電気303/電情303 数値解析 (4)

連立一次方程式の解法 (1)

Scilab で
連立一次方程式を解く

はじめに

これから 3 回の講義の典拠は教科書および以下の文献:

- 杉原, 室田: 線形計算の数理, 岩波書店, 2009
- 斎藤: 数値解析入門, 東京大学出版会, 2012
- 久保田: 工学基礎 数値解析とその応用, 数理工学社, 2010

- 伊理: 線形代数汎論, 浅倉書店, 2009
- 伊理, 藤野: 数値計算の常識, 共立出版, 1996
- 渡部: 連立１次方程式の基礎知識, 九州大学大型計算機センター広報, Vol.28, No.4, pp.291–349, 1995.

連立一次方程式

- 数学を使って応用上の問題を解くということは, 方程式を立てて解を求めることに相当. .
- 多くの場合, 少なくとも局所的には, 線形近似が有効.
- 線形の数学モデルは非線形の数学モデルより圧倒的に取り扱いやすい.

- 必要に応じて線形近似してから問題を解くということがよく行われる.
- このような場合には, 最終的に線形方程式を解けばもとの問題の (近似?) 解が得られる.
- 今回および次回の講義では, 方程式に微分演算が含まれない場合について考える.
- 微分が含まれる場合については第 11 回から第 14 回で取り扱う.

- 変数が 1 個の線形方程式の解法について悩むことは何もないので …
- はじめから多変数の場合を考える.
- 多変数の微分を含まない線形方程式を連立一次方程式ともいう.

Scilab による線形代数の基礎

- 今回の講義では, 受講者が Scilab の実行環境を持っていることを前提とする.
- Scilab で線形方程式を解くためには, Scilab の線形代数関連の機能を使うための知識が必要になる. まず, それについて述べる.

行列の定義

- 行列を定義するには,
  [第１行；第２行；．．．第ｎ行]
  という記法を用いる.
- 各行の要素は空白あるいはコンマで区切る.

- ; のかわりに改行を用いてもよいし, ; と改行を併用してもよい.
- 角括弧 [] を他の括弧に変更することはできない.
- 次ページに以下の行列を定義する例を挙げる.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

```
A=[1 2 3;4 5 6]; //要素を空白区切り
```

```
A=[1,2,3;4,5,6]; //要素をコンマ区切り
                 //実行結果は上と同じ
```

```
A=[1 2 3; //行の区切りにセミコロンと改行
4 5 6]    //実行結果は上と同じ
```

- //はScilabのコメントを表す記号.
- //から行末までがコメントと見做される.
- C言語のようにコメントの最初と終わりを指定する流儀ではなく, //から行末までが自動的にコメントと見做されるので注意.

- Scilab では, 定義されている変数名を打ち込んでからリターンキーを押すと, その変数の内容が表示される.
- よって, 上記の次に A(リターン) と入力すると, 次ページの結果が得られる.

```
-->A
 A  =
    1.    2.    3.
    4.    5.    6.
```

- -->は Scilab のプロンプトである.
- 表示例では, 空行や空白を適宜省略している.

- 行末にセミコロン (;) を打つと, 計算は実行されるが, 計算結果が画面に表示されなくなる.
- 行末にセミコロン (;) がない場合には, 計算結果が変数に記憶され, 同時に画面に表示される.

- 上記の `A=[1 2 3;4 5 6]` のように等号の左辺に変数名を明示した場合, 指定した変数に計算結果が代入される.
- 等号による変数の指定がない場合,

  `ans`

  という変数に計算結果が格納される.

- 変数 ans の内容は計算をする毎に更新される. 計算結果が後で必要になる場合には, 他の変数に計算結果を格納しておく必要がある.

ベクトルの定義

- 行ベクトルを定義するには,
  [要素１ 要素２ ... 要素n]
  のように, [] 内に要素を空白 (あるいはコンマ) で区切って並べる.

- 列ベクトルを定義するには,
  [要素 1; 要素２ ...; 要素 n]
  のように, [] 内に要素をセミコロン (;) あるいは改行で区切って並べる.

- $a = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}, b = \begin{pmatrix} 4 \\ 5 \\ 7 \end{pmatrix}$ を定義している例は, 次の通り.

```
a=[1 2 3];
```

```
b=[4;5;6];
```

行列やベクトルの要素を参照・変更する

- 行列の要素を参照するには,
  `A(i,j)`
  のように丸括弧内に参照したい要素の第1および第2の添字を書く.

