線形代数演算ライブラリBLASとLAPACKの基礎と実践 (I)
BLAS, LAPACK入門編

中田 真秀
理化学研究所 情報システム本部
2019/5/23 計算科学技術特論A

13:00

BLAS, LAPACK入門編 講義目的

- 線形代数演算をコンピュータで行うには、必ずBLAS LAPACKのお世話になる。
- 使うには(若干)知識がいる。
- 実際にUbuntu Linuxで試せる形で提示し、使えるようになる。
  - 例: 行列-行列積(BLAS) + 行列の対角化 (LAPACK)

BLAS, LAPACK入門編 講義内容

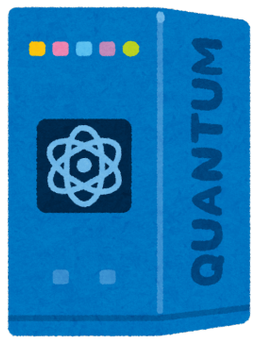
- 線形代数を勉強しよう!
  - 様々な応用があり、とても重要
- 線形代数とコンピュータの歴史
  - 有史以来ずっと人類は線形代数してる。
  - コンピュータでも線形代数演算 & コンピュータは速いが正義
- コンピュータでの数値計算
- コンピュータでの線形代数演算
- BLAS, LAPACKの紹介
  - 珠玉のライブラリ
- BLAS LAPACK豆知識
- BLASを使ってみる: 行列-行列積
- LAPACKを使ってみる: 対称行列の対角化
- まとめと次回予告

線形代数を勉強しよう!

- いまからでも遅くないから線形代数ちゃんと勉強しとこう
  - 線形代数≒連立一次方程式を解く
  - かなり抽象的 — 応用先がたくさんあり、つぶしが利く
- 重要な応用例
  - 機械学習

  - 三次元コンピュータグラフィックス

  - 量子コンピュータ



機械学習で必要になる線形代数

- 行列の定義　$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$
- ベクトル、基底、一次独立　$b = (b_1, b_2, b_3, \cdots, b_n)$
- 行列同士の足し算、スカラー倍
- ベクトル同士の足し算、スカラー倍、内積　$a \cdot b = \sum_{i}^{n} a_i b_i$
- 行列と行列の掛け算
- 連立一次方程式を解く、固有値、逆行列

13:05

量子コンピュータで必要な線形代数

- 量子コンピュータは無限次元の線形代数≒Hilbert空間論
  - ざっくり有限次元の線形代数と思って良い(数学の先生の前では言わないように)
- ベクトル、行列の基本的知識

$$H_{ij} = H^*_{ji}$$

- 出てくる行列は、エルミート行列とユニタリ行列

$$U^{-1}_{ij} = U^*_{ji}$$

- 行列の固有値と固有ベクトル

$$Hx = \lambda x$$

  - エルミート行列をユニタリ行列で対角化

$$U^\dagger H U = \begin{pmatrix} \lambda_1 & & & & 0 \\ & \lambda_2 & & & \\ & & \lambda_3 & & \\ & & & \ddots & \\ 0 & & & & \lambda_\infty \end{pmatrix}$$

線形代数とコンピュータの歴史

- 人類は線形代数を有志以来やってきた。エジプトが最古 (パピルス)、中国もガウスの消去法は 1000 年以上前に知っていた (九章算術; 紀元前1世紀から紀元後2世紀ころ)。

- あらゆる分野に線形代数がでてくる:物理、化学、工学、生物学、経済、経営...

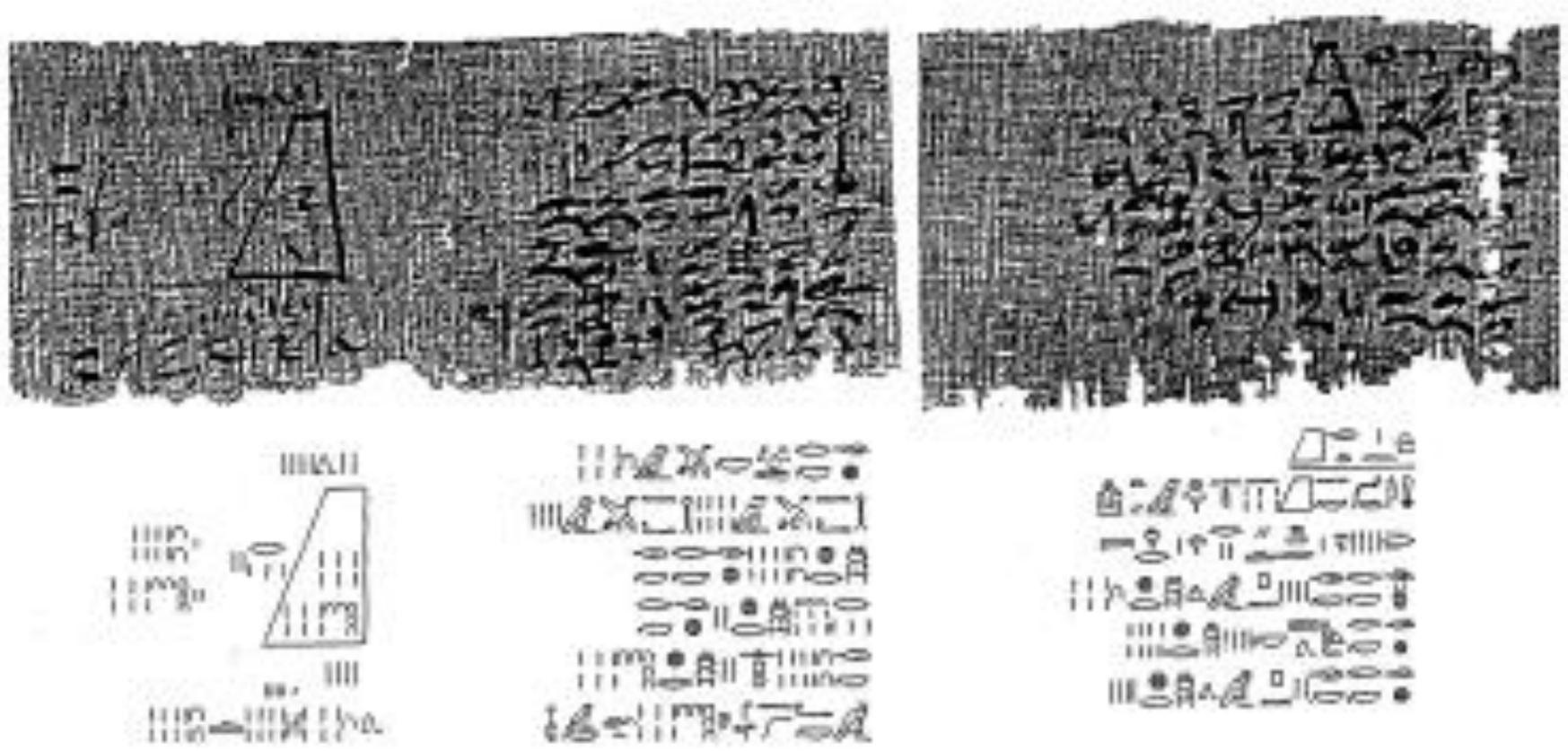
- コンピュータが生まれ、線形代数演算をコンピュータ上で高速に、大量にやらせることが重要になってきた

- コンピュータの歴史を振り返ると、処理速度のみを追求してきてる。
  - 遅いコンピュータは市場から消える

The Rhind Papyrus

紀元前1550頃。Rhindによりエジプトのルクソールで発見された。
Book I, II, III とあり、Book I のProblem 32 に、
現在の表記で x + 1/3 x + 1/4 x = 2 をxについて解け
という問題が出ている

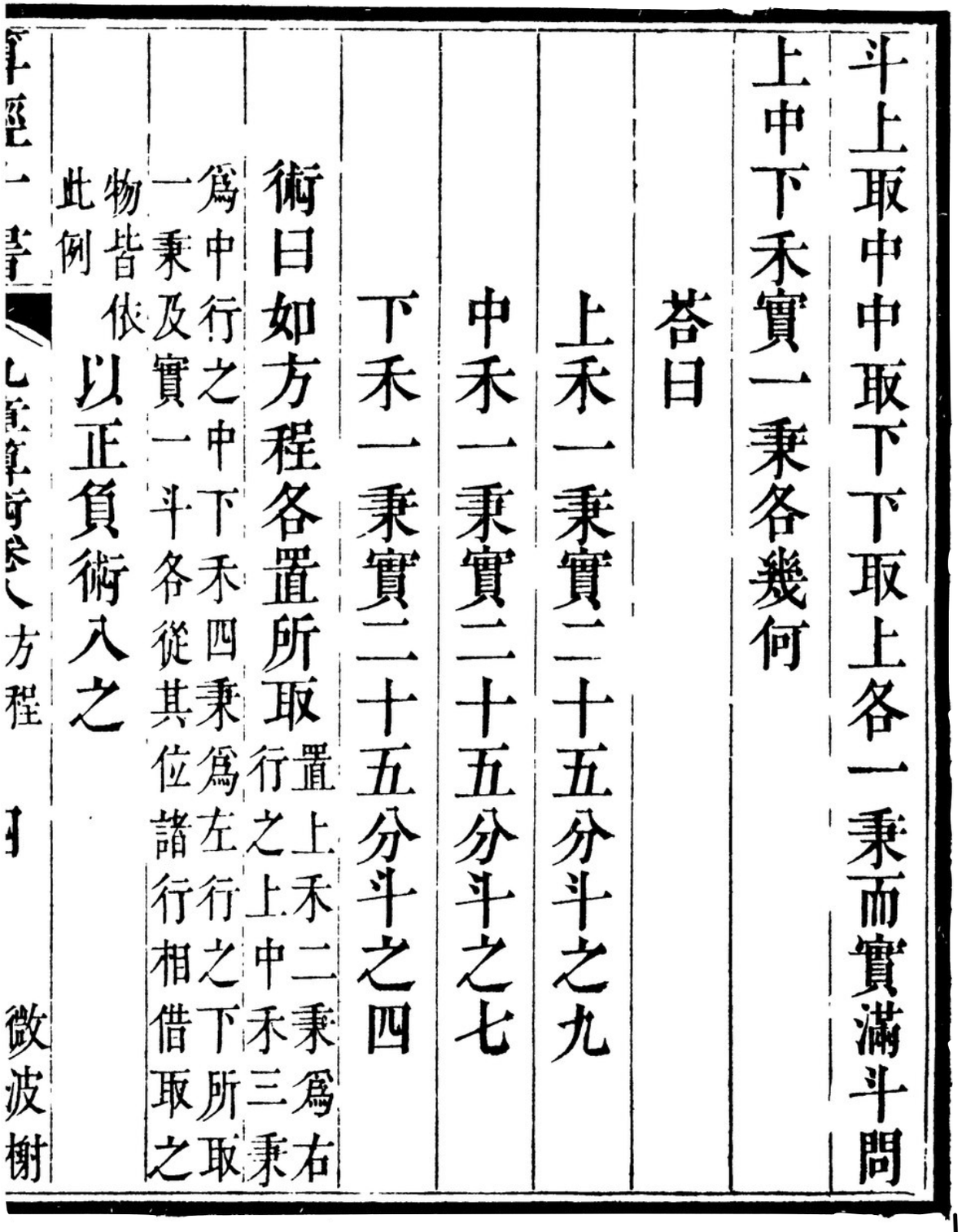
The Moscow Mathematics Papyrus



紀元前1850頃。ゴレニシチョフよりエジプトより持ち帰った。
Problem 1, 19, 25 に アハ問題 (アハは未知数の意味)がある。
19は(11/2)x + 4 = 10を解けという問題
図は四角の台の体積を求める問題

九章算術 方程より

- 中国、紀元前1世紀から紀元後2世紀ころ、著者不
- 263年頃魏の時代に劉徽（りゅうき）によって整理と注釈が加えられた。
- https://ctext.org/nine-chapters/fang-cheng/zh
- 人類史上初めての連立一次方程式をGaussの消去法で解いたと思われる。
- 今でも1000年以上前の文が何となく読めるのは凄い


