HPL-AI行列の観察

理化学研究所　CCS
工藤 周平

第8回HPC-Phys勉強会 9/24 オンライン

概要

- HPL-AIベンチマークの背景
- ルールの詳細
- 行列の解析
- 我々の実装上の工夫点
- 議論

HPL-AIベンチマークの背景

https://icl.bitbucket.io/hpl-ai/

- AIにおける低精度演算の性能要求の増大
  - 単精度や半精度など
  - AI向けHWを持つスパコンも登場している
- HPCにおける事実上標準のベンチマークHPLは“古典的”な倍精度演算のみ
- HPL-AIはこれら2つの存在を連続的につなげる存在

HPL-AIの計算

- 巨大な線型方程式を解く

$$A = LU, \qquad x_{s+1} = x_s + U^{-1}L^{-1}(b - Ax_s)$$

  - Aは巨大な“乱数”行列（詳細後記）
    - 行列の次元nは数千万次元になる
  - LU分解
    - 低精度演算を利用しても良い
  - 反復改良
    - 最終結果がHPLと同程度の精度を持つまで繰り返す

なぜ他のAIベンチマークではないのか

- HPL-AIは “AI” というわりにAIとの関係性がない
- MLPerfなどの専用ベンチマークは
  - 現在行われている計算に近く, AI性能の評価としてより適切だが
  - 巨大　具体的な仕事, 計算に縛られる
  - 動作させるだけで一苦労
- HPL-AIは
  - 単純　安定性のある理論を持ち
  - 現在のAIにおける特定の演算のみに着目
  - AIとのかかわりはない

ルールの詳細

- 計算は混合精度のLU分解と解の反復改良で構成される
- LU分解
  - $\frac{2}{3}n^3 + O(n^2)$ 回の**浮動小数点数演算**を行わなければならない
  - 倍精度演算を行う必要はない
- 反復改良
  - アルゴリズムは任意だが, LU分解の結果を使う**べき（should）**
  - 改良後の誤差が以下を満たす必要がある

$$\frac{\|Ax - b\|_\infty}{\|A\|_\infty \|x\|_\infty + \|b\|_\infty} \times (n \times \epsilon)^{-1} \leq 16$$

  - ϵは倍精度の計算機イプシロン　反復回数の上限は49回
- 性能数値は $(2/3n^3 + 3/2n^2)/t_{\text{total}}$ で換算する

HPL-AI行列の定義

Random matrix with elements in $[-0.5, 0.5]$

"The benchmark should use the HPL benchmark harness with a modification of the matrix generator. The modified generator should produce a **non-symmetric** matrix with **the diagonal entries being the sum of the row's off-diagonal elements,** this will force the matrix to be **diagonally dominant**."

```
for (j = 0; j < m; j++) {
      for (i = 0; i < m; i++) {
            A(i, j) = mcg_rand_double();
            diag[i] += fabs(A(i, j));  }}
for (i = 0; i < m; i++) {
      A(i, i) = diag[i] - fabs(A(i, i));}
```