- ベクトルの要素を参照するには,
  `a(i)`
  のように, 丸括弧内に参照したい要素の添字を書く.
- 行ベクトルと列ベクトルで要素を参照する方法は同じ.

- 今まで述べたように A, a, b がすでに定義されていることを前提とし, 使用例を次ページに示す.

```
-->A(2,3)
 ans  =
    6.
```

```
-->a(2)
 ans  =
    2.
```

- ベクトル a がすでに定義されているとき, たとえば

  a(2)=100

  などのようにすると, 行列やベクトルの指定した要素の内容を変更することができる.

- ベクトルや行列の要素数を超えた添字を指定すると, 行列が拡張され, 指定した添字に指定された数値が代入され, 値が指定されていない添字の部分には零が代入される (Scilab6.1.1 の場合)

- 直観に反する挙動であるが, たとえば, a=8 としてから, a(4)=3 とすると, a=[8;0;0;3] としたのと同じことになる.
- ただし, このような使い方をすることは, 自分が混乱するだけなので, 推奨しない.

ベクトル・行列の加減算

- ベクトル・行列の加算には演算子

  $+$

  , 減算には演算子

  $-$

  を用いる.
- 型が合っていないと計算ができない.

以下のようにAとBが定義されているという前提のもとで…

```
A=[1 2;3 4];
B=[5 6;7 8];
```

```
--> A+B
 ans  =
   6.    8.
   10.   12.
```

```
--> A-B
 ans  =
  -4.   -4.
  -4.   -4.
```

以下のようにaとbが定義されているという前提のもとで…

```
a=[1;2];
b=[3;4];
```

```
--> a+b
 ans  =
   4.
   6.
```

```
--> a-b
 ans  =
  -2.
  -2.
```

ベクトル・行列の乗算

- ベクトル・行列の乗算には演算子

  *

  を用いる.

- A, B が行列あるいはベクトルのとき, A*B が計算できるためには, A の列の数と B の行の数が一致することが必要かつ十分である.

以下のようにAとBが定義されているという前提のもとで…

```
A=[1 2;3 4];
B=[5 6;7 8];
```

```
--> A*B
 ans  =
   19.   22.
   43.   50
```

ベクトル・行列に対する除算演算子の効果

- Scilab における除算の演算子は / である.
- スカラーに対してこの演算子を用いると通常の除算となるが…
- ベクトルや行列にこれを適用すると, 型が合えば, 今回の講義の主題である, 連立一次方程式の解が得られる (後述).

- 演算子

  \

  も連立一次方程式の解を得る演算子であるが,
  方程式を解く「向き」が異なる (後述).

- どちらかというと\の方がよく使われる.
- 日本語キーボードや端末では\が￥と表示されることがあるので注意.

ベクトル・行列の成分ごとの乗除算

- A と B が同じ型 (行と列の大きさが同じ) の行列あるいはベクトルとしたとき…

- `A.*B`
  とすると, 第 (i, j) 要素が
  `A(i,j) * B(i,j)`
  の行列 (対応する成分どうしを乗算した行列)
  が得られる.

- `A./B`
  とすると第 (i, j) 要素が
  `A(i,j) / B(i,j)`
  の行列 (対応する成分どうしを除算した行列)
  が得られる.

- ベクトルについても同様.
- .*と./は, 2 文字が連続することで意味を持つ演算子であることに注意.
- .*や./の 2 文字のあいだに空白を入れるとエラーになる.

以下のようにAとBが定義されているという前提のもとで…

```
A=[1 2;3 4];
B=[5 6;7 8];
```

```
--> A.*B
 ans  =
    5.    12.
   21.    32.
```

```
--> A./B
 ans  =
   0.2          0.3333333
   0.4285714    0.5
```

以下のようにaとbが定義されているという前提のもとで…

```
a=[1;2];
b=[3;4];
```

```
--> a.*b
 ans  =
    3.
    8.
```

```
--> a./b
 ans  =
   0.3333333
   0.5
```

ベクトルの内積

- Scilab には, ベクトルの内積を計算するための演算子は用意されていない.
- a と b をともに列ベクトルとしたとき, a と b をの内積を求めるには, a を転置することで得られる行ベクトルと b の積を計算する.

- 後で改めて述べるが, 演算子’によって, 行列やベクトルを転置することができる.
- 以下に計算例を示す.

