連続系アルゴリズム レポート課題7

- 宿題(レポート)
  - 提出状況に応じて，期末試験の点に最大 50 点を加算します
    - まったく提出しなくても，期末試験の点で成績が出ます
    - ただし，試験を受けなければ単位は出ません
  - プログラムだけはメールの添付ファイルで
    reiji@is.s.u-tokyo.ac.jp
    に送付してください
    - 所属学科，学年，学籍番号，氏名をメール本文に明記すること
    - メールの題名(Subject)は「連続系レポート課題第7回 提出者氏名」とすること
  - 手計算をするものや，計算結果と考察など，プログラム以外は A4 の紙(裏もつかってよい)にまとめてください
    - 所属学科，学年，学籍番号，氏名をレポートの最初に明記すること
    - 次回の講義の前に集めます
    - あるいは PDF で A4 サイズでもよい(プログラムと一緒にメールで送付してください)
    - ページ数制約を緩和しました
- 問題1
  - 下記のように、三重対角行列の LU 分解(枢軸選択なし)を考える

$$\begin{pmatrix} 1 & & & & \\ l_2 & 1 & & & \\ & l_3 & 1 & & \\ & & \ddots & \ddots & \\ & & & l_n & 1 \end{pmatrix} \begin{pmatrix} d_1 & u_1 & & & \\ & d_2 & u_2 & & \\ & & d_3 & \ddots & \\ & & & \ddots & u_{n-1} \\ & & & & d_n \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & & & \\ c_2 & a_2 & b_2 & & \\ & c_3 & a_3 & \ddots & \\ & & \ddots & \ddots & b_{n-1} \\ & & & c_n & a_n \end{pmatrix}$$

  - (a_i), (b_i), (c_i) から、(l_i), (d_i), (u_i) を求める式を示せ
  - LU 分解の計算量のオーダーはいくらか
- 問題2
  - $n \times m$ 行列 A と $m \times l$ 行列 B の積 C = AB を計算する関数を書け
    - 関数は，任意の n, m, l に対応できるように書くこと
    - 適当な行列(例:2 ×3 行列と 3 ×4 行列)を与えて、正しく計算されていることを確認せよ

数値計算で使わない有名な定理

数値計算の理論では使う；浮動小数点数による計算に使わないということ

- Jordan 標準形：任意の正方行列 A に対し，ある正則行列 X があって $X^{-1} A X = \mathrm{diag}(J_i)$
  - わずかな差で Jordan ブロックが変わってしまうので，丸め誤差の存在下では計算できない
- Cramer の公式

$$x_i = \frac{\det\begin{pmatrix}\mathbf{a}_1 & \cdots & \mathbf{a}_{i-1} & \mathbf{b} & \mathbf{a}_{i+1} & \cdots & \mathbf{a}_n\end{pmatrix}}{\det A}$$

  - ただし，行列式が解析的に容易に求められる場合は使えることがある

26

本日の C 言語

- 密行列の表現方法

(i) 2 次元配列を用いる

- 宣言

```
double a[10][10];
```

  - 昔はサイズが固定でなければいけなかった
  - 新しい C 言語では，サイズに式を入れてよい
    - ただし大域変数，static 変数などはだめ

- 参照

```
for (i=0; i< 10; i++)
  for (j=0; j< 10; j++)
    a[i][j] = i;
```

  - a[i, j] ではないことに注意！
    - やっかいなことに，a[i, j] は違う意味になる
  - 最初の要素は a[0, 0] であることに注意！

(ii) 1 次元配列を用いる

- A_{ij} を a[i * n + j] に入れる
  - n は A の列数
  - 2 次元配列は結局これに翻訳される
  - やはり i, j は 0 から $n-1$ まで

- 密行列の配列引数

(i) 2 次元配列を用いる

- 使い方

```
void func(int m, int n, double a[m][n]) {
  ... a[i][j] ...
}
```

  - 昔はサイズが固定でなければいけなかった
  - 新しい言語仕様に対応したコンパイラなら上記で OK
  - サイズを表す m, n が先に来ないといけない
  - 古い言語仕様のコンパイラでは，可変長にするには 1 次元配列でなければならない（下記）

(ii) 1 次元配列を用いる

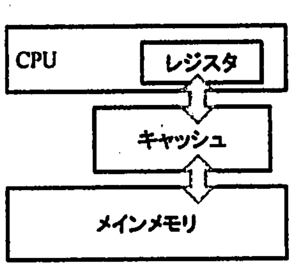
- 使い方

```
void func(int m, int n, double *a) {
  ... a[i * n + j] ...
}
```

