精度保証と
スーパーコンピュータ

大石 進一
早稲田大学

「flop/s」から「flop/sと精度」へ

- 単位時間あたりに何回浮動小数点数演算が行えるかでは能力ははかれない
- ある仕事をするときの速さに目標を変更しよう
- どんな目的がよいのか

ベンチマークの変更

- 行列積計算における速さを競っている
- これを連立一次方程式の正しい解を得るのに必要な時間にしてはどうか

2　高速精度保証法

IEEE754の倍精度浮動小数点規格

1. 考えている浮動小数点数の全体を$\mathbb{F}$と表す.
2. 丸めの演算を切り上げΔ，切り捨て∇，最近点$\square$，チョッピング$\bowtie$とする．すなわち，それぞれ，実数を一定条件下で一番近い浮動小数点数に丸める演算子であるが,
   (a) 切り上げはその実数以上の浮動小数点数の中から,
   (b) 切り捨ては，それ以下の浮動小数点数の中から.
   (c) 最近点は条件なし,
   (d) チョッピングはその実数より絶対値が同じか小さい浮動小数点数の中からという定義である.

   このとき，$\bigcirc \in \{\Delta, \nabla, \square, \bowtie\}$，$\cdot \in \{+, -, \times, /\}$，$a, b \in \mathbb{F}$としてその丸めのモードでの四則演算を次の規則で定める

2　高速精度保証法

IEEE754の倍精度浮動小数点規格

1. 考えている浮動小数点数の全体を$\mathbb{F}$と表す.
2. 丸めの演算を切り上げΔ，切り捨て∇，最近点$\square$，チョッピング$\bowtie$とする．すなわち，それぞれ，実数を一定条件下で一番近い浮動小数点数に丸める演算子であるが,
   (a) 切り上げはその実数以上の浮動小数点数の中から,
   (b) 切り捨ては，それ以下の浮動小数点数の中から.
   (c) 最近点は条件なし,
   (d) チョッピングはその実数より絶対値が同じか小さい浮動小数点数の中からという定義である.

   このとき，$\bigcirc \in \{\Delta, \nabla, \square, \bowtie\}$，$\cdot \in \{+, -, \times, /\}$，$a, b \in \mathbb{F}$としてその丸めのモードでの四則演算を次の規則で定める

のがホモモルフィズムの定義である：

$$a \bigodot b = \bigcirc(a \cdot b)$$

左辺の定義を右辺で与える訳である.

3. プログラム中では，切り上げ，切り捨て，最近点，チョッピングに丸める命令をup(), down(), near(), chop()で書くことにする.
4. 丸めのモードを変更すると，次に丸めのモードが変更されるまで同じ丸めのモードで計算されるのがIEEE754規格の仕様である.

著者らによる，ブレークスルー

2つの行列$A, B \in \mathbb{F}^{n\times n}$の積を包み込む（正しい値が入る区間を計算すること）アルゴリズムを次のよう与えた:

```
function [L,U]=prod(A,B)
up();
U=A*B;
down();
L=A*B;
```

ただし，ここではMATLABの記号を用いた.

- 従来は，区間演算を演算単位で用いていたために，行列積の包み込みを得るのに普通の行列計算の数千倍の手間がかかっていたが,
- 上の算法では，２倍の手間で行列積の包み込みができる. これによって，一気に精度保証付き数値計算が実用レベルになった.
- このとき，A, Bの真の積ABは区間行列$[L, U]$に含まれ

現行のベンチマークの解釈

- 行列積が速く計算できると，実は連立一次方程式の精度保証も早くできる

  例　連立一次方程式

$$Ax = b, \quad (A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n)$$

  の精度保証

1. MATLABの記法では

```
>> B=inv(A);
```

とするとAの逆行列の近似$B \in \mathbb{F}^{n\times n}$が得られる.

