数理工学実験レポート

第 1 章（数値線形代数）

京都大学 工学部情報学科 数理工学コース
学年：2 回生
学籍番号：1029366161　中塚 一瑳

科目名：	数理工学実験
実験テーマ：	数値線形代数
実験の実施年月日：	2025 年 10 月 21 日
レポート提出年月日：	2025 年 10 月 21 日

要旨（Abstract）

1　はじめに

本章では，数値計算の基礎となる数値線形代数の手法を取り上げる．連立一次方程式や固有値問題の数値的解法を通じて，計算精度・安定性・効率性の観点から各アルゴリズムの特徴を理解することを目的とする．また，理論的背景と実装結果を比較し，数値誤差や収束挙動の要因を考察することで，線形代数の理論と計算機実装の関係を体系的に把握することを目指す．

2　課題 1：ガウスの消去法による連立一次方程式の解法

2.1　原理・方法

ガウスの消去法は，連立一次方程式

$$A\boldsymbol{x} = \boldsymbol{b}$$

を上三角化によって逐次的に解く手法である．行基本変形により係数行列 A を上三角行列 U に変換し，対応して右辺 $\boldsymbol{b}$ にも同じ操作を施す．これにより

$$U\boldsymbol{x} = \boldsymbol{c}$$

の形が得られ，最後に後退代入（back substitution）を行うことで未知ベクトル $\boldsymbol{x}$ を求める．

各ステップでは，ピボット要素 A_{kk} を基準に第 k 列の下部要素を消去する．ただし A_{kk} が小さいと丸め誤差の影響で数値不安定となるため，絶対値が最大の要素を選んで行を入れ替える「主成分選択（partial pivoting)」を導入する．これにより除算時の誤差伝播が抑制され，計算の安定性が向上する．

Julia による実装例を以下に示す．

```
function gaussian_elimination(A)
    A = float.(copy(A))             # 浮動小数点型への変換
    n = size(A, 1)
    for k = 1:n-1
        # --- 主成分選択 ---
        pivot = k - 1 + argmax(abs.(A[k:end, k]))
        if pivot != k
            A[k, :], A[pivot, :] = A[pivot, :], A[k, :]
        end

        # --- 消去操作 ---
        for i = k+1:n
            if A[i,k] != 0
                factor = A[i,k] / A[k,k]
```

```
                A[i, k:end] .-= factor .* A[k, k:end]
            end
        end
    end
    A
end
```

得られた上三角行列 U に対しては後退代入法を適用，上の行に依存する各変数を逆順に計算していくことで x_i を逆順に求める．後退代入の実装を以下に示す．

```
function back_substitution(A) # [U b]
    n = size(A, 1)
    x = zeros(eltype(A), n)
    for i = n:-1:1
        x[i] = A[i, n+1]
        for j = i+1:n
            x[i] -= A[i, j] * x[j]
        end
        x[i] /= A[i, i]
    end
    x
end
```

2.2 実験方法

本課題では，前節で実装したガウス消去法と後退代入法を組み合わせ，連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の数値解を求める実験を行った．

このアルゴリズムを様々な行列サイズ N に対して繰り返し実行し，計算精度と計算時間を評価する関数 exp1 を実装した．

```
function exp1(N; num_trials=100)
    residuals = zeros(num_trials)
    relerrs   = zeros(num_trials)
    times     = zeros(num_trials)
    for t in 1:num_trials
        A = rand(N,N); b = rand(N)
        C = hcat(A,b)
        times[t] = @elapsed begin
            U = gaussian_elimination(C)
            x̂ = back_substitution(U)
            xref = A \ b
```

```
            residuals[t] = norm(A*x̂ - b)
            relerrs[t]   = norm(x̂ - xref) / norm(xref)
        end
    end
    (; residuals, relerrs, times,
       res_median=median(residuals), rel_median=median(relerrs), time_median=median(times))
end
```

この実験関数では，行列 A とベクトル $\boldsymbol{b}$ を乱数で生成し，100 回の試行を行った．各試行で以下を計測した：

- 残差 $\|A\hat{\boldsymbol{x}} - \boldsymbol{b}\|$：数値解の再現精度
- 相対誤差 $\|\hat{\boldsymbol{x}} - \boldsymbol{x}_{\mathrm{ref}}\|/\|\boldsymbol{x}_{\mathrm{ref}}\|$：Julia 組込み演算子 $A\backslash b$ による真値との誤差比較
- 計算時間（`@elapsed` により測定）

試行ごとの結果から，中央値（median）を代表値としてまとめることで，外れ値の影響を抑えて全体的な傾向を評価した．これにより，ガウス消去法の数値安定性および計算コストを統計的に検証できる構成とした．

2.3 結果 (小問 1)

本節では，ガウスの消去法による連立一次方程式の数値解法について，行列サイズ $N = 100, 200, 400, 800$ の場合に対する実験結果を示す．各サイズにおいて 100 回の試行を行い，残差ノルム・相対誤差・計算時間の中央値を求めた．

