固有値計算アルゴリズムの最近の進展
I. 対称密行列向け並列解法

2003年3月27日

山本有作

目次

1. 固有値計算の概要
2. 高性能化のための手法
3. 逆反復法の並列化
4. 三重対角化部分の並列化と高性能化
5. 今後の展開

1．はじめに

- 固有値計算とその応用
- 固有値・固有ベクトル計算の流れ
- 二分法・逆反復法による固有値・固有ベクトル計算

固有値計算とその応用

- 対象とする問題
  - 標準固有値問題　$\mathbf{Ax} = \lambda \mathbf{x}$　（A: 実対称またはエルミート行列）
- 対象とする計算機
  - 共有メモリ型並列計算機
  - 分散メモリ型並列計算機
  - キャッシュマシン
- 固有値計算の応用分野
  - 構造解析，電子状態計算，導波路解析など

固有値･固有ベクトル計算の流れ

実対称行列 A

計算内容

計算手法

三重対角化

$Q^{*}AQ = T$
(Q: 直交行列)

ハウスホルダー法

三重対角行列 T

三重対角行列の
固有値・固有ベクトル計算

$Ty_i = \lambda_i y_i$

QR法
DC法
二分法・逆反復法

Tの固有値 $\{\lambda_i\}$,
固有ベクトル $\{y_i\}$

逆変換

$x_i = Qy_i$

逆変換

Aの固有ベクトル $\{x_i\}$

三重対角行列Tに対する各計算手法の特徴

解法	概要	速度	並列化	一部の固有値・固有ﾍﾞｸﾄﾙのみの計算
QR法	相似変換によりTを対角行列に変換	○	容易	不可
DC法（分割統治法）	Tを再帰的に半分のｻｲｽﾞの行列に分割	◎	容易	不可
二分法・逆反復法	二分法で固有値，逆反復法で固有ﾍﾞｸﾄﾙを計算	○	非自明	可

今回は二分法・逆反復法を取り上げる。

二分法・逆反復法による固有値・固有ベクトル計算

実対称行列 A

計算内容

↓

三重対角化

$Q^{*}AQ = T$

(Q: 直交行列)

↓ 三重対角行列 T

二分法

$|T-\lambda_i I| = 0$ を解いて λ_i を求める。

↓ Tの固有値 $\{\lambda_i\}$

逆反復法

$Ty_i = \lambda_i y_i$ を解いて y_i を求める。

↓ Tの固有ベクトル $\{y_i\}$

逆変換

$x_i = Qy_i$

↓

Aの固有ベクトル $\{x_i\}$

二分法・逆反復法による固有値・固有ベクトル計算

実対称行列 A

⬇

三重対角化　　$Q^{*}AQ = T$ (Q: 直交行列)　　第４章で説明

⬇ 三重対角行列 T

二分法　　$|T-\lambda_i I| = 0$　　固有値毎に別のプロセッサを用いて探索

⬇ Tの固有値 $\{\lambda_i\}$

逆反復法　　$Ty_i = \lambda_i y_i$　　第３章で説明

⬇ Tの固有ベクトル $\{y_i\}$

逆変換　　$x_i = Qy_i$　　固有ベクトル毎に独立に逆変換

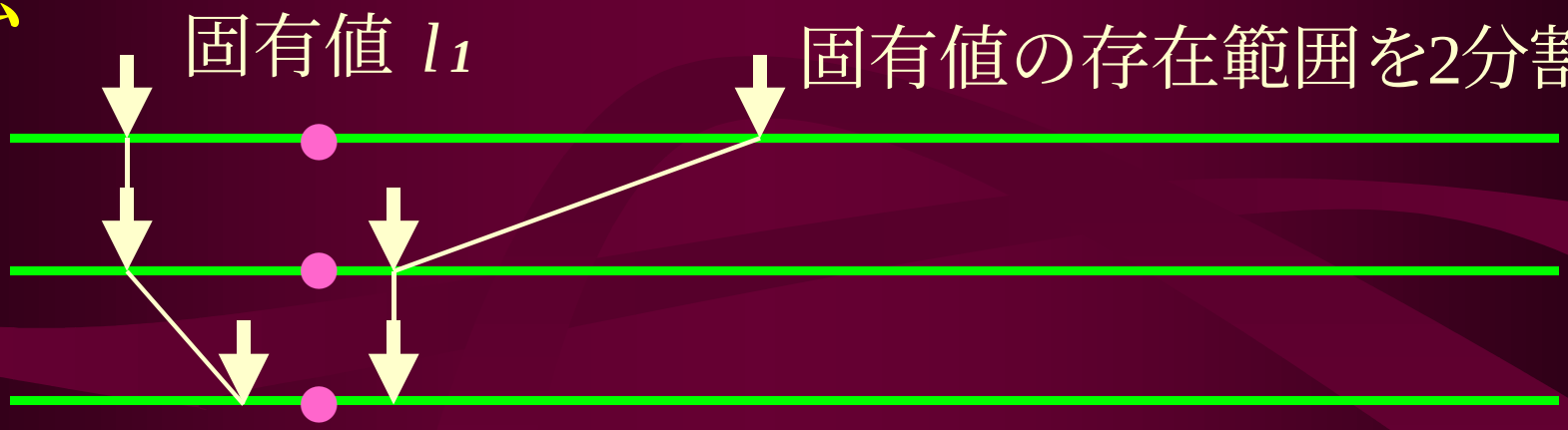
⬇

Aの固有ベクトル $\{x_i\}$

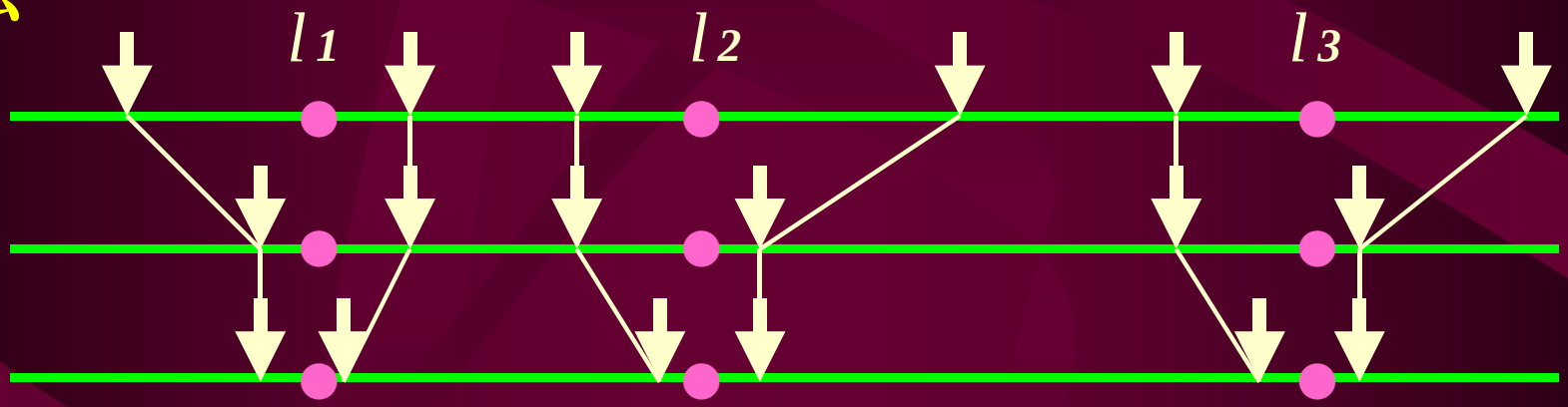
計算内容　　並列化手法

二分法の並列化

二分法



多分法



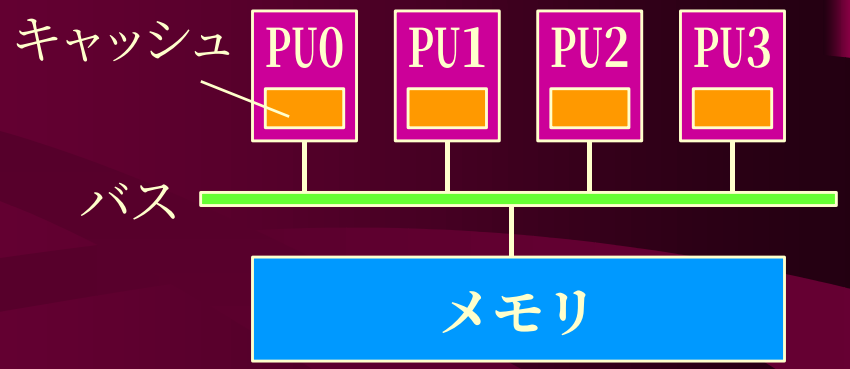
固有値ごとに別プロセッサで検索

2. 高性能化のための手法

- 共有メモリ型並列計算機
- 分散メモリ型並列計算機
- キャッシュメモリの有効利用
- BLASの利用による高性能化

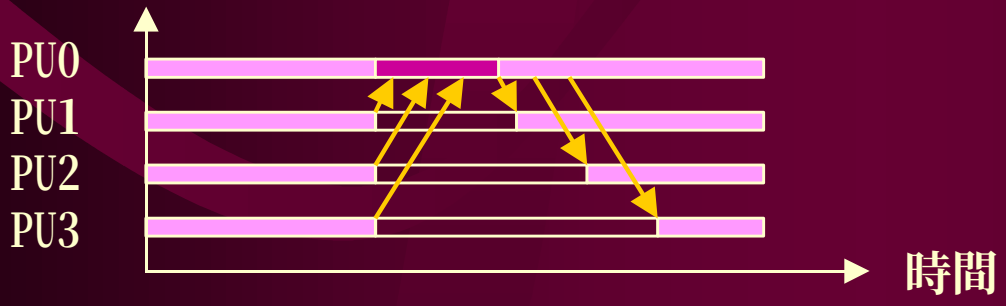
共有メモリ型並列計算機

- 構成
  - 複数のプロセッサ（PU）がバスを通してメモリを共有
  - PUはそれぞれキャッシュを持つ。



- 特徴
  - メモリ空間が単一のためプログラミングが容易
  - PUの数が多すぎる（＞20個）と，アクセス競合により性能が低下

- PU間同期
  - 他PUの計算結果を使って計算を行う場合は，同期が必要



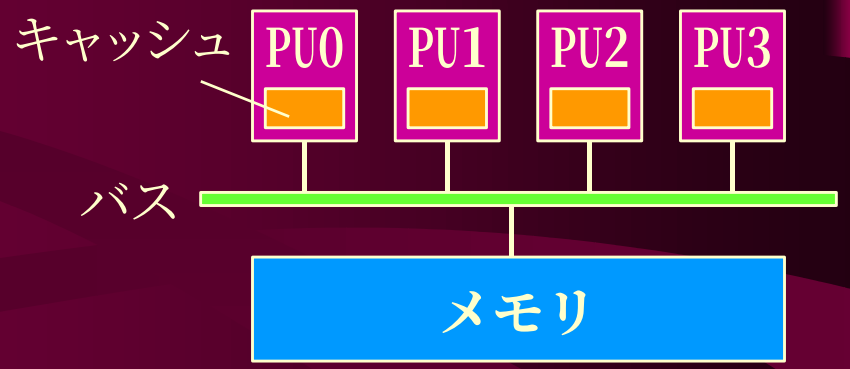
内積計算の例

```
DO IP = 0, 3
  DO I = 1, 25
    S(IP) = S(IP)
      + A(IP*25+I)*B(IP*25+I)
  END DO
END DO
S = S(0)+S(1)+S(2)+S(3)
```