  - これは今も昔も OK
  - m と n を間違えないように（たいてい間違える）

おまけ

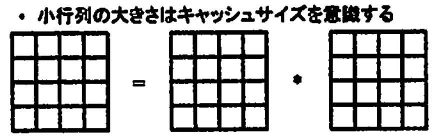
- コンピュータのメモリ階層
  - データはメインメモリに格納されているが、計算は CPU で行われる
  - 計算の前にデータをメインメモリから CPU に持ってきて、計算した結果は CPU からメインメモリに格納しなければならない
    - 実は計算そのものよりも、メインメモリと CPU の間でのデータの移動のほうが時間がかかる
    - だから、メインメモリと CPU の間でのデータのやり取りをできるだけ少なくするほうがよい
  - このため、速くて小容量のメモリが準備されている
    - CPU 内に、レジスタ
    - CPU とメインメモリの間に、キャッシュ



- 大まかに言って、配列はメインメモリに、スカラーの変数はレジスタに配置される
- 行列積 C += A * B

```
for (i=0; i< n; i++)
  for (j=0; j< n; j++)
    for (k=0; k< n; k++)
      c[i][j] += a[i][k] * b[k][j];
```

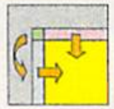
  - $3n^3$ 回の読み出しと、n^3 回の書きこみがある
    - コンパイラによっては n^2 回の書きこみにしてくれる
- ブロッキング
  - 行列 A, B, C を m × m 個の小行列にわける
    - 小行列の大きさはキャッシュサイズを意識する



  - 小行列ごとに計算する
    - 簡単のため，小行列単位で主記憶アクセスとすると，$3(n/m)^2$ の読み出しと $(n/m)^2$ の書き込みが m^3 回起こる
    - つまり，$3n^2m$ の読み出しと n^2m の書き込み・・・ブロッキングしない場合の m/n 倍に減少！

0になるかもで，丸め誤差もあるし → 枢軸選択を行う．

LU 分解のプログラム



これからUkiは でてこない．
あらかじめ逆数にしておく．

- for $i = 1$ to n

  枢軸選択

  $a[i, i] = 1.0 / a[i, i]$ // 逆数に変えておく

  for $j = i + 1$ to n

  $a[j, i] = a[j, i] * a[i, i]$ // $l_{ji} = a_{ji}\,a_{ii}^{-1}$

  for $k = i + 1$ to n // $u_{ik} = a_{ik}$

  $a[j, k] = a[j, k] - a[j, i] * a[i, k]$ // $a_{jk} - l_{ji}\,u_{ik}$
- 計算量 $O(n^3)$

1

枢軸選択を記憶する

- 須田のやり方

  for $i = 1$ to n

  idx[i] = i; } 準備

  for $i = 1$ to n

  k = 枢軸行

  行列の i 行目と k 行目を交換

  idx[i] と idx[k] を交換

  } LU 分解の最初の部分（このあと掃き出し計算のコードが続く（前頁））

  交換のしかた
  tmp = idx[i];
  idx[i] = idx[k];
  idx[k] = tmp;

  前進・後退代入をどうするかは頭の体操
- LU 分解終了後，idx[i] は行列の i 行目に入っているデータが，もともと何行目だったかを示す

何行目がもともと何行目にあったか

2

LU 分解で枢軸選択をしないと(1)

- 行列 A として

A =

0.5032252	- 0.0480207	- 0.5072656	0.6280916	0.0426332	0.1191979
- 0.0480207	1.9820117	0.1306984	0.5911376	0.6442172	0.3851798
- 0.5072656	0.1306984	0.5147631	0.5402085	0.9464921	0.7291645
0.0792822	0.5488684	0.3071979	0.4240016	0.4416274	0.1832391
0.4202746	0.2775685	0.9295958	0.9422982	0.8307754	0.857476
0.8474012	0.2176310	0.1175817	0.0091902	0.5821865	0.0649050

- 枢軸選択なしの LU 分解をして，$L \times U$ を計算

LU =

0.5032252	- 0.0480207	- 0.5072656	0.6280916	0.0426332	0.1191979
- 0.0480207	1.9820117	0.1306984	0.5911376	0.6442172	0.3851798
- 0.5072656	0.1306984	0.5147631	0.5402085	0.9464921	0.7291645
0.0792822	0.5488684	0.3071979	0.4240112	0.4416504	0.1832581
0.4202746	0.2775685	0.9295958	0.9422607	0.8308105	0.8574219
0.8474012	0.2176310	0.1175817	0.0092773	0.5822144	0.0648804