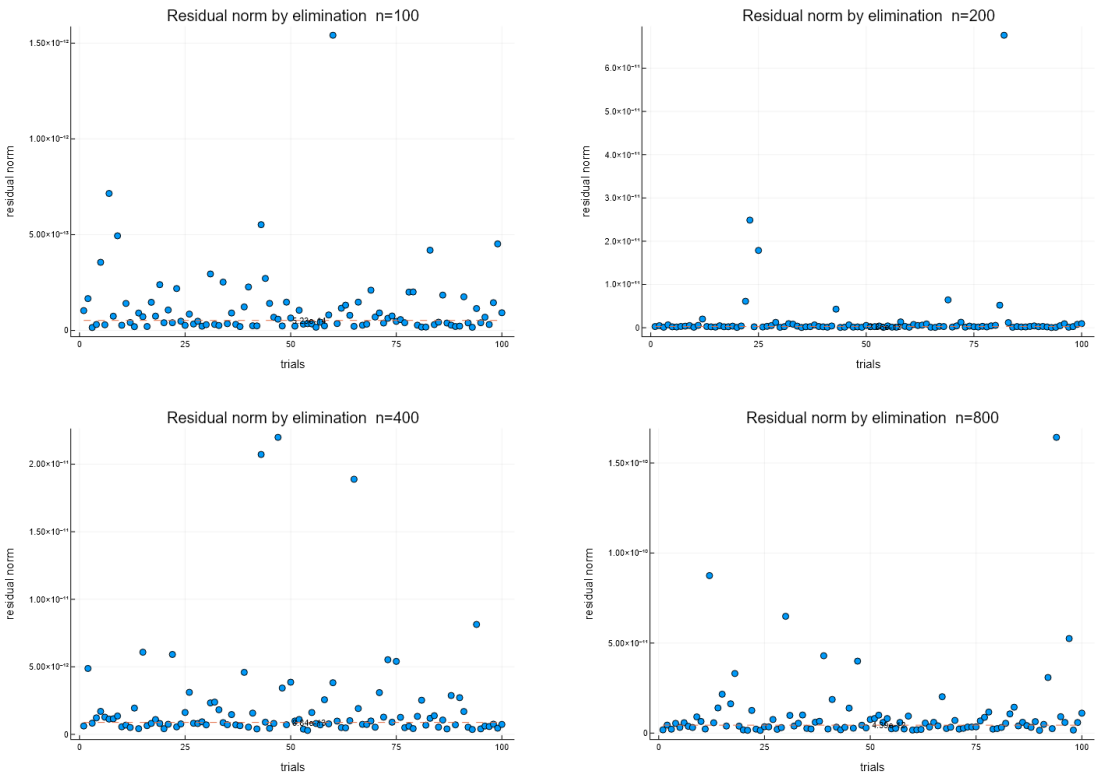

図 1　ガウス消去法における残差ノルム分布（各 N）

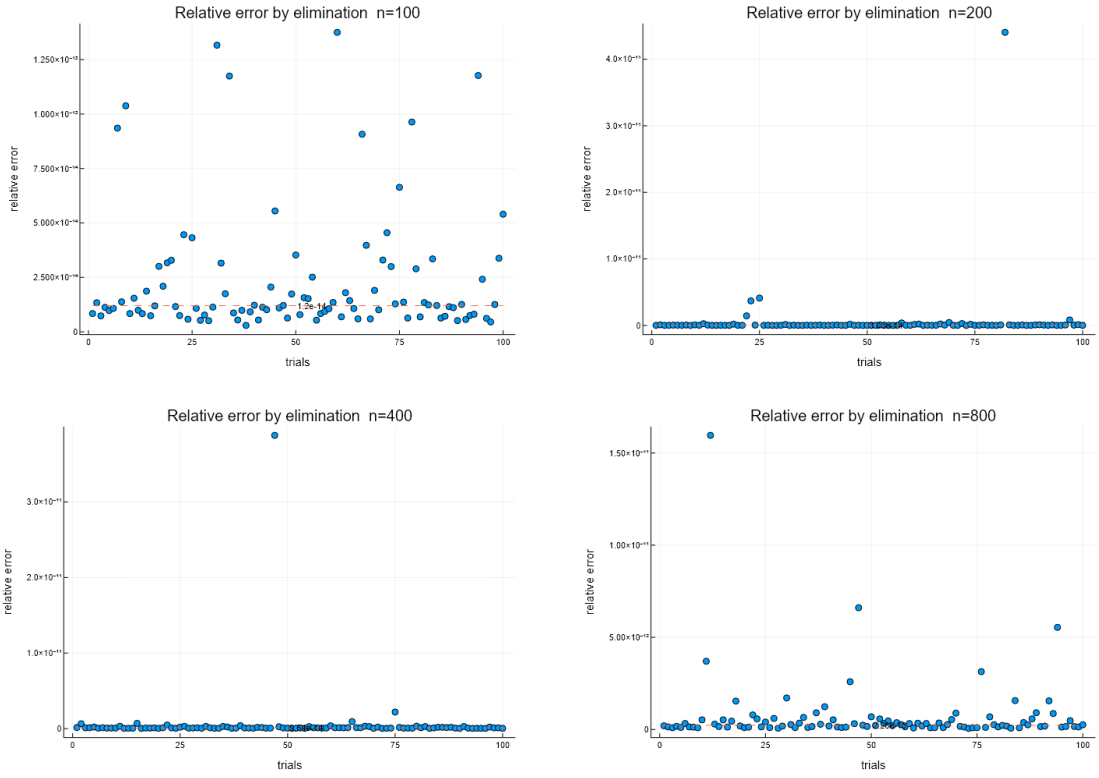

図 2　ガウス消去法における相対誤差分布（各 N）

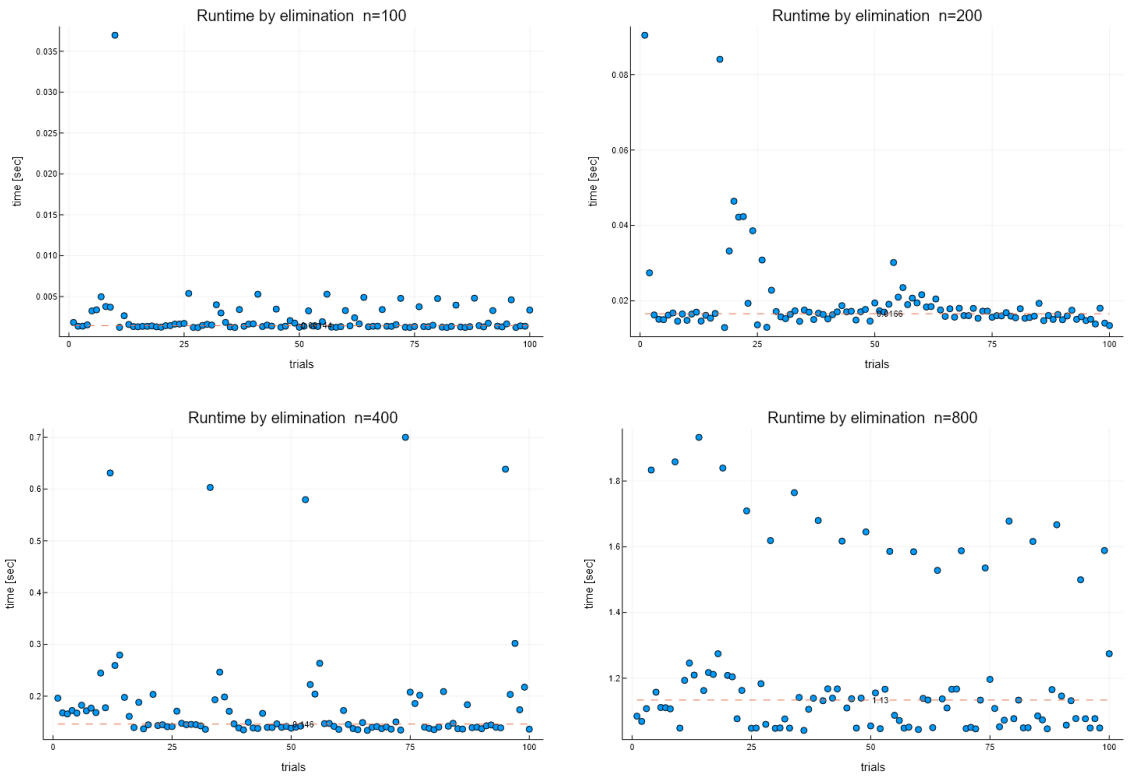

図 3　ガウス消去法における計算時間分布（各 N）

表 1　ガウス消去法の結果まとめ（100 試行の中央値）

N	残差ノルム	相対誤差	計算時間 [s]
100	5.62×10^{-14}	1.59×10^{-14}	0.00226
200	2.31×10^{-13}	3.51×10^{-14}	0.0161
400	1.06×10^{-12}	1.12×10^{-13}	0.179
800	3.83×10^{-12}	2.13×10^{-13}	1.20

2.4　考察 (小問 2,3)

図 1〜図 3 より，各試行において一部の外れ値が存在し，特に残差ノルムや相対誤差においてまれに大きな値が観測された．これは，乱数生成された行列の条件数が悪化した試行で丸め誤差が増幅されるためと考えられる．

表 1 の中央値から，各物理量のオーダーを見積もる．$y(N) \approx C N^{p}$ と仮定すると，連続するサイズ比から

$$\hat{p} \;=\; \frac{\log\bigl(y(2N)/y(N)\bigr)}{\log 2}$$

で指数を推定できる．実測値を用いた比と推定値は次の通り：

	$y(200)/y(100)$	$y(400)/y(200)$	$y(800)/y(400)$
残差ノルム $\|A\hat{x}-b\|$	4.11	4.59	3.61
$\hat{p}$	2.04	2.20	1.85
相対誤差 $\|\hat{x}-x_{\mathrm{ref}}\|/\|x_{\mathrm{ref}}\|$	2.21	3.19	1.90
$\hat{p}$	1.15	1.67	0.93
計算時間 t [s]	7.12	11.12	6.70
$\hat{p}$	2.83	3.47	2.74

これより，平均的には

$$\boxed{\|A\hat{x}-b\| = O(N^{\approx 2})}, \qquad \boxed{\|\hat{x}-x_{\mathrm{ref}}\|/\|x_{\mathrm{ref}}\| = O(N^{\approx 1.3-1.7})}, \qquad \boxed{t = O(N^{\approx 3})}$$

と推定され，とくに計算時間は理論どおり $O(N^3)$ に整合する．

理論的な見込みについても補足する．ガウス消去＋後退代入では，丸め誤差の累積が各列あたり概ね $O(N)$，全体で $O(N^2)$ に達するモデルが妥当で，残差ノルムは $O(N^2)$ 規模が期待される．一方，相対誤差は分母 $\|x_{\mathrm{ref}}\|$ のスケーリングに依存する．乱数ベクトルで $\|x_{\mathrm{ref}}\| \sim O(\sqrt{N})$ とみなすと，$O(N^2)/O(\sqrt{N}) = O(N^{3/2})$ が理論上の基準となるが，実測の $\hat{p}$ は 1〜1.7 の範囲であり分散が大きいものの理論値の 3/2 を裏付けていると考えられる．

3　課題 2：LU 分解

3.1　原理・方法

LU 分解は，正則行列 A を下三角行列 L と上三角行列 U の積として表す手法である：

$$A = LU.$$

これにより，連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ は

$$LU\boldsymbol{x} = \boldsymbol{b}$$

と書け，まず $L\boldsymbol{y} = \boldsymbol{b}$ を順方向代入（forward substitution）で解き，次に $U\boldsymbol{x} = \boldsymbol{y}$ を後退代入（backward substitution）で解くことで $\boldsymbol{x}$ を求める．

LU 分解のアルゴリズムでは，各列のピボット要素を基準にその下の要素を消去し，除算係数を L に格納する．これを列ごとに繰り返すことで A は LU に分解される．数値誤差を抑制するため，主成分選択（partial pivoting）を導入し，行入れ替え行列 P を用いた

$$PA = LU$$

の形に変形する．ピボット選択により，分母が極端に小さくなる除算を回避し，安定な計算が実現される．

順方向代入では，既に求めた未知数を使って次の項を計算していく．つまり L が下三角行列であることを利用し，式

$$y_i = \frac{1}{L_{ii}}\left(b_i - \sum_{j=1}^{i-1} L_{ij}y_j\right)$$

を $i = 1$ から順に求める.

後退代入も同様に，U の上三角構造を用いて

$$x_i = \frac{1}{U_{ii}} \left(y_i - \sum_{j=i+1}^{n} U_{ij} x_j \right)$$

を $i = n$ から逆順に計算する．これらにより，同じ係数行列 A に対して複数回の右辺 $\boldsymbol{b}$ に対応する解 $\boldsymbol{x}$ を効率的に求めることが可能となる

本課題では，資料に基づく LU 分解を実装し，得られた L, U, P を用いて数値解を求めた.

主成分選択付きの LU 分解アルゴリズムを以下に示す：

```
function lu_pp(Ain)
    U = float.(copy(Ain))
    n = size(U,1)
    L = Matrix{eltype(U)}(I, n, n)
    p = collect(1:n)  # permutation vector representing P

    for k = 1:n-1
        piv = k - 1 + argmax(abs.(U[k:end, k]))
        if abs(U[piv, k]) == 0
            error("Matrix is singular to working precision.")
        end
        if piv != k
            U[k, :], U[piv, :] = U[piv, :], U[k, :]
            if k > 1
                L[k, 1:k-1], L[piv, 1:k-1] = L[piv, 1:k-1], L[k, 1:k-1]
            end
            p[k], p[piv] = p[piv], p[k]
        end
        for i = k+1:n
            L[i, k] = U[i, k] / U[k, k]
            U[i, k:end] .-= L[i, k] .* U[k, k:end]
        end
    end
    return L, U, p  # so that P*b == b[p]
end
```

順方向代入および後退代入の実装を以下に示す：

```
function forward_sub(L, b)
    n = size(L,1)
    y = similar(b, n)
    for i = 1:n
```

```
        s = zero(eltype(L))
        for j = 1:i-1
            s += L[i,j]*y[j]
        end
        y[i] = (b[i] - s) / L[i,i]
    end
    y
end

function backward_sub(U, y)
    n = size(U,1)
    x = similar(y, n)
    for i = n:-1:1
        s = zero(eltype(U))
        for j = i+1:n
            s += U[i,j]*x[j]
        end
        x[i] = (y[i] - s) / U[i,i]
    end
    x
end
```

これにより，$PA = LU$ が得られた後，順方向代入で $L\boldsymbol{y} = P\boldsymbol{b}$ を解き，後退代入で $U\boldsymbol{x} = \boldsymbol{y}$ を解く流れとなる．

3.2 実験方法

本課題でも，課題 1 と同様の評価手順を用いた．行列サイズ N ごとに乱数行列 $A \in \mathbb{R}^{N\times N}$ と $\boldsymbol{b} \in \mathbb{R}^N$ を生成し，100 回の試行を実施する．各試行で，主成分選択付き LU 分解 $PA = LU$ を計算し，順方向代入で $L\boldsymbol{y} = P\boldsymbol{b}$，続いて後退代入で $U\boldsymbol{x} = \boldsymbol{y}$ を解いて $\hat{\boldsymbol{x}}$ を得る．

評価指標は，(i) 残差ノルム $\|A\hat{\boldsymbol{x}} - \boldsymbol{b}\|$，(ii) 相対誤差 $\|\hat{\boldsymbol{x}} - \boldsymbol{x}_{\mathrm{ref}}\| / \|\boldsymbol{x}_{\mathrm{ref}}\|$（参照解は Julia の `A\b b`），(iii) 計算時間（`@elapsed`）である．各指標について中央値を代表値として記録する．

実装は以下の関数で行った：

```
function exp2(N; num_trials=100)
    residuals = zeros(num_trials)
    relerrs   = zeros(num_trials)
    times     = zeros(num_trials)

    for t in 1:num_trials
        A = rand(N,N); b = rand(N)
```

```
        xref = A \ b  # reference by Julia solver

        times[t] = @elapsed begin
            L, U, p = lu_pp(A)
            bp = b[p]
            y  = forward_sub(L, bp)
            xhat = backward_sub(U, y)
            residuals[t] = norm(A*xhat - b)
            relerrs[t]   = norm(xhat - xref) / norm(xref)
        end
    end

    (; residuals, relerrs, times,
       res_median = median(residuals),
       rel_median = median(relerrs),
       time_median = median(times))
end
```

3.3 結果 (小問 1)

LU 分解による連立一次方程式の数値解法の結果を図 4 および 5、6 に示す．さらに，試行ごとの分布の概要を 表 2 に示す．各サイズ $N = 100, 200, 400, 800$ に対して 100 回の試行を行い，残差ノルム・相対誤差・計算時間の平均値をまとめた．

