

数値解析

第2回: メモリの動的確保、行列・ベクトルとノルム

本日の目標

1. メモリの動的確保について理解する
2. ライブラリ関数を使えるようになる
3. 配列を用いた行列演算をプログラムできるようにする

領域(メモリ, 配列)の動的確保

- プログラム実行中に, 必要な量の領域を作る
 - malloc (Memory Allocation)
- 使わなくなったときには解放する
 - free

例)

- 卒業証書は, 3月になってから作成する
- 実行(後期日程完了)するまで, 何人卒業できるかわからない
- 4月に十分多い枚数を作っておくことも可能だが非効率

復習: ポインタと配列

`double b[4] = {2.54, 1.33, 0.52, 2.824}`

`double *a = (b+1)`



`*a : 1.33 (=b[1])`

2.54	1.33	0.52	2.824
------	-----------------	------	-------

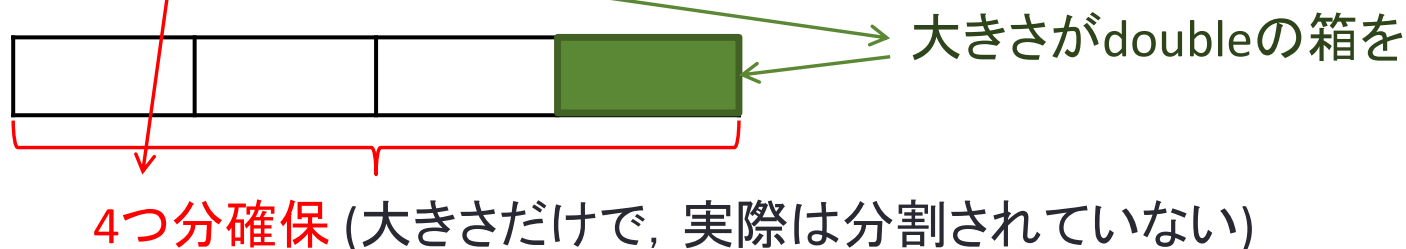
3.0

`*a = 3.0`

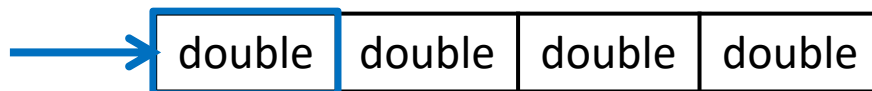
- 要素番号を伴わずに出てきた配列は, 先頭要素へのポインタ
- ポインタの加減は, オブジェクトのサイズ分ずつアドレスを変化
- 間接参照演算子と, ポインタ型と, 積演算子はどれも*なので注意する

1次元配列の動的確保 (malloc)

① `malloc(4 * sizeof(double))`



② 確保できたら先頭要素を指すポインタを返します



```
double *a = (double *)malloc(4 * sizeof(double))
```