斗上取中中取下下取上各一秉而實滿斗問
上中下禾實一秉各幾何
荅曰
上禾一秉實二十五分斗之九
中禾一秉實二十五分斗之七
下禾一秉實二十五分斗之四
術曰如方程各置所取置上禾二秉爲右
爲中行之中下禾四秉爲左行之下所取
一秉及實一斗各從其位諸行相借取之
物皆依此例以正負術入之
方程 一
徽波樹

https://zh.wikisource.org/wiki/Page:Sibu_Congkan0392-劉徽-九章算術-3-3.djvu/8 より

九章算術 方程より 問題

- 有上禾三秉，中禾二秉，下禾一秉，實三十九斗；上禾二秉，中禾三秉，下禾一秉，實三十四斗；上禾一秉，中禾二秉，下禾三秉，實二十六斗。問上、中、下禾實一秉各幾何？答曰:上禾一秉，九斗、四分斗之一，中禾一秉，四斗、四分斗之一，下禾一秉，二斗、四分斗之三。

- 方程術曰，置上禾三秉，中禾二秉，下禾一秉，實三十九斗，於右方。中、左禾列如右方。以右行上禾遍乘中行而以直除。又乘其次，亦以直除。然以中行中禾不盡者遍乘左行而以直除。左方下禾不盡者，上為法，下為實。實即下禾之實。求中禾，以法乘中行下實，而除下禾之實。餘如中禾秉數而一，即中禾之實。求上禾亦以法乘右行下實，而除下禾、中禾之實。餘如上禾秉數而一，即上禾之實。實皆如法，各得一斗。

九章算術 方程より 現代語訳

Powered by Google翻訳

- 問: 3束の上質のキビ、2束の中質のキビ、1 束の低質のキビが39個のバケツに入っている。2束の上質のキビ、3束の中質のキビ、1束の低質のキビが34個のバケツに入っている。1束の上質のキビ、2束の中質のキビ、3束の低質のキビが26個のバケツに入っている。上質、中質、低質のキビ1束はそれぞれバケツいくつになるか。

- 答: 上質 ９ ¼ , 中質 ４ ¼ , 低質 ２ ¾ 個づつ

- 上質のキビ3束、中質のキビ3束、低質のキビ1束を39バケツを右行に置く。中行、左行も右のように並べる。右の上質を中行にかけ、右行で引く。また左行にもかけて右行から引く。次に、中行の中質のキビの余りを左行にかけて、中行で引く。左の低質に余りがあるのでそして、割れば求まる(実を法で割る)。以下略

九章算術 方程より 現代語訳

Powered by Google翻訳

- 問

$$\begin{cases} 3x+2y+z=39\cdots(\text{右}) \\ 2x+3y+z=34\cdots(\text{中}) \\ x+2y+3z=26\cdots(\text{左}) \end{cases}$$

- (右)はそのまま、(中)は(中)を3倍したものから(右)を2倍したものを引き、(左)を3倍して(左)から(右)を引く。

$$\begin{array}{ll} 3(2x+3y+z=34) & 3(x+2y+3z=39) \\ 2(3x+2y+z=39) & 3x+2y+z=39 \\ \hline 5y+z=24\cdots(\text{中}) & \hline 4y+8z=39\cdots(\text{左}) \end{array}$$

- それから(左)を5倍する

$$\begin{cases} 3x+2y+z=39\cdots(\text{右}) \\ 5y+z=24\cdots(\text{中}) \\ 20y+40z=195\cdots(\text{左}) \end{cases}$$

$$36c=99$$

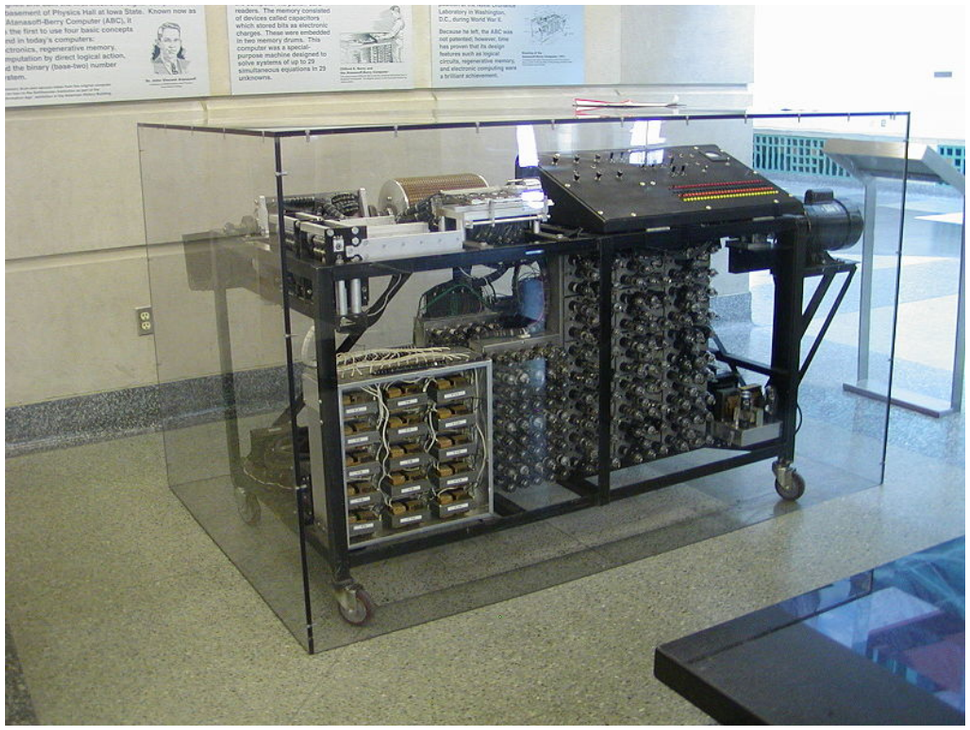
あとは略

近現代の線形代数

- 1693年ライプニッツ、1750年頃クラメール、1888年ペアノ
- 1900年～1920頃 無限次元の線形代数=ヒルベルト空間(=量子力学)
- 1950年代～コンピュータ上での線形代数の発達(LU分解、固有値分解など)

ABC: Atanasoff-Berry コンピュータ

- 世界初のコンピュータとされる
  - 1937-1942年頃
  - ENIACは1946年
- 1秒30回の加減算
- 50ビット固定小数点
- 60Hz
- 最大29元連立一次方程式を解けた



ABCの復元機@アイオワ州立大学

- 論理は真空管で実装
- 機械やアナログ部分がない
- 2進数の概念、史上初導入

13:45

コンピュータでの数値計算

コンピュータでの実数演算はどうするか

- コンピュータは有限の整数しか扱えないため、特別な表記(フォーマット)を使う。

- 浮動小数点数表記 2進数で32桁、64桁などのビット列を実数とみなす。

- 浮動小数点数は、符号、仮数部(fraction)、指数部(exponent)、a_nは0 or 1から成る

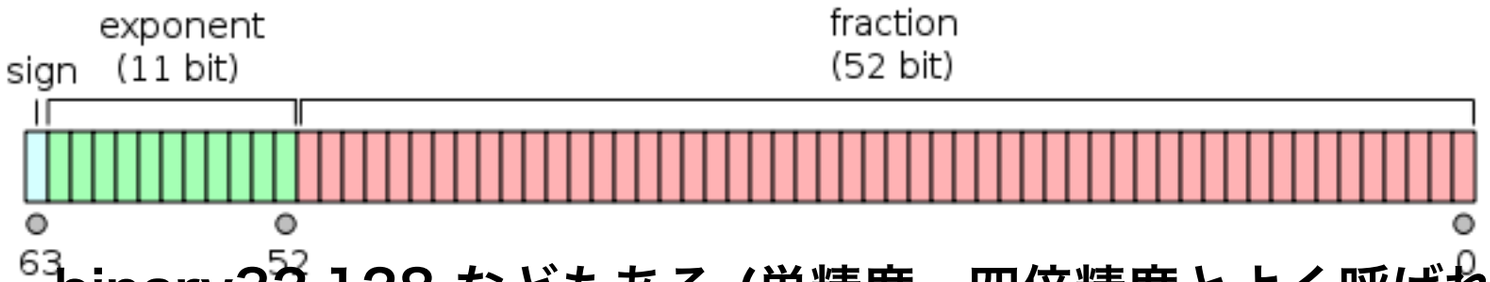
$$\pm\left(1+\sum_{n=1}^{k} a_n \overbrace{\left(\frac{1}{2}\right)^n}^{fraction} \times \overbrace{2^m}^{exponent}\right)$$

- 浮動小数点数を10進数で表す例。4ビット2進数-1.101×2^5　を10進数になおす。

$$-1.101\times 2^5 = -\left(1+1\times 0.5+0\times 0.25+1\times 0.125\right)\times 32 = 52$$

コンピュータでの数の取扱いbinary64 (倍精度)

- “754-2008 IEEE Standard for Floating-Point Arithmetic”
  - binary64 フォーマットは 10進16桁の有効桁がある (よく倍精度とよぶ)



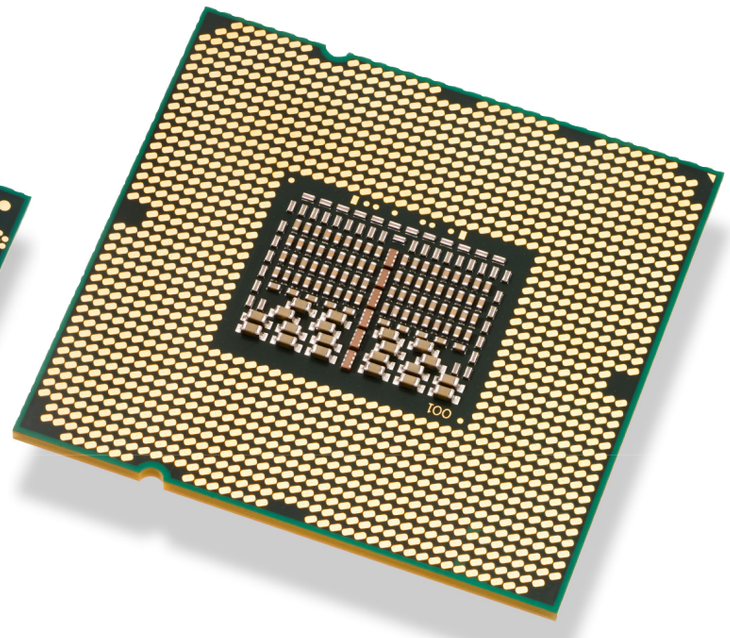
$$\pm\left(1+\sum_{n=1}^{52} a_n \overbrace{\left(\frac{1}{2}\right)^n}^{fraction} \times \overbrace{2^{-1022\sim1023}}^{exponent}\right)$$