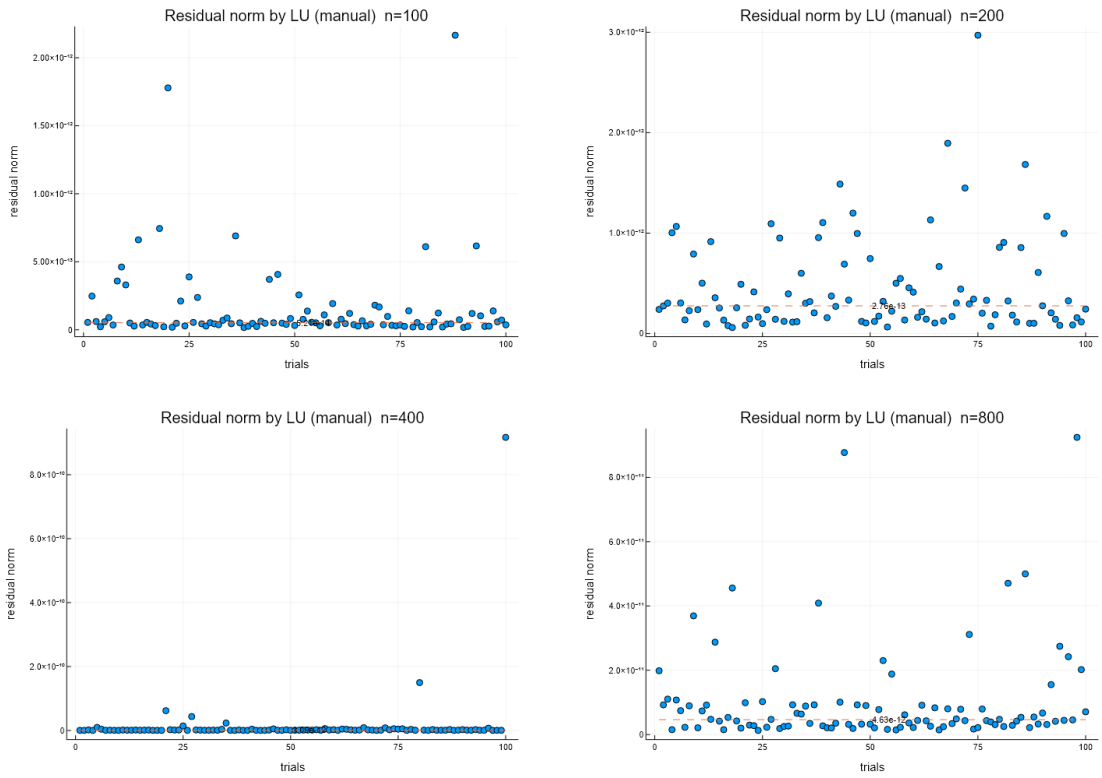

図 4　LU 分解における残差ノルム分布（各 N）

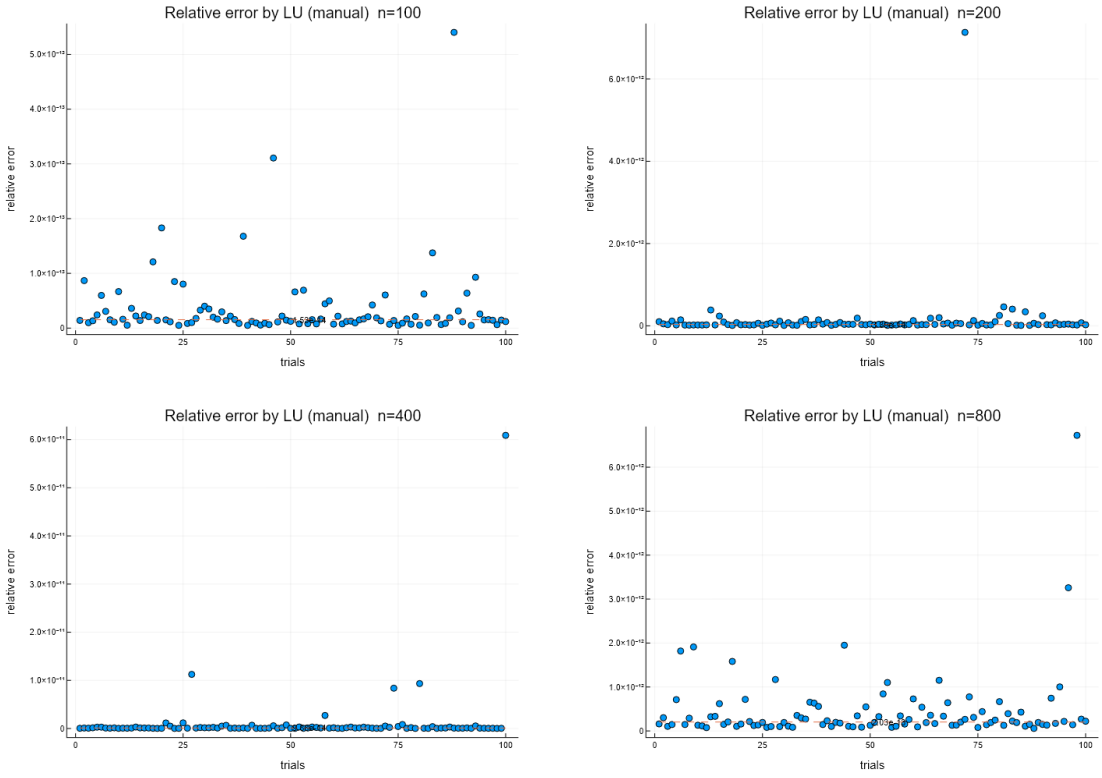

図 5　LU 分解における相対誤差分布（各 N）

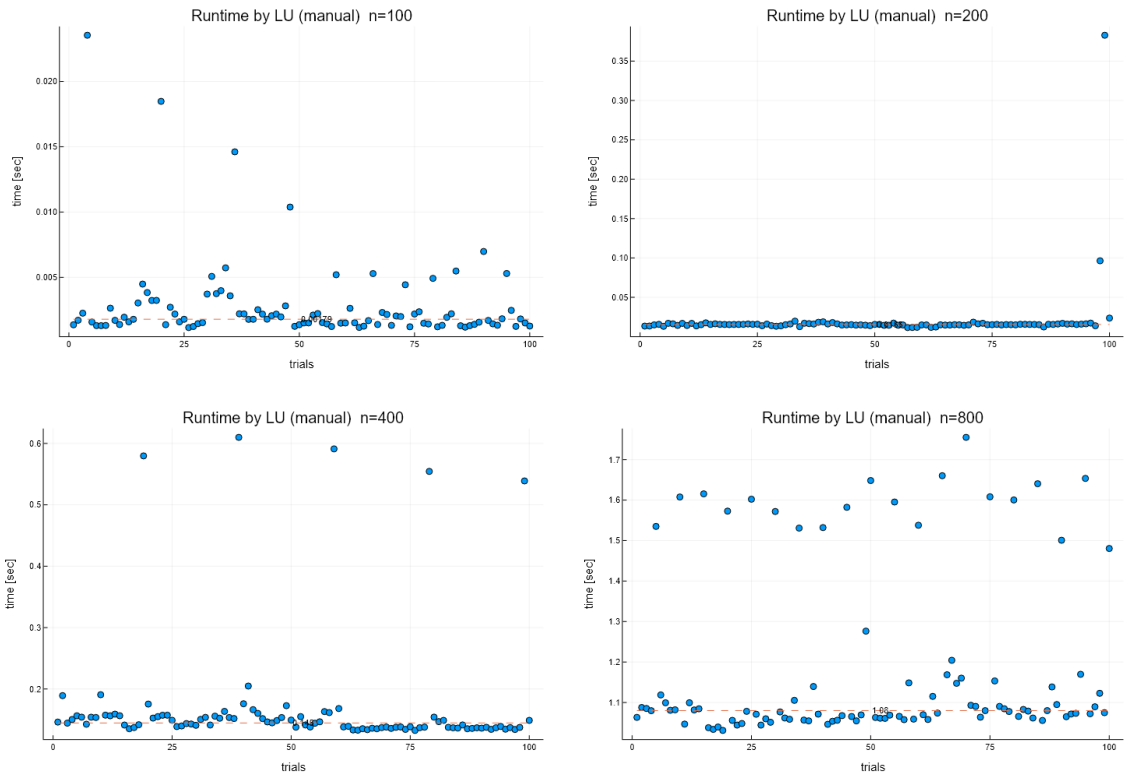

図 6　LU 分解における計算時間分布（各 N）

表 2　LU 分解による実験結果（100 試行の平均）

N	残差ノルム	相対誤差	計算時間 [s]
100	6.888982×10^{-14}	1.706824×10^{-14}	0.001811
200	2.784646×10^{-13}	3.878147×10^{-14}	0.014543
400	9.748238×10^{-13}	8.028622×10^{-14}	0.133908
800	5.362099×10^{-12}	2.353337×10^{-13}	1.077296

3.4　考察 (小問 2,3)

課題 1 と同様に、図から外れ値が一部存在することがわかる。表 2 の中央値から、各物理量のオーダーを見積もると、残差ノルムが $O(N^{\approx 2.1})$、相対誤差が $O(N^{\approx 1.3})$、計算時間が $O(N^{\approx 3})$ となり、ガウス消去法とほぼ同等の性能を示している。

4　課題 3：ガウスの消去法と LU 分解による解法の比較

4.1　結果

課題 1（ガウス消去法）と課題 2（LU 分解）の結果を表 3 にまとめる。

表 3　ガウス消去法と LU 分解の比較（中央値）

N	残差ノルム (GE)	残差ノルム (LU)	相対誤差 (GE)	相対誤差 (LU)	時間 [sec](GE)	時間 [sec](LU)
100	5.62×10^{-14}	6.72×10^{-14}	1.59×10^{-14}	1.77×10^{-14}	0.00226	0.00165
200	2.31×10^{-13}	2.53×10^{-13}	3.51×10^{-14}	3.74×10^{-14}	0.0161	0.0158
400	1.06×10^{-12}	9.15×10^{-13}	1.12×10^{-13}	8.13×10^{-14}	0.179	0.137
800	3.83×10^{-12}	5.08×10^{-12}	2.13×10^{-13}	2.58×10^{-13}	1.20	1.08

4.2　考察

表 3 より，ガウス消去法と LU 分解の数値精度および計算時間には大きな差は見られない．残差ノルム・相対誤差はいずれも 10^{-13} オーダーであり，LU 分解では行入れ替えによる安定性が維持されていることが確認できる．一方で，計算時間は両者とも $O(N^3)$ に比例して増加しており，実測でも N 倍ごとにおよそ 8 倍の増加傾向を示している．

