+001 1.41503906250000e+000 8.25500126904855e-004
1.10000000000000e+001 1.41455078125000e+000 3.37218876904855e-004
1.20000000000000e+001 1.41430664062500e+000 9.30782519048545e-005
1.30000000000000e+001 1.41418457031250e+000 2.89920605951455e-005
1.40000000000000e+001 1.41424560546875e+000 3.20430956548545e-005
1.50000000000000e+001 1.41421508789062e+000 1.52551752985453e-006
1.60000000000000e+001 1.41419982910156e+000 1.37332715326455e-005
1.70000000000000e+001 1.41420745849609e+000 6.10387700139547e-006
1.80000000000000e+001 1.41421127319336e+000 2.28917973577047e-006
1.90000000000000e+001 1.41421318054199e+000 3.81831102957975e-007
2.00000000000000e+001 1.41421413421631e+000 5.71843213448275e-007
>>
繰り返し回数
x_Mの値
誤差
Command Window
Editor
Documentation

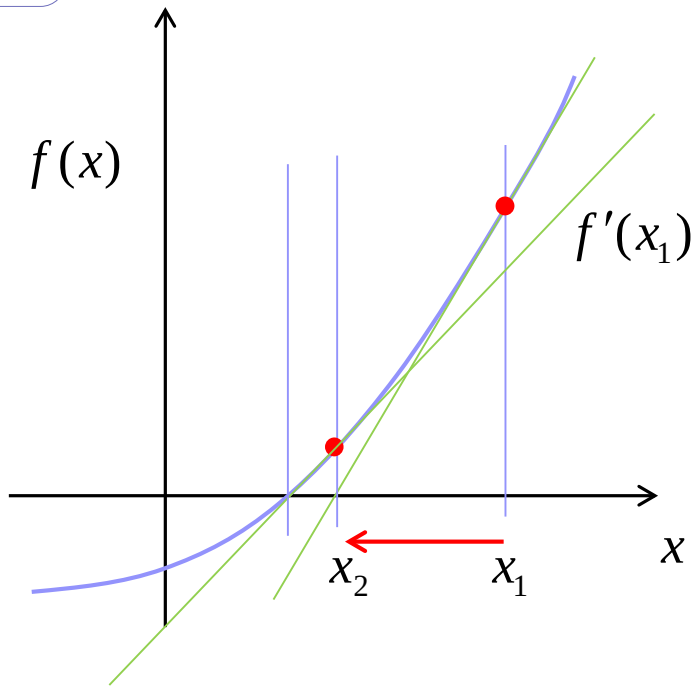
ニュートン法 (Newton's Method)

初期値 x_1 に対して x_1 における $f(x)$ の接線を引く.

接線と x 軸の交点を x_2 とする.

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

所望の近似値が得られるまで繰り返す.



これもやってみよう. 初期値は例えば 2. 繰返しは5回ぐらいで十分.
二分法の結果とあわせて, 次週簡単なレポートにまとめて提出.