PU間同期

共有メモリ型並列計算機向けの高性能化手法

- PU間の負荷分散均等化
  - 各PUの処理量が均等になるよう処理を分割
  - 密行列向けのアルゴリズムでは比較的容易

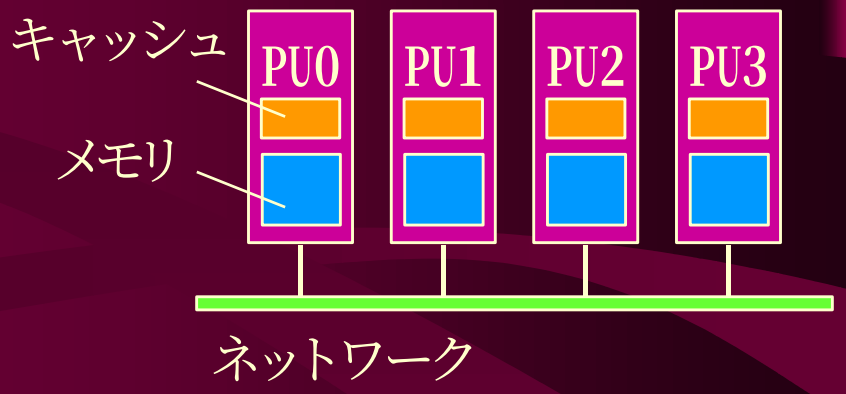


- PU間同期の削減
  - 同期には通常，数百サイクルの時間が必要
  - 並列粒度の増大による同期回数の削減が性能向上の鍵

- キャッシュメモリの有効利用
  - 演算順序の変更等により，キャッシュ中のデータの再利用性を向上
  - PU間でのアクセス競合を防ぎ，アクセスを高速化

分散メモリ型並列計算機

- 構成
  - 各々がメモリを持つ複数のPUをネットワークで接続
  - PUはそれぞれキャッシュを持つ。

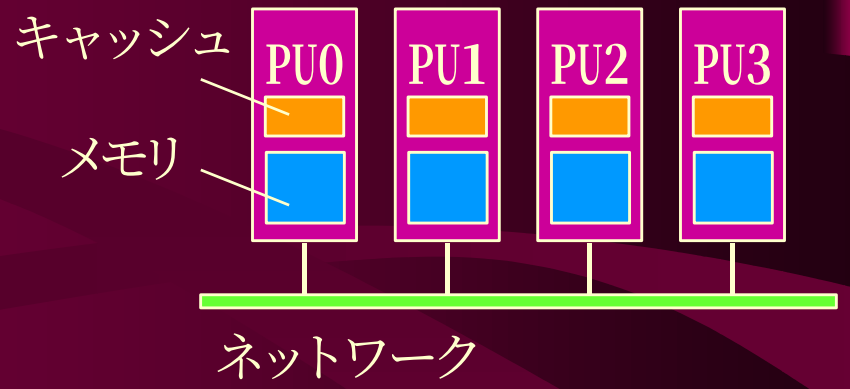


- 特徴
  - 数千～数万PU規模の並列が可能
  - PU間へのデータ分散を意識したプログラミングが必要

- PU間でのデータ転送
  - 他PUの計算結果，あるいは他PUの持つデータを使って計算を行う場合は，PU間でのデータ転送が必要

分散メモリ型並列計算機向けの高性能化手法

- PU間の負荷分散均等化
  - 共有メモリ型の場合と同様



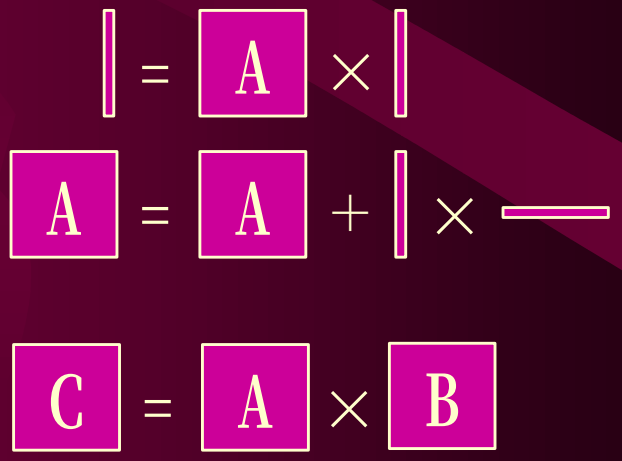
- データ転送の削減
  - 転送には通常，数千サイクルのセットアップ時間が必要
  - データ１個の転送には，演算１回の数十倍の時間が必要
  - アルゴリズムとデータ分散方法の工夫により，データ転送量・転送回数を削減することが性能向上の鍵

- キャッシュメモリの有効利用
  - 演算順序の変更等により，キャッシュ中のデータの再利用性を向上
  - 相対的に遅い主メモリへのアクセスを削減し，計算を高速化

BLASの利用による高性能化

- BLASとは
  - Basic Linear Algebra Subprogramsの略
  - 個々のマシン向けにチューニングした基本行列演算のライブラリ

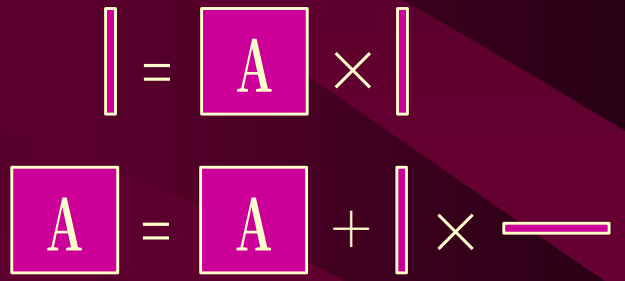
- BLASの種類
  - BLAS1: ベクトルとベクトルの演算
    - 内積　$c := \mathbf{x}^t \mathbf{y}$
    - AXPY演算　$\mathbf{y} := a\mathbf{x} + \mathbf{y}$ など
  - BLAS2: 行列とベクトルの演算
    - 行列ベクトル積　$\mathbf{y} := A\mathbf{x}$
    - 行列のrank-1更新　$A := A + \mathbf{x}\mathbf{y}^t$
  - BLAS3: 行列と行列の演算
    - 行列乗算　$C := AB$ など



BLASにおけるデータ再利用性と並列粒度

- BLAS1
  - 演算量： O(N)，　データ量： O(N)
  - データ再利用性： なし
  - 並列粒度： O(N/p)　（N: ベクトルの次元，p: プロセッサ台数）

- BLAS2
  - 演算量： $O(N^2)$，　データ量： $O(N^2)$
  - ベクトルデータのみ再利用性あり
  - 並列粒度： $O(N^2/p)$



- BLAS3
  - 演算量： $O(N^3)$，　データ量： $O(N^2)$
  - O(N)回のデータ再利用が可能
  - 並列粒度： $O(N^3/p)$

演算をできる限り BLAS2，3で行うことが高性能化のポイント

3. 逆反復法の並列化

- 逆反復法による固有ベクトル計算
- 従来の並列化手法とその問題点
- マルチカラー逆反復法
- ハウスホルダー逆反復法
- 性能評価

逆反復法による固有ベクトル計算 (1)

- 逆反復法の原理
  - T:　三重対角行列
  - λ'_i : Tの固有値 λ_i の近似値
  - 適当な初期ベクトル $\mathbf{v}_i^{(0)}$ から出発し，次の反復を行う。

$$\mathbf{v}_i^{(m)} := (T - \lambda'_i I)^{-1} \mathbf{v}_i^{(m-1)}$$

  - $\mathbf{v}_i$ に平行な成分が，１反復毎に $(\lambda_i - \lambda_i')^{-1}$ 倍に拡大される。
  - $\mathbf{v}_i^{(m)}$ は $\mathbf{v}_i$ に収束

逆反復法による固有ベクトル計算 (2)

- 直交化の問題
  - 実対称行列（エルミート行列）の異なる固有値に属する固有ベクトルは，本来直交する。
  - しかし，2個の固有値 λ_i と λ_j とが近接するとき，逆反復法では $\mathbf{v}_i$ だけでなく $\mathbf{v}_j$ の成分も拡大される。

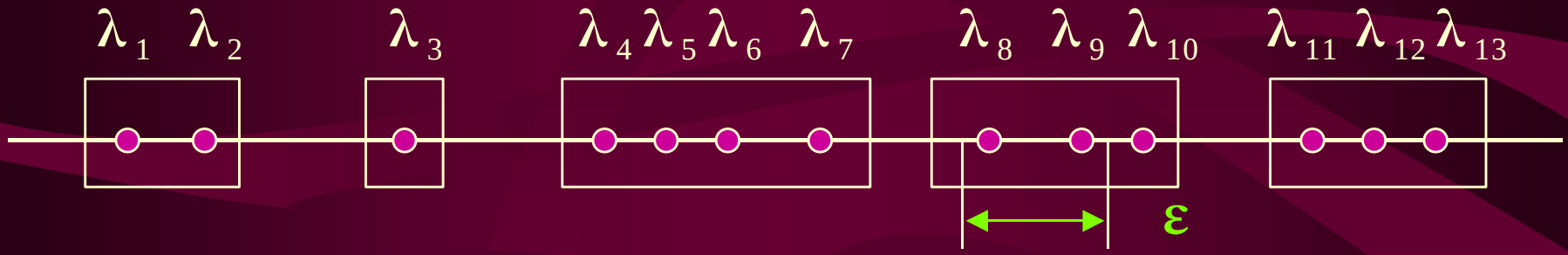
計算により得られる固有ベクトルでは，$\mathbf{v}_i$ と $\mathbf{v}_j$ の成分とが混じり合い，直交性が崩れる。

固有ベクトルの直交性を保証する何らかの方法が必要

この部分が並列化阻害要因となる。

逆反復法による固有ベクトル計算 (3)