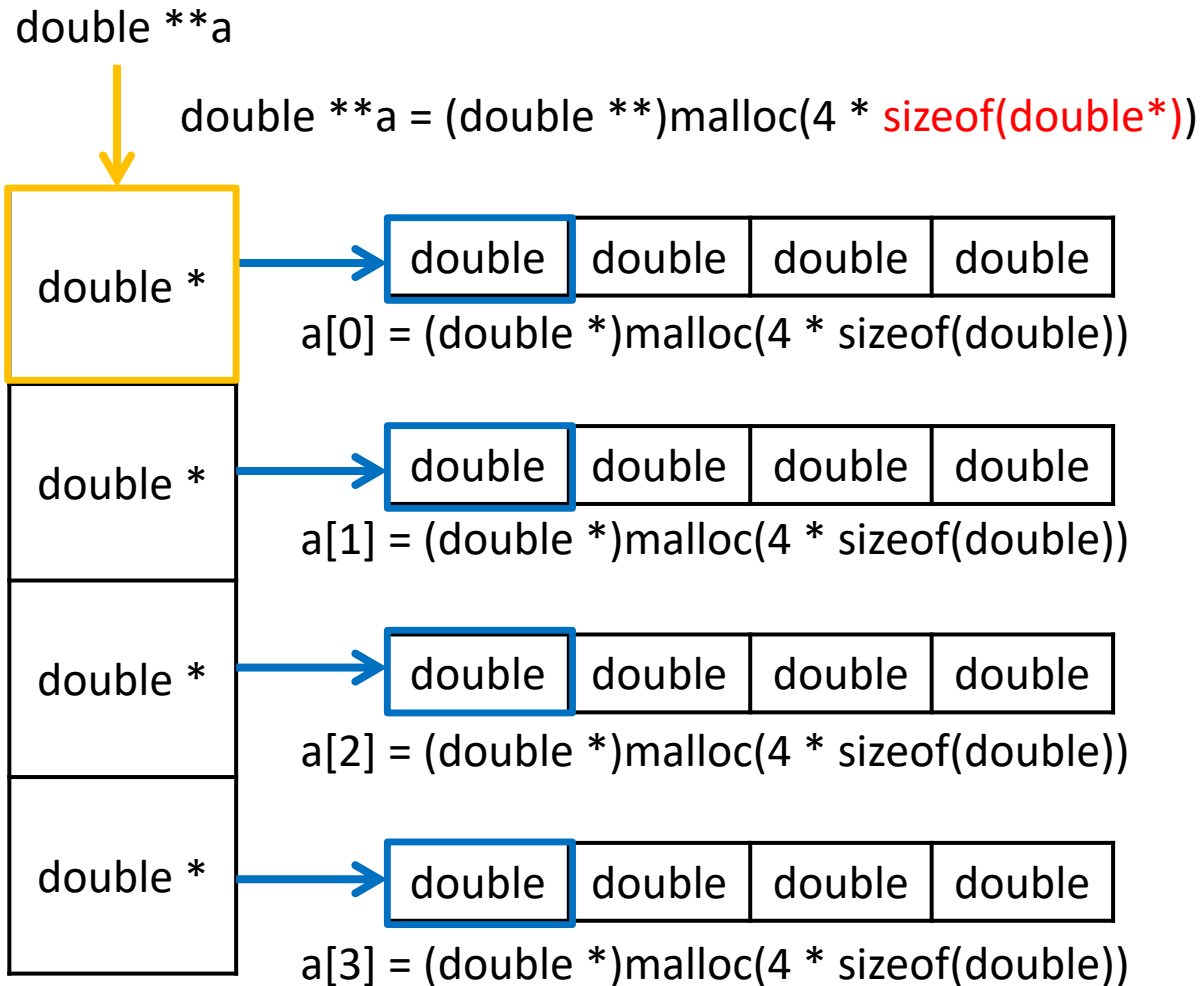
先頭はaで表す

1箱の大きさはdouble: キャスト(`double *`)