  - binary32,128 などもある (単精度、四倍精度とよく呼ばれる)。

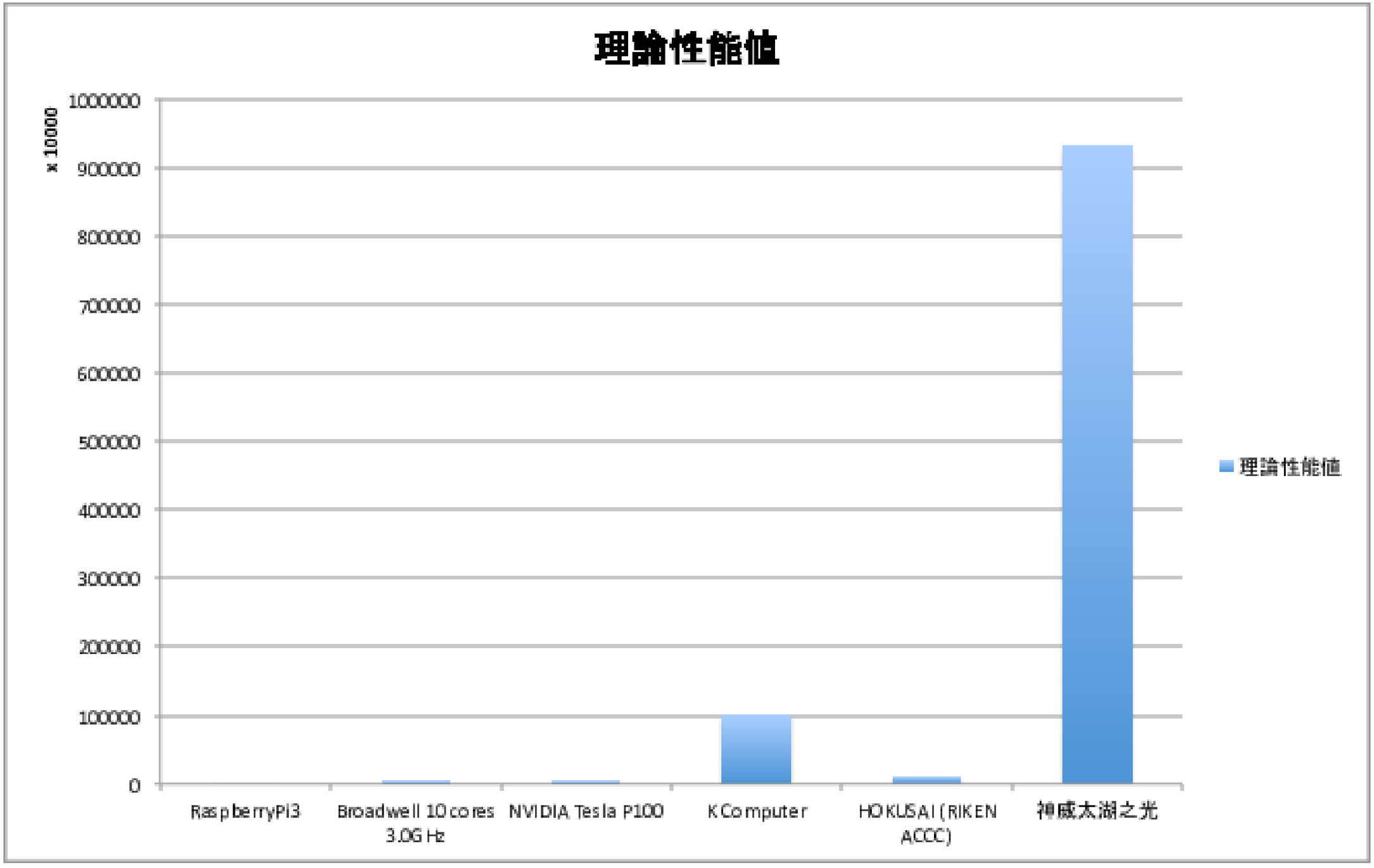
- この規格に則って演算する場合がほとんど(最近の例外:PlayStation2)
  - 規格が無いときはコンピュータ毎に違う結果が得られたりした

- FLOPS(フロップス）という単語が頻出
  - 1秒間に1回浮動小数点数が計算できること=Floating point operation per second
  - 速さ:Core i7 (Broadwell, 10 cores, 3.5GHz): 560GFlops, NVIDIA TESLA P100 5.3TFLOPS, 京コンピュータ10PFLOPS, HOKUSAI (1PFlops), 神威太湖之光 (93.01PFlops)

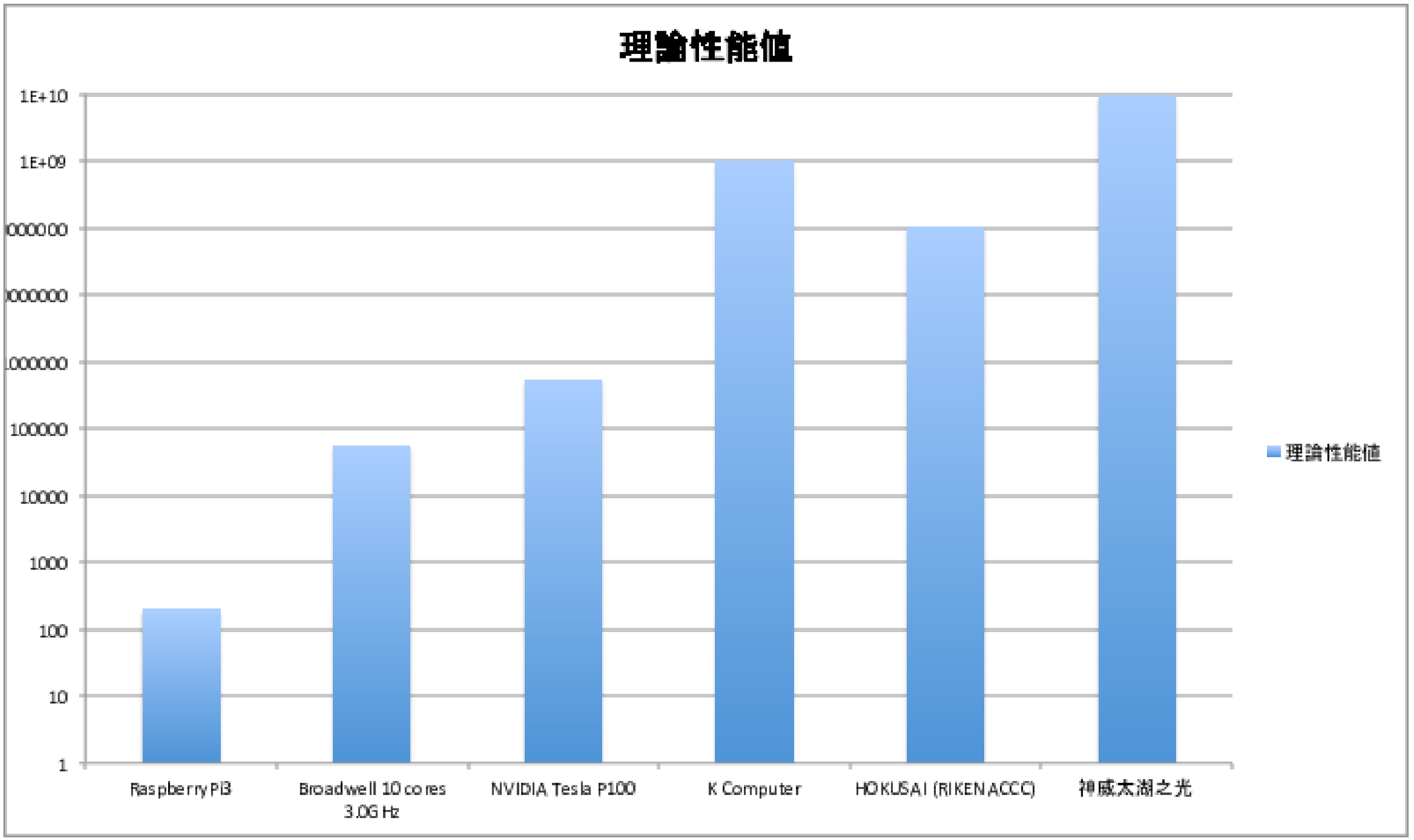
現在のコンピュータ



倍精度の計算スピードの比較



倍精度の計算スピードの比較:logスケールで



コンピュータでの実数演算の注意点

- 精度が有限であるので誤差が入る。
  - 例えば倍精度は10進16桁の精度をもつので、以下が成り立ってしまう

$$1 + 0.0000000000000001 = 1$$

  - 結合法則が成り立たない場合がある

$$a + (b + c) \neq (a + b) + c$$

  - どのように演算結果を丸めるか(=四捨五入みたいな感じ)で、一番下のbitの0,1が変化する。

コンピュータでの数値計算に再現性はあるか?

- 問題 C=AB という行列の掛け算について、同じコンピュータで同じ計算を何回か行ったときの結果を考える。このとき、どうなるか?

- a) 毎回全くbit単位で同じ結果が出る。

- b) 毎回違った結果が出る。

- c) bit単位で全く同じ結果が出る場合もあるが違う結果が出る場合もある。

コンピュータでの数値計算に再現性はあるか?

- 答え
- c) 違う結果が出る場合がある。最近のコンピュータはマルチスレッドで、足し算の順序がコンピュータの都合で変わることがある。従って、結合法則が成り立たないことがあることより、違う結果が出る場合がある。

コンピュータでの実数演算で変な値が出る例

- 驚くべき例!

問: a を変えた場合、float ((18+a) - a) はどんな値を取りうるか。

答:

(a) 18のみ。

(b) 0を取る場合がある

(c) それ以外

コンピュータでの実数演算で変な値が出る例

答えは(c)でした。

```
$ cat test.c
#include <math.h>
#include <stdio.h>

int main()
{
   double a = 18.0;
   double b = pow(2,57);
   printf("%lf\n", (a+b) - b);
}
$ gcc test.c ; ./a.out
32.000000
```

18に同じ数を足して引いただけなのにおかしい結果がでてきた。

special thx to 山中先生

コンピュータの数値計算に再現性はあるか? Wii版 Super Mario64

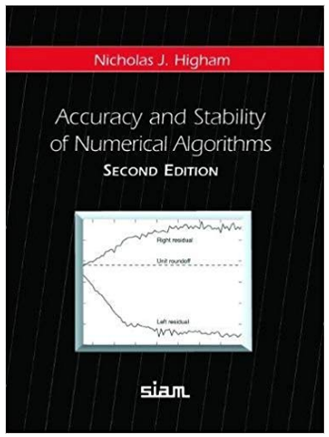
- いかにAボタンを使わずにsuper Mario 64をクリアするという競技があるらしい。
- Wii版Super Mario64で、「ほのおの うみ のクッパ」で3日待機すると、Aボタンを一度も押さずにクリアできるとのこと。
- スタート地点に近い足場がどんどん浮上して行くバグが混入された
- 理由は
  - 倍精度から単精度に変換する場合、Nintendo 64とWii Virtual Consoleで違っていた。
  - 64では最近接丸め、VCではゼロ方向への丸めを選んだ。→ 最下位 1bit ずつずれる
  - この違いによりWii版では、少しずつ浮上する。 → 1bit 誤差を3日つみかさねたら...

https://sbfl.net/blog/2018/06/10/wii-mario64-platform-bug/

コンピュータでの線形代数

自作は避けたほうがいい

- 線形代数演算をコンピュータでやるとき、プログラムを自作する場合があるかもしれないが、**自作は避けたほうがいい**
- クラメールの公式で線形連立一次方程式を解く。
  - 行列が少し大きくなるとすぐ解けなくなる
- 行列式を求める。
  - 誤差が大きくなる。
- 固有値を求めるとき安定しない。
- 行列-行列の積を求める。
  - カタログに出てる理論性能値と比べて、大変遅くなる。
- 他にもノウハウがいっぱいある…



数値アルゴリズムにおける精度と安定性

Accuracy and Stability of Numerical Algorithms, N. Higham 2002

計算科学のためのHPC技術1

安直だが **ライブラリを用いるのが正義**

自作して失敗する例:連立一次方程式をクラメールの公式で解く

- 以下のような連立一次方程式を解く。

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \quad A := \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \quad x := \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad b := \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$$Ax = b$$

- 解は

$$A_i := \begin{pmatrix} a_{1,1} & \cdots & a_{1,i-1} & b_1 & a_{1,i+1} & \cdots & a_{1,n} \\ a_{2,1} & \cdots & a_{2,i-1} & b_2 & a_{2,i+1} & \cdots & a_{2,n} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,i-1} & b_n & a_{n,i+1} & \cdots & a_{n,n} \end{pmatrix}$$

- として、行列Bに対する行列式det(B)を用いて、以下のようにかける。

$$x_i = \frac{\det(A_i)}{\det(A)}$$

18:00

自作して失敗する例:連立一次方程式をクラメールの公式で解く

- 理論的にはクラメールの公式を使ってプログラムを作ると、コンピュータで解ける。

- しかしながら、行列のサイズ n が大きくなるとすぐ解けなく￥

- n=10で 36288000 回の演算

- n = 100 で 9.3×10^{157}回の演算

- あっというまに宇宙時間より長い時間が必要になる。

分野の違いと意識の違い

(偏見あり)

- 数学者の意識：原理的に可能, 解の存在のみ興味ある場合が多い
- 情報系の数学より：アルゴリズムが多項式程度なものを考えたがる
- 自然科学系研究者：ともかく答えが求まる方法なら何でも良い
    - とりあえず求まればよい。問題が出るまで放置。よく指数関数的なアルゴリズムを意識せずにゴリ押しする。
- HPC or 数値解析系研究者：1 clockでも速い方法 1bitでも転送量が少ない方法、1桁でも精度の良い方法などから選択
    - ハード依存高め。さまざまな現実的な制限を考慮し、なるべく良い結果を出す。

分野の違いと意識の違い (偏見あり)

連立一次方程式をどう解くか

- 数学者の意識：行列式が0でないなら解ける。終わり

- 情報系の数学より：LU分解も逆行列求める方法もオーダーは同じO(n^3)なので違いはない、クラメールはO(n!)なんで論外。

- 自然科学系研究者：とりあえず逆行列求めとこう。LU分解って何? 解きたい問題は別だし他に考えるべきことが多いし後まわしにしよう。

- HPC or 数値計算系研究者：LU分解経由一択。基本であるdgemmだからキャッシュヒットも高いし、逆行列を使うより誤差も少ない。よってLU分解経由以外あり得ない。クラメールは誤差も大きい。

線形代数演算ならBLAS, LAPACKを使おう

- 行列、ベクトルなどの線形代数演算のライブラリはある?
  - BLASやLAPACKを用いましょう。
- コンピュータの仕組みを軽く知っておくと、手軽に高速なライブラリの利用方法がわかる
  - 来週やります
- 今回はBLAS, LAPACKをとりあえず使ってみる、ということをやります

BLAS, LAPACK:世界最高の線形代数演算パッケージ

- コンピュータで線形代数演算するならBLAS+LAPACKを使いましょう。
- 品質、信頼性がとても高く、無料で入手できる
  - http://www.netlib.org/lapack/
- 標準で高速版がインストールされていることが多い
  - http://www.openblas.net/
- 密行列のみ対応(疎行列は他のライブラリを利用する)。
- コンピュータでの行列線形代数演算の基礎中の基礎!