3

LU 分解で枢軸選択をしないと(2)

- 部分枢軸選択つきなら，ほぼ完全に戻る

LU =

0.5032252	- 0.0480207	- 0.5072656	0.6280916	0.0426332	0.1191979
- 0.0480207	1.9820117	0.1306984	0.5911376	0.6442172	0.3851798
- 0.5072656	0.1306984	0.5147631	0.5402085	0.9464921	0.7291645
0.0792822	0.5488684	0.3071979	0.4240016	0.4416274	0.1832391
0.4202746	0.2775685	0.9295958	0.9422982	0.8307754	0.857476
0.8474012	0.2176310	0.1175817	0.0091902	0.5821865	0.0649050

- 方程式を解いてみても・・・

真の解	右辺ベクトル	枢軸選択なし	枢軸選択あり
x =	b = A x =	x =	x =
0.5649126	0.3090275	0.5648804	0.5649126
0.1029868	0.4853791	0.1029924	0.1029868
0.2313628	0.2087399	0.2313411	0.2313628
0.1993843	0.3571309	0.1993860	0.1993843
0.1772244	0.9188450	0.1771746	0.1772244
0.1197162	0.6411050	0.1197715	0.1197162

4

固有値の計算法？（1）

- 行列 A の固有値はどうやって計算する？
- 数学の教科書には

  $\det(A - \lambda I)$ の根が固有値　　次には使はない

- 実験
  - 10×10 の対称行列 A を乱数から生成
  - $A' = A$ の各要素に 1e-4 程度の誤差を加えたもの
  - $p' = \det(A - \lambda I)$ の各 λ のべきの係数に 1e-4 程度の誤差を加えたもの

5

固有値の計算法？（2）

例えば(λ)
について
λがちょっとずれると
大きいerrorになる

A の固有値	A の固有値と A' の固有値の差	A の固有値と p' の根の差
31.330341	- 0.0013956	0.0029217
1.6645463	0.0000047	- 0.0180362
1.4518537	0.0000858	0.0307866
1.1886373	- 0.0000386	- 0.0159929
0.6544475	- 0.0000810	0.0119644
0.4800675	0.0000426	- 0.0143673
0.3401261	0.0000968	0.0057316
0.1824360	0.0000599	- 0.0006934
0.0799798	0.0000791	0.0000374
0.0139027	0.0000460	- 5.211D-09

6

逆反復法

- $(A - \sigma I)^{-1}$ の固有値は $1 / (\lambda - \sigma)$ となる
  - σ にもっとも近い固有値が，絶対値最大になる
- 逆反復法：$(A - \sigma I)^{-1}$ に冪乗法を適用することで，σ に最も近い固有対を得る
- $\sigma \leftarrow$ 得られた近似固有値で，よりはやく収束
  - Rayleigh 商反復法（2 次または 3 次収束）

$$y_k = (A - \rho_{k-1} I)^{-1} x_{k-1}, \quad x_k = \frac{y_k}{\|y_k\|}, \quad \rho_k = \frac{y_k^H A y_k}{y_k^H y_k}$$

7

逆反復：悪条件方程式

- A ：10x10 の乱数行列
  - ひとつの固有対を選んだ：$A\boldsymbol{u} = \lambda \boldsymbol{u}$
  - $A - \lambda I$ はほとんど特異（条件数 $\kappa = 1.2\mathrm{e}{+}16$）
- 乱数ベクトル $\boldsymbol{b}$ に対し $(A - \lambda I)\,\boldsymbol{x} = \boldsymbol{b}$
  - $\|\boldsymbol{b} - A\boldsymbol{x}\| / \|\boldsymbol{b}\| = 0.04$
    - 解としてはかなり悪い
- $\boldsymbol{y} = \boldsymbol{x} / \|\boldsymbol{x}\|$
  - $\|\boldsymbol{u} - \boldsymbol{y}\| = 5.6\mathrm{e}{-}15$
    - ほぼ完全な固有ベクトル

8

数値アルゴリズム 12/5

連立一次方程式

n元の連立方程式

$$
\begin{cases}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\
\cdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n
\end{cases}
$$

を行列, ベクトルで書くと.

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

Aは正則.

簡単に解ける連立一次方程式

下三角行列

$$
\begin{pmatrix}
a_{11} & & & & \\
a_{21} & a_{22} & & & O \\
a_{31} & a_{32} & a_{33} & & \\
\vdots & \vdots & \vdots & \ddots & \\
a_{n1} & a_{n2} & a_{n3} & & a_{nn}
\end{pmatrix}
\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}
=
\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}
$$

$x_1 = \frac{a_1}{b_1}$ とする.