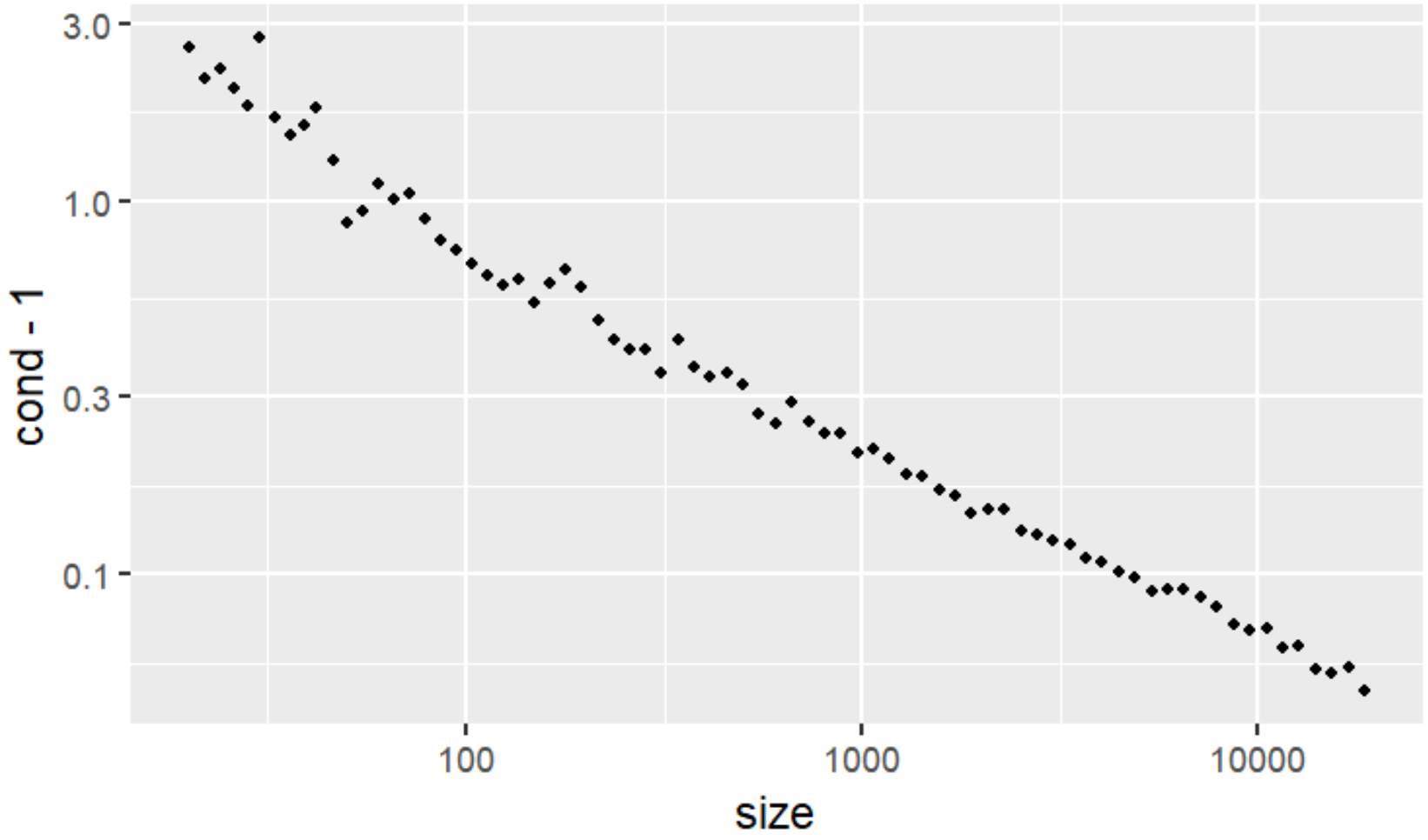
HPL-AI行列の性質

- 劣優対角
  - 対角が$O(n)$, 非対角が$O(1)$
    - LU分解での1ステップでの更新量が$O(1/n)$
- 未証明だが具体的な行列で観察された性質
  - 乱数を仮定すると証明も可能だろうと推測している
  - 良条件
  - ヤコビ法の収束レートが良い
  - 容易かつ低演算量, 高精度にLU分解を近似できる