BLAS, LAPACKは人類の至宝

14:05

BLAS：基本的な線形代数サブプログラム群

- BLASはBasic Linear Algebra Subprogramsの略
- 基礎的な線形代数の「サブ」プログラム Level 1 〜 3まである
  - Level 1：ベクトル-ベクトルの内積など　$x \cdot y$
  - Level 2：行列-ベクトル積など　$y \leftarrow \alpha Ax + \beta y$
  - Level 3：行列-行列積など　$C \leftarrow \alpha AB + \beta C$
- FORTRAN77でさまざまなルーチンの仕様を提供している。
- 参照実装の形で提供されている (Reference BLAS)
- BLASのルーチンを「ブロック」にしてより高度なことをする。
- この実装を「お手本」とする
- とても美しいコード!
- 高速版もある。

Level 1 BLAS

- **Level 1:ベクトル-ベクトル演算(+そのほか)のルーチン**
- **ベクトルの加算 DAXPY**

$$y \leftarrow \alpha A x + \beta y$$

- **内積計算: DDOT**

$$< x, y > = \sum_{i}^{N} x_i y_i$$

- **2-ノルム計算**

$$||x|| = \sqrt{\sum_{i} |x_i|^2}$$

- など15種類あり, さらに単精度, 倍精度, 複素単精度, 複素数倍精度についての4通りの組み合わせがある.

Level 2 BLAS

- **Level 2:行列-ベクトル演算のルーチン**
- **行列-ベクトル積: DGEMV**

$$y \leftarrow \alpha Ax + \beta y$$

- **上三角行列とベクトルの積:DTRMV**

$$x \leftarrow Ax$$

- **上三角行列の連立一次方程式を解く:DTRSV**

$$x \leftarrow A^{-1}x$$

- **列ベクトルと行ベクトルの積: DGER**

$$A \leftarrow \alpha xy^t + A$$

- **など25種類あり, 同じように単精度、倍精度、複素数の4通りの組み合わせがある。**

Level 3 BLAS

- **Level 3 BLASは行列-行列演算のルーチン群**
- **行列-行列積: DGEMM**

$$C \leftarrow \alpha AB + \beta C$$

- **対称行列-行列積: DSYMM**

$$C \leftarrow \alpha AB + \beta C$$

- **上(下)三角行列と行列の積: DTRMM**

$$B \leftarrow \alpha AB$$

- **対称行列の階数nの更新: DSYRK**

$$C \leftarrow \alpha AA^T + \beta C$$

- **上三角行列の連立一次方程式を解く: DTRSM**

$$B \leftarrow \alpha A^{-1}B$$

- **など9種類ある。**

BLAS Quick Reference

Level 1 BLAS

	dim	scalar	vector	vector	scalars		5-element array		prefixes
SUBROUTINE xROTG (					A, B, C, S)			Generate plane rotation	S, D
SUBROUTINE xROTMG(					D1, D2, A, B,		PARAM)	Generate modified plane rotation	S, D
SUBROUTINE xROT (	N,		X, INCX,	Y, INCY,		C, S)		Apply plane rotation	S, D
SUBROUTINE xROTM (	N,		X, INCX,	Y, INCY,			PARAM)	Apply modified plane rotation	S, D
SUBROUTINE xSWAP (	N,		X, INCX,	Y, INCY)				$x \leftrightarrow y$	S, D, C, Z
SUBROUTINE xSCAL (	N,	ALPHA,	X, INCX)					$x \leftarrow \alpha x$	S, D, C, Z, CS, ZD
SUBROUTINE xCOPY (	N,		X, INCX,	Y, INCY)				$y \leftarrow x$	S, D, C, Z
SUBROUTINE xAXPY (	N,	ALPHA,	X, INCX,	Y, INCY)				$y \leftarrow \alpha x + y$	S, D, C, Z
FUNCTION xDOT (	N,		X, INCX,	Y, INCY)				$dot \leftarrow x^T y$	S, D, DS
FUNCTION xDOTU (	N,		X, INCX,	Y, INCY)				$dot \leftarrow x^T y$	C, Z
FUNCTION xDOTC (	N,		X, INCX,	Y, INCY)				$dot \leftarrow x^H y$	C, Z
FUNCTION xxDOT (	N,		X, INCX,	Y, INCY)				$dot \leftarrow \alpha + x^T y$	SDS
FUNCTION xNRM2 (	N,		X, INCX)					$nrm2 \leftarrow \|\|x\|\|_2$	S, D, SC, DZ
FUNCTION xASUM (	N,		X, INCX)					$asum \leftarrow \|\|re(x)\|\|_1 + \|\|im(x)\|\|_1$	S, D, SC, DZ
FUNCTION IxAMAX(	N,		X, INCX)					$amax \leftarrow 1^{st} k \ni \lvert re(x_k)\rvert + \lvert im(x_k)\rvert = max(\lvert re(x_i)\rvert + \lvert im(x_i)\rvert)$	S, D, C, Z

Level 2 BLAS

	options		dim	b-width	scalar	matrix	vector	scalar	vector		
xGEMV (		TRANS,	M, N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$	S, D, C, Z
xGBMV (		TRANS,	M, N,	KL, KU,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$	S, D, C, Z
xHEMV (	UPLO,		N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xHBMV (	UPLO,		N,	K,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xHPMV (	UPLO,		N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xSYMV (	UPLO,		N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xSBMV (	UPLO,		N,	K,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xSPMV (	UPLO,		N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xTRMV (	UPLO,	TRANS, DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTBMV (	UPLO,	TRANS, DIAG,	N,	K,		A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTPMV (	UPLO,	TRANS, DIAG,	N,			AP,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTRSV (	UPLO,	TRANS, DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow A^{-1} x, x \leftarrow A^{-T} x, x \leftarrow A^{-H} x$	S, D, C, Z
xTBSV (	UPLO,	TRANS, DIAG,	N,	K,		A, LDA,	X, INCX)			$x \leftarrow A^{-1} x, x \leftarrow A^{-T} x, x \leftarrow A^{-H} x$	S, D, C, Z
xTPSV (	UPLO,	TRANS, DIAG,	N,			AP,	X, INCX)			$x \leftarrow A^{-1} x, x \leftarrow A^{-T} x, x \leftarrow A^{-H} x$	S, D, C, Z

	options		dim	scalar	vector	vector	matrix			
xGER (			M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)			$A \leftarrow \alpha xy^T + A, A - m \times n$	S, D
xGERU (			M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)			$A \leftarrow \alpha xy^T + A, A - m \times n$	C, Z
xGERC (			M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)			$A \leftarrow \alpha xy^H + A, A - m \times n$	C, Z
xHER (	UPLO,		N,	ALPHA,	X, INCX,		A, LDA)			$A \leftarrow \alpha xx^H + A$	C, Z
xHPR (	UPLO,		N,	ALPHA,	X, INCX,		AP)			$A \leftarrow \alpha xx^H + A$	C, Z
xHER2 (	UPLO,		N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)			$A \leftarrow \alpha xy^H + y(\alpha x)^H + A$	C, Z
xHPR2 (	UPLO,		N,	ALPHA,	X, INCX,	Y, INCY,	AP)			$A \leftarrow \alpha xy^H + y(\alpha x)^H + A$	C, Z
xSYR (	UPLO,		N,	ALPHA,	X, INCX,		A, LDA)			$A \leftarrow \alpha xx^T + A$	S, D
xSPR (	UPLO,		N,	ALPHA,	X, INCX,		AP)			$A \leftarrow \alpha xx^T + A$	S, D
xSYR2 (	UPLO,		N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)			$A \leftarrow \alpha xy^T + \alpha yx^T + A$	S, D
xSPR2 (	UPLO,		N,	ALPHA,	X, INCX,	Y, INCY,	AP)			$A \leftarrow \alpha xy^T + \alpha yx^T + A$	S, D

Level 3 BLAS

	options			dim	scalar	matrix	matrix	scalar	matrix		
xGEMM (		TRANSA,	TRANSB,	M, N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha op(A) op(B) + \beta C, op(X) = X, X^T, X^H, C - m \times n$	S, D, C, Z
xSYMM (	SIDE, UPLO,			M, N,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^T$	S, D, C, Z
xHEMM (	SIDE, UPLO,			M, N,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^H$	C, Z
xSYRK (	UPLO,	TRANS,		N, K,	ALPHA,	A, LDA,		BETA,	C, LDC)	$C \leftarrow \alpha AA^T + \beta C, C \leftarrow \alpha A^T A + \beta C, C - n \times n$	S, D, C, Z
xHERK (	UPLO,	TRANS,		N, K,	ALPHA,	A, LDA,		BETA,	C, LDC)	$C \leftarrow \alpha AA^H + \beta C, C \leftarrow \alpha A^H A + \beta C, C - n \times n$	C, Z
xSYR2K(	UPLO,	TRANS,		N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB^T + \bar{\alpha} BA^T + \beta C, C \leftarrow \alpha A^T B + \bar{\alpha} B^T A + \beta C, C - n \times n$	S, D, C, Z
xHER2K(	UPLO,	TRANS,		N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB^H + \bar{\alpha} BA^H + \beta C, C \leftarrow \alpha A^H B + \bar{\alpha} B^H A + \beta C, C - n \times n$	C, Z
xTRMM (	SIDE, UPLO,	TRANSA,	DIAG,	M, N,	ALPHA,	A, LDA,	B, LDB)			$B \leftarrow \alpha op(A) B, B \leftarrow \alpha B op(A), op(A) = A, A^T, A^H, B - m \times n$	S, D, C, Z
xTRSM (	SIDE, UPLO,	TRANSA,	DIAG,	M, N,	ALPHA,	A, LDA,	B, LDB)			$B \leftarrow \alpha op(A^{-1}) B, B \leftarrow \alpha B op(A^{-1}), op(A) = A, A^T, A^H, B - m \times n$	S, D, C, Z

BLAS Quick Reference

Meaning of prefixes

S - REAL	C - COMPLEX
D - DOUBLE PRECISION	Z - COMPLEX*16 (this may not be supported by all machines)

For the Level 2 BLAS a set of extended-precision routines with the prefixes ES, ED, EC, EZ may also be available.

Level 1 BLAS
In addition to the listed routines there are two further extended-precision dot product routines DQDOTI and DQDOTA.

Level 2 and Level 3 BLAS
Matrix types:

GE - GEneral	GB - General Band	
SY - SYmmetric	SB - Sym. Band	SP - Sum. Packed
HE - HErmitian	HB - Herm. Band	HP - Herm. Packed
TR - TRiangular	TB - Triang. Band	TP - Triang. Packed

Level 2 and Level 3 BLAS Options
Dummy options arguments are declared as CHARACTER*1 and may be passed as character strings.