以下のようにaとbが定義されているという前提のもとで…

```
a=[1;2];
b=[3;4];
```

```
--> a'*b
 ans  =
    11.
```

- 先の例について注意を述べる.
- a と b がともに 2 次の列ベクトル (すなわち 2 行 1 列の行列) なので, a’ (a の転置) は 2 次の行ベクトル (すなわち 1 行 2 列の行列) で, a’ の列の数と b の行の数が一致するから積が計算でき, 結果は 1 行 1 列の行列, すなわちスカラーとなる.

- Scilabでベクトルの内積を計算するときには,「転置してから積を取る」という手順を取らねばならない.
- なお, 先の例では, a は2次の列ベクトル (すなわち2行1列の行列), b’(b の転置) は2次の列ベクトル (すなわち1行2列の行列) で, a の列の数と b’ の行の数が一致するから, a*b’ も計算できる. 結果は2行2列の行列となる.

以下のようにaとbが定義されているという前提のもとで…

```
a=[1;2];
b=[3;4];
```

```
--> a*b'
 ans  =
   3.    4.
   6.    8.
```

行列に関するいくつかの関数や演算子

- 以下の関数や演算子はよく用いられる:
  - ▷ `det`　正方行列の行列式を与える
  - ▷ `inv`　正方行列の逆行列を与える
  - ▷ `rank`　行列のランクを与える
  - ▷ `size`　行列やベクトルサイズを与える
  - ▷ `'`　行列やベクトルを転置する

- 以下に使用例を示す.

- 行列

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

  の行列式を求める.

- A の行列式は-2 になる筈であるが, 確かに…

```
--> A=[1 2; 3 4]
 A  =
    1.    2.
    3.    4.
--> det(A)
 ans  =
  -2.
```

- 行列

$$A = \begin{pmatrix} -2 & 1 \\ 0 & 1 \end{pmatrix}$$

の逆行列を求める.

- A の逆行列は

$$\begin{pmatrix} -\frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix}$$

になる筈であるが, 確かに…

```
--> A=[-2 1;0 1]
 A  =
   -2.    1.
    0.    1.
```

```
--> inv(A)
 ans  =
  -0.5   0.5
   0.    1.
```

- 行列

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

のランクを求める.

- A のランク (線形独立な行あるいは列の数) は 2 になる筈であるが, 確かに…

```
--> A=[1 0 0 0;0 2 0 0;0 0 0 0]
 A  =
    1.    0.    0.    0.
    0.    2.    0.    0.
    0.    0.    0.    0.
```

```
--> rank(A)
 ans  =
    2.
```

- 行列

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

を転置する.

- A の転置行列は

$$A^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

となる筈であるが, 確かに…

```
--> A=[1 2 3;4 5 6]
 A  =
    1.    2.    3.
    4.    5.    6.
```

```
--> A'
 ans  =
    1.    4.
    2.    5.
    3.    6.
```

単位行列

- 単位行列を定義するには, 組み込み関数

  eye

  を用いる.

- n と m をある数値としたとき,

  `eye(n,m)`
  とすると…

  - ▷ $n = m$ のときには `n` 次の単位行列
  - ▷ $n \neq m$ のときには, 対角要素がすべて 1 でそれ以外の要素が零の行列

- 使用例を示す.

```
--> eye(2,2)
 ans  =
    1.    0.
    0.    1.
```

```
--> eye(2,4)
 ans  =
   1.    0.    0.    0.
   0.    1.    0.    0.
```

零行列

- 零行列を定義するには, 組み込み関数
  `zeros`
  を用いる.

- n と m をある数値としたとき…
  - ▷ `zeros(n,m)`
    n 行 m 列の零行列
  - ▷ `zeros(n,1)`
    n 次の全要素が零の列ベクトル
  - ▷ `zeros(1,m)`
    m 次の全要素が零の行ベクトル

- 使用例を示す

```
--> zeros(2,3)
 ans  =
   0.    0.    0.
   0.    0.    0.
```

```
--> zeros(2,1)
 ans  =
   0.
   0.
```

```
--> zeros(1,3)
 ans  =
   0.    0.    0.
```

全要素が１の行列やベクトル

- 数値計算では全要素が1の行列やベクトルをしばしば用いるため, それを生成するための関数

  `ones`

  が用意されている.