狭義優対角性

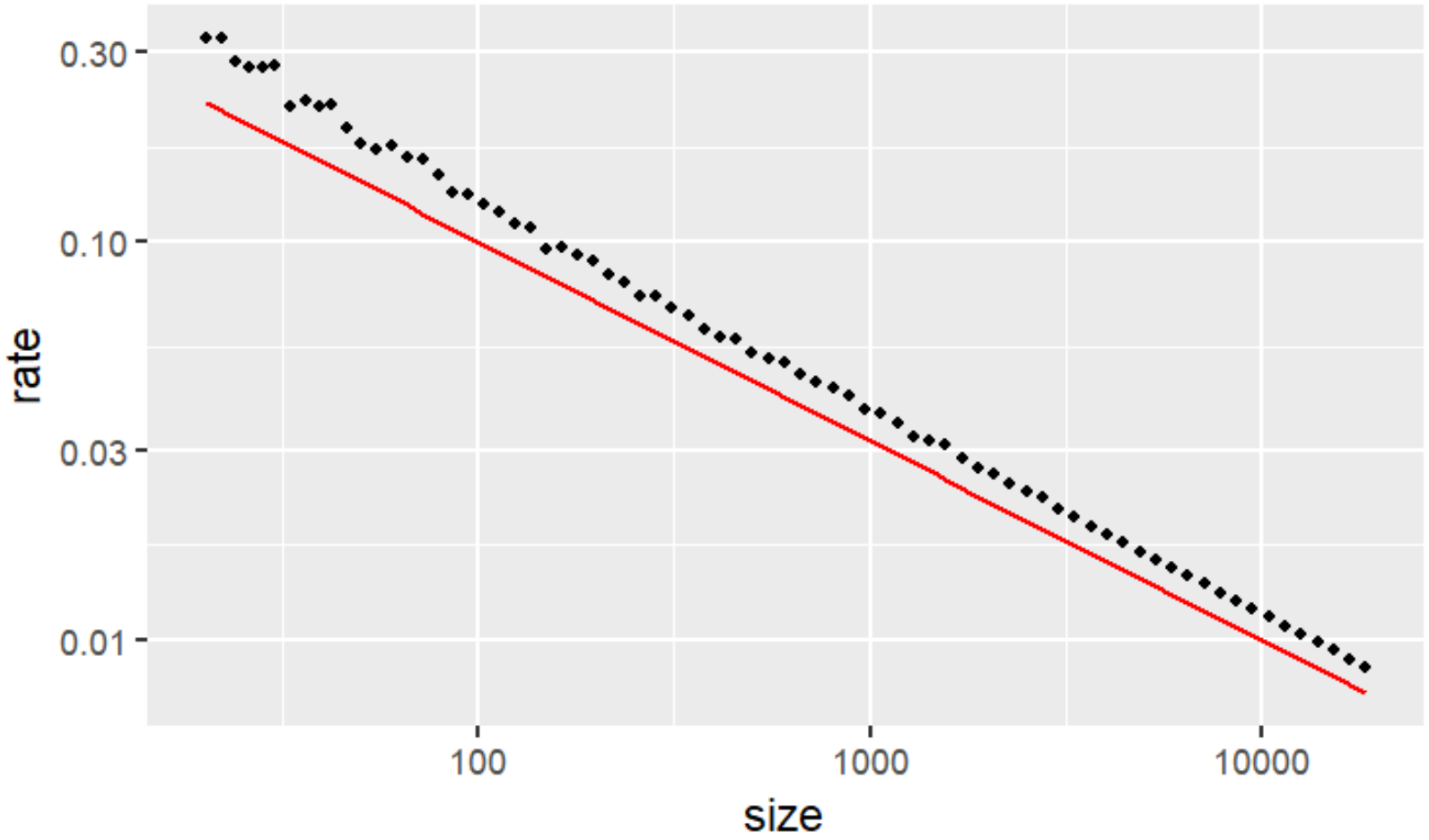
- 行列Aが狭義優対角のとき
  - 非特異
  - LU分解時にPivotingを行わなくとも
    growth factorが小さな定数で抑えられる
- HPL-AI行列は劣優対角のため上記の性質はすぐにはいえない
  - きわめて運が悪い場合に行列が特異になるかもしれない
    - 疑似乱数生成器が非常に悪い性質をもつ場合
      - 疑似乱数の周期と行列の次元が同期してしまうなどのケース
        - 古いHPLにあるバグ
    - ベンチマークにおいてこの不運を特別注目する意義はない