- 直交性の保証方法
  - 2個の固有値の距離が基準値 ε 以内の場合，同じグループに属すると定義し，固有値のグループ分けを行う。



  - 同一グループに属する固有ベクトルすべてに対し，直交化を行う。
  - 直交化には，修正 Gram-Schmidt 法を用いる。

```
DO i = IS, IE
    DO k = IS, i -1
        v_i^(m) := v_i^(m) – (v_i^(m) ・ v_k) v_k
    END DO
    v_i^(m) := v_i^(m) / ‖ v_i^(m) ‖
END DO
```

グループ内での修正Gram-Schmidt法による直交化（２重ループ）

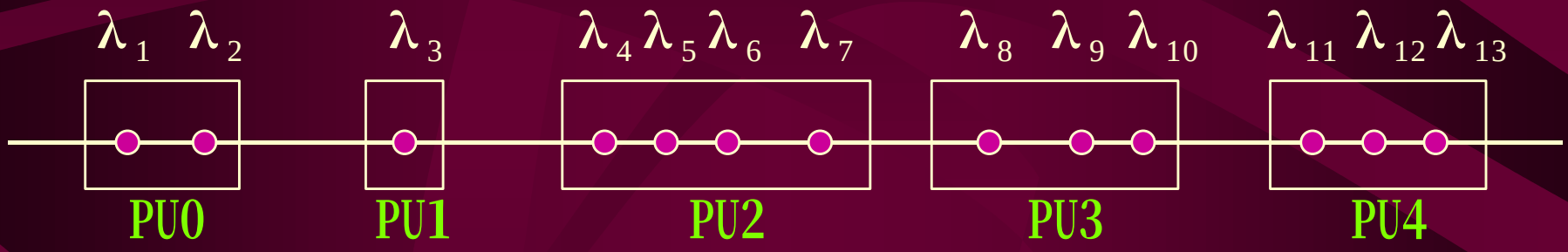
従来の並列化手法

- 計算の並列性
  - 異なるグループに属する固有ベクトルは独立に計算可能
  - 修正Gram-Schmidt法は最内側ループに関してのみ並列性あり

- 2種の並列化方法

(1) 各固有値グループを1個のプロセッサに割り当てて並列化

- ScaLAPACK で採用　（L. Blackford et al., 1997）



(2) 修正Gram-Schmidt法の最内側ループで並列化

- $\mathbf{v}_i^{(m)} := \mathbf{v}_i^{(m)} - (\mathbf{v}_i^{(m)} \cdot \mathbf{v}_k)\,\mathbf{v}_k$ において内積とAXPYを並列化

従来の並列化手法の問題点

- 並列化手法 (1) の問題点
  - 大規模問題では，固有値が密集し，全固有値が1個のグループになることが多い。
  - この場合，この方法では並列化不可
- 並列化手法 (2) の問題点
  - BLAS1レベル（内積，AXPY演算）の並列化
  - 並列粒度は O(N/p)
  - PU間同期のオーバーヘッド大

これらの手法では，効率の良い並列化は困難

新しい並列化手法

- Dhillonのアルゴリズム（Dhillon, 1997）
  - 直交化を行うことなしに，直交性の高い固有ベクトルを計算
- マルチカラー逆反復法（Naono et al., 2000）
  - 誤差評価に基づき直交化を必要な部分のみに限定し，同時に演算の並列性を抽出
- ハウスホルダー逆反復法（山本他, 2001）
  - 直交化の手法として，修正 Gram-Schmidt 法に代わる並列性の高い手法を用いる。

Dhillonのアルゴリズム

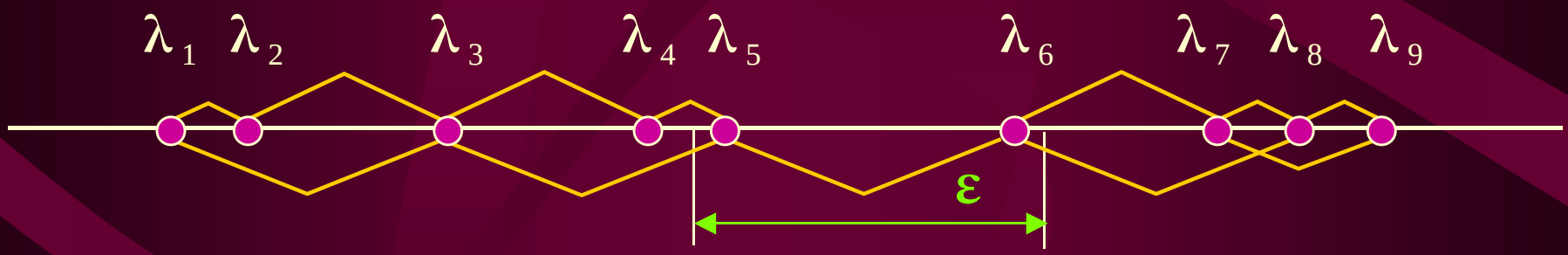
- 原理（Dhillon, 1997）
  - 逆反復法において，Twisted LU分解と呼ぶ特殊な手法を利用
  - 直交化を行うことなしに，直交性の高い固有ベクトルを計算可
  - 固有ベクトルの本数分の並列性を持つ。
- 問題点
  - 固有値を極めて高い精度で求めておく必要あり。
    - 従来の二分法は適用不可
  - 固有値が極めて縮重に近いとき，直交性が悪化する場合あり。

マルチカラー逆反復法 (1)

- マルチカラー逆反復法（Naono et al., 2000）
  - 従来の逆反復法における固有値のグループ分けを廃止
  - 2個の固有値の距離が基準値以下の場合に限り，対応する固有ベクトルのペアに対して直交化を行う。

直交化すべき固有ベクトルの数を大きく削減

直交化を行う固有値ペア



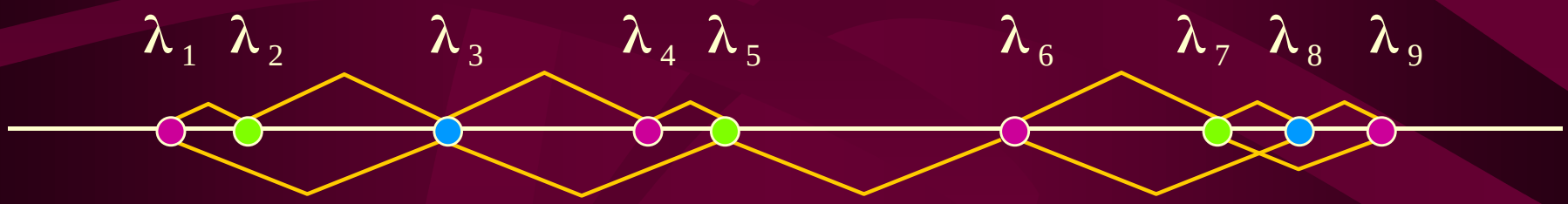
・線で結ばれた固有値に対する固有ベクトルのペアを直交化
・古典的逆反復法では，全ペアに対する直交化が必要

マルチカラー逆反復法 (2)

- マルチカラー逆反復法における並列化
  - 線でつながっていない固有ベクトルは同時に計算可能
  - グラフ理論の頂点彩色アルゴリズムを用いて並列性を抽出

数多くの例題に対して高い並列化効率を達成

固有ベクトルの計算順序



- **ステップ１で計算（並列度４）**
- **ステップ２で計算（並列度３）**
- **ステップ３で計算（並列度２）**

- 問題点
  - 直交化削減の影響で，固有ベクトルの直交性が悪化する場合あり。

ハウスホルダー逆反復法 (1)

- アルゴリズムの開発方針
  (1) 直交化を削減／廃止するのではなく，直交化計算自体を並列化
    - これにより，逐次計算機での逆反復法と同等の精度を目指す。
  (2) 直交化に必要なPU間同期回数を削減
  (3) 上記(1), (2)を満たせれば，演算量の多少の増加は許容

修正Gram-Schmidt 法の代わりに，ハウスホルダー変換（鏡像変換の一種）による直交化手法に着目

- **ハウスホルダー変換による直交化の特徴**
  - **直交性は極めてよい。**
  - **計算はBLAS2（行列ベクトル積，行列のrank-1更新）からなる。並列粒度は $O(N^2/p)$ で，Gram-Schmidt 法のN倍**
  - **演算量は修正Gram-Schmidt法の1.5倍**

ハウスホルダー逆反復法 (2)

- 原理（山本他，2001）
  - 既に求めた固有ベクトルの張る空間 $V_{i-1} = \mathrm{span}\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_{i-1}\}$ とその直交補空間 $V_{i-1}^{\perp}$ の正規直交基底 Q_{i-1} を陽的に保持
  - 逆反復で求めたベクトルを $V_{i-1}^{\perp}$ に射影することにより，V_{i-1} に直交する固有ベクトル $\mathbf{v}_i$ を求める。
  - Q_{i-1} の第1列を $\mathbf{v}_i$ に変換するハウスホルダー変換を求める。
  - Q_{i-1} にこのハウスホルダー変換を作用させ，行列 Q_{i-1}' に変換
  - Q_{i-1}' の第1列（$\mathbf{v}_i$）を除去してできる行列を Q_i とする。
  - 除去した第1列を V_{i-1} に加え，V_i とする。

ハウスホルダー逆反復法のアルゴリズム

$Q_1 = I_N$
$V_0 = \phi$（N×0の行列）
do i=1, N
　逆反復の初期値$\mathbf{v}_i^{(0)}$を設定する。
　m=1
　固有ベクトル$\mathbf{v}_i^{(m)}$が収束するまで以下を繰り返す。
　　$\mathbf{v}_i := (T-\lambda_i I)^{-1} \mathbf{v}_i^{(m-1)}$　：逆反復により直交化前の $\mathbf{v}_i$ を求める。
　　$\mathbf{p}_i = Q_i^t \mathbf{v}_i$　：直交補空間 $V_{i-1}^{\perp}$ への射影
　　$\mathbf{p}_i$の第２成分以下を０にするハウスホルダー変換
　　$H_i = I_{N-i+1} - \alpha_i \mathbf{w}_i \mathbf{w}_i^t$を求める。
　　$\mathbf{q}_i = \alpha_i Q_i \mathbf{w}_i$　：ハウスホルダー変換前半
　　$\mathbf{v}_i^{(m)} = (Q_i)_1 - \mathbf{q}_i (\mathbf{w}_i^t)_1$
　　m := m+1
　$Q_i' = Q_i - \mathbf{q}_i \mathbf{w}_i^t$　：ハウスホルダー変換後半（基底Q_iの更新）
　$V_i = [V_{i-1} | (Q_i')_1]$
　Q_i'の第１列を取り去ってできる行列をQ_{i+1}とする。
end do