- n と m をある数値としたとき…
  - ▷ `ones(n,m)`
    n 行 m 列の全要素が 1 の行列
  - ▷ `ones(n,1)`
    n 次の全要素が 1 の列ベクトル
  - ▷ `ones(1,m)`
    m 次の全要素が 1 の行ベクトル

- 使用例を示す

```
--> ones(2,3)
 ans  =
    1.    1.    1.
    1.    1.    1.
```

```
--> ones(2,1)
 ans  =
    1.
    1.
```

```
--> ones(1,3)
 ans  =
    1.    1.    1.
```

全要素が擬似乱数の行列やベクトル

- 擬似乱数を成分として持つ行列やベクトルを生成するには, 組み込み関数

  `rand`

  を用いる.

- 起動時の設定では, `rand` は $[0, 1]$ の範囲に一様分布する擬似乱数 (あるいは, それを成分として持つ行列やベクトル) を生成する.
- 以下では, n と m をある数値とする. このとき…

- `rand(n,m)`
  とすると，n 行 m 列で，各成分が擬似乱数 ($[0,1]$ の範囲に一様分布する母集団から抽出されたもの) の行列が生成される．

- `rand(n,1)`
  あるいは
  `rand(1,m)`
  とすると, 各成分が擬似乱数 ($[0,1]$ の範囲に一様分布する母集団から抽出されたもの) の行ベクトルあるいは列ベクトルが生成される.

`rand(n,m,'normal')`
とすると, n 行 m 列で, 各成分が擬似乱数 (期待値零, 分散 1 で正規分布する母集団から抽出されたもの) の行列が生成される.

- `rand()`
  とすると, 擬似乱数 ($[0, 1]$ の範囲に一様分布する母集団から抽出されたもの) が 1 個生成される.
- 使用例を以下に示す.
- 各要素の値は実行するたびに変わる. 受講者が以下と同一のコマンドを入力しても, 同一の結果が得られるわけではないので注意せよ.

```
--> rand(2,3)
 ans  =
   0.2113249   0.0002211   0.6653811
   0.7560439   0.3303271   0.6283918
```

```
--> rand(2,1)
 ans  =
   0.8497452
   0.6857310
```

```
--> rand(1,3)
 ans  =
   0.8782165   0.068374   0.5608486
```

```
--> rand(2,3,'normal')
 ans  =
   0.8915736  -1.3925211  -0.7414363
   1.2429915   0.2044185  -0.7437915
```

```
--> rand(2,1,'normal')
 ans  =
  -0.2589641
   0.3501626
```

```
--> rand(1,3,'normal')
 ans  =
   1.0478272  -1.3218007  -1.4061926
```

乱数に関する注意

- コンピュータが生成する乱数は擬似乱数と呼ばれるもので, 見掛け上規則性を持たない数値が生成されるが, 実は周期を持っている.
- 乱数の「種 (seed)」と呼ばれる値を変えると, 生成される擬似乱数を変更することができる.

- ランダムに近い試行をおこないたいときには, 乱数の種を別の手段 (たとえば時刻情報をミリ秒単位で取得するなど) で毎回異なる数値に定めるとよい.
- 逆に数値実験に再現性を持たせたいときには, 乱数の種を毎回同じ値に設定する.

- Scilab では, `rand('seed',n)` とすることで, 乱数の種を数値 `n` にすることができる.

- Scilab の `rand()` は, 起動時の設定では母集団として一様分布を用いるが…
- `rand('normal')`
  とすることにより, 母集団を正規分布に変更することができる.

- 上記によって母集団を正規分布に変更した場合, Scilab を一旦終了するか, 明示的に母集団を一様分布に戻すまで, 母集団として正規分布がずっと利用されるので注意せよ.

- `rand('uniform')`
  とすることにより, 母集団を一様分布に変更することができる.

- `rand('info')`
  とすることにより, 現在の擬似乱数に関する設定を表示することができる.

コロン (:) 演算子

- Scilab で

  `m:n`

  とすると, `m` から `n` まで 1 刻みで増える数のリストが生成される. $m > n$ のときには空リストが生成される.

- Scilab で

  `m:k:n`

  とすると, `m` から `n` まで `k` 刻みで増える数のリストが生成される. `k` は負でもよい. $k > 0$ かつ $m > n$, あるいは $k < 0$ かつ $m < n$ のときには空リストが生成される.

- 行列の要素を参照するとき，： を単独で用いると，行あるいは列の要素すべてという意味になる．たとえば …
  - A(1,:) とすると行列 A の第 1 行の全要素から成る行ベクトルが得られる
  - A(:,1) とすると行列 A の第 1 列の全要素から成る列ベクトルが得られる

- 使用例を示す.