理論的な計算量については，ガウス消去法が加減算および乗除算をそれぞれ約 $\frac{2}{3}N^3$ 回要するのに対し，LU 分解では同程度の $O(N^3)$ の演算を分解段階で行い、順方向代入、後退代入に要する $O(N^2)$ の演算の時間はほとんど無視できる．ただし LU 分解は一度の分解後に複数の右辺 $\boldsymbol{b}$ に対して再利用でき，その際の追加計算は前進・後退代入に要する $O(N^2)$ のみで済む．今回の場合は単一の $\boldsymbol{b}$ に対する解法であるため，両者の計算時間に大きな差は見られなかったと考えられる．

5　課題 4：べき乗法

5.1　原理・方法

べき乗法（Power Method）は，正方行列 $A \in \mathbb{R}^{n\times n}$ の最大固有値と対応固有ベクトルを近似する反復法である。初期ベクトル $\boldsymbol{b}^{(0)}$ を任意に与え，以下を反復する：

$$\boldsymbol{b}^{(k+1)} = \frac{A\boldsymbol{b}^{(k)}}{\|A\boldsymbol{b}^{(k)}\|}.$$

十分な反復で $\boldsymbol{b}^{(k)}$ は A の最大固有値 $\lambda_{\max}$ に対応する固有ベクトル方向に収束する。A が対角化可能で $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$，かつ初期成分 $c_1 \neq 0$ のとき，

$$A^k\boldsymbol{b}^{(0)} = \sum_{i=1}^{n} c_i\,\lambda_i^k\,\boldsymbol{v}_i$$

より $|\lambda_1|$ の項が支配的になるためである。

収束後の固有値はレイリー商で推定する：

$$\lambda = \frac{\boldsymbol{b}^\top A\boldsymbol{b}}{\boldsymbol{b}^\top\boldsymbol{b}}.$$

以下が今回のべき乗法の Julia 実装である：

```
function powEigen(A, max_iter = 10000, tol = 1e-12)
    n = size(A, 1)
```

```
    b_k = rand(n)
    b_k /= norm(b_k)
    itr = 0
    for iter in 1:max_iter
        itr += 1
        b_k1 = A * b_k
        b_k1 /= norm(b_k1)
        if norm(b_k1 - b_k) < tol * norm(b_k)
            break
        end
        b_k = b_k1
    end
    eigenvalue = dot(b_k, A * b_k) / dot(b_k, b_k)
    return eigenvalue, b_k, itr
end
```

5.2 実験方法

本課題では，べき乗法を用いて行列 A の第 1 固有値および第 1 固有ベクトルを数値的に求める実験を行った．行列サイズ $N = 50, 100, 200, 400$ について，それぞれランダムに生成した対称行列 $A \in \mathbb{R}^{N\times N}$ に対して 100 回の試行を実施した．行列は $A = (A + A^T)/2$ として対称化した．

各試行では，べき乗法により推定した固有値 $\tilde{\lambda}_1$ および固有ベクトル $\tilde{\boldsymbol{v}}_1$ を求め，Julia の `eigen(A)` により算出した厳密な最大固有値・固有ベクトル $(\lambda_1, \boldsymbol{v}_1)$ と比較した．異なる向きの固有ベクトルを同一視するため，厳密な固有ベクトルとの内積が負の場合は $\tilde{\boldsymbol{v}}_1$ を反転させた．

評価指標は以下の通りである：

- 固有方程式の残差ノルム：$\|A\tilde{\boldsymbol{v}}_1 - \tilde{\lambda}_1\tilde{\boldsymbol{v}}_1\|$
- 固有値の相対誤差：$|\tilde{\lambda}_1 - \lambda_1|/|\lambda_1|$
- 固有ベクトルの相対誤差：$\|\tilde{\boldsymbol{v}}_1 - \boldsymbol{v}_1\|/\|\boldsymbol{v}_1\|$
- 計算時間（`@elapsed`）
- 収束までの反復回数

実験は次の関数により実施した：

```
function exp3(N; num_trials=100)
    eigenvalues = zeros(num_trials)
    times       = zeros(num_trials)
    resnorms    = zeros(num_trials)
    relerrs_lambda = zeros(num_trials)
    relerrs_v      = zeros(num_trials)
    itrs           = zeros(Int, num_trials)
```

```
    for t in 1:num_trials
        A = randn(N,N)
        A = (A + A') / 2
        λ ref, vref = eigen(A).values[end], eigen(A).vectors[:, end]
        times[t] = @elapsed begin
            λ, v, itr = powEigen(A)
        end
        eigenvalues[t] = λ
        itrs[t] = itr
        resnorms[t] = norm(A*v - λ*v)
        relerrs_lambda[t] = abs(λ - λ ref) / abs(λ ref)

        v_normalized = v / norm(v)
        vref_normalized = vref / norm(vref)
        if dot(v_normalized, vref_normalized) < 0
            v_normalized = -v_normalized
        end
        relerrs_v[t] = norm(v_normalized - vref_normalized) / norm(vref_normalized)
    end

    (; resnorms, relerrs_lambda, relerrs_v, times, itrs)
end
```

5.3 結果

べき乗法による最大固有値・固有ベクトルの推定結果を示す．行列サイズ $N = 50, 100, 200, 400$ に対して 100 回の試行を行い，残差ノルム・固有値相対誤差・固有ベクトル相対誤差・計算時間・反復回数を評価した．