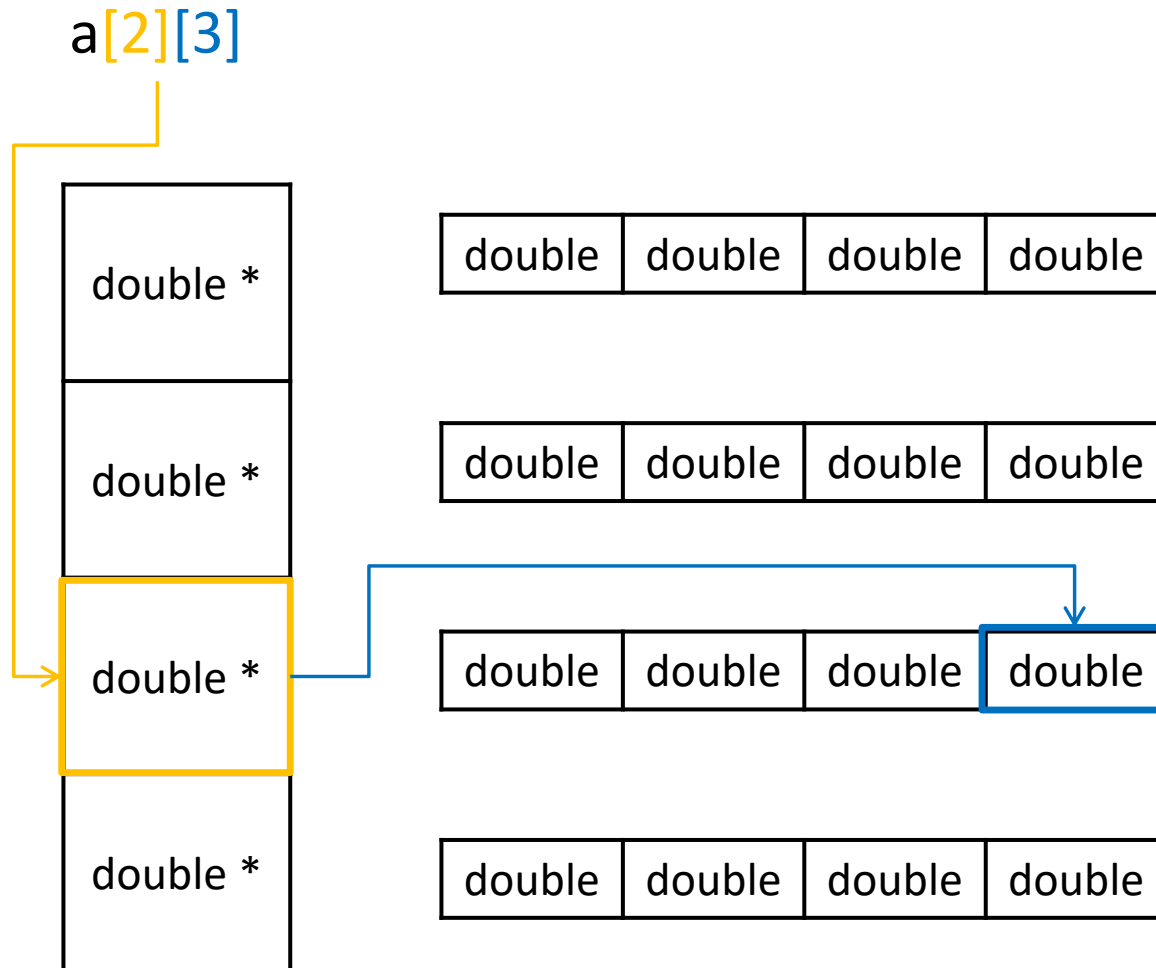
③ 使い終わったら後片付けをします.

```
free(a)
```

2次元配列



2次元配列(使用時)



3次元以上も同様にして作成可能

補足

領域解放する際は、末端から順に解放する.

例えば、`double **matrix` に2次元配列を割り当てた場合

`free(matrix)`

を先にやってしまうと`matrix[0]` で作った領域が解放できない.

```
for (行)
```

```
{
```

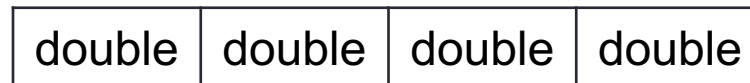
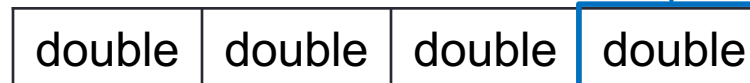
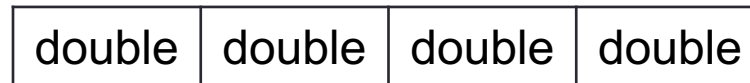
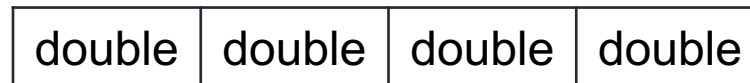
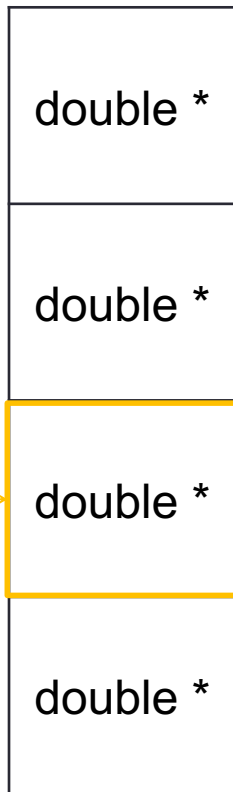
```
    free(matrix[行])
```

```
}
```

```
free(matrix)
```


a[2][3]

領域は消えていない



どこ？

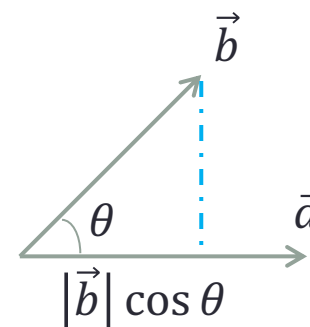
ベクトルの演算

- 和・差

成分ごとの和・差を計算.