HPL-AI行列の条件数



２ノルム条件数
対角要素の最大値と最小値，
非対角部分の2ノルムから
推定
ノルムはべき乗法で推定

HPL-AI行列のヤコビ法収束レート



ヤコビ法の収束レート
$\|D^{-1}(A-D)\|_2$
ノルムはべき乗法で推定

HPL-AI行列のLU因子近似手法

- HPL-AIはとくに対角成分が大きいため

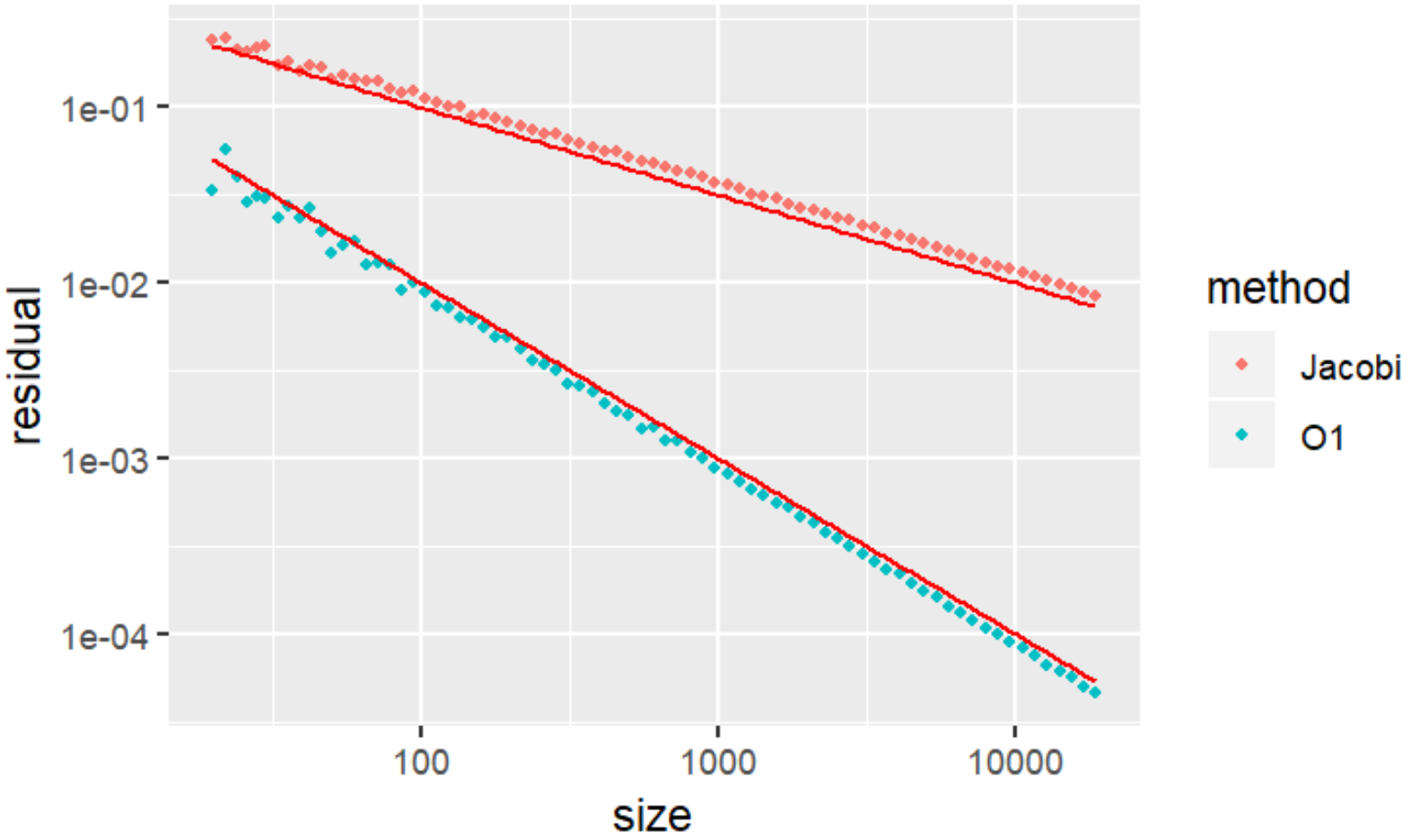
$$A \approx L_{\mathrm{J}} U_{\mathrm{J}} = I_n D, \qquad D = \mathrm{diag}(A).$$

  - 対角要素のみ残し, ほかは0にする
  - 対角は$O(n)$, 非対角は乱数行列なので$O(\sqrt{n})$程度の2ノルムを持つ
    - 対角と非対角が関係を持つため, まったくいい加減な議論
- 非対角成分を多少考慮する

$$A \approx L_O U_O = (TD^{-1} + I)(D + S), \qquad T = \mathrm{tril}(A), S = \mathrm{triu}(A).$$

  - T, S は A の狭義下(上)三角部分
  - $(TD^{-1} + I)(D + S) = (T + D + S) + TD^{-1}S = A + TD^{-1}S$

HPL-AI行列のLU因子近似誤差



残差
$\|A - L_X U_X\|_2 / \|A\|_2$
ノルムはべき乗法で推定
Jacobiは$L_J U_J$を使うもの
O1は$L_O U_O$を使うもの
赤線はそれぞれ上から
$1/\sqrt{x}$と$1/x$

HPL-AI行列は簡単

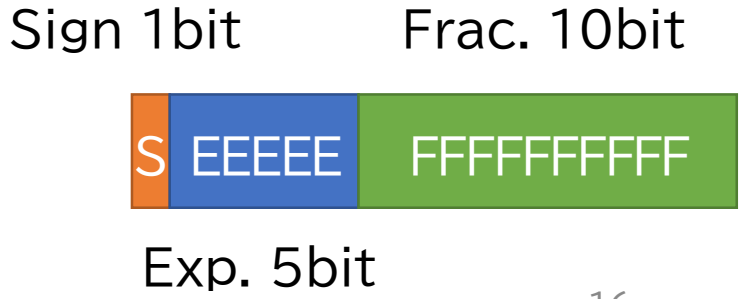
- あまりに簡単すぎるため, チートは容易
  - チートをしたということではない
- 手順上はただしく実装しても, 無意味な計算となることがある
  - 結果的にチートと同じ分解をした場合, 何の意味があるのか
  - 無意味な計算となることで性能数値が良くなってはならない
    - 関連: ルールにピボットがない理由