(2行目) x_2 は x_1 を用いて, $x_2 = (b_2 - a_{21}x_1)/a_{22}$.

(3行目) x_3 は x_1, x_2 を用いて

$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}.$$

$\vdots$

$$x_i = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j\right)/a_{ij}$$

「前進代入」, 計算量 $O(n^2)$.

上三角行列

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & a_{22} & \cdots & a_{2n} \\ & 0 & \ddots & \vdots \\ & & & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}.$$

$x_n = b_n / a_{nn}$ 「後退代入」

一般の場合

① $LU = A$ となるような
下三角行列 L と
上三角行列 U を求める.
(LU分解)

② $L\mathbb{y} = \mathbb{b}$ を解く. (前進代入)

③ $U\mathbb{x} = \mathbb{y}$ を解く. (後退代入)

②, ③ から. $L(U\mathbb{x}) = \mathbb{b}$

これで解ける

---

① LU分解は $O(n^3)$ かかる

②, ③ は $O(n^2)$.

☆ 同じ係数行列 A で右辺 $\mathbb{b}^{(1)}, \mathbb{b}^{(2)}, \mathbb{b}^{(3)}, \cdots, \mathbb{b}^{(m)}$

$A\mathbb{x}^{(1)} = \mathbb{b}^{(1)}$.

$A\mathbb{x}^{(2)} = \mathbb{b}^{(2)}$ $\rightarrow$ $O(n^2 m)$.

$\vdots$

LU分解はもうしているので, $O(n^2)$ だけですんでしまう.

逆行列は不要.

逆行列を求める $O(n^3)$

〃 ・ベクトル積 $O(n^2)$

$x = A^{-1}b$ これは「逆行列をかける」ではない.

（何らかの方法で 式を x について解くというイミ）

LU分解.

$n \times n$.

$$\left(\begin{array}{c|c} a_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) = \left(\begin{array}{c|c} l_{11} & l_{12}^T \\ \hline l_{21} & L_{22} \end{array}\right) \left(\begin{array}{c|c} u & u_{12}^T \\ \hline 0 & U_{22} \end{array}\right)$$

$$A = L \quad U$$

$$\shortparallel$$

$$\left(\begin{array}{c|c} l_{11} u_{11} & l_{11} u_{12}^T \\ \hline l_{21} u_{11} & l_{21} \cdot u_{12}^T + L_{22} U_{22} \end{array}\right)$$

$l_{11} = 1$ とおく.　∵ 自由度は n個あるので, 1に固定

$a_{11} = l_{11} u_{11}$ より.　$a_{11} = u_{11}$

$a_{12}^T = l_{11} u_{12}^T$ より.　$a_{12}^T = u_{12}^T$.

$a_{21} = l_{21} u_{11}$ より　$l_{21} = \dfrac{1}{a_{11}} a_{21}$

$A_{22} = l_{21} u_{12}^T + L_{22} U_{22}$ より.

$$L_{22} U_{22} = A_{22} - l_{21} u_{12}^T$$

これは $n-1$ 次の LU分解.

☆ よく使う メモリ配置.

行列 A に L と U を上書き.

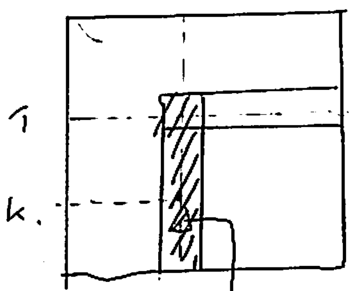
$l_{ii}=1$ なので 対角は必要なし.

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ l_{21} & u_{22} & u_{23} & u_{24} \\ l_{31} & l_{32} & u_{33} & u_{34} \\ l_{41} & l_{42} & l_{43} & u_{44} \end{pmatrix}$$

すくめている.
これなら 計算 プログラム で 目がいらない.

枢軸選択.

部分枢軸選択.



枢軸列.

枢軸列中 .. 絶対値が最大の要素 (k行目).

i行目 と k行目 を交換.

☆ 並列なら分解できる ← メモ.

☆ なぜか精度よい.

固有値の計算，

べき乗法

適当な初期値 x_0.

$$x_{k+1} = \frac{A x_k}{\|A x_k\|}$$

をくり返す．

$x_k \to A$の固有ベクトルに収束

$A x = \lambda x$

$$\hat{\lambda} = \frac{x^T A x}{x^T x}$$ が固有値の近似．

得られるのは 絶対値が最大の 固有値．