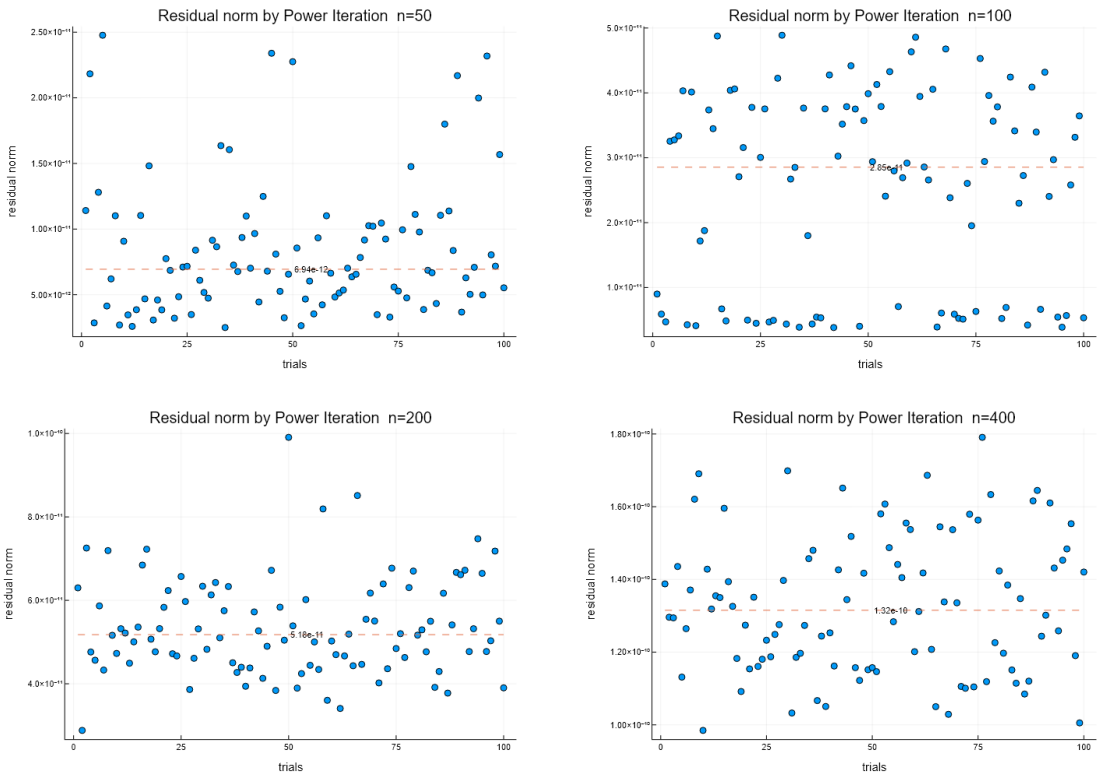

図 7　べき乗法における固有方程式残差ノルム分布（各 N）

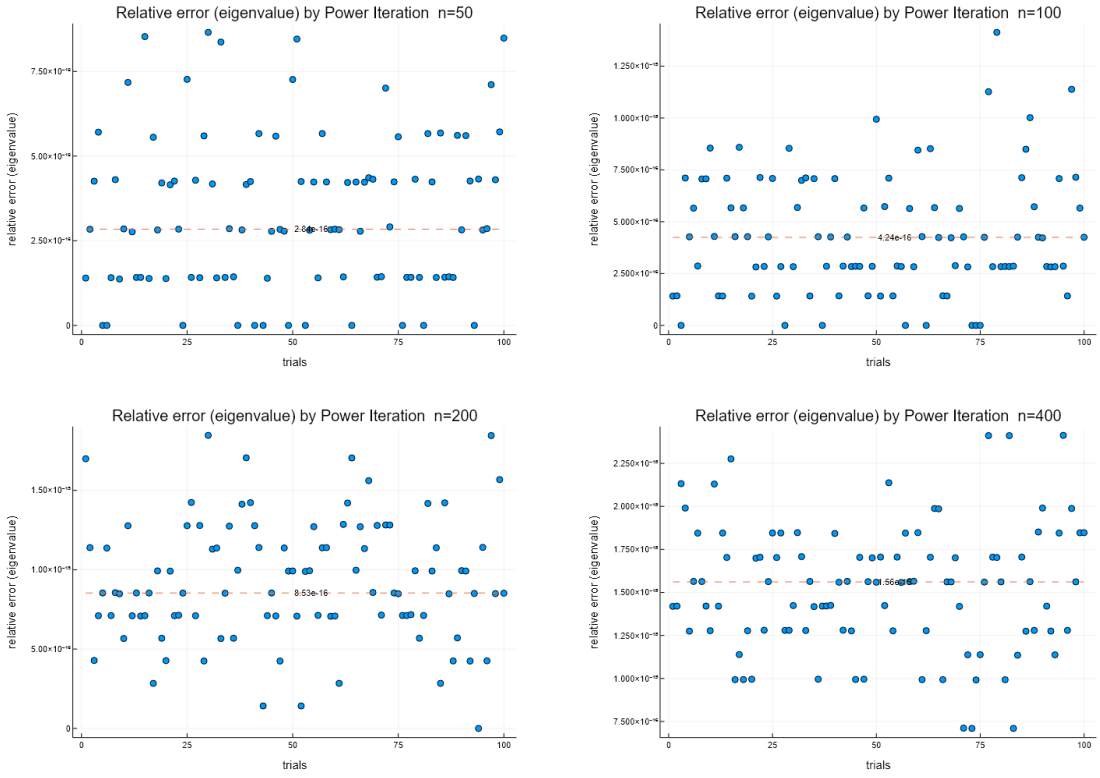

図 8　最大固有値の相対誤差分布（各 N）

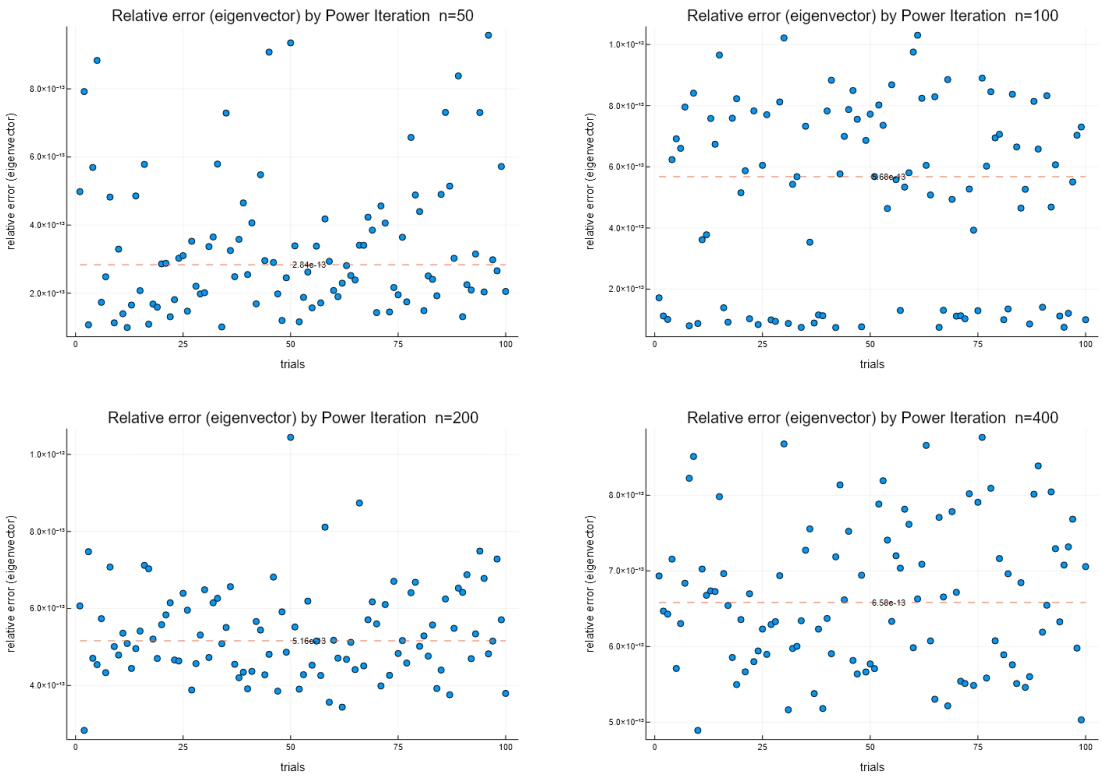

図 9　最大固有ベクトルの相対誤差分布（各 N）

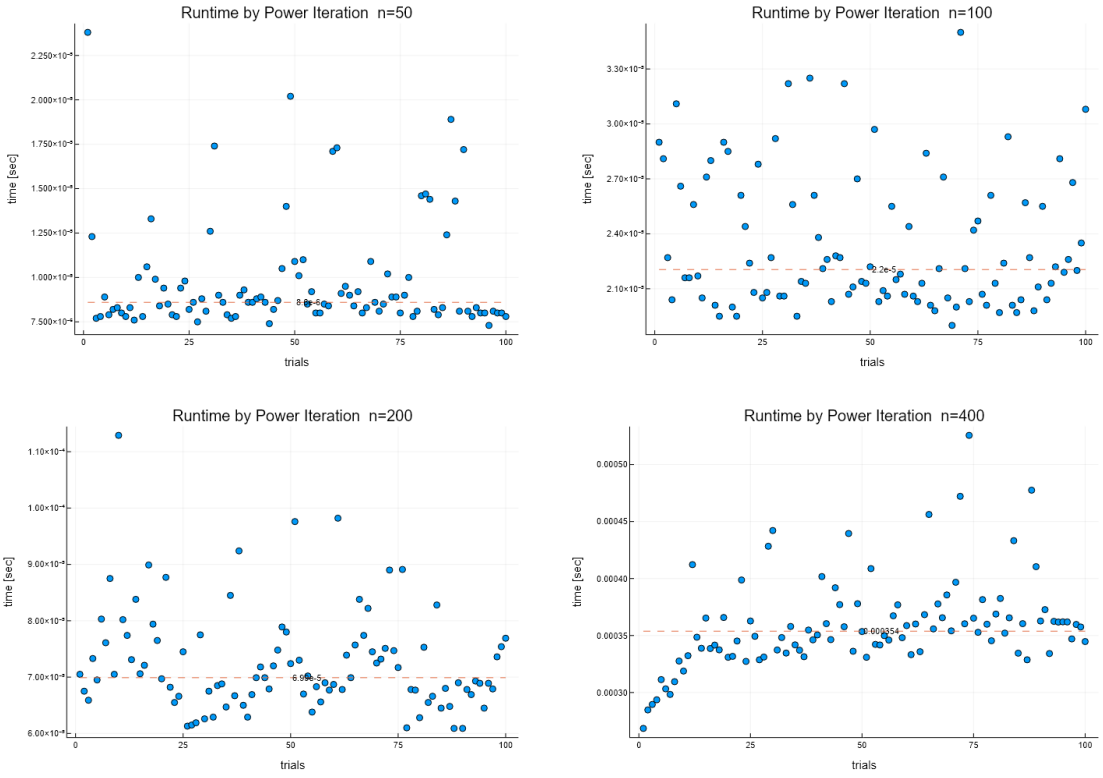

図 10　べき乗法の計算時間分布（各 N）

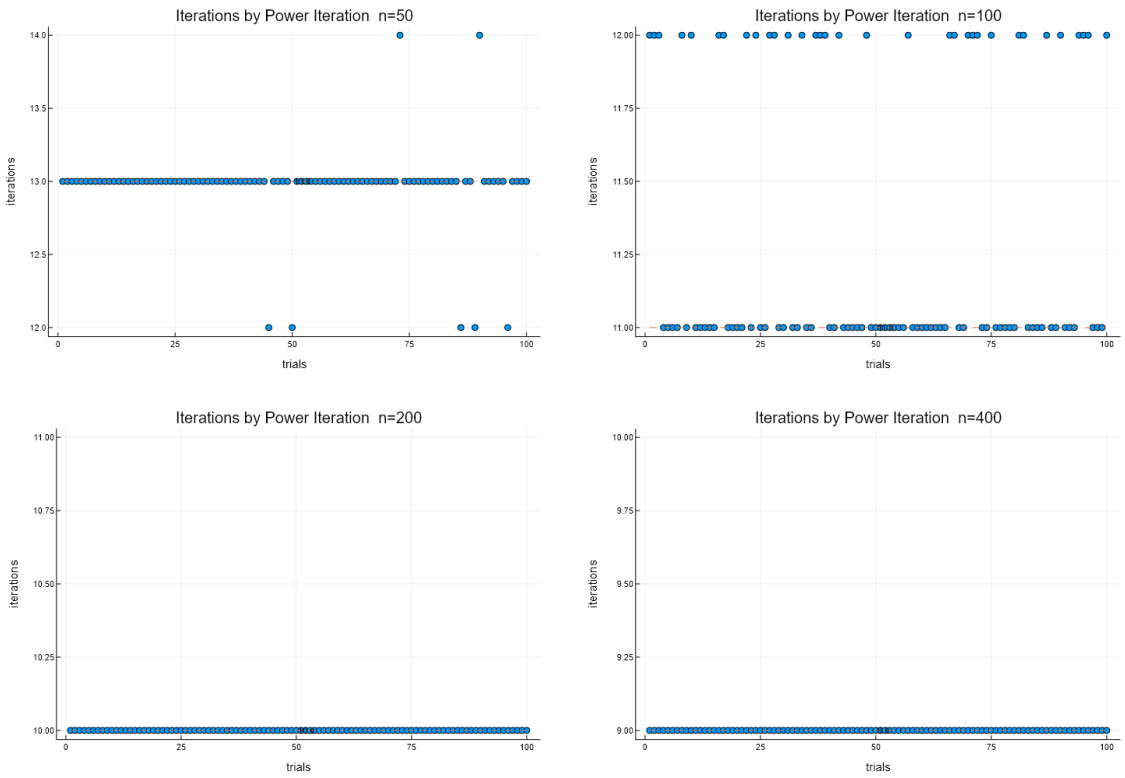

図 11　べき乗法における収束までの反復回数分布（各 N）

表 4　べき乗法の結果まとめ（100 試行の中央値）

N	残差ノルム	固有値相対誤差	固有ベクトル相対誤差	計算時間 [s]	反復回数
50	7.26×10^{-12}	2.82×10^{-16}	2.87×10^{-13}	0.000014	13.00
100	2.16×10^{-11}	4.25×10^{-16}	4.36×10^{-13}	0.000036	11.00
200	5.45×10^{-11}	9.94×10^{-16}	5.43×10^{-13}	0.000093	10.00
400	1.29×10^{-10}	1.56×10^{-15}	6.42×10^{-13}	0.000353	9.00

5.4　考察

実行時間のオーダーは $O(N^1.7)$ 程度と推定され，理論的な $O(N^2)$ よりも若干良好である．これは、N が大きくなると反復回数が減少する傾向があり、全体の計算量に影響を与えたためと考えられる。また，残差ノルム 、固有値相対誤差 及び 固有ベクトル相対誤差は，いずれも N の増加に伴いわずかに増加するものの，収束判定の許容誤差 10^{-12} を一様に設定しているため、大きな悪化は見られなかった。反復回数は N の増加に伴い減少傾向が見られ，これは固有値の分離度が高まることで収束が速まったためと考えられる。

6　課題 5：逆反復法とべき乗法を用いた固有値問題の高速化

6.1　原理・方法

逆反復法（Inverse Iteration Method）は，行列 $A \in \mathbb{R}^{n\times n}$ の固有値のうち，あらかじめ与えたシフト値 μ に最も近い固有値および対応する固有ベクトルを求める手法である．

シフト μ を用いて変換行列

$$B = (A - \mu I)^{-1}$$

を考えると，B の固有値は

$$\frac{1}{\lambda_i - \mu} \quad (i = 1, \ldots, n)$$

である．したがって，この B に対してべき乗法を適用すれば，μ に最も近い固有値 λ_i が支配的となる．

このとき，反復計算の手順は次のように表される：

$$(A - \mu I)\boldsymbol{y}_k = \boldsymbol{x}_{k-1}, \quad \boldsymbol{x}_k = \frac{\boldsymbol{y}_k}{\|\boldsymbol{y}_k\|}.$$

十分な反復の後，$\boldsymbol{x}_k$ は μ に最も近い固有値に対応する固有ベクトルに収束し，固有値はレイリー商

$$\lambda = \frac{\boldsymbol{x}^T A \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}}$$

により近似される．

計算量は，初回に $(A - \mu I)$ の LU 分解を行うために $O(n^3)$，各反復で前進・後退代入を行うために $O(n^2)$ を要する．したがって，全体として $O(n^3 + kn^2)$ となる（k は反復回数）．

本課題では，数値的安定性を確保するため，シフト値に微小量 $\varepsilon = 10^{-6}$ を加え，Julia により次のように実装した：

```
function invEigen(A, mu, max_iter=1000, tol=1e-12)
    n = size(A, 1)
    ε = 1e-6  # near-singular avoidance
    A_shifted = A - (mu + ε) * I
    F = lu(A_shifted)
    b_k = rand(n)
    b_k /= norm(b_k)
    for _ in 1:max_iter
        y = F \ b_k
        y /= norm(y)
        if norm(y - b_k) < tol
            b_k = y
            break
        end
        b_k = y
```

```
    end
    λ = dot(b_k, A * b_k) / dot(b_k, b_k)
    return λ, b_k
end
```

逆反復法は，特定の固有値に近いシフト値を与えることで，$\frac{\lambda_i+1}{\lambda_i-\mu}$ の項を小さくし、べき乗法よりも速く収束することが期待される.