TRANx	= '**N**o transpose', '**T**ranspose', '**C**onjugate transpose' (X, X^T, X^H)
UPLO	= '**U**pper triangular', '**L**ower triangular'
DIAG	= '**N**on-unit triangular', '**U**nit triangular'
SIDE	= '**L**eft', '**R**ight' (A or op(A) on the left, or A or op(A) on the right)

For real matrices, TRANSx = 'T' and TRANSx = 'C' have the same meaning.
For Hermitian matrices, TRANSx = 'T' is not allowed.
For complex symmetric matrices, TRANSx = 'H' is not allowed.

References

C. Lawson, R. Hanson, D. Kincaid, and F. Krogh, "Basic Linear Algebra Subprograms for Fortran Usage," *ACM Trans. on Math. Soft.* 5 (1979) 308-325

J.J. Dongarra, J. DuCroz, S. Hammarling, and R. Hanson, "An Extended Set of Fortran Basic Linear Algebra Subprograms," *ACM Trans. on Math. Soft.* 14,1 (1988) 1-32

J.J. Dongarra, I. Duff, J. DuCroz, and S. Hammarling, "A Set of Level 3 Basic Linear Algebra Subprograms," *ACM Trans. on Math. Soft.* (1989)

Obtaining the Software via `netlib@ornl.gov`

To receive a copy of the single-precision software, type in a mail message:

```
send sblas from blas
send sblas2 from blas
send sblas3 from blas
```

To receive a copy of the double-precision software, type in a mail message:

```
send dblas from blas
send dblas2 from blas
send dblas3 from blas
```

To receive a copy of the complex single-precision software, type in a mail message:

```
send cblas from blas
send cblas2 from blas
send cblas3 from blas
```

To receive a copy of the complex double-precision software, type in a mail message:

```
send zblas from blas
send zblas2 from blas
send zblas3 from blas
```

Send comments and questions to `lapack@cs.utk.edu` .

Basic Linear Algebra Subprograms

A Quick Reference Guide

University of Tennessee
Oak Ridge National Laboratory
Numerical Algorithms Group Ltd.

May 11, 1997

LAPACKとは?

- LAPACK(Linear Algebra PACKage) : 線形代数パッケージ
- BLASをビルディングブロックとして使いつつ、より高度な問題である連立一次方程式、 最小二乗法固有値問題、固有値問題、特異値問題を解くことができる.
- 下請けルーチン群も提供する: 行列の分解(LU分解, コレスキー分解, QR分解, 特異値分解, Schur分解, 一般化Schur分解)、条件数の推定ルーチン, 逆行列計算など。
- 品質保証も非常に精密かつ系統的で、信頼がおける。
- パソコンからスーパーコンピュータまで様々なCPU、OS上で動く。
- Fortran 90で書かれ、3.8.0は1900以上のルーチンからなっている。
- webサイトはなんと約1億7000万ヒットである
- githubで開発が続いている https://github.com/Reference-LAPACK

http://www.netlib.org/lapack

LAPACK公式ドキュメント

- http://www.netlib.org/lapack/lug/ : ユーザーガイド

- http://www.netlib.org/lapack/faq.html : FAQ

- http://www.netlib.org/lapack/lawns/index.html
  LAWN: LAPACK Working Notes : 実装の詳細、アルゴリズム、パフォーマンスの比較など。

線形代数+コンピュータで最重要タスクたち

- **連立一次方程式問題：Ax=b**
- **最小二乗法 min ||b-Ax||**
- **固有値問題 $Ax=\lambda x$**
- **特異値問題 $M = U\Sigma V^*$**

規模、精度、行列のタイプ、解き方に多様な応用がある

LAPACKのルーチンの種類

- Driver routines : 先程あげた固有値、連立一次方程式を解く
  - Simple driver:
  - Expert driver: Simple driverに比べて、条件数推定、解の改善、エラーバウンド、行列の平衡化などを行う
- Computational routines
  - 上記タスクなどのために行うLU分解や三角行列のリダクションを行うがBLASよりは高級な処理を行う
- Auxiliary routines
  - blockアルゴリズムのサブタスク、行列ノルム、スケーリングなどBLASの拡張またはBLASに入れたほうがいいルーチンなど低レベル処理

14.'15

LAPACKで連立一次方程式を解く simple driverたち

Simple Drivers

Simple Driver Routines for Linear Equations

Matrix Type	Routine								
General	SGESV(	N,	NRHS,	A,	LDA,	IPIV,	B,	LDB,	INFO)
	CGESV(	N,	NRHS,	A,	LDA,	IPIV,	B,	LDB,	INFO)
General Band	SGBSV(	N,	KL,	KU,	NRHS,	AB,	LDAB,	IPIV,	B,	LDB,	INFO)
	CGBSV(	N,	KL,	KU,	NRHS,	AB,	LDAB,	IPIV,	B,	LDB,	INFO)
General Tridiagonal	SGTSV(	N,	NRHS,	DL,	D,	DU,	B,	LDB,	INFO)
	CGTSV(	N,	NRHS,	DL,	D,	DU,	B,	LDB,	INFO)
Symmetric/Hermitian Positive Definite	SPOSV(	UPLO,	N,	NRHS,	A,	LDA,	B,	LDB,	INFO)
	CPOSV(	UPLO,	N,	NRHS,	A,	LDA,	B,	LDB,	INFO)
Symmetric/Hermitian Positive Definite (Packed Storage)	SPPSV(	UPLO,	N,	NRHS,	AP,		B,	LDB,	INFO)
	CPPSV(	UPLO,	N,	NRHS,	AP,		B,	LDB,	INFO)
Symmetric/Hermitian Positive Definite Band	SPBSV(	UPLO,	N,	KD,	NRHS,	AB,	LDAB,	B,	LDB,	INFO)
	CPBSV(	UPLO,	N,	KD,	NRHS,	AB,	LDAB,	B,	LDB,	INFO)
Symmetric/Hermitian Positive Definite Tridiagonal	SPTSV(	N,	NRHS,	D,	E,		B,	LDB,	INFO)
	CPTSV(	N,	NRHS,	D,	E,		B,	LDB,	INFO)
Symmetric/Hermitian Indefinite	SSYSV(	UPLO,	N,	NRHS,	A,	LDA,	IPIV,	B,	LDB,	WORK,	LWORK,	INFO)
	CSYSV(	UPLO,	N,	NRHS,	A,	LDA,	IPIV,	B,	LDB,	WORK,	LWORK,	INFO)
	CHESV(	UPLO,	N,	NRHS,	A,	LDA,	IPIV,	B,	LDB,	WORK,	LWORK,	INFO)
Symmetric/Hermitian Indefinite (Packed Storage)	SSPSV(	UPLO,	N,	NRHS,	AP,		IPIV,	B,	LDB,			INFO)
	CSPSV(	UPLO,	N,	NRHS,	AP,		IPIV,	B,	LDB,			INFO)
	CHPSV(	UPLO,	N,	NRHS,	AP,		IPIV,	B,	LDB,			INFO)

http://www.netlib.org/lapack/lapackqref.ps

LAPACKで最小二乗法を解く
simple driverたち

Simple Driver Routines for Standard and Generalized Linear Least Squares Problems

Problem Type	Routine
Solve Using Orthogonal Factor, Assuming Full Rank	`SGELS( TRANS, M, N, NRHS, A, LDA, B, LDB, WORK, LWORK, INFO )`
	`CGELS( TRANS, M, N, NRHS, A, LDA, B, LDB, WORK, LWORK, INFO )`
Solve LSE Problem Using GRQ	`SGGLSE( M, N, P, A, LDA, B, LDB, C, D, X, WORK, LWORK, INFO )`
	`CGGLSE( M, N, P, A, LDA, B, LDB, C, D, X, WORK, LWORK, INFO )`
Solve GLM Problem Using GQR	`SGGGLH( N, M, P, A, LDA, B, LDB, D, X, Y, WORK, LWORK, INFO )`
	`CGGGLH( N, M, P, A, LDA, B, LDB, D, X, Y, WORK, LWORK, INFO )`

http://www.netlib.org/lapack/lapackqref.ps

LAPACKで一般化固有値問題、一般化特異値問題を解く simple & dvide and conqure driverたち

Simple and Divide and Conquer Driver Routines for Generalized Eigenvalue and Singular Value Problems

Matrix/Problem Type	Routine																		
Symmetric-definite	SSYGV(	ITYPE,	JOBZ,	UPLO,	N,	A,	LDA,	B,	LDB,	W,				WORK,	LWORK,				INFO)
Eigenvalues/vectors	CHEGV(	ITYPE,	JOBZ,	UPLO,	N,	A,	LDA,	B,	LDB,	W,				WORK,	LWORK,	RWORK,			INFO)
Divide and Conquer	SSYGVD(	ITYPE,	JOBZ,	UPLO,	N,	A,	LDA,	B,	LDB,	W,				WORK,	LWORK,		IWORK,	LIWORK,	INFO)
	CHEGVD(	ITYPE,	JOBZ,	UPLO,	N,	A,	LDA,	B,	LDB,	W,				WORK,	LWORK,	RWORK, LRWORK,	IWORK,	LIWORK,	INFO)
Symmetric-definite	SSPGV(	ITYPE,	JOBZ,	UPLO,	N,	AP,		BP,		W,	Z,	LDZ,		WORK,					INFO)
(Packed Storage)	CHPGV(	ITYPE,	JOBZ,	UPLO,	N,	AP,		BP,		W,	Z,	LDZ,		WORK,		RWORK,			INFO)
Eigenvalues/vectors																			
Divide and Conquer	SSPGVD(	ITYPE,	JOBZ,	UPLO,	N,	AP,		BP,		W,	Z,	LDZ,		WORK,	LWORK,		IWORK,	LIWORK,	INFO)
	CHPGVD(	ITYPE,	JOBZ,	UPLO,	N,	AP,		BP,		W,	Z,	LDZ,		WORK,	LWORK,	RWORK, LRWORK,	IWORK,	LIWORK,	INFO)
Symmetric-definite	SSBGV(		JOBZ,	UPLO,	N,	KA,	KB,	AB,	LDAB,	BB,	LDBB,	W,	Z, LDZ,	WORK,					INFO)
(Band Storage)	CHBGV(		JOBZ,	UPLO,	N,	KA,	KB,	AB,	LDAB,	BB,	LDBB,	W,	Z, LDZ,	WORK,		RWORK,			INFO)
Eigenvalues/vectors																			
Divide and Conquer	SSBGVD(		JOBZ,	UPLO,	N,	KA,	KB,	AB,	LDAB,	BB,	LDBB,	W,	Z, LDZ,	WORK,	LWORK,		IWORK,	LIWORK,	INFO)
	CHBGVD(		JOBZ,	UPLO,	N,	KA,	KB,	AB,	LDAB,	BB,	LDBB,	W,	Z, LDZ,	WORK,	LWORK,	RWORK, LRWORK,	IWORK,	LIWORK,	INFO)
General	SGGES(	JOBVSL,	JOBVSR,	SORT,	SELCTG,	N,	A,	LDA,	B,	LDB,	SDIM,	ALPHAR,	ALPHAI,	BETA,	VSL,	LDVSL, VSR, LDVSR,	WORK,	LWORK,	BWORK, INFO)
Schur Factorization	CGGES(	JOBVSL,	JOBVSR,	SORT,	SELCTG,	N,	A,	LDA,	B,	LDB,	SDIM,	ALPHA,		BETA,	VSL,	LDVSL, VSR, LDVSR,	WORK,	LWORK, RWORK,	BWORK, INFO)
General	SGGEV(	JOBVL,	JOBVR,			N,	A,	LDA,	B,	LDB,		ALPHAR,	ALPHAI,	BETA,	VL,	LDVL, VR, LDVR,	WORK,	LWORK,	INFO)
Eigenvalues/vectors	CGGEV(	JOBVL,	JOBVR,			N,	A,	LDA,	B,	LDB,		ALPHA,		BETA,	VL,	LDVL, VR, LDVR,	WORK,	LWORK, RWORK,	INFO)
General	SGGSVD(	JOBU,	JOBV,	JOBQ,	H,	N,	P,	K,	L,	A,	LDA,	B,	LDB,	ALPHA,		BETA, U, LDU, V, LDV, Q, LDQ,	WORK,		IWORK, INFO)
Singular Values/Vectors	CGGSVD(	JOBU,	JOBV,	JOBQ,	H,	N,	P,	K,	L,	A,	LDA,	B,	LDB,	ALPHA,		BETA, U, LDU, V, LDV, Q, LDQ,	WORK,	RWORK,	IWORK, INFO)

http://www.netlib.org/lapack/lapackqref.ps

LAPACKで標準固有値問題、特異値問題を解く simple & dvide and conqure driverたち

Simple and Divide and Conquer Driver Routines for Standard Eigenvalue and Singular Value Problems