ハウスホルダー逆反復法の特徴

- 長所
  - 直交化にハウスホルダー変換を用いているため，直交性は従来法と同等かそれ以上
  - 計算の主要部は BLAS2 であり，並列粒度は従来法のN倍
  - 更に，複数の固有ベクトルをまとめて処理することで，計算の主要部をBLAS3 にでき，キャッシュの有効利用が可能

- 短所
  - 全固有値が１個のグループに属するとき，合計の演算量は $3N^3$ であり，従来の逆反復法の1.5倍

  - 共有メモリ型並列機におけるPU間同期のコストを考えると，長所が短所を上回り，従来法より高速なアルゴリズムが得られると予想

ハウスホルダー逆反復法の性能評価 (1)

- 評価環境
  - 日立 SR8000の１ノード（8プロセッサの共有メモリ型並列機）で評価
  - ピーク性能は８GFLOPS（E0モデル）または12GFLOPS（F1モデル）

- 評価例題

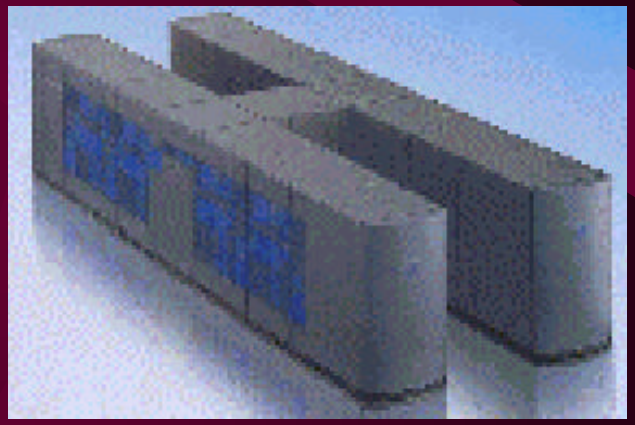
  (1) フランク行列 $A_{ij} = \min(i, j)$

  (2) $\lambda_i = i$ なる対角行列を直交変換してできた密行列

  (3) 一般固有値問題
  - 次元数 N はそれぞれ1000, 2000, 4000 の３通り

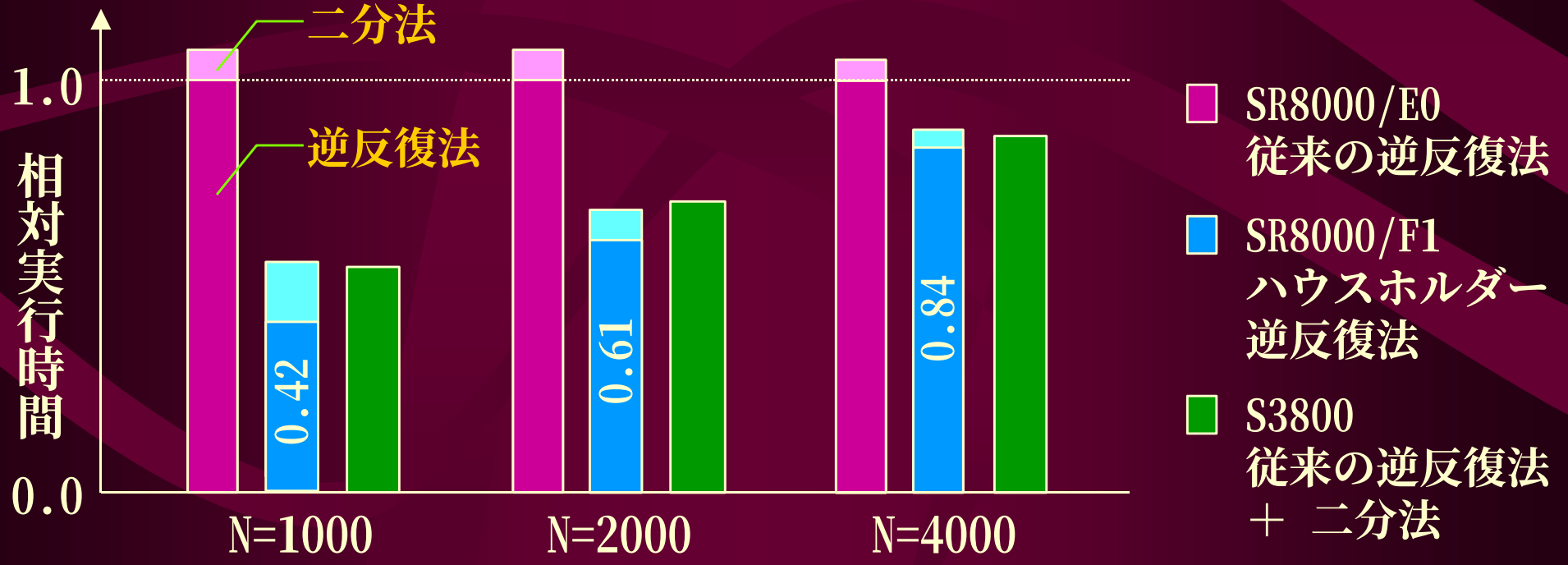
- 比較対象
  - SR8000での従来の逆反復法
  - ベクトル計算機 日立 S3800
    （ピーク性能8GFLOPS）での
    従来の逆反復法

SR8000



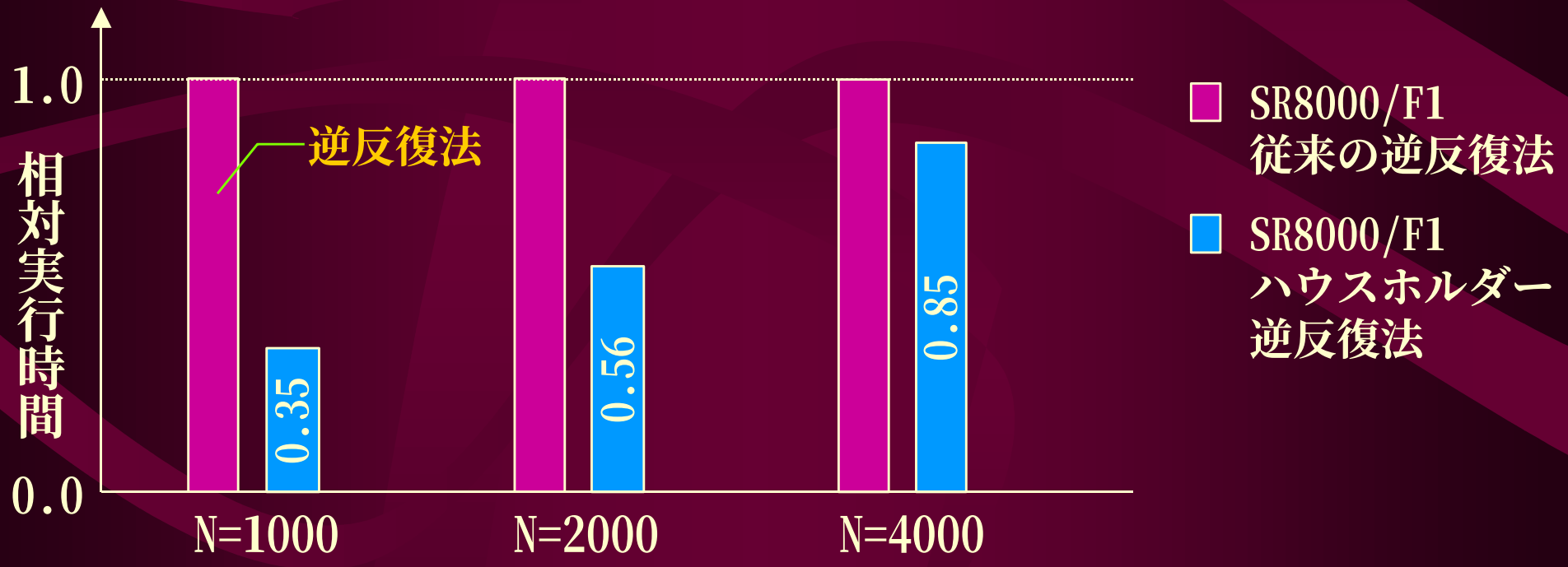
ハウスホルダー逆反復法の性能評価 (2)

- 8GFLOPSのマシン（SR8000/E0）での評価結果
  - ハウスホルダー逆反復法は，従来の逆反復法の1.2 ～ 2.4倍の性能
  - SR8000の二分法 + ハウスホルダー逆反復法では，S3800と同等の性能が得られる。
  - 3種類の例題に対し，実行時間はほぼ同じ。



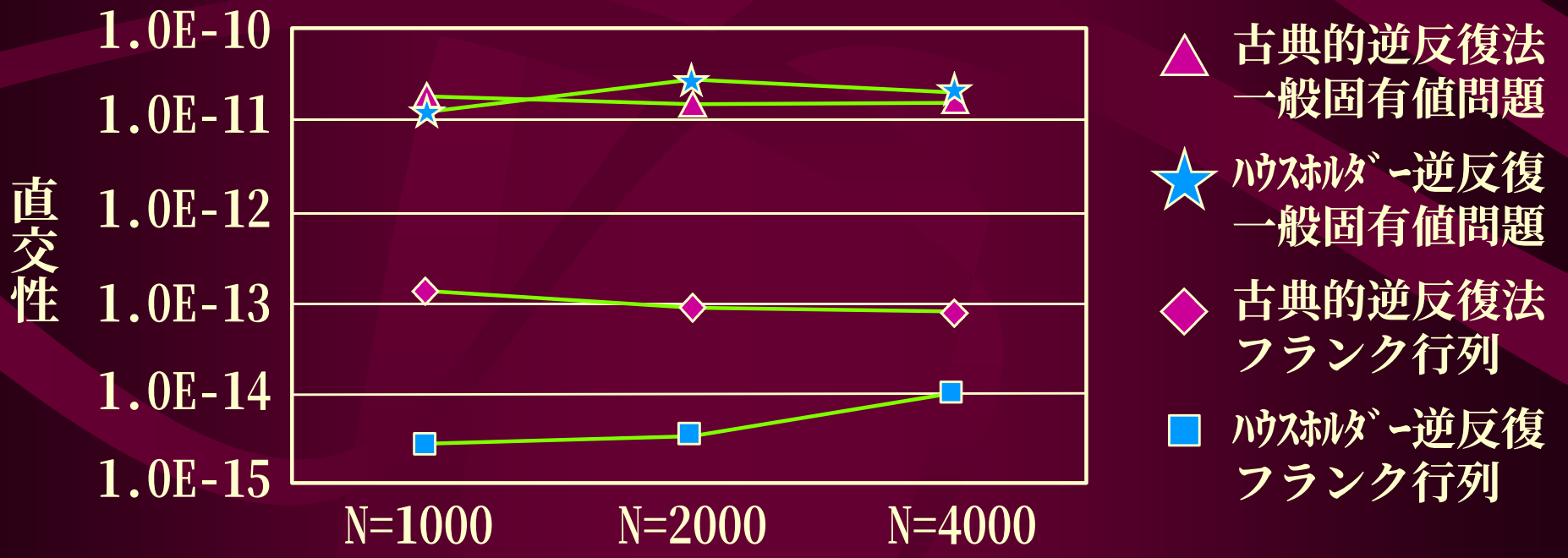
ハウスホルダー逆反復法の性能評価 (3)