6.2 実験方法

本課題では、固有値問題を高速化するため、べき乗法で得られた最大固有値をシフト値として逆反復法を適用し、その性能を評価した。行列サイズ $N = 50, 100, 200, 400$ に対して、ランダムに生成した対称行列 $A \in \mathbb{R}^{N\times N}$ に対して 100 回の試行を実施した。行列は $A = (A + A^T)/2$ として対称化した。以下が今回の Julia 実装である。:

```
function pow_andthen_invEigen(A, min_turning_iter= 50, max_iter = 1000, tol = 1e-
12)
    n = size(A, 1)
    b_k = rand(n)
    b_k /= norm(b_k)
    itr = 0
    L = Matrix{eltype(A)}(I, n, n)
    mu = 0.0
    for iter in 1:min_turning_iter
        itr += 1
        # Calculate the matrix-by-vector product Ab
        b_k1 = A * b_k

        # Re-normalize the vector
        b_k1_norm = norm(b_k1)
        b_k1 /= b_k1_norm
        if norm(b_k1 - b_k) < 0.001*norm(b_k)
            break
        end
        b_k = b_k1

    end
    mu = dot(b_k, A * b_k) / dot(b_k, b_k)  # Rayleigh quotient for eigenvalue approximatior
    # b_k リセット
    b_k = rand(n)
    b_k /= norm(b_k)
```

```
    F = lu(A - mu * I)
    for iter in 1:(max_iter - min_turning_iter)
        itr += 1
        # Solve (A - mu*I) y = b_k
        y = F \ b_k

        # Re-normalize the vector
        y_norm = norm(y)
        b_k1 = y / y_norm

        # Check for convergence
        if norm(b_k1 - b_k) < tol*norm(b_k)
            b_k = b_k1
            break
        end

        b_k = b_k1
    end
    return mu, b_k, itr
end

function exp4(N;num_trials=100)
    eigenvalues = zeros(num_trials)
    times       = zeros(num_trials)
    resnorms    = zeros(num_trials)
    relerrs_lambda = zeros(num_trials)
    relerrs_v      = zeros(num_trials)
    itrs           = zeros(Int, num_trials)

    for t in 1:num_trials
        A = randn(N,N)
        A = (A + A') / 2
        λ_ref, vref = eigen(A).values[end], eigen(A).vectors[:, end]
        times[t] = @elapsed begin
            λ, v, itr = pow_andthen_invEigen(A)
        end
        eigenvalues[t] = λ
        itrs[t] = itr
        resnorms[t] = norm(A*v - λ*v)
        relerrs_lambda[t] = abs(λ - λ_ref) / abs(λ_ref)
```

```
        # Normalize both vectors and ensure same sign direction
        v_normalized = v / norm(v)
        vref_normalized = vref / norm(vref)
        # Check if vectors point in opposite directions and flip if needed
        if dot(v_normalized, vref_normalized) < 0
            v_normalized = -v_normalized
        end
        relerrs_v[t] = norm(v_normalized - vref_normalized) / norm(vref_normalized)

    end

    (; resnorms, relerrs_lambda, relerrs_v, times, itrs)
```

6.3 結果

行列サイズ $N = 50, 100, 200, 400$ に対して 100 回の試行を行い，残差ノルム・固有値相対誤差・固有ベクトル相対誤差・計算時間・反復回数を評価した.