2. ここで，スペクトル半径を

$$\rho(A) = \max\{|\lambda| \in \sigma(A)\}$$

と定義する．ただし，$\sigma(A)$ は A の固有値の集合である．

3. 次の定理が成立することが知られている：

定理 **1** $A, B \in \mathbb{F}^{n\times n}$ とする．$\rho(BA - I) < 1$ なら A は正則である． □

4. さらに，$A \in \mathbb{F}^{n\times n}$ に対して

$$\rho(A) \leqq \|A\|_\infty = \max_{i=1,2,\cdots} \sum_{j=1}^{n} |A_{i,j}|$$

となることを利用すると，$\|BA - I\|_\infty < 1$ は A が正則となる十分条件となることがわかる．

5. $\|BA - I\|_\infty \leqq k$ となる厳密な上界 k はMATLABでは

```
>> down();
>> L=B*A-eyn(n);
>> up();
>> U=B*A-eye(n);
>> U=max(abs(L),abs(U));
>> k=norm(U,inf);
```

と計算できる．したがって，定理1から $k < 1$ ならば，A が正則なことが証明されることになる．

6. A が正則であるとすると，$\tilde{x} \in \mathbb{F}^n$ を任意の近似解として，$Ax = b$ の真の解 $x^* = A^{-1}b$ との誤差は

$$\|\tilde{x} - x^*\|_\infty \leqq \|A^{-1}(A\tilde{x} - b)\|_\infty \leqq \|A^{-1}\|_\infty \|A\tilde{x} - b\|_\infty$$

問題点

- 問題の条件数によって，精度が悪くなる
- 場合によっては精度保証ができなくなる
- したがって，行列積をベンチマークにすると連立一次方程式を速く解くこととは異なる

提案

- いろいろな連立一次方程式の解を最大精度（倍精度の場合は10進数換算で15桁まで）得るのに最高速なスーパーコンピュータを作るのがよいのではないか

ベンチマークの例1

- ランダム行列の密行列を係数とする連立一次方程式(10万次元から100万次元程度)を解くベンチマーク　(一般にランダム行列は良条件問題を与える)

- 正定値対称行列を係数とする連立一次方程式を解くベンチマーク
- 少なくとも解の存在と一意性を保証

問題規模が大きくなると

- 条件数が大きくなり10^16を越えることが想定される
- この場合は一桁もあっていなくなり，計算の意味が失われる

並列計算と精度保証

- 共有メモリ方式の計算機での最適化BLASでは丸めのモードの変更ができないものがある
- 最近点への丸めのモードのみを使う！
- 事前誤差評価を使えば最近点の丸めのモードのみで区間演算はできるが過大評価となる

浮動小数点数での無誤差変換

$$[x, y] = \mathbf{TwoSum}(a, b) \quad \Rightarrow \quad a + b = x + y \quad (|x| \geq |y|)$$

$\oplus, \ominus$: floating point addition and subtraction

function $[x, y] = \mathbf{TwoSum}(a, b)$

$x = a \oplus b;$

$c = x \ominus a;$

$y = (a \ominus (x \ominus c)) \oplus (b \ominus c);$

Computational cost: 6 flops

Dekkerの定理

$$[x, y] = \mathbf{TwoProduct}(a, b) \quad \Rightarrow \quad a \times b = x + y \quad (|x| \geq |y|)$$

function $[x, y] = \mathbf{TwoProduct}(a, b)$

$x = a \otimes b;$

$[a_H, a_L] = \mathbf{Split}(a); \quad \% \; a_H + a_L \leftarrow a$

$[b_H, b_L] = \mathbf{Split}(b); \quad \% \; b_H + b_L \leftarrow b$

$y = a_L \otimes b_L \ominus (((x \ominus a_H \otimes b_H) \ominus a_L \otimes b_H) \ominus a_H \otimes b_L);$

17 flops

Split

$$[a_H, a_L] = \mathbf{Split}(a) \quad \Rightarrow \quad a = a_H + a_L \quad (|a_H| \geq |a_L|)$$

$\otimes$: floating point multiplication

function $[a_H, a_L] = \mathbf{Split}(a)$

$c = \texttt{factor} \otimes a;$ % $\texttt{factor} = 2^{\lceil \frac{t}{2} \rceil} + 1$

$a_H = c \ominus (c \ominus a);$

$a_L = a \ominus a_H;$

4 flops

最近点の丸めのみの区間演算

- [a,b]+[c,d]の計算
- [l,u]=IntervalSum([a,b],[c,d])