Matrix/Problem Type	Routine																
Symmetric/Hermitian	SSYEV(	JOBZ,	UPLO,	N,	A,	LDA,	W,				WORK,	LWORK,					INFO)
Eigenvalues/vectors	CHEEV(	JOBZ,	UPLO,	N,	A,	LDA,	W,				WORK,	LWORK,	RWORK,				INFO)
Divide and Conquer	SSYEVD(	JOBZ,	UPLO,	N,	A,	LDA,	W,				WORK,	LWORK,			IWORK,	LIWORK,	INFO)
	CHEEVD(	JOBZ,	UPLO,	N,	A,	LDA,	W,				WORK,	LWORK,	RWORK,	LRWORK,	IWORK,	LIWORK,	INFO)
Symmetric/Hermitian	SSPEV(	JOBZ,	UPLO,	N,	AP,		W,	Z,	LDZ,		WORK,						INFO)
(Packed Storage)	CHPEV(	JOBZ,	UPLO,	N,	AP,		W,	Z,	LDZ,		WORK,		RWORK,				INFO)
Eigenvalues/vectors																	
Divide and Conquer	SSPEVD(	JOBZ,	UPLO,	N,	AP,		W,	Z,	LDZ,		WORK,	LWORK,			IWORK,	LIWORK,	INFO)
	CHPEVD(	JOBZ,	UPLO,	N,	AP,		W,	Z,	LDZ,		WORK,	LWORK,	RWORK,	LRWORK,	IWORK,	LIWORK,	INFO)
Symmetric/Hermitian Band	SSBEV(	JOBZ,	UPLO,	N,	KD,	AB, LDAB,	W,	Z,	LDZ,		WORK,						INFO)
Eigenvalues/vectors	CHBEV(	JOBZ,	UPLO,	N,	KD,	AB, LDAB,	W,	Z,	LDZ,		WORK,		RWORK,				INFO)
Divide and Conquer	SSBEVD(	JOBZ,	UPLO,	N,	KD,	AB, LDAB,	W,	Z,	LDZ,		WORK,	LWORK,			IWORK,	LIWORK,	INFO)
	CHBEVD(	JOBZ,	UPLO,	N,	KD,	AB, LDAB,	W,	Z,	LDZ,		WORK,	LWORK,	RWORK,	LRWORK,	IWORK,	LIWORK,	INFO)
Symmetric Tridiagonal	SSTEV(	JOBZ,		N,	D,	E,		Z,	LDZ,		WORK,						INFO)
Eigenvalues/vectors																	
Divide and Conquer	SSTEVD(	JOBZ,		N,	D,	E,		Z,	LDZ,		WORK,	LWORK,			IWORK,	LIWORK,	INFO)
General	SGEES(	JOBVS,	SORT, SELECT,	N,	A,	LDA, SDIM,	WR, WI,	VS,	LDVS,		WORK,	LWORK,				BWORK,	INFO)
Schur Factorization	CGEES(	JOBVS,	SORT, SELECT,	N,	A,	LDA, SDIM,	W,	VS,	LDVS,		WORK,	LWORK,	RWORK,			BWORK,	INFO)
General	SGEEV(	JOBVL,	JOBVR,	N,	A,	LDA,	WR, WI,	VL,	LDVL,	VR, LDVR,	WORK,	LWORK,					INFO)
Eigenvalues/vectors	CGEEV(	JOBVL,	JOBVR,	N,	A,	LDA,	W,	VL,	LDVL,	VR, LDVR,	WORK,	LWORK,	RWORK,				INFO)
General	SGESVD(	JOBU,	JOBVT,	H, N,	A,	LDA,	S,	U,	LDU,	VT, LDVT,	WORK,	LWORK,					INFO)
Singular Values/Vectors	CGESVD(	JOBU,	JOBVT,	H, N,	A,	LDA,	S,	U,	LDU,	VT, LDVT,	WORK,	LWORK,	RWORK,				INFO)
Divide and Conquer	SGESDD(	JOBZ,		H, N,	A,	LDA,	S,	U,	LDU,	VT, LDVT,	WORK,	LWORK,			IWORK,		INFO)
	CGESDD(	JOBZ,		H, N,	A,	LDA,	S,	U,	LDU,	VT, LDVT,	WORK,	LWORK,	RWORK,		IWORK,		INFO)

http://www.netlib.org/lapack/lapackqref.ps

様々な解法が存在していて、様々なルーチンが存在する

- たくさんLAPACKのルーチンを提示したが、これにそれぞれExpert driverや、RRR (relatively robust representation) 版などが存在する。

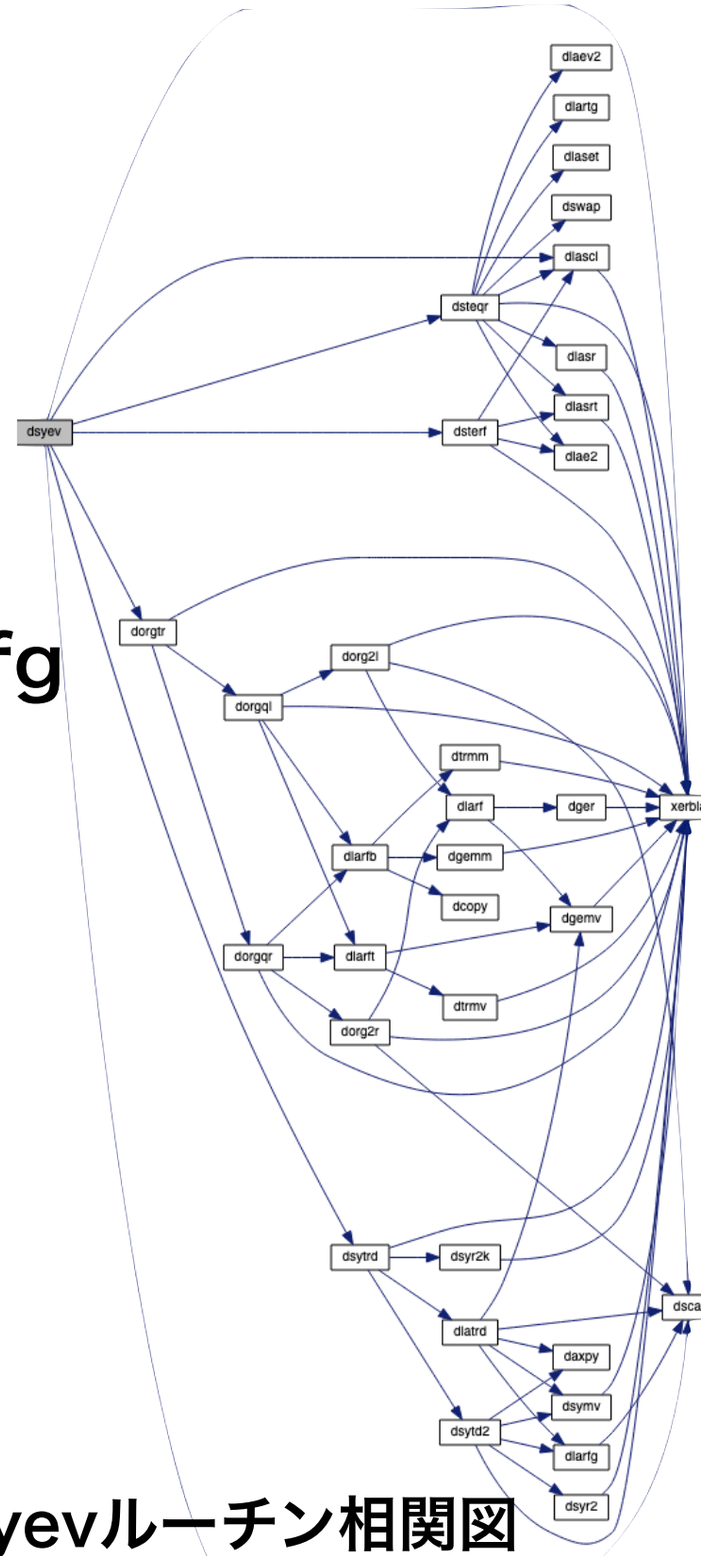
- simple/divede and conqure/RRR/Expertからどうやって選べばよいか?

  - これは問題に応じて個々人が選ぶ必要が出てくる。

14:20

LAPACKのルーチン構造

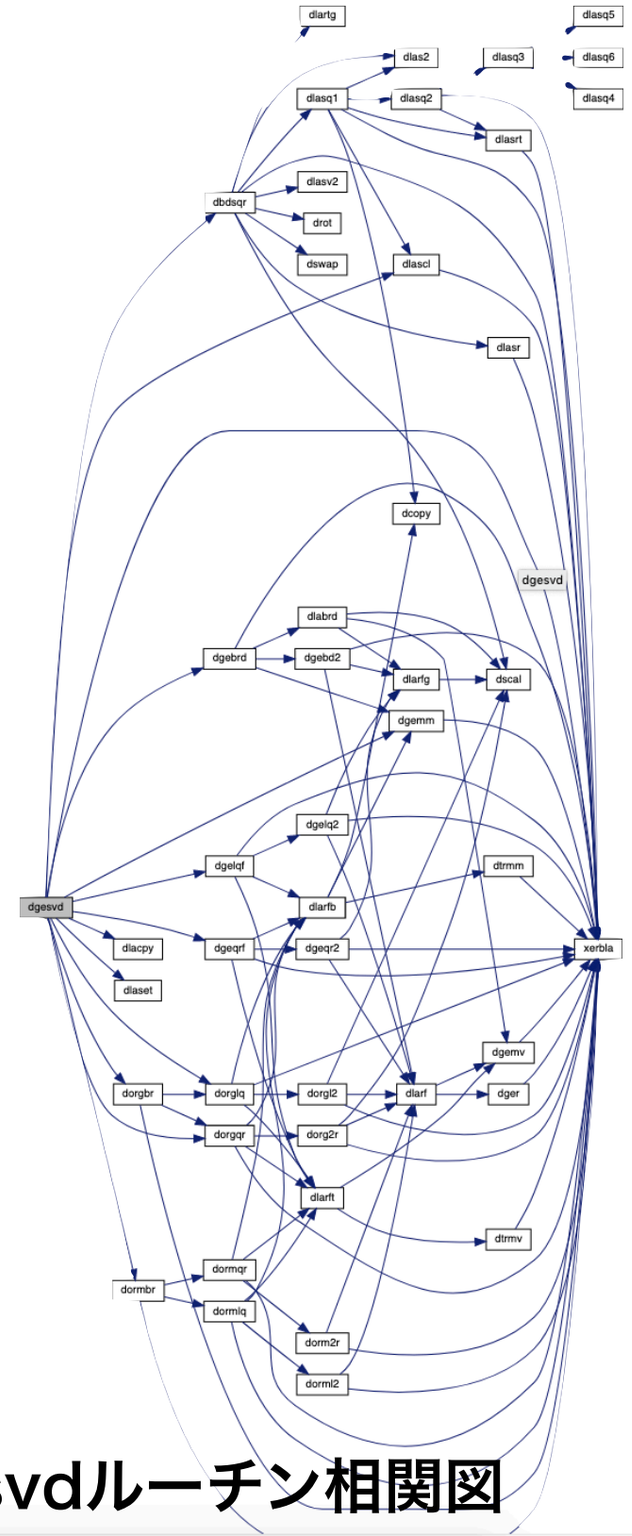
- 例えば実対称行列の固有値を求めるのにはdsyevを使ったが下請けには34のルーチン群がある。
  - dorgtr, dorgql, dorg2l, dorgqr, dlarfb
  - dlarf, dgemm, dcopy, dtrmv, dgemv, dger
  - dsyr2k, dlatrd, dsytd2, daxpy, dsymv, dlarfg
  - dsyr2, dscal, dsteqr, dsterf, dlaev2, dlartg,
  - dlaset, swap, dlascl, dlasr, dlasrt, dlae2



dsyevルーチン相関図

LAPACKのルーチン構造

- 実特異値分解はもっと複雑。dgesvdだけでも3503行あるが、殆どが総計46の下請けルーチンをコールしている。



dgesvdルーチン相関図

BLAS LAPACK豆知識

BLAS, LAPACKを利用したソフトウェア

- 著名な計算プログラムパッケージは大抵BLAS, LAPACKを利用している.
    - 物理、化学ではGaussian, Gamess, ADF, VASP
    - 線形計画問題のCPLEX, NUOPT, GLPKなど..
- 高級言語からも利用可能
- Ruby, Python (numpy), Perl, Java, C, Mathematica, Maple, Matlab, R, octave, SciLab