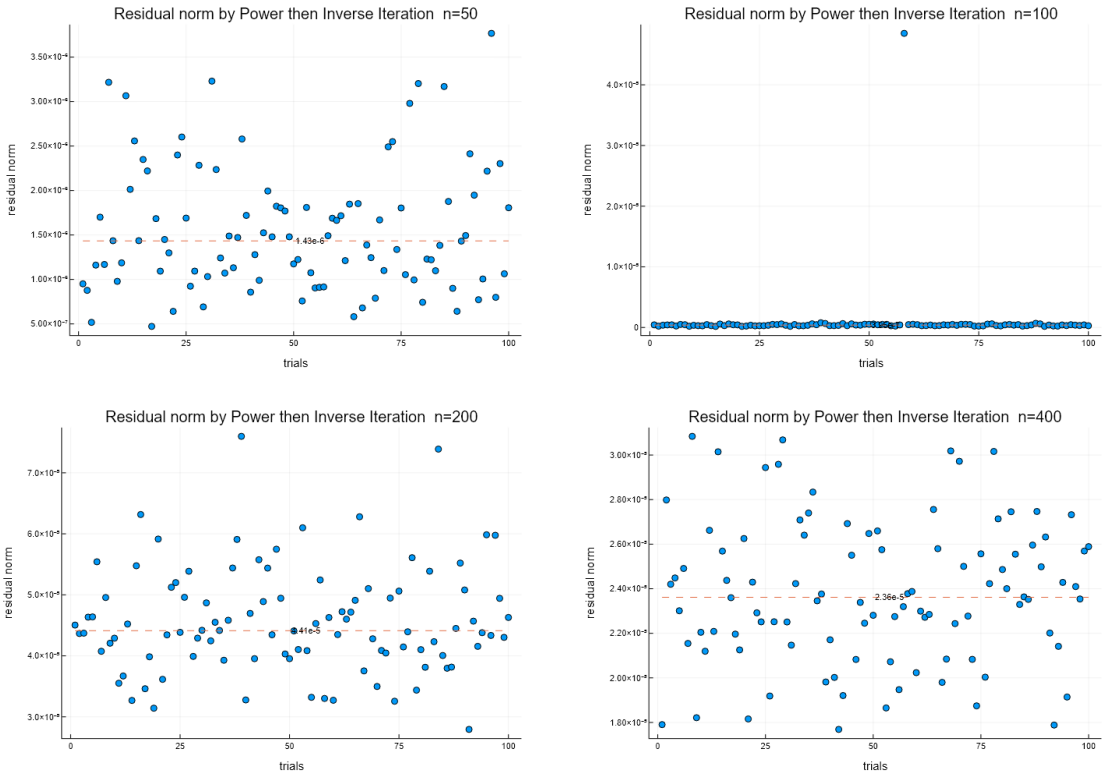

図 12　逆反復法における固有方程式残差ノルム分布（各 N）

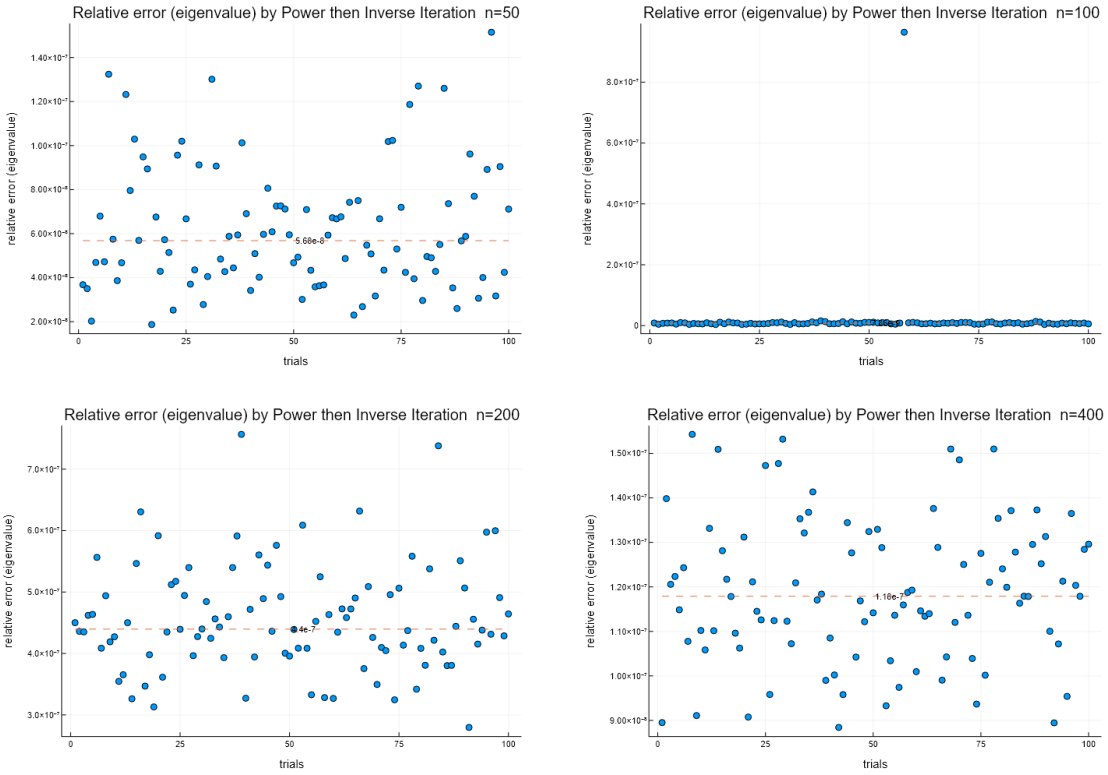

図 13　最大固有値の相対誤差分布（各 N，逆反復法）

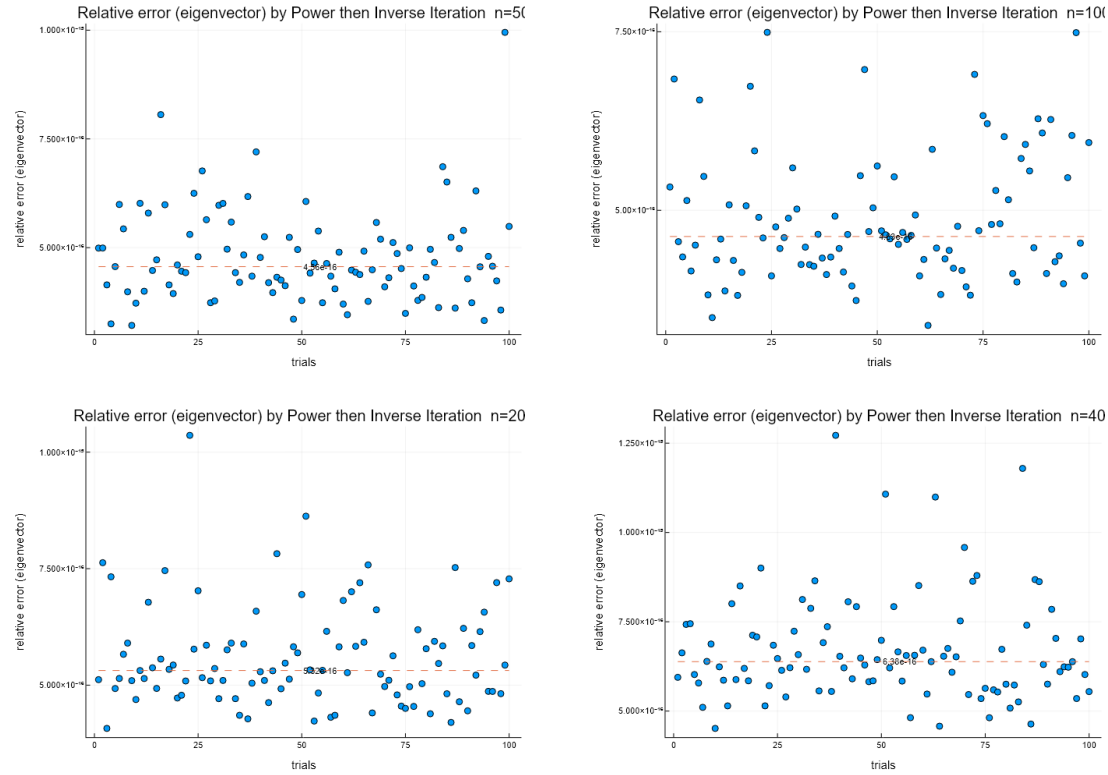

図 14　最大固有ベクトルの相対誤差分布（各 N，逆反復法）

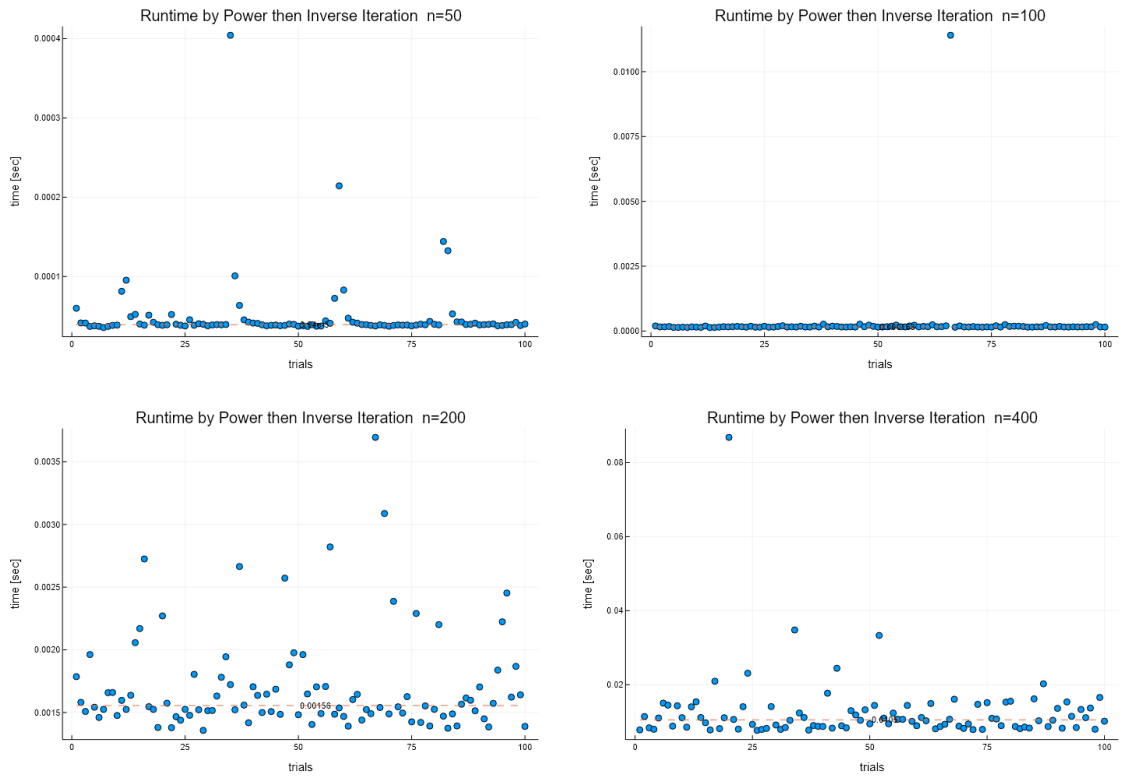

図 15　逆反復法の計算時間分布（各 N）

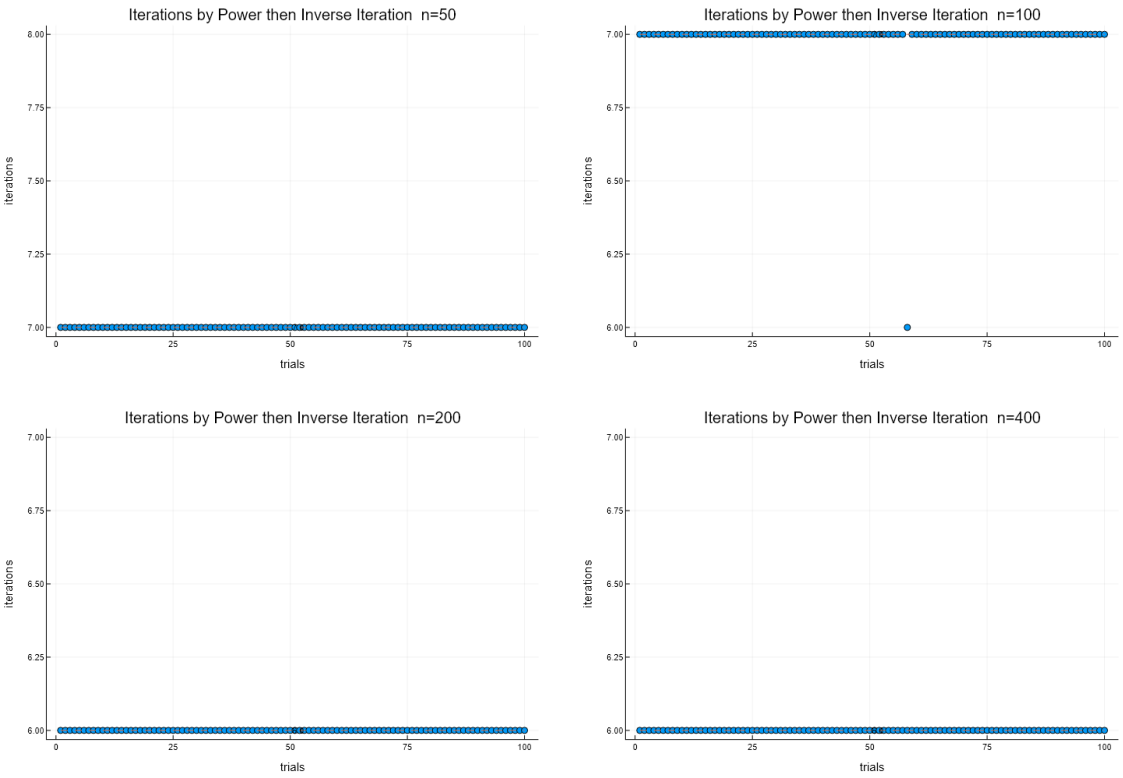

図 16　逆反復法における収束までの反復回数分布（各 N）

表 5 に高速化の結果をまとめる。

表 5　逆反復法を用いた固有値計算の結果まとめ（100 試行の中央値）

N	残差ノルム	固有値相対誤差	固有ベクトル相対誤差	計算時間 [s]	反復回数
50	1.396741×10^{-6}	5.550151×10^{-8}	4.561756×10^{-16}	0.000046	7.00
100	3.569149×10^{-7}	7.120842×10^{-9}	4.624595×10^{-16}	0.000204	7.00
200	4.561472×10^{-5}	4.549618×10^{-7}	5.279023×10^{-16}	0.001780	6.00
400	2.302051×10^{-5}	1.150639×10^{-7}	6.452753×10^{-16}	0.010692	6.00

6.4　考察

逆反復法を用いることで、収束までの反復回数は減少したものの、計算時間は増加した。これは、逆反復法が各反復で線形方程式を解く必要があり、そのための LU 分解が計算コストを押し上げたためである。

具体的には、LU 分解に要する計算量は $O(N^3)$ であり全体の計算量は少なくとも $O(N^3)$ である、これに対しべき乗法の各反復は $O(N^2)$ で済むので反復回数を k とすると全体の計算量は、$O(N^2k)$ となる。今回の場合、べき乗法の反復回数 k が 10 回程度のため、次元 n が k に対して支配的であり、全体の計算時間が LU 分解のコストに引きずられたと考えられる。

反復回数が極端に少ない理由として考えられるのは、実験方法で行列の生成を一様乱数である rand() 関数を用いているため、固有値同士の分離度が高く、収束が速まったためと考えられる。恣意的に固有値同士の分離度が低い行列を選択したり、正規分布の乱数 randn() など他の分布を用いた場合、反復回数が増加し高速化が実現する可能性があるため、今後の課題として検討する価値がある。

7　課題 6：QR 分解と QR 法による固有値問題の解法

7.1　原理・方法

QR 分解（QR Decomposition）は，任意の正方行列 $A \in \mathbb{R}^{n \times n}$ をユニタリ行列 Q と上三角行列 R の積

$$A = QR$$

として表す手法である:contentReference[oaicite:0]index=0．ここで $Q^TQ = I$ が成り立つ．QR 分解はグラム・シュミットの直交化に基づいて構成されるが，数値的安定性を向上させるため，「修正グラム・シュミット法」が用いられる:contentReference[oaicite:1]index=1.