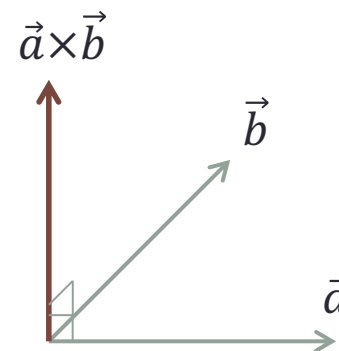
- 内積 $\vec{a} \cdot \vec{b}$

$$|\vec{a}| |\vec{b}| \cos \theta = \sum_{i=1}^n a_i b_i$$



- 外積 $\vec{a} \times \vec{b}$

大きさ $|\vec{a}| |\vec{b}| \sin \theta$, \vec{a} と \vec{b} に直交

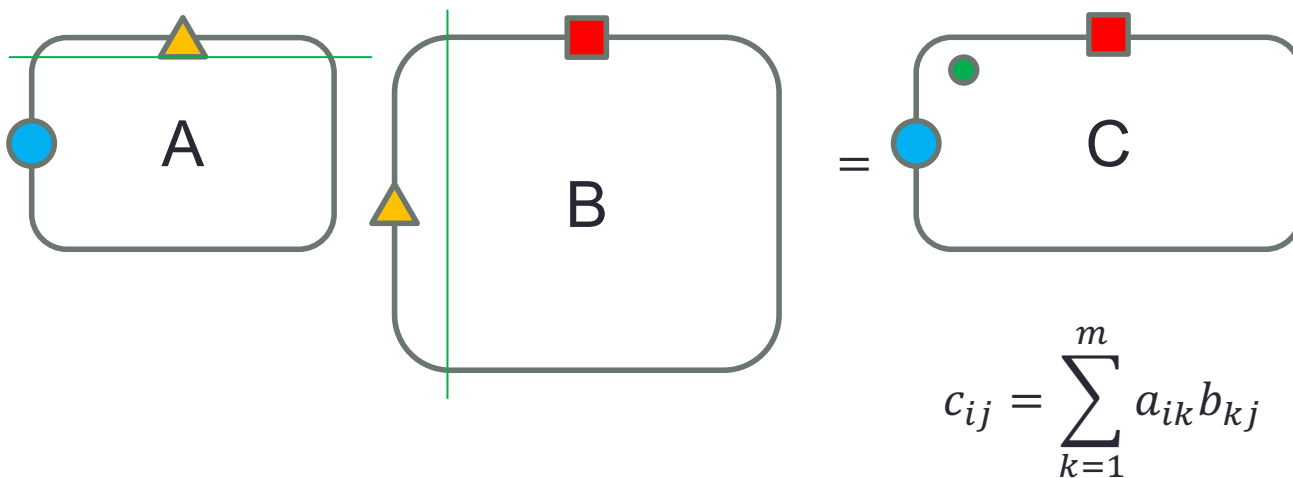


行列の演算

- 和・差

成分ごとの和・差を計算.

- 積



行列の演算

- 転置

行と列を入れ替える

逆行列

正方行列Aに対して,

$$AX = XA = E$$

となる行列Xを逆行列といい, A^{-1} と記述する.

逆行列が存在するとき行列Aは正則であるといい,
逆行列が存在しなければ特異であるという.

ベクトルノルム

長さ(距離), 大きさの概念の一般化

$$\sqrt[p]{\sum_{i=1}^n |x_i|^p}$$

- $p=2$ (2-ノルム), $n=2$ (2次元ベクトル)の場合

$\sqrt{(x_1)^2 + (x_2)^2}$: ベクトルの長さ(大きさ), ユークリッドノルム

ベクトルノルム

- $p=1$ の場合 (1-ノルム), $n=2$ では

$$|x_1| + |x_2| : \text{絶対値の総和}$$

- $p=\infty$ の場合 (∞ -ノルム, 最大値ノルム), $n=2$ では

$$(|x_1|^\infty + |x_2|^\infty)^{\frac{1}{\infty}} = \max_{i=1,2} |x_i|$$

最大値ノルムの証明

$$(|x_1|^\infty + |x_2|^\infty)^{\frac{1}{\infty}}$$

最大値では1
最大値以外では1より小さい

最大値を仮に x_m と置く

$$(|x_1|^\infty + |x_2|^\infty)^{\frac{1}{\infty}} = \left\{ |x_m|^\infty \left(\frac{|x_1|}{|x_m|} \right)^\infty + |x_m|^\infty \left(\frac{|x_2|}{|x_m|} \right)^\infty \right\}^{\frac{1}{\infty}}$$

$$\begin{aligned} &= \{ |x_m|^\infty (\text{1})^\infty + |x_m|^\infty (\text{1以下})^\infty \}^{\frac{1}{\infty}} \\ &= |x_m|^{\infty \frac{1}{\infty}} \\ &= |x_m| \end{aligned}$$

ソート

数(データ)をあるルールに従って並べかえること.

ex) 学籍番号順, 50音順, アルファベット順

- バブルソート
- クイックソート
- 選択ソート
- ヒープソート
- etc.