歴史:高速なBLAS, LAPACKの実装について

- Reference BLAS/LAPACKはある意味仕様書そのままなので、非常に低速である。メモリの階層構造などは非常に意識して書かれているが、CPUに最適化は、各々がやることになる。

- 2000年までは、だいたいSUN/IBM/DEC/Intel/Fujitsu/Hitachi/NECなどCPU、OS,システムごとにベンダーの実装があり、高価で販売されていた、またはマシンを買うとついてきたりした。

- ATLAS:R. Clint Whaley氏による, オートチューニング機構で高速化したBLAS。それまでの2001年より多くのコンピュータのBLAS環境を劇的に改善した, パイオニア的存在。ハンドチューニングしたBLASよりは数%から数10%低速程度(オープンソース)

- GotoBLAS/GotoBLAS2：後藤和茂氏が、アセンブラで様々なCPUに対応したBLASのソースコードを公開。マシンの性能の限界近くまで性能を追求(非オープンソース)。

2000年ころから高速なBLAS/LAPACKの環境が良くなってきた

[4:25

現状: 高速なBLAS, LAPACKの実装について

- OpenBLAS: Zhang Xianyi氏がGotoBLAS2の開発を引き継いだ。開発はアクティブでSandyBridge以降のプロセッサ，OSにも対応している。ARM各種、AMD、Power, ICT Loongson-3A, 3Bにも対応。オープンソース https://www.openblas.net/

- Intel Math Kernel Library: Intelが開発している加速されたBLASおよびLAPACK。2012年から後藤氏がIntelに移籍してチューニングしているのでIntel系では最速と思われる。FFTなどもはいっている。https://software.intel.com/en-us/mkl “Free to use for personal and commercial applications.”

- GPU向けBLAS, LAPACK

  - MAGMAプロジェクトはCUDA, Xeon Phi OpenCLなどGPUやアクセラレータ向けBLAS, LAPACKを開発している。NVIDIAのcuBLASよりも高速。

  - https://icl.cs.utk.edu/magma/

Top500:コンピュータの速度ランキング

- **Top 500:世界で一番高速なコンピューターを決めるTop 500では,LINPACKを使って、連立一次方程式を解くスピードを競う**

$$Ax = b$$

- **DGEMMのスピードが重要となる。**

最新(2018/11)のランク

USが1,2 中国が3,4位, 5位がスイス、7位が産総研ABCI,京は18位

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	DOE/SC/Oak Ridge National Laboratory United States	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM	2,397,824	143,500.0	200,794.9	9,783
2	DOE/NNSA/LLNL United States	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM / NVIDIA / Mellanox	1,572,480	94,640.0	125,712.0	7,438
3	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
4	National Super Computer Center in Guangzhou China	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 NUDT	4,981,760	61,444.5	100,678.7	18,482
5	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	387,872	21,230.0	27,154.3	2,384

BLAS、LAPACKを使ってみる

- Ubuntu 16.04 デスクトップ版で実際にBLAS, LAPACKを実際に使ってみる。

C++から

- 行列-行列積
- 対称行列の対角化

を行う。思ったより設定が必要

BLAS、LAPACKのインストール

- Ubuntu 16.04 で次のようにすると、BLAS、LAPACKの開発環境が整う。

$ sudo apt-get install gfortran g++ libblas-dev liblapack-dev liblapacke-dev

パッケージリストを読み込んでいます... 完了

依存関係ツリーを作成しています

状態情報を読み取っています... 完了

- 成功したら二回目の実行で

$ sudo apt-get instll gfortran g++ libblas-dev liblapack-dev liblapacke-dev

...

g++ はすでに最新バージョンです。

gfortran はすでに最新バージョンです。

libblas-dev はすでに最新バージョンです。

liblapack-dev はすでに最新バージョンです。

アップグレード: 0 個、新規インストール: 0 個、削除: 0 個、保留: 172 個。

- こんな感じであればok

行列-行列の積 DGEMMを使ってみる

- 行列-行列積DGEMMを使ってみる。ここでは

$$A = \begin{pmatrix} 1 & 8 & 3 \\ 2 & 10 & 8 \\ 9 & -5 & -1 \end{pmatrix} \quad B = \begin{pmatrix} 9 & 8 & 3 \\ 3 & 11 & 2.3 \\ -8 & 6 & 1 \end{pmatrix} \quad C = \begin{pmatrix} 3 & 3 & 1.2 \\ 8 & 4 & 8 \\ 6 & 1 & -2 \end{pmatrix}$$

$$\alpha = 3, \beta = -2$$

$$C \leftarrow \alpha AB + \beta C$$

- を計算するプログラムを書いてみる.

- 答えは以下のようになる

$$\begin{pmatrix} 21 & 336 & 70.8 \\ -64 & 514 & 95 \\ 210 & 31 & 47.5 \end{pmatrix}$$

14:30

行列-行列の積:DGEMMの詳細

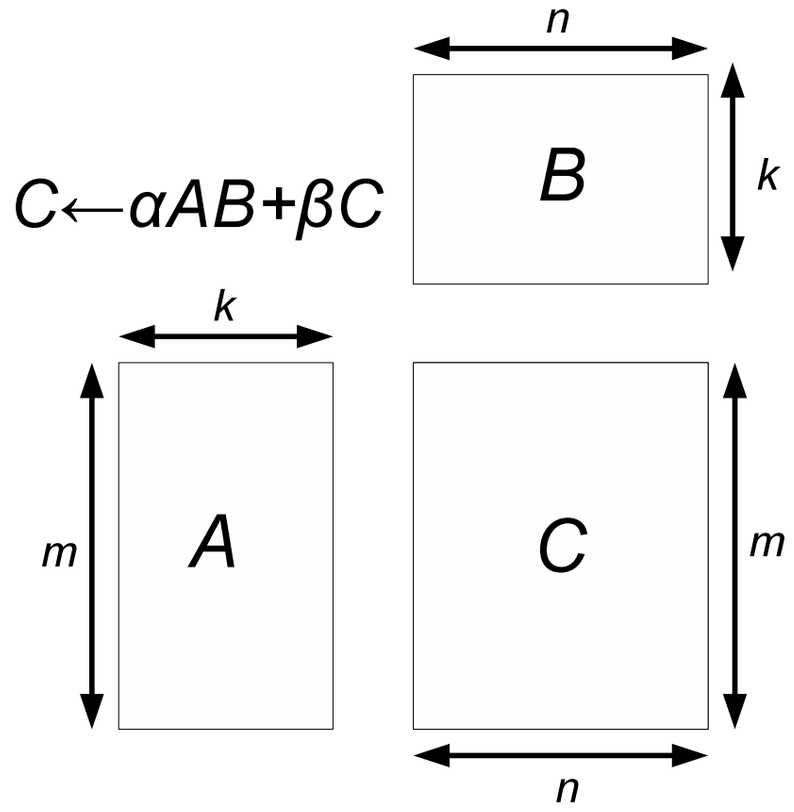
$$C \leftarrow \alpha AB + \beta C$$

- 今回はCBLASから、BLASを呼んでみる。

- BLASはFORTRANで書かれているが、Cから呼び出すことも可能。プロトタイプ宣言は以下のようになる

void cblas_dgemm(CBLAS_LAYOUT layout, CBLAS_TRANSPOSE TransA, CBLAS_TRANSPOSE TransB, const int M, const int N, const int K, const double alpha, const double *A, const int lda, const double *B, const int ldb, const double beta, double *C, const int ldc);

- layout : Column major or Row major
- “transa”, “transb”, “transc” で行列を転置するか否かを指定。
- m, n, k は行列の次元。左図参照
- alpha, betaは行列積に対する掛けるスカラー
- A, B, CはRow major形式で格納した行列へのポインタ
- lda, ldb, ldc は 行列A, B, CへのLeading dimensionたち
- 配列はゼロから始まるか1からはじまるか



ここらへんがわかりにくいところ。あとで説明

行列-行列の積のリスト

```
#include <stdio.h>
#include <cblas.h>
//Matlab/Octave format
void printmat(int N, int M, double *A, int LDA) {
  double mtmp;
  printf("[ ");
  for (int i = 0; i < N; i++) {
    printf("[ ");
    for (int j = 0; j < M; j++) {
      mtmp = A[i + j * LDA];
      printf("%5.2e", mtmp);
      if (j < M - 1) printf(", ");
    } if (i < N - 1) printf("]; ");
    else printf("] ");
  } printf("]");
}
int main()
{
  int n = 3; double alpha, beta;
  double *A = new double[n*n];
  double *B = new double[n*n];
  double *C = new double[n*n];

  A[0+0*n]=1; A[0+1*n]= 8; A[0+2*n]= 3;
  A[1+0*n]=2; A[1+1*n]=10; A[1+2*n]= 8;
  A[2+0*n]=9; A[2+1*n]=-5; A[2+2*n]=-1;

  B[0+0*n]= 9; B[0+1*n]= 8; B[0+2*n]=3;
  B[1+0*n]= 3; B[1+1*n]=11; B[1+2*n]=2.3;
  B[2+0*n]=-8; B[2+1*n]= 6; B[2+2*n]=1;

  C[0+0*n]=3; C[0+1*n]=3; C[0+2*n]=1.2;
  C[1+0*n]=8; C[1+1*n]=4; C[1+2*n]=8;
  C[2+0*n]=6; C[2+1*n]=1; C[2+2*n]=-2;
  printf("# dgemm demo...\n");
  printf("A
=");printmat(n,n,A,n);printf("\n");
  printf("B
=");printmat(n,n,B,n);printf("\n");
  printf("C
=");printmat(n,n,C,n);printf("\n");
  alpha = 3.0; beta = -2.0;


cblas_dgemm(CblasColMajor,CblasNoTrans,Cbl
asNoTrans, n, n, n, alpha, A, n, B, n,
beta, C, n);
  printf("alpha = %5.3e\n", alpha);
  printf("beta  = %5.3e\n", beta);
  printf("ans="); printmat(n,n,C,n);
  printf("\n");
  printf("#check by Matlab/Octave by:\n");
  printf("alpha * A * B + beta * C =\n");
  delete[]C; delete[]B; delete[]A;
}
```

行列-行列の積のコンパイルと実行

- 先ほどのリストを"dgemm_demo.cpp"などと保存する。

```
$ g++ dgemm_demo.cpp -o dgemm_demo -lblas -lapack
```

- でコンパイルができる. 何もメッセージが出ないなら, コンパイルは成功である。実行は以下のようになっていればよい。OctaveやMatlabにこの結果をそのままコピー&ペーストとすれば答えをチェックできるようにしてある

```
alpha = 3.000e+00

beta  = -2.000e+00

ans=[ [ 2.10e+01, 3.36e+02, 7.08e+01]; [ -6.40e+01, 5.14e+02, 9.50e+01];

     [ 2.10e+02, 3.10e+01, 4.75e+01] ]

#check by Matlab/Octave by:

alpha * A * B + beta * C
```

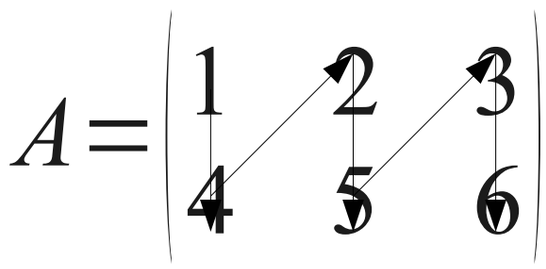
行列をColumn majorでメモリに格納する

- 行列は2次元だが、コンピュータのメモリは1次元的である。次のような行列を

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

- 考えるとき、どのようにメモリに格納するか? column major式では、
- アドレスの小さい順から

  1, 4, 2, 5, 3, 6



- のように格納する。

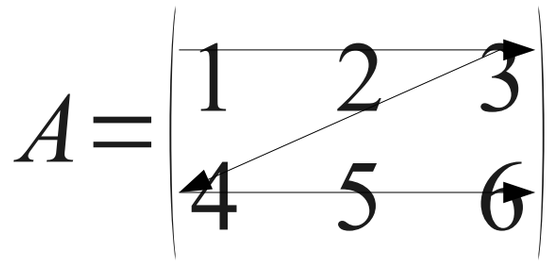
- FORTRANや、Matlab, octaveはcolumn majorである。

行列をRow majorでメモリに格納する

同じように、行列A

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

- について、row majorでは、行方向に
- 1, 2, 3, 4, 5, 6



- のように格納する。C/C++では普通row majorで格納。意識的にcolumn majorにしたほうが良い

- BLAS, LAPACKを呼び出すときは、行列格納方法に注意!