  [p,q]=TwoSum[a,c];

  if q<0, l=pred(p), end;

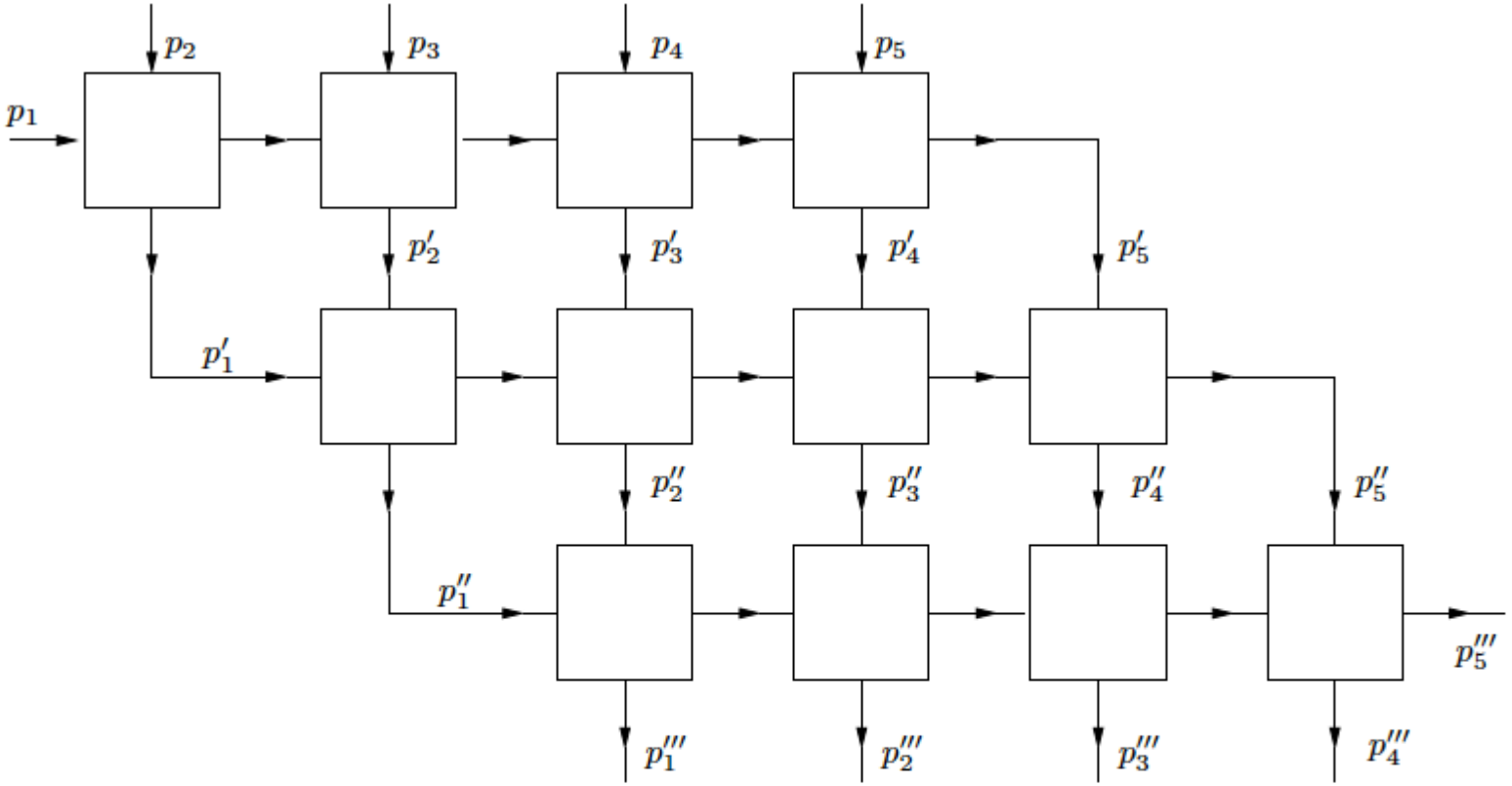
  [s,t]=TwoSum[b,d];

  if t>0, u=succ(s);

ハードウエアかソフトウエアか

- Ogita, Rump, Oishiの高速高精度内積計算法
- Kulischの高精度内積計算ハードウエア

SumKアルゴリズム



Example: $n = 5$, $K = 4$.