バブルソート

3	1	5	9	2
---	---	---	---	---

1個目と2個目の要素を比較し、小さい順に並べる.

1	3	5	9	2
---	---	---	---	---

2個目と3個目の要素を比較し、小さい順に並べる.

1	3	5	9	2
---	---	---	---	---

3個目と4個目の要素を比較し、小さい順に並べる.

1	3	5	9	2
---	---	---	---	---

4個目と5個目の要素を比較し、小さい順に並べる.

1	3	5	2	9
---	---	---	---	---

n個の要素に対し、**n-1回の比較**で1セット. 1セット終わると最大値が最後尾に来る.

バブルソート

2セット目, また先頭から順に同じことを実行する.

1	3	5	2	9
---	---	---	---	---

2番目に大きい数が最後から2番目の要素に入る.

1	3	2	5	9
---	---	---	---	---

n-1セット繰り返すことで, データは小さい順に整列される.

計算量: $(n - 1)^2$

実際には不要な比較・交換を省けるのでもう少し下がる

クイックソート

3	1	5	9	2
---	---	---	---	---

ピボットを選び、数列をピボット以下と以上に分割

3	1	2	5	9
---	---	---	---	---

分割されたそれぞれで同じことを繰り返す

1	2	3
---	---	---

1	2
---	---

計算量: $Q_n = n - 1 + Q_a + Q_b$
(Q_a と Q_b はサブセットの計算量)

最悪値: バブルソートと同じ, 平均値: $2n \log n$

計算量, Order

計算量の概略を比較するとき, O という表記が使われる.

- 高い次数のものを抜き出す
- 係数は省く

ex)

バブルソート $(n - 1)^2 \rightarrow O(n^2)$

クイックソートの平均 $2n \log n \rightarrow O(n \log n)$

参考: qsort関数

```
void qsort(void *array, size_t n, size_t m,  
           int (*comp)(const void *a1, const void *a2) )
```

比較関数へのポインタ `int (*comp)(const void *a1, const void *a2))` が必要

講義中にソートが必要な際は,

- qsort関数
 - 自作のソート関数(過去のプログラム授業で作ったもの)
- どちらを使っても良い.

演習課題2

(予備) 以下の計算をおこなえ

$$(1) \begin{pmatrix} 1 \\ 5 \\ 8 \end{pmatrix} + \begin{pmatrix} 3 \\ 7 \\ -2 \end{pmatrix}$$

$$(2) \begin{pmatrix} 1 \\ 5 \\ 8 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 7 \\ -2 \end{pmatrix}$$

k2-input.csvから行列Aを, k2-input2.csvからベクトルxを入力し, 行列積Axの計算を行うプログラムを作成せよ。

例

$$\begin{pmatrix} 2 & 8 & 4 \\ 3 & 2 & -1 \\ 7 & -1 & 3 \end{pmatrix} \begin{pmatrix} 3 \\ 7 \\ -2 \end{pmatrix}$$

余談

- 行列積って役に立つの？

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 1/3(x_1 + x_2 + x_3) \\ 1/3(x_2 + x_3 + x_4) \\ 1/3(x_3 + x_4 + x_5) \\ 0 \end{pmatrix}$$

移動平均の計算

課題について

- 2週間に1回程度の課題
- 2週目終了後2週間で提出締切(今回は10/25)
- 提出ファイルが複数になる場合はzip圧縮して1ファイルにする
- Blackboard上の提出ページか, readme.txtにて説明を加える
 - コンパイル環境, コンパイル方法, 実行方法等
- 配布したライブラリに関しては変更していなければ添付不要
 - 説明にライブラリ使用について記載すること
- 動作確認用のデータファイルは課題と同時に掲載
- 評価用データファイルは評価の時間に公開