- 12GFLOPSのマシン（SR8000/F1）での評価結果
  - ハウスホルダー逆反復法は，従来の逆反復法の1.2 ～ 2.9倍の性能
  - 8GFLOPSのマシンの場合より，更に大きい性能向上効果



ハウスホルダー逆反復法の精度評価

- 評価例題
  - フランク行列と一般固有値問題で固有ベクトルの直交性 $|V^t V - I|$ と残差 $|Av - \lambda v|$ を評価
- 評価結果
  - フランク行列では直交性が一桁向上，一般固有値ではほぼ同等
  - 残差は両例題とも，両手法でほとんど一致



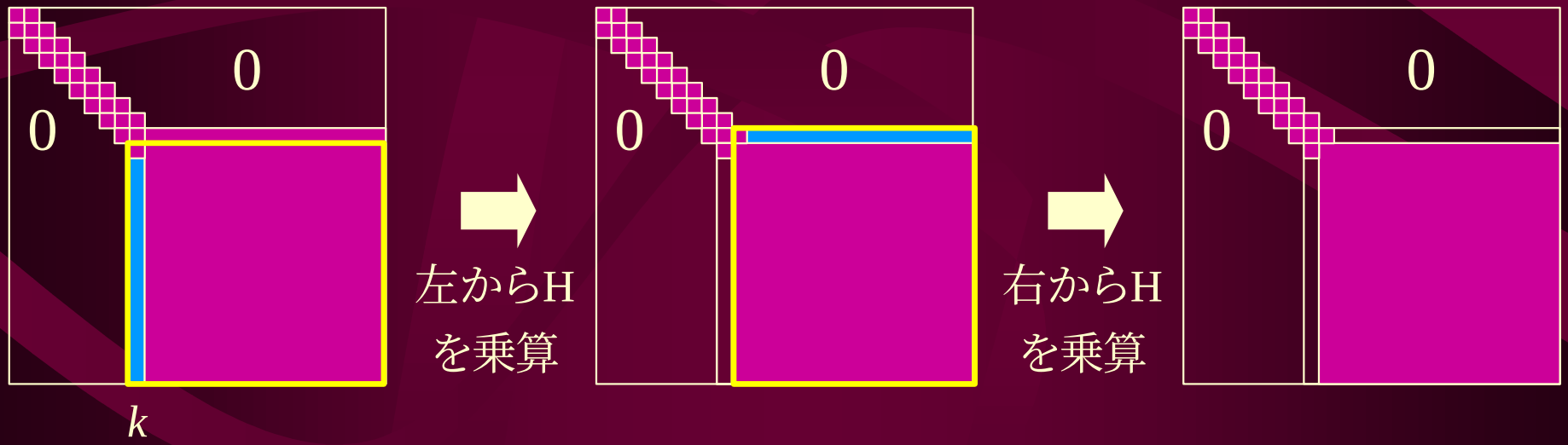
3. 三重対角化部分の並列化と高性能化

- ハウスホルダー法による三重対角化
- 並列化アルゴリズム
- ブロック化ハウスホルダー法 I
- ブロック化ハウスホルダー法 II

ハウスホルダー法による三重対角化

- 鏡像変換による列の消去
  - 鏡像変換 $H = I - \mathbf{w}\mathbf{w}^t$
    - Hは対称な直交行列
    - 与えられたベクトルの第１成分以外をゼロにする。

- 第kステップでの処理



ハウスホルダー法のアルゴリズム

[Step 1]　k = 1からN –2まで以下の[Step 2] ～ [Step 8]を繰り返す。

[Step 2]　内積 $\sigma^{(k)} = \mathrm{sqrt}(\boldsymbol{d}^{(k)t}\boldsymbol{d}^{(k)})$の計算

[Step 3]　鏡像変換ベクトル作成：

$$\boldsymbol{u}^{(k)} = (d^{(k)}_1 - \mathrm{sgn}(d^{(k)}_1)\sigma^{(k)}, d^{(k)}_2, \cdots, d^{(k)}_{N-k})$$

[Step 4]　規格化定数の計算：　$\alpha^{(k)} = 2 / \| \boldsymbol{u}^{(k)} \|^2$

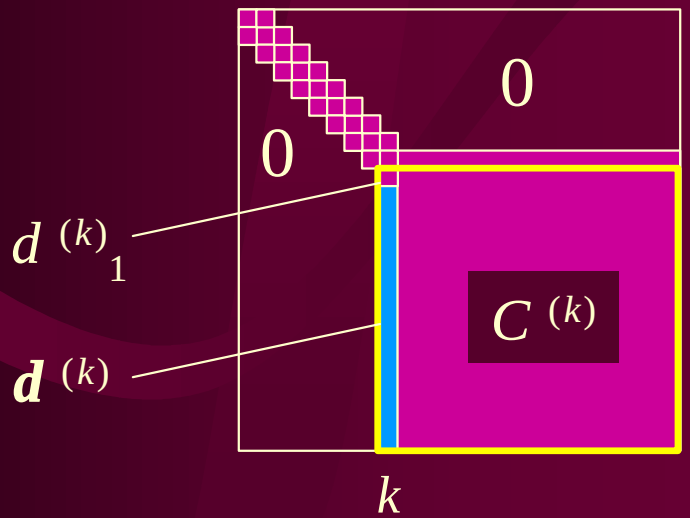
[Step 5]　行列ベクトル積：　$\boldsymbol{p}^{(k)} = \alpha^{(k)} C^{(k)} \boldsymbol{u}^{(k)}$

[Step 6]　ベクトルの内積：　$\beta^{(k)} = \alpha^{(k)} \boldsymbol{u}^{(k)t} \boldsymbol{p}^{(k)} / 2$

[Step 7]　ベクトルの加算：　$\boldsymbol{q}^{(k)} = \boldsymbol{p}^{(k)} - \beta^{(k)} \boldsymbol{u}^{(k)}$

[Step 8]　行列のrank-2更新：　$C^{(k)} = C^{(k)} - \boldsymbol{u}^{(k)}\boldsymbol{q}^{(k)t} - \boldsymbol{q}^{(k)}\boldsymbol{u}^{(k)t}$

ピボット列　　ピボット行



ハウスホルダー法の特徴

- 演算量
  - 行列サイズが N のときの演算量:　約 $(4/3)N^3$
    - 行列ベクトル積の演算量:　約 $(2/3)N^3$
    - rank-2更新の演算量:　約 $(2/3)N^3$

- データの再利用性
  - 演算はほとんどBLAS2のため，行列データの再利用性はなし
  - キャッシュマシンでは高い性能が期待できない。

⇩

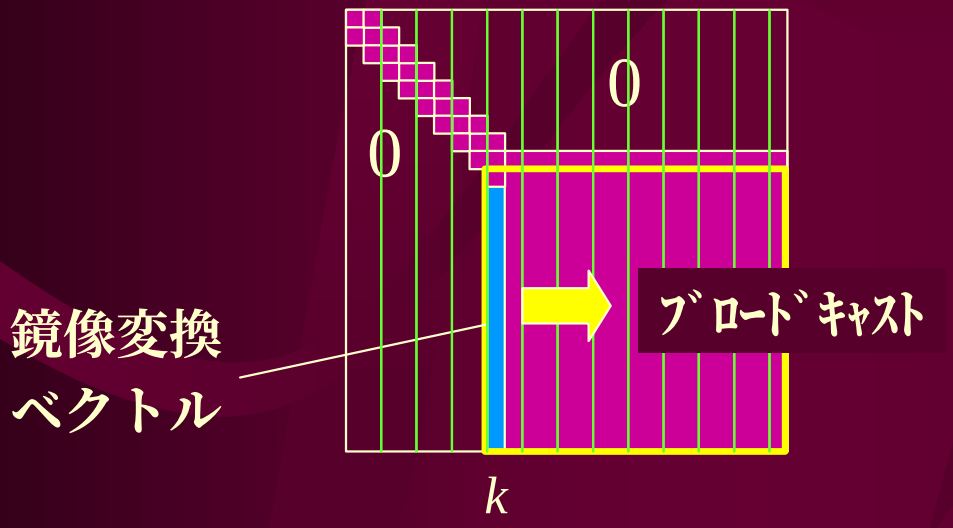
ブロック化など，アルゴリズム上の工夫が必要

- 並列粒度
  - BLAS2のため $O(N^2/p)$ であり，比較的高い。

ハウスホルダー法の並列化 (1)

- サイクリック列分割による並列化（Dongarra et al., 1992）
  - 鏡像変換ベクトルの作成部では通信不要
  - 作成したベクトル全体を全ﾌﾟﾛｾｯｻにﾌﾞﾛｰﾄﾞｷｬｽﾄする必要あり
  - 二分木を使った場合，各段での通信量は O(N log p)