誤差解析

$$\gamma_n := \frac{n \cdot \mathbf{u}}{1 - n \cdot \mathbf{u}} \quad (\mathbf{u} := 2^{-53} \text{ in IEEE 754 double precision})$$

$$\mathrm{cond}(\sum p_i) := \frac{\sum |p_i|}{|\sum p_i|} \quad \text{(condition number of summation)}$$

Denote by $\mathtt{res} \in \boldsymbol{F}$ the result obtained by **SumK**, then

$$\frac{|\mathtt{res} - \sum p_i|}{|\sum p_i|} \leq \mathbf{u} + 3\gamma_{n-1}^2 + \gamma_{2n-2}^K \mathrm{cond}(\sum p_i).$$

$\Longrightarrow$ Basically, relative error bound is $\mathbf{u} + c_n \mathbf{u}^K \mathrm{cond}(\sum p_i)$

$\Longrightarrow$ K-fold working precision arithmetic

線形方程式は最後の桁まで正確に

Consider $Ax = b$ with $A \in \boldsymbol{R}^{n\times n}, b \in \boldsymbol{R}^n$. Denote by x^0 an approximate solution obtained by LU decomposition ($\frac{2}{3}n^3$ flops).
Then, **Dot2** is very useful for iterative refinement:

1. Calculate residual $\tilde{r} \approx b - Ax^k$ using **Dot2** ($\mathcal{O}(n^2)$ flops)
2. Solve $Ay = \tilde{r}$ using LU factors ($\mathcal{O}(n^2)$ flops)
3. Update $x^{k+1} = x^k + \tilde{y}$

高速かつ正確に

Results without and with residual iteration ($n = 1000$)

condition number $\mathrm{cond}(A)$	10^5	10^9	10^{13}
max. rel. err. x^0	4.7e-12	2.8e-08	1.9e-04
max. rel. err. x^k	1.8e-16	1.8e-16	1.8e-16
number of iterations k	3	3	5
ratio computing time	1.38	1.38	1.65

For larger dimensions, the ratio of additional cost decreases.

最大精度を得るベンチマーク

- 倍精度であれば10進15桁の精度持つ解を一番速く計算できるスーパーコンピュータ

FusedMultipleAddとAddThree

- 再び何を標準化するか？
- [x,y]=TwoSum(a,b)
  x=a+b;
  y=AddThree(a,b,-x);

  [p,q]=TwoProduct(s,t)
  p=s+t;
  q=FMA(s,t,-p);

高速･高精度行列積

- Fを浮動小数点数の集合とし，$A \in F^{m \times n}$, $B \in F^{n \times p}$とする.
- $A=A^{(1)}+A^{(2)}+\ldots+A^{(r)}$, $B=B^{(1)}+B^{(2)}+\ldots+B^{(s)}$と分解
- （ここで，$A^{(i)} \in F^{m \times n}$, $B^{(j)} \in F^{n \times p}$である．）
- その後
- $AB=(A^{(1)}+A^{(2)}+\ldots+A^{(r)})(B^{(1)}+B^{(2)}+\ldots+B^{(s)})$を浮動小数点演算で計算.
- ただし$fl(A^{(i)}B^{(j)})=A^{(i)}B^{(j)}$が成立する.

高速・高精度行列積

- この行列の和をRumpらのアルゴリズムで計算すると最近点への丸めを達成したことが保証される.
- 行列積計算には最適化された関数が用意されている. その関数への依存度が非常に高い
- この手法の応用は, 悪条件問題の解法で活躍する.