- 1から5まで

```
--> 1:5
 ans  =
   1.    2.    3.    4.    5.
```

- 1 から 5 まで (2 刻み)

```
--> 1:2:5
 ans  =
   1.    3.    5.
```

- 行列の行や列の抽出: まず行列を作る

```
--> A=rand(3,3)
 A   =
    0.2113249    0.3303271    0.8497452
    0.7560439    0.6653811    0.685731
    0.0002211    0.6283918    0.8782165
```

- 第１行を抽出

```
--> A(1,:)
 ans  =
   0.2113249    0.3303271    0.8497452
```

- 第 2 行を抽出

```
--> A(2,:)
 ans  =
   0.7560439   0.6653811   0.685731
```

- 第 3 行を抽出

```
--> A(3,:)
 ans  =
    0.0002211    0.6283918    0.8782165
```

- 第１列を抽出

```
--> A(:,1)
 ans  =
   0.2113249
   0.7560439
   0.0002211
```

- 第 2 列を抽出

```
--> A(:,2)
 ans  =
   0.3303271
   0.6653811
   0.6283918
```

- 第 3 列を抽出

```
--> A(:,3)
 ans  =
   0.8497452
   0.6857310
   0.8782165
```

行列やベクトルを並べて新しい行列を作る

- A や B が行列やベクトルであるとき, Scilab で
  `[A B]`
  とすると A と B を横に並べたベクトルが生成される (A と B の行の数が合わないとエラーになる).

- A や B が行列やベクトルであるとき, Scilab で
  [A;B]
  とすると A と B を縦に並べたベクトルが生成される (A と B の列の数が合わないとエラーになる).

- 使用例を示す.

- 2 行 2 列の行列 A と B を作る

```
--> A=[1 2;3 4]
 A  =
    1.    2.
    3.    4.
```

```
--> B=[5 6;7 8]
 B  =
    5.    6.
    7.    8.
```

- A と B を横に並べる

```
--> [A B]
 ans  =
    1.    2.    5.    6.
    3.    4.    7.    8.
```

- A と B を縦に並べる

```
--> [A;B]
 ans  =
    1.    2.
    3.    4.
    5.    6.
    7.    8.
```

Scilab で連立一次方程式を解く

- $\boldsymbol{A}$ を m 行 n 列の行列, $\boldsymbol{x}$ を n 次のベクトル, $\boldsymbol{b}$ を m 次のベクトルとする.
- 数学的には, 連立一次方程式を解くとは,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

を満たす $\boldsymbol{x}$ をすべて求めることを意味する.

- Scilab で連立一次方程式を解くには演算子\ (あるいは/; 後述) を使う.

- 行列 $\boldsymbol{A}$ および $\boldsymbol{b}$ がScilabにおいて変数 `A` および `b` に格納されているとき, Scilabで $\boldsymbol{Ax} = \boldsymbol{b}$ の解を求めて結果を `x` に格納するには,

  ```
  x=A\b
  ```

  のようにする.

- 連立一次方程式の解には一意解, 不定解, 不能解の 3 種類があるが …
- Scilab は一意解, 不定解, 不能解のすべての場合について (近似的な) 解を与える.

- 一意解の場合には, $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ の解と Scilab の `x=A\b` は数値計算の誤差を除いて一致.
- 不定解の場合は, Scilab の `x=A\b` は $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ を満たす解のひとつを返す.
- 不能解の場合, Scilab は $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|$ が最小となる $\boldsymbol{x}$ を近似解として返す.

- 一意解の例を示す.
- 連立一次方程式

$$\begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \end{pmatrix}$$

  の解は $(x_1, x_2) = (2/3, 5/3)$ である.
- Scilab で計算すると…

```
-->A=[1 2;0 3];
-->b=[4;5];
-->x=A\b
 x  =
    0.6666667
    1.6666667
```

- 不定解の例を示す.
- 連立一次方程式

$$\begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 2$$

  の解は不定で, $(x_1, x_2) = (2, 0)$ はひとつの解
- Scilab で計算すると…

```
-->A=[1 0];
-->b=2;
-->x=A\b
 x  =
    2.
    0.
```

- 不能解の例を示す.
- 連立一次方程式

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

  は解を持たない.
- Scilab で計算すると…

```
-->A=[1;1];
-->b=[2;3];
-->x=A\b
 x  =
    2.5
```

- $x = 2.5$ は $Ax = b$ を満たさないが,

$$Ax = \begin{pmatrix} 2.5 \\ 2.5 \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

だから, `A*x` が `b` の最小二乗近似になっていることが確認できる.