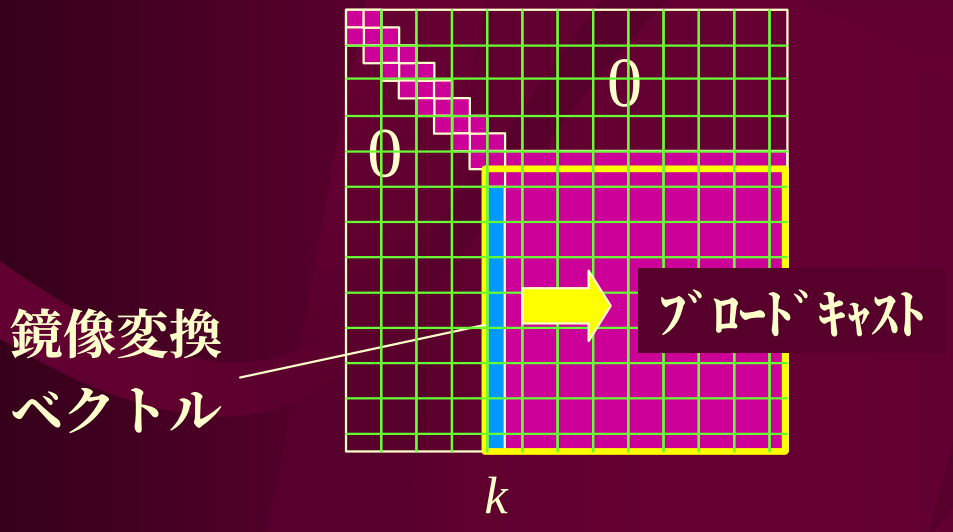
通信量はプロセッサ台数に従って増加



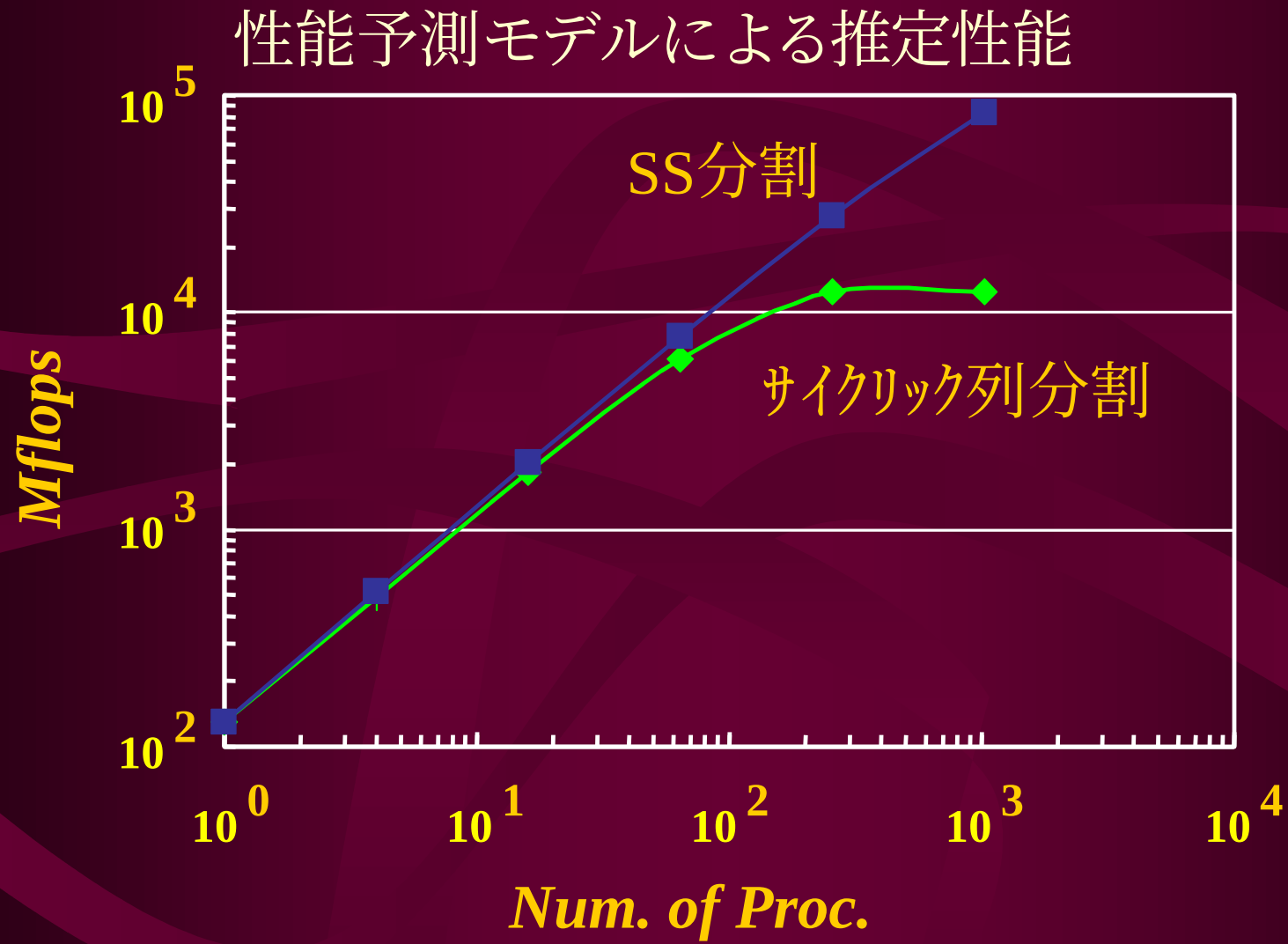
ハウスホルダー法の並列化 (2)

- Scattered Square 分割による並列化（Choi et al., 1995）
  - 鏡像変換ベクトルの作成部で通信が必要
  - 作成したベクトルを√p 個のﾌﾟﾛｾｯｻにﾌﾞﾛｰﾄﾞｷｬｽﾄする必要あり
    - ただし，1台のﾌﾟﾛｾｯｻは全体の 1/ √p のみを担当
  - 二分木を使った場合，各段での通信量は O(N log p / √p)

通信量はプロセッサ台数に従って減少



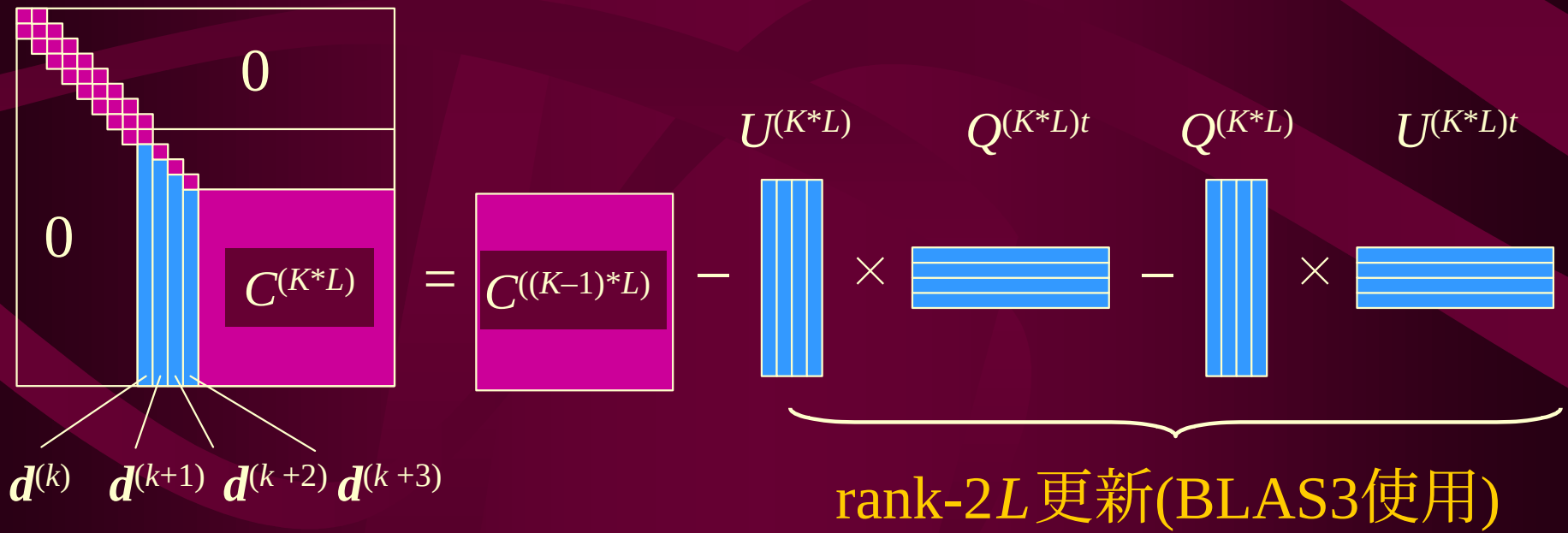
サイクリック列分割とSS分割の性能比較



超並列ではScattered Square分割が優位

ブロック化ハウスホルダー法 I

- 原理（Dongarra et al., 1992）
  - 複数のピボット列・行を溜めておき，後でまとめて更新
  - rank-2更新の部分を行列乗算（BLAS3）にでき，キャッシュマシンで有効



ブロック化ハウスホルダー法 I のアルゴリズム

[Step 1]　K = 1からN/Lまで以下の[Step 2,3,12] を繰り返す。

[Step 2]　$U^{((K-1)*L)} = \Phi$,　$Q^{((K-1)*L)} = \Phi$ (0行0列の行列)

[Step 3]　$k = (K-1)*L+1$から$K*L$まで以下の[Step 3-1〜8]を繰り返す。

[Step 3-1]　部分ハウスホルダ変換:

$$\boldsymbol{d}^{(k)} := \boldsymbol{d}^{(k)} - U^{(k-1)}(Q^{(k-1)t})_{k-(K-1)*L} - Q^{(k-1)}(U^{(k-1)t})_{k-(K-1)*L}$$

[Step 3-2]　内積 $\sigma^{(k)} = \mathrm{sqrt}(\boldsymbol{d}^{(k)t}\,\boldsymbol{d}^{(k)})$の計算

[Step 3-3]　鏡像変換ベクトル作成:

$$\boldsymbol{u}^{(k)} = (d^{(k)}_1 - \mathrm{sgn}(d^{(k)}_1)\,\sigma^{(k)},\ d^{(k)}_2,\ \ldots,\ d^{(k)}_{N-k})$$

[Step 3-4]　規格化定数の計算:　$\alpha^{(k)} = 2 / \| \boldsymbol{u}^{(k)} \|^2$

[Step 3-5]　行列ベクトル積: $\boldsymbol{p}^{(k)} = \alpha^{(k)} (C^{(k)} - U^{(k-1)}Q^{(k-1)t} - Q^{(k-1)}U^{(k-1)t})\,\boldsymbol{u}^{(k)}$

[Step 3-6]　ベクトルの内積:　$\beta^{(k)} = \alpha^{(k)}\,\boldsymbol{u}^{(k)t}\boldsymbol{p}^{(k)}/2$

[Step 3-7]　ベクトルの加算:　$\boldsymbol{q}^{(k)} = \boldsymbol{p}^{(k)} - \beta^{(k)}\boldsymbol{u}^{(k)}$

[Step 3-8]　$U^{(k)} = [U^{(k-1)} \mid \boldsymbol{u}^{(k)}]$,　$Q^{(k)} = [Q^{(k-1)} \mid \boldsymbol{q}^{(k)}]$