実装戦略

- 三精度混合
  - LU分解 – 単精度
    - ただし**GEMMのみ半精度**
  - 反復改良 – 倍精度
- 2つの実装上の注意
  - FP16の値域
    - スケーリングによってover/underflowを排除する
  - FP16の精度
    - 丸めによって演算が無意味にならないように, 演算順序を工夫する

FP16のフォーマット

- 符号1, 指数5, 仮数10bit
- 0より大きい正規化数の値域: 2^{-14} から 2^{16}より小まで
  - HPL-AI行列は対角が$O(n)$, 更新量が$O(1/n)$なのでn^2の比があり, スケーリングなしではover/underflowが発生する
- 丸め誤差の単位: 2^{-11}
  - HPL-AI行列では$O(1)$の非対角に$O(1/n)$の更新量を足すため, なにもしなければ更新幅が(ほぼ)すべて丸められる
  - 計算結果が前記の近似分解と一致する



演算順序の並び替え

- LU分解中の(i,j)ブロックは初期値$A_{i,j}^{(0)}$に更新幅を引いたもの

$$A_{i,j}^{(m)} = A_{i,j}^{(0)} - \sum_{k=1}^{m} L_i^{(k)} U_j^{(k)}$$

  - $L_i^{(k)}, U_j^{(k)}$は左・右パネル
- 初期値と比べて更新幅が小さい→更新幅の和に初期値をたす

$$A_{i,j}^{(m)} = \left(- \sum_{k=1}^{m} L_i^{(k)} U_j^{(k)} \right) + A_{i,j}^{(0)}$$

  - 更新幅どうしは同程度の大きさ→和の結果が無視されにくくなる

スケーリングの戦略

- Static single scaling

$$A_{i,j}^{(m)} = s^{-1}\left(-\sum_{k=1}^{m}\left(sL_i^{(k)}\right)U_j^{(k)}\right) + A_{i,j}^{(0)}$$

  - スカラーsは行列サイズnのみに依存する数値で$s = n^{\frac{3}{4}}$
    - $\left\|sL_i^{(k)}\right\|_{\max} \sim O\left(n^{-1/4}\right)$
    - 最大$n-1$回の和で$\sqrt{n-1}$倍程度値が拡大しても$O\left(n^{\frac{1}{4}}\right)$に収まる
  - Lパネルのみスケール：初期値と足すときに逆スケーリング

議論

- 問題設定は適切か
  - 簡単すぎやしないか
  - なぜPivotを不要にするのか
    - 意図的なのか？ HPLの技術の大半を忘れる理由は？
- これほど簡単でなければAI向けHWを数値計算に活用できない？
  - 数値計算にもいろいろあるので……
  - 密行列計算にもいろいろあるが
- AI性能の評価として適切か
  - GEMMの性能ぐらいしか参考にならない
    - GEMMも大きさが異なれば性質が大きく違う

Fasi and HighamのHPL-AI行列

- N. J. Higham HPL-AIのcollaboratorの一人
- HighamのGithub repository https://github.com/higham/hpl-ai-matrix
- M. Fasi, N. J. Higham, “Matrices with **Tunable** Infinity-Norm **Condition Number** and **No Need for Pivoting** in LU Factorization”, MIMS Eprint 2020.17.
  - コードが先行して公開されていた
  - ペーパーは8月はじめ

FH行列の詳細

$$A(\alpha,\beta)=\begin{bmatrix} 1 & & & & \\ -\alpha & 1 & & & \\ -\alpha & -\alpha & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ -\alpha & -\alpha & \cdots & -\alpha & 1 \end{bmatrix}\begin{bmatrix} 1 & -\beta & -\beta & \cdots & -\beta \\ & 1 & -\beta & \cdots & -\beta \\ & & \ddots & \ddots & \vdots \\ & & & 1 & -\beta \\ & & & & 1 \end{bmatrix}$$

- $\alpha,\beta\sim O(1/n)$：パラメーター
- $\kappa_\infty\big(A(\alpha,\beta)\big)=f(\alpha,\beta)$ を陽的に計算可能
  - $\alpha=0.5\beta$ のようにパラメーター数を減らしてゼロ点探索すれば, 任意の条件数の行列を生成するパラメーターを簡単に見つけることができる
- プログラミング上の取り扱いが簡単で数値的には難しくできる

FH行列の特徴

- 容易に生成可能
  - HPL-AIのように大規模にも対応できる
- 条件数を任意に設定可能
- 良い数値的性質
  - LU分解時にピボット不要かつGrowth factorが小さい
- **定数の和**が出現する
  - 計算精度が大きく悪化　FP32のアキュムレーターが必要？
  - FP16を用いても計算できる方法を思案中