超高速マルチ精度化

並列計算においてはデータの転送がメインの計算時間になる

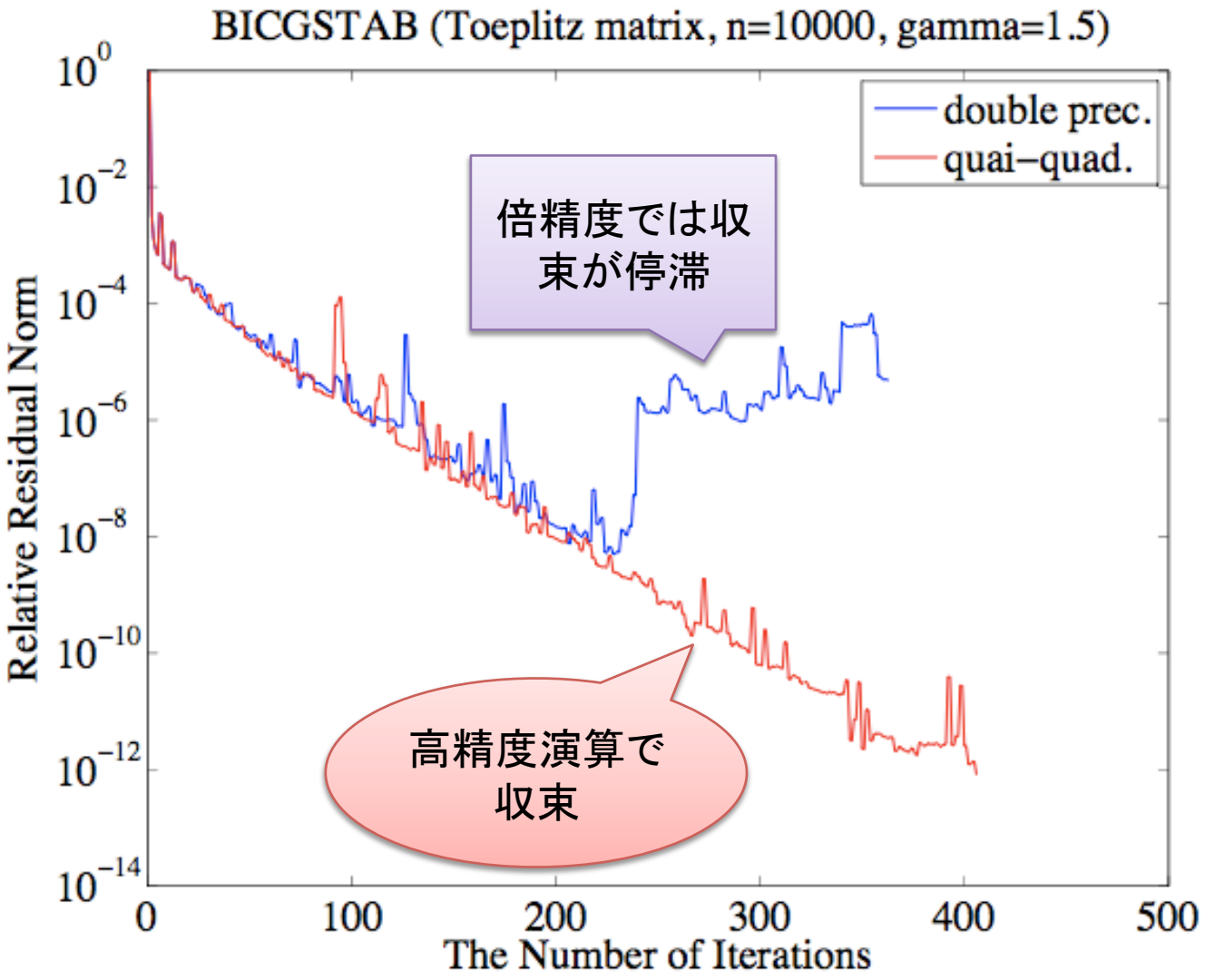
DMV	Data load/other overhead	+	fl-pt
QMV	Data load/other overhead	+	floating-point arithmetic

エラーフリー変換により高精度内積計算を導入し、行列・ベクトル積を高精度に実行させる（次頁を参照）。

トータルで従来計算の手間と比べてほぼ同じ手間でCG法が実行できるように。

超高速なマルチ精度線形アルゴリズムの創造

残差の収束履歴


BICGSTAB (Toeplitz matrix, n=10000, gamma=1.5)
double prec.
quai−quad.
倍精度では収束が停滞
高精度演算で収束
Relative Residual Norm
The Number of Iterations
10^0
10^−2
10^−4
10^−6
10^−8
10^−10
10^−12
10^−14
0
100
200
300
400
500