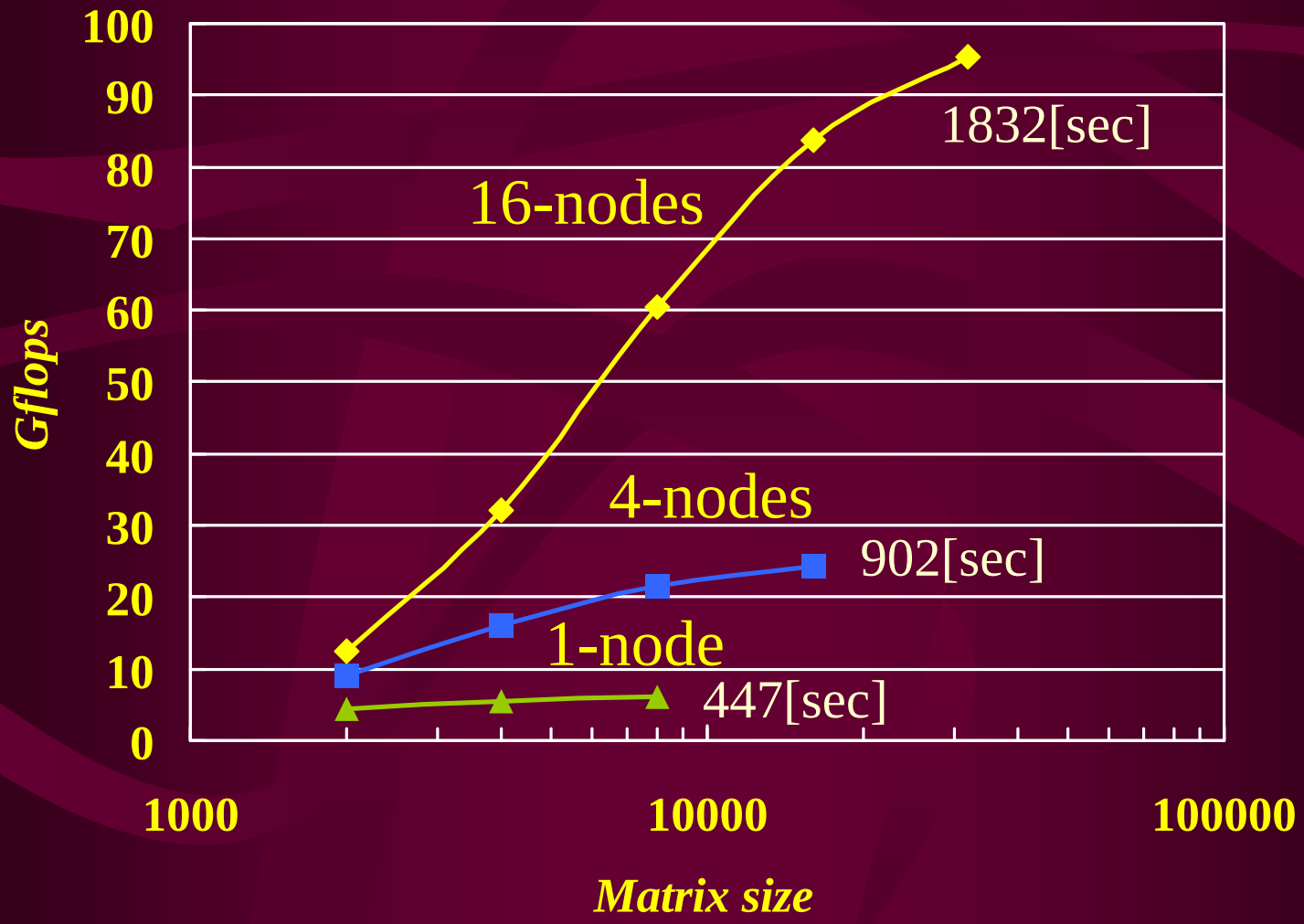
[Step 12]　行列のrank-$2L$更新:

$$C^{(K*L)} = C^{((K-1)*L)} - U^{(K*L)}Q^{(K*L)t} - Q^{(K*L)}U^{(K*L)t}$$

BLAS-3使用

ブロック化ハウスホルダー法Ｉの性能評価

- Scattered Square 分割による並列化（MATRIX/MPP　HZES1MDP使用）
- SR8000/F1の１～16ノードで評価
- N=2000 ～ 32000 のエルミート行列での性能



ブロック化ハウスホルダー法Ｉの問題点

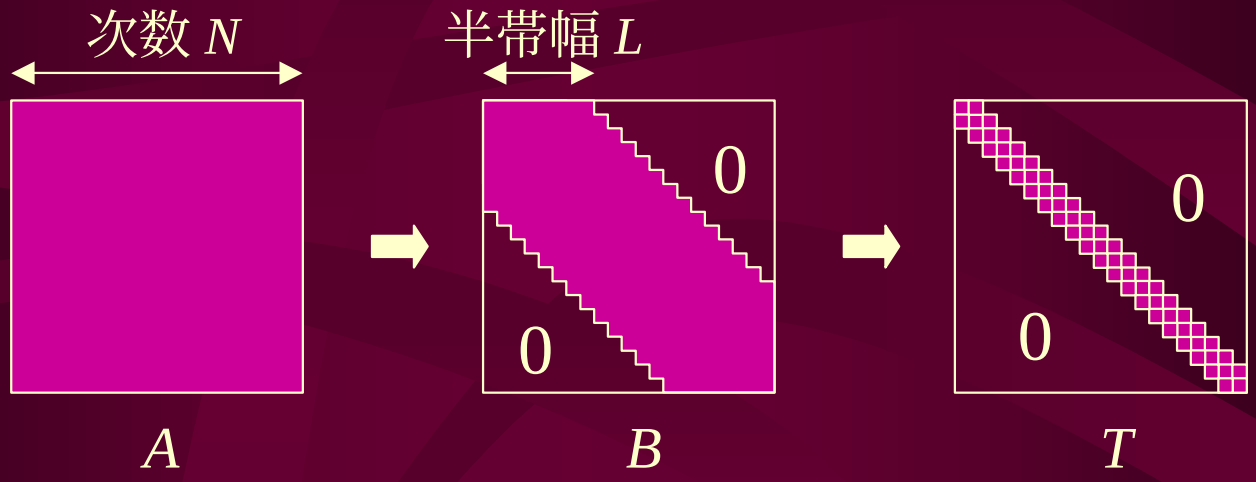
- データの再利用性
  - 演算量の半分を占める rank-1 更新は BLAS3 化
  - しかし，残り半分を占める行列ベクトル積は BLAS2 のまま

SR8000などの（擬似）ベクトル機では高い性能が出せるが，
キャッシュマシンでは性能を出すことが困難

キャッシュマシン向けには，行列ベクトル積の部分も BLAS3
に置き換えられるアルゴリズムが必要

ブロック化ハウスホルダー法 II

- 原理（Bishof et al., 1993, 1994）
  - 密行列Aをまず半帯幅Lの帯行列Bに変換し，次にこの帯行列を三重対角行列Tに変換する。

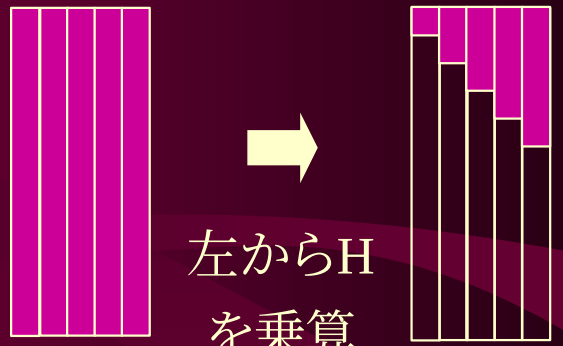


- 三重対角化を２段階で行うことの利点
  - 帯行列への変換は，BLAS3のみを使って実行可能。
  - 帯行列から三重対角行列への変換の演算量は$O(N^2L)$であり，前半部に比べて無視できる。

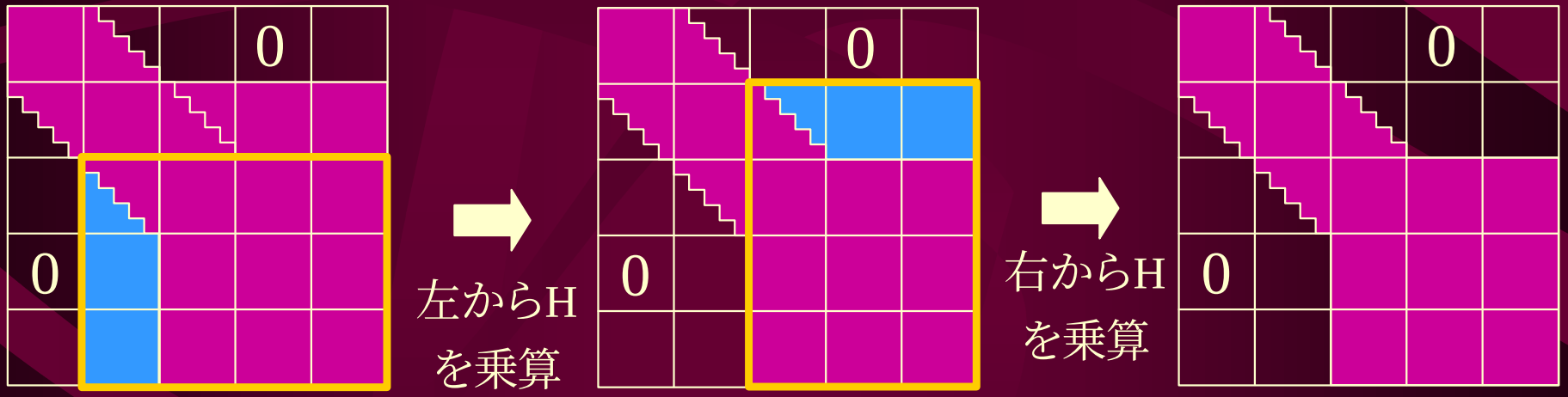
密行列から帯行列への変換

ブロック鏡像変換によるブロック列の消去
- ブロック鏡像変換 $H = I - U \alpha U^t$
  - Hは対称な直交行列
  - 与えられたブロックベクトルの第1ブロックを上三角行列にし，それ以外をゼロにする。

ブロックベクトル



第kステップでの処理



非ゼロ要素　ゼロにしたい部分　影響を受ける部分

ブロック化ハウスホルダー法 II のアルゴリズム

[Step 1]　K = 1からN/L–1まで以下の[Step 2] ～ [Step 6]を繰り返す。

[Step 2]　$D^{(K)}$ を第1ブロックが上三角行列で第2ブロック以下がゼロ行列であるブロックベクトルに変換するブロック鏡像変換 $I - U^{(K)}\alpha^{(K)}U^{(K)t}$を求める。