§ 連立一次方程式の右辺が行列のとき

- A と B がともに行列であるとき,
  A\B
  とすると, B の各列に対して連立一次方程式を解いた結果をまとめた行列が得られる.

§ 演算子 \ のまとめ

- X=A\B は, A*X=B となるベクトルあるいは行列 X を求める演算子である.

- 対応関係は以下の通り.

$$\boldsymbol{X} = \underbrace{\boldsymbol{A}\backslash\boldsymbol{B}}_{(\frac{B}{A}\text{のようなもの})} \iff \boldsymbol{AX} = \boldsymbol{B}$$

- 上式右辺で, $\boldsymbol{X}$ が右から掛けられていることに注意. 演算子/を使った場合には, この部分が変更になる.

§ 演算子 /

- X=A/B は, X*B=A となるベクトルあるいは行列 X を求める演算子である.

- 対応関係は以下の通り.

$$\boldsymbol{X} = \underbrace{\boldsymbol{A}/\boldsymbol{B}}_{(\frac{A}{B}\text{のようなもの})} \iff \boldsymbol{X}\boldsymbol{B} = \boldsymbol{A}$$

  という対応.

- 上式右辺で, $\boldsymbol{X}$ が左から掛けられていることに注意. この部分が演算子\と異なる.

- 演算子\と比較すると, ?/?の方が機能が少ない. 演算子\が不定解に対して解のひとつ, 不能解に対して最小二乗近似解を返すのに対して, 演算子/が解けるのは一意解の場合だけで, 他はエラーになる.
- 以下に使用例を示す.

```
--> A=[1 2;0 3];
--> b=[4 5];
--> x=b/A
 x  =
    4.   -1.
```

連立一次方程式の一般論

- $\boldsymbol{A}$ を m 行 n 列の行列, $\boldsymbol{x}$ を n 次のベクトル, $\boldsymbol{b}$ を m 次のベクトルとする.
- 数学的には, 連立一次方程式を解くとは,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

を満たす $\boldsymbol{x}$ をすべて求めることを意味する.

- 行列 $\boldsymbol{A}$ とベクトル $\boldsymbol{b}$ を結合して作った行列 $\boldsymbol{B} = (\boldsymbol{A}|\boldsymbol{b})$ を, 連立一次方程式 $\boldsymbol{Ax} = \boldsymbol{b}$ の拡大係数行列という.
- $\mathrm{rank}\boldsymbol{A} < \mathrm{rank}\boldsymbol{B}$ なら不能解.
- $\mathrm{rank}\boldsymbol{A} = \mathrm{rank}\boldsymbol{B} = \dim \boldsymbol{x}$ なら一意解
- $\mathrm{rank}\boldsymbol{A} = \mathrm{rank}\boldsymbol{B} < \dim \boldsymbol{x}$ なら不定解.

- 行列に行基本変形を施すことで, 以下のような階段行列が得られる.

$$\begin{pmatrix} 0 & \cdots & 0 & 1 & * & & & \cdots & & \\ 0 & & \cdots & & & 0 & 1 & * & \cdots & \\ 0 & & & \cdots & & & & 0 & 1 & * & \cdots \\ & & & & & \cdots & & & & & \end{pmatrix}$$

- 階段行列の特徴は以下の通り: 1 の左はすべて零, 1 の右側は任意, 1 を含まない行はすべて零だけ, 行列全体を見ると 1 の系列は右斜め下に進む (真下不可)
- 第 1 行左端に零があるかないかは行列によって変わる.

- 一意解に対応する拡大係数行列を階段行列に変形すると (今回の講義ではこの場合を取り扱う)

$$\left(\begin{array}{cccc|c} 1 & * & \cdots & & * \\ 0 & 1 & * & \cdots & * \\ \vdots & \ddots & \ddots & * & * \\ 0 & \cdots & 0 & 1 & * \end{array}\right)$$

- 拡大係数行列を階段行列に変換する手順は行基本変形.
- 行基本変形は基本行列を拡大係数行列に左から掛けることに対応.
- 行基本変形によって, 係数行列 $\boldsymbol{A}$ が上三角行列 (後述) に変形されていることになる.

- 上三角行列とはこういう行列:

$$\begin{pmatrix} * & \cdots & \cdots & * \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & * \end{pmatrix}$$

ただし $*$ は任意の数 (零でもよい).