14:35

Leading dimension (I)

- 行列をさらに小さい行列に分けて考えることがある。
- これらを区分行列、小行列、ブロック行列などとよぶ。
- たとえば 以下のように、A, B, C, Dという行列を考えて

$$A = \begin{pmatrix} 2 & -1 & 5 \\ -1 & 4 & 1 \\ 8 & 1 & -2 \end{pmatrix}, \quad B = \begin{pmatrix} -3 & 6 \\ 1 & 3 \\ 4 & 1 \end{pmatrix}, \quad C = (-4 \quad 2 \quad 6), \quad D = (9 \quad 1)$$

- それらを組み合わせてより大きな行列を作ることができる。

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} 2 & -1 & 5 & -3 & 6 \\ -1 & 4 & 1 & 1 & 3 \\ 8 & 1 & -2 & 4 & 1 \\ -4 & 2 & 6 & 9 & 1 \end{pmatrix}$$

Block行列が便利になる例

- 行列の積を考える

$$C = AB, C_{ij} = \sum_k A_{ik} B_{kj}$$

- 行列積の定義は、要素ごとに積をとって和を取るだが、区分行列にわけても、そのまま「数」のように積をとってよい

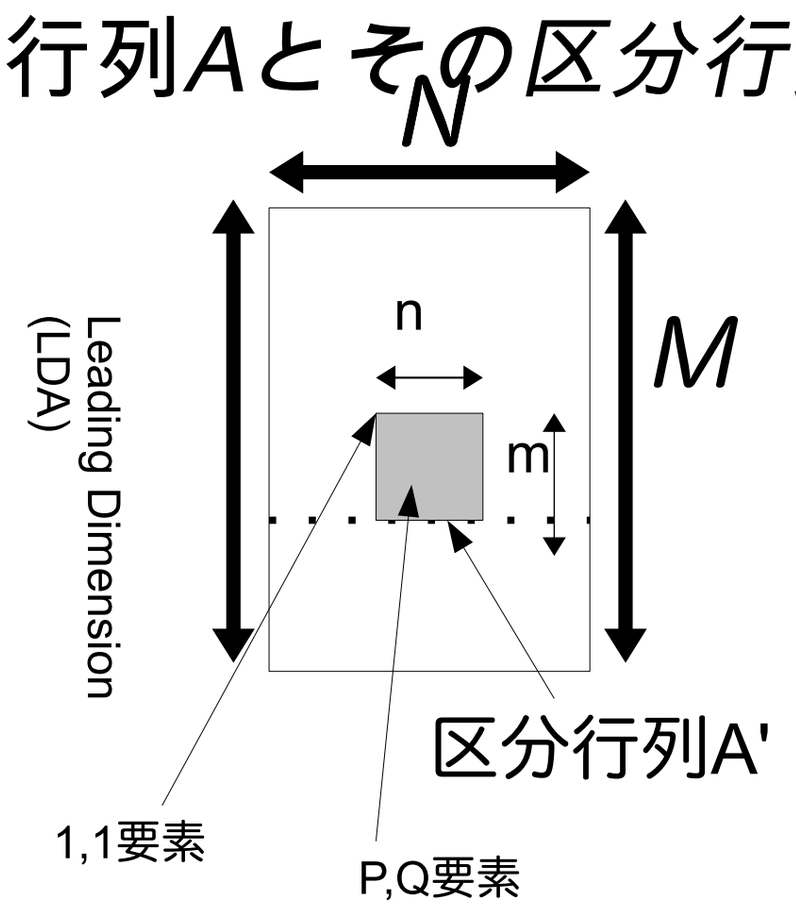
$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1q} \\ A_{21} & A_{22} & \cdots & A_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1} & A_{p2} & \cdots & A_{pq} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1r} \\ B_{21} & B_{22} & \cdots & B_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ B_{q1} & B_{q2} & \cdots & B_{qr} \end{pmatrix}$$

$$AB = \begin{pmatrix} C_{11} & C_{12} & \cdots & C_{1r} \\ C_{21} & C_{22} & \cdots & C_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ C_{p1} & C_{p2} & \cdots & C_{pr} \end{pmatrix} \qquad C_{ij} = \sum_{k=1}^{q} A_{ik} B_{kj}$$

Leading dimension (III)

- 行列Aの区分行列A’にアクセスするにはどうしたらよいか? A’のサイズはn x m とし (p, q)要素とする。これにアクセスするには “leading dimension”を使うと便利

- A’[1,1] のアドレスから、A’[P,Q]はA’[1,1]+P*m+Qではなくて、A’[1,1]+P*LDA+Qとなる。



配列は0か1どちらから始まるか?

- FORTRANでは配列は1からスタートするが, C, C++では, 0からスタートする. 例えば

- ループの書き方が一般的には1からNまで(FORTRAN)か, 0からn未満か(C,C++).

- ベクトルのx_i要素へのアクセスはFORTRANでは**X(I)**だが, Cでは**x[i-1]**となる.

- 行列のA_{ij}要素へのアクセスはFORTRANでは**A(I,J)**だが, Cではcolumn majorとして**A[i-1+ (j-1)*lda]**とするとよい

LAPACK実習:行列の固有ベクトル、固有値を求める:DSYEV

- 3x3 の実対称行列の固有ベクトル、固有値を求めよう。これらは三つあり、

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 4 \\ 3 & 4 & 6 \end{pmatrix} \qquad Av_i = \lambda_i v_i \ \ (i = 1,2,3)$$

という関係式が成り立つ。

- 固有値$\lambda_1, \lambda_2, \lambda_3$ は　$-0.40973, 1.57715, 10.83258$

- で、 固有ベクトルは、

$$v1 = (-0.914357, 0.216411, 0.342225)$$

$$v2 = (0.040122, -0.792606, 0.608413)$$

$$v3 = (0.402916, 0.570037, 0.716042)$$

となる

14:40

行列の固有ベクトル、固有値を求めるDSYEV

- C/C++からLAPACKをコールするにはLAPACKEを用いる。プロトタイプは以下の通り
- `lapack_int LAPACKE_dsyev( int matrix_layout, char jobz, char uplo, lapack_int n, double* a, lapack_int lda, double* w );`
- matrix_layout : LAPACK_COL_MAJOR: column majorかrow majorか
- jobz:固有値、固有ベクトルが必要か、固有値だけでよいか指定。
- uplo:行列の上三角、下三角を使うか。
- A, lda:行列Aとそのleading dimension
- w:固有値を返す配列(昇順)

(注意) 以下はFORTRANを直接呼ぶ場合は必要だったが、LAPACKEではなくなった

- work, lwork:ワーク領域への配列またはポインタ、とそのサイズ
- info: =0 正常終了。<0: INFO=-iではi番目の引数が不適当。>0: INFO=i 収束せず。

対称行列の対角化dsyevの例

```
#include <iostream>
#include <stdio.h>
#include <lapacke.h>

//Matlab/Octave format
void printmat(int N, int M,
double *A, int LDA) {
  double mtmp;
  printf("[ ");
  for (int i = 0; i < N; i++) {
    printf("[ ");
    for (int j = 0; j < M; j++) {
      mtmp = A[i + j * LDA];
      printf("%5.2e", mtmp);
      if (j < M - 1) printf(", ");
    } if (i < N - 1) printf("]; ");
    else printf("] ");
  } printf("]");
}
```

```
int main()
{
  int n = 3;
  double *A = new double[n*n];
  double *w = new double[n];
//setting A matrix
  A[0+0*n]=1;A[0+1*n]=2;A[0+2*n]=3;
  A[1+0*n]=2;A[1+1*n]=5;A[1+2*n]=4;
  A[2+0*n]=3;A[2+1*n]=4;A[2+2*n]=6;

  printf("A ="); printmat(n, n, A, n);
  printf("\n");
  LAPACKE_dsyev(LAPACK_COL_MAJOR, 'V', 'U', n
//print out some results.
  printf("#eigenvalues \n"); printf("w =");
  printmat(n, 1, w, 1); printf("\n");
  printf("#eigenvecs \n"); printf("U =");
  printmat(n, n, A, n); printf("\n");
printf("#Check Matlab/Octave by:\n");
  printf("eig(A)\n");
  printf("U'*A*U\n");
  delete[]w;
  delete[]A;
}
```

対称行列の対角化dsyevの例

- 先ほどのリストを"eigenvalue_demo.cpp"などと保存する。

- **`g++ dsyev_demo.cpp -o dsyev_demo -llapacke -lblas -llapack`**

でコンパイルができる。何もメッセージが出ないなら, コンパイルは成功である。

- 実行は以下のようになっていればよい。同様にOctaveやMatlabにこの結果をそのままコピー&ペーストすれば答えをチェックできるようにしてある。

```
$ ./dsyev_demo
A =[ [ 1.00e+00, 2.00e+00, 3.00e+00]; [ 2.00e+00, 5.00e+00, 4.00e+00]; [ 3.00e+00,
4.00e+00, 6.00e+00] ]
#eigenvalues
w =[ [ -4.10e-01]; [ 1.58e+00]; [ 1.08e+01] ]
#eigenvecs
U =[ [ -9.14e-01, 2.16e-01, 3.42e-01]; [ 4.01e-02, -7.93e-01, 6.08e-01]; [ 4.03e-01,
5.70e-01, 7.16e-01] ]
#Check Matlab/Octave by:
eig(A)
U'*A*U
```

まとめと次回予告

まとめ

- 線形代数は有史以来ずっと人類はやってきている。重要で応用の幅が広い。コンピュータができても相変わらず人類は線形代数をやりつづけている。コンピュータは高速なものが正義
- 線形代数演算などコンピュータで数値計算を行うにはライブラリを用いたほうが良い
- BLAS, LAPACKは線形代数演算ライブラリで重要。高速な実装もある。
- BLAS, LAPACKについて行列-行列式、および対称行列の対角化を行う場合の例を示した。column/row major/ leading dimension/ 配列0,1 という注意点3つを説明した。

次回予告

- コンピュータの簡単なしくみ。
- なぜそのコードは高速/低速に動くのか。
- BLAS, LAPACKを高速につかうにはどうしたらよいか。
- GPUでBLASを使う
- 性能評価の例

参考図書

- BLAS, LAPACKチュートリアル パート1 (基礎編) BLAS, LAPACKチュートリアル パート2 (GPU編)
  - http://nakatamaho.riken.jp/blas_lapack_tutorial_part1.pdf
  - http://nakatamaho.riken.jp/blas_lapack_tutorial_part2.pdf
- LAPACK/BLAS入門 幸谷智紀 (https://www.morikita.co.jp/books/book/2226)
- Matrix Computations, Gene H. Golub and Charles F. Van Loan
  - http://web.mit.edu/ehliu/Public/sclark/Golub%20G.H.,%20Van%20Loan%20C.F.-%20Matrix%20Computations.pdf
- Accuracy and Stablity of Numerical Algorithms, Nicholas J. Higham
- LAPACK Working NOTE : 実装上の詳細ノートたちがある
  - http://www.netlib.org/lapack/lawns/index.html
- 数値計算の常識, 伊理 正夫, 藤野 和建

14:45