[Step 3]　行列・ブロックベクトル積：　$P^{(K)} = C^{(K)}U^{(K)}\alpha^{(K)}$

[Step 4]　ブロックベクトルの内積：　$\beta^{(K)} = \alpha^{(K)t}U^{(K)t}P^{(K)} / 2$

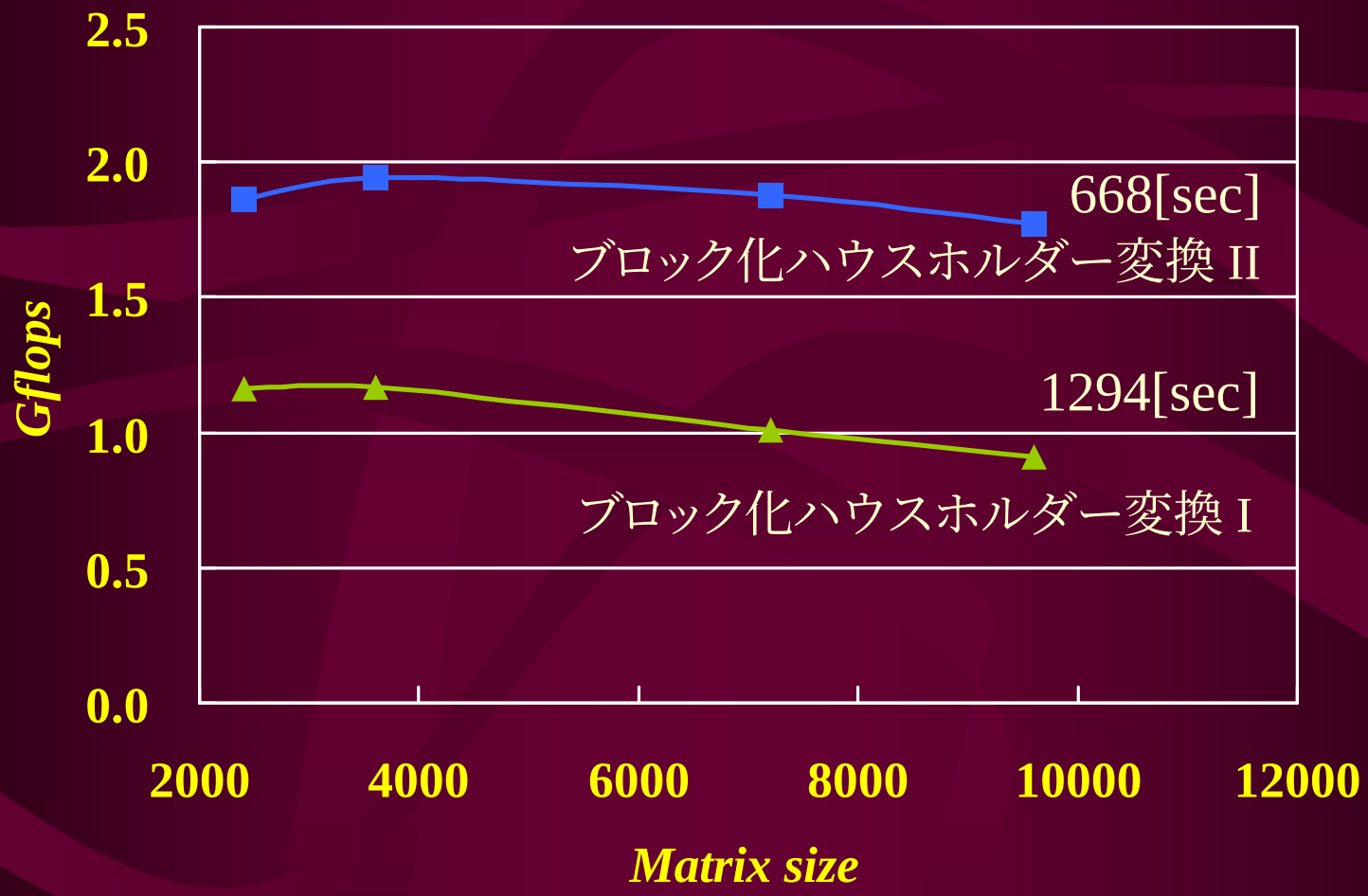
[Step 5]　ブロックベクトルの加算：　$Q^{(K)} = P^{(K)} - U^{(K)}\beta^{(K)}$

[Step 6]　行列のrank-2L更新：　$C^{(K)} = C^{(K)} - U^{(K)}Q^{(K)t} - Q^{(K)}U^{(K)t}$

すべてBLAS3（行列乗算）

ブロック化ハウスホルダー法 II の性能評価

- IBM Power 4 の 1CPU（1.3GHz，ピーク性能5.2GFLOPS）で評価
- N = 2400 ～ 9600 の実対称行列での性能



⇒ II は I に比べて約２倍の性能を達成

ブロック化ハウスホルダー法 II の限界

- 固有ベクトルを求める際の問題点
  1. 前半部（帯行列への変換）と後半部（帯行列から三重対角行列への変換）の両方の相似変換の行列を保存しておく必要があるため，記憶領域が２倍必要
  2. 逆変換のための演算量も，固有ベクトル１本あたりハウスホルダー法の約２倍

  - 求める固有ベクトル本数が多い場合は，BLAS3使用による性能向上効果よりも２のデメリットのほうが大きい。
  - 求める固有ベクトル本数が少ない（～N/10）場合に最適

5. おわりに

- まとめ
  - 密行列向けの並列固有値解法では，三重対角化部分と逆反復法による固有ベクトル計算部分の並列化が課題
  - 逆反復法の並列化手法として，Dhillonのアルゴリズム，マルチカラー逆反復法，ハウスホルダー逆反復法を紹介
  - 三重対角化部分の高性能化手法として，2種類のブロック化ハウスホルダー法を紹介
  - これらの利用により，並列計算機，キャッシュマシンで高い性能が達成できる。

おわりに

- 今後の展開
  - 自動チューニング型ライブラリ
  - 数値解の精度保証

参考文献 (1/4)

固有値計算全般

- J. W. Demmel: “Numerical Linear Algebra”, SIAM, Philadelphia, 1996.
- G. H. Golub and C. F. van Loan: “Matrix Computations”, The Johns Hopkins University Press, 1989.
- B. N. Parlett: “The Symmetric Eigenvalue Problem”, Prentice-Hall, 1980.
- J. H. Wilkinson and C. Reinsch (eds.): “Linear Algebra”, Springer-Verlag, 1971.
- J. H. Wilkinson: “The Algebraic Eigenvalue Problem”, Claredon Press, Oxford, 1965.

並列固有値計算全般

- T. Katagiri: “A Study on Large Scale Eigensolvers for Distributed Memory Parallel Machines”, Ph. D. Thesis, Information Science Division, University of Tokyo, Tokyo, Japan, 2001.
- K. S. Stanley: “Execution Time of Symmetric Eigensolvers”, Ph. D. Thesis, Computer Science Division, University of California at Berkeley, 1997.
- B. Hendrickson, E. Jessup and C. Smith: “Toward an Efficient Parallel Eigensolver for Dense Symmetric Matrices”, SIAM Journal on Scientific Computing, Vol. 20, No. 3, pp. 1132-1154 (1999).

参考文献(2/4)

DC法

- J. J. M. Cuppen: “A Divide and Conquer Method for the Symmetric Tri-diagonal Eigenproblem”, *Numerische Mathematik*, Vol. 36, pp. 177-195 (1981).
- M. Gu and S. C. Eisenstat: “A Stable and Efficient Algorithm for the Rank-1 Modification of the Symmetric Eigenproblem”, SIAM Journal on Matrix Analysis and Applications, Vol. 15, pp. 1266-1276 (1994).
- M. Gu and S. C. Eisenstat: “A Divide-and Conquer Algorithm for the Symmetric Tridiagonal Eigenproblem”, SIAM Journal on Matrix Analysis and Applications, Vol. 16, pp. 172-191 (1995).
- F. Tisseur and J. Dongarra: “A Parallel Divide and Conquer Algorithm fot the Symmetric Eigenvalue Problem on Distributed Memory Architectures”, SIAM Journal on Scientific Computing, Vol. 20, No. 6, pp. 2223-2236 (1999).

従来法による逆反復法の並列化

- J. Choi et. al.: “ScaLAPACK: A Portable Linear Algebra Library for Distributed Memory Computers – Design Issues and Performance”, LAPACK Working Notes 95 (1995).

参考文献(3/4)

Dhillonのアルゴリズム

- I. Dhillon: “A New O(n2) Algorithm for the Symmetric Tri-diagonal Eigenvalue /Eigenvector Problem”, Ph. D. Thesis, Computer Science Division, University of California at Berkeley, 1997.

マルチカラー逆反復法

- 直野健, 猪貝光祥, 山本有作: “並列固有値ソルバーの開発と性能評価”, 並列処理シンポジウムJSPP ’96論文集, pp. 9-16 (1996).
- K. Naono, Y. Yamamoto, M. Igai, H. Hirayama and N. Ioki: “A Multi-color Inverse Iteration for a High Performance Real Symmetric Eigensolver”, presented at *Euro-Par 2000*, Munich, August 2000 (in *Lecture Notes in Computer Science*, No. 1900, pp. 527-531, Springer-Verlag, 2000).

ハウスホルダー逆反復法

- 山本有作, 猪貝光祥, 直野健: “共有メモリ型並列計算機向けの高並列固有ベクトル解法とSR8000での評価”, 情報処理学会論文誌, Vol. 42, No. 4, pp. 771-778 (2001).

三重対角化部分の並列化

- H. Chang, S. Utku, M. Sakama and D. Rapp: “A Parallel Householder Tridiagonalization Stratagem using Scattered Square Decomposition”, Parallel Computing, Vol. 6, pp. 297-311 (1988).

参考文献(4/4)

ブロック化ハウスホルダー法 I

- J. J. Dongarra and R. A. van de Geijn: “Reduction to Condensed Form for the Eigenvalue Problem on Distributed Architectures”, Parallel Computing, Vol. 18, No. 9, pp. 973-982 (1992).
- K. Naono, M. Igai and Y. Yamamoto: “High Performance Implementation of Tridiagonalization on the SR8000”, presented at *HPC-Asia 2000*, Beijing, May 2000 (in *Proceedings of the Fourth International Conference on High-Performance Computing in the Asia-Pacific Region*, Vol. 1, pp. 206-219, IEEE Computer Society, 2000).

ブロック化ハウスホルダー法 II

- C. Bishof, M. Marques and X. Sun: “Parallel Bandreduction and Tridiagonalization”, Technical Report 8, PRISM Working Note, 1993.
  http://www-unix.mcs.anl.gov/prism/lib/tech.html
- C. Bishof, B. Lang and X. Sun：“Parallel Tridiagonalization through Two-step Band Reduction”, Technical Report 17, PRISM Working Note, 1994.
  http://www-unix.mcs.anl.gov/prism/lib/tech.html.
- 村田健郎, 小国力, 唐木幸比古: “スーパーコンピュータ：科学技術計算への適用”, 丸善, 1985.