修正版グラム・シュミット法では，行列 $A = (a_1, \ldots, a_n)$ の各列ベクトル a_j に対して次の手順を繰り返す：

$$
\begin{aligned}
&\text{for } j = 1, \ldots, n \text{ do} \\
&\quad R_{j,j} = \|a_j\|, && (1)\\
&\quad q_j = \frac{a_j}{R_{j,j}}, && (2)\\
&\quad \text{for } k = j+1, \ldots, n \text{ do} \\
&\qquad R_{j,k} = q_j^T a_k, && (3)\\
&\qquad a_k \leftarrow a_k - R_{j,k} q_j, && (4)\\
&\quad \text{end for} \\
&\text{end for}
\end{aligned}
$$

この手続きにより，$Q = (q_1, \ldots, q_n)$ は直交行列，R は上三角行列となり，

$$A = QR$$

が成り立つ。

本実験ではこの修正版アルゴリズムを Julia で次のように実装した：

```
function qr_manual(A)
    n = size(A, 1)
    Q = copy(A)
    R = Matrix{eltype(A)}(I, n, n)
    for k = 1:n
        for j = 1:k-1
            R[j, k] = dot(Q[:, j], Q[:, k])
            Q[:, k] .-= R[j, k] * Q[:, j]
        end
        R[k, k] = norm(Q[:, k])
        Q[:, k] /= R[k, k]
    end
    return Q, R
end
```

次に，QR 法（QR Algorithm）は，この QR 分解を反復的に適用して行列の固有値を求める手法である．初期行列 $A_0 = A$ に対して以下の手順を繰り返す：

$$A_k = Q_k R_k, \tag{5}$$

$$A_{k+1} = R_k Q_k (= Q_k^{-1} A_k Q_k). \tag{6}$$

これはユニタリ行列による相似変換の繰り返しであり，各 A_k は A と同じ固有値を持つ．反復を続けると A_k は上三角行列に近づき，その対角成分が固有値に収束する．本実装では，収束判定条件

$$\|A_{k+1} - A_k\|_\infty < \mathrm{tol} \cdot \|A_k\|_\infty$$

を満たした時点で反復を停止し，対角成分を固有値として採用した．また，求めた固有値を初期値として逆反復法を適用し，対応する固有ベクトルを得るようにした．

```
function qrEigens(A; max_iter=100000, tol=1e-10)
    n = size(A, 1)
    Ak = copy(A)
    eigenvectors = Matrix{Float64}(I, n, n)
    itr = 0
    for k in 1:max_iter
        itr += 1
        Q, R = qr_manual(Ak)
        Ak1 = R * Q
        if norm(Ak1 - Ak, Inf) < tol * norm(Ak, Inf)
            break
        end
        Ak = Ak1
    end
    eigenvalues = diag(Ak)
    for k in 1:n
        mu = eigenvalues[k]
        _, v_k = invEigen(A, mu, max_iter, tol)
    end
    return eigenvalues, eigenvectors, itr
end
```

QR 法の理論的計算量は，各反復において $O(n^3)$（QR 分解部分）を要するため，全体では $O(kn^3)$ である．収束率はべき乗法と同様に一次であり，固有値の絶対値が近い場合は収束が遅くなる

7.2 実験方法

本課題では，QR 分解を用いた固有値計算法（QR 法）の性能を評価した．対称行列 $A \in \mathbb{R}^{n\times n}$ を乱数により生成し，QR 法を適用して全固有値を求め，Julia の標準固有値関数 `eigen(A)` による真値と比較した．

次の関数 `exp5` により，複数の行列サイズ N に対して 100 回の試行を行い，各試行における最大残差ノルム，固有値誤差，計算時間を記録した．

```
function exp5(N; num_trials=100)
    eigenvalues = zeros(num_trials, N)
    times       = zeros(num_trials)
    max_resnorms    = zeros(num_trials)
    max_relerrs_lambda = zeros(num_trials)
    itrs           = zeros(Int, num_trials)

    for t in 1:num_trials
```

```
        A = rand(N,N)
        A = (A + A') / 2              # 対称化
        λ_ref = sort(eigen(A).values) # 参照値
        times[t] = @elapsed begin
            λ, V, itr = qrEigens(A)
        end
        eigenvalues[t, :] = λ
        itrs[t] = itr

        # 残差ノルム max_i ||(A - λ_i I)v_i||
        resnorms_i = zeros(N)
        for i in 1:N
            v_i = V[:, i]
            resnorms_i[i] = norm(A * v_i - λ[i] * v_i)
        end
        max_resnorms[t] = maximum(resnorms_i)

        # 相対誤差 max_i |λ_i - λ^_i| / |λ_i|
        λ_sorted = sort(λ)
        relerrs_i = abs.(λ_sorted .- λ_ref) ./ abs.(λ_ref)
        max_relerrs_lambda[t] = maximum(relerrs_i)
    end

    (; max_resnorms, max_relerrs_lambda, times, itrs)
end
```

ここで測定した量は次の通りである：

- 最大残差ノルム $\max_i \|(A - \lambda_i I)\boldsymbol{v}_i\|$：固有対の再現精度を評価.
- 最大相対誤差 $\max_i \dfrac{|\lambda_i - \hat{\lambda}_i|}{|\lambda_i|}$：真値に対する固有値誤差.
- 計算時間：QR 反復の収束までの所要時間.

各試行で得られた値の統計的変動を考慮するため，中央値ではなく最大値を評価指標とし，QR 法の数値安定性と収束性能を比較できるよう設計した.

7.3 結果

コードを実行したものの、1 時間経っても終わらなかったため、やむなく num_trials を 10 に減らして実行した。N = 10, 20, 40, 80 に対して 10 回の試行を行い、各試行における最大残差ノルム、固有値誤差、計算時間を記録し。

7.4　考察

古典／改良 GS とハウスホルダーの安定性差。

8　課題 7：QR 法

8.1　原理・方法

QR 法の高速化には，「シフト（shift）」と「減次（deflation）」の 2 つの戦略が用いられる:contentReference[oaicite:0]index=0:contentReference[oaicite:1]index=1.

■シフトによる収束の加速　行列 A に対して定数 σ を引いた

$$A' = A - \sigma I$$

という変換を行う操作をシフトという．A の固有値が $\lambda_1, \ldots, \lambda_n$ のとき，シフト後の行列 A' の固有値は $\lambda_i - \sigma$ である．

QR 法の反復中で，例えば第 n 固有値 λ_n の近似値 σ が得られたとき，次の反復を $A_k - \sigma I$ に対して行うことで収束を加速できる．特に σ が λ_n に近ければ，行列 A_k の下三角成分の収束率

$$\rho'_{i,j} = \frac{|\lambda_i - \sigma|}{|\lambda_j - \sigma|}$$

はシフト前の収束率 $\rho_{i,j} = |\lambda_i|/|\lambda_j|$ より小さくなり，第 n 行要素の収束が著しく速くなる．このような「レイリー商シフト（Rayleigh quotient shift）」は，実際の反復で有効な加速手段である．

■減次による計算量削減　QR 法を繰り返すと，行列 A_k の第 n 行成分がほぼゼロになる（すなわち下三角要素が十分に小さい）時点が現れる．すなわち，ある閾値 ε に対して

$$\frac{|a_{k;n,j}|}{|a_{k;n,n}|} < \varepsilon \quad (1 \le j < n)$$

が成立すれば，A_k は

$$A_k \approx \begin{pmatrix} B_k & * \\ 0 & a_{k;n,n} \end{pmatrix}$$

の形となる．ここで B_k は A_k の左上 $(n-1)\times(n-1)$ の部分行列であり，固有値は $a_{k;n,n}$ と B_k の固有値で近似できる．このとき $a_{k;n,n}$ を 1 つの固有値として確定させ，以後の反復を B_k に対してのみ続けることで，行列のサイズを 1 つ減らし計算量を削減できる．この手続きを「減次」と呼ぶ．

本課題では，これらの戦略を組み合わせた高速 QR 法を Julia で実装した．以下にその主要部を示す：

```
function qrEigens_fast(A; max_iter=100000, tol=1e-10)
    n = size(A, 1)
    Ak = copy(A)
    itr = 0
    mu = 0.0
```

```
    dimension = n
    eigenvalues = zeros(n)
    eigenvectors = Matrix{Float64}(I, n, n)

    for k in 1:max_iter
        itr += 1
        L = Matrix{eltype(Ak)}(I, dimension, dimension)
        # --- 減次判定 ---
        if norm((Ak - mu*L)[dimension, 1:dimension-1], 1) < tol
            eigenvalues[dimension] = Ak[dimension, dimension]
            dimension -= 1
            Ak = Ak[1:dimension, 1:dimension]
            if dimension == 0
                break
            end
        end
        # --- シフト付き QR 反復 ---
        mu = Ak[dimension, dimension]
        Q, R = qr_manual(Ak - mu*L)
        Ak1 = R * Q + mu * L
        if norm(Ak1 - Ak, Inf) < tol * norm(Ak, Inf)
            eigenvalues[1:dimension] = diag(Ak1)
            break
        end
        Ak = Ak1
    end
    # --- 固有ベクトルは逆反復法で算出 ---
    for k in 1:n
        mu = eigenvalues[k]
        _, v_k = invEigen(A, mu, 1000, tol)
        eigenvectors[:, k] = v_k
    end
    return eigenvalues, eigenvectors, itr
end
```

この方法により，従来の QR 法に比べて収束速度が向上し，計算量を効果的に削減することができる．

8.2 実験方法

課題 6 と同一の評価系を用いる．対称行列 $A \in \mathbb{R}^{N\times N}$ を乱数から生成・対称化し，高速化 QR 法（シフト＋減次）`qrEigens_fast` を適用する．各 N について 100 回試行し，(i) 最大残差ノルム $\max_i \|(A-\lambda_i I)\boldsymbol{v}_i\|$，(ii) 固有値の最大相対誤差 $\max_i |\lambda_i - \hat{\lambda}_i|/|\lambda_i|$（参照は `eigen(A)`），(iii) 計算時間，(iv) 反復回数を記録し，代表値として中央値を用いる．

```
function exp6(N; num_trials=100)
    eigenvalues = zeros(num_trials, N)
    times       = zeros(num_trials)
    max_resnorms    = zeros(num_trials)
    max_relerrs_lambda = zeros(num_trials)
    itrs           = zeros(Int, num_trials)

    for t in 1:num_trials
        A = rand(N,N)
        A = (A + A') / 2            # 対称化
        λref = sort(eigen(A).values)
        times[t] = @elapsed begin
            λ, V, itr = qrEigens_fast(A)
        end
        eigenvalues[t, :] = λ
        itrs[t] = itr
        # 残差ノルム（最大）
        resnorms_i = zeros(N)
        for i in 1:N
            v_i = V[:, i]
            resnorms_i[i] = norm(A * v_i - λ[i] * v_i)
        end
        max_resnorms[t] = maximum(resnorms_i)
        # 固有値の最大相対誤差
        λ_sorted = sort(λ)
        relerrs_i = abs.(λ_sorted .- λref) ./ abs.(λref)
        max_relerrs_lambda[t] = maximum(relerrs_i)
    end
    (; max_resnorms, max_relerrs_lambda, times, itrs)
end
```

違いはアルゴリズム本体のみ（シフトと減次を併用）であり，評価指標・統計処理は課題 6 と同じである．

8.3 結果

8.4 考察

シフト導入の効果。収束速度と計算コストのトレードオフ。

9 結論

主要な知見を要約。今後の改良点を簡潔に述べる。

参考文献（References）

参考文献

[1] 数理工学実験（2025 年度配布資料）.

付録（Appendix）

- 使用したソースコード（Julia/Python）
- 追加ログ・出力データ