MATH 543 – Homework 8

Parham Khodadi

1 Trefethen & Bau 24.3

1.1 Problem

Let A be a 10×10 random matrix with entries from the standard normal distribution, minus twice the identity. Write a program to plot $||e^{tA}||_2$ against t for $0 < t < 20$ on a log scale, comparing the result to the straight line $e^{t\alpha(A)}$, where $\alpha(A) = \max_j \mathrm{Re}(\gamma_j)$ is the spectral abscissan of A. Run the program for (at least) ten random matrices A and comment on the results. What property of a matrix leads to a $||e^{tA}||_2$ curve that remains oscillatory as $t \to \infty$?

- Hint-0: Consider the structure of the eigenvalues.
- Hint-1: Use `expm` for matrix exponentiation e^{tA}.
- Hint-2: Make sure you have many points in the interval of interest, e.g. use `linspace`
- Hint-3: It is useful to (additionally) plot $\frac{||e^{tA}||_2}{e^{t\alpha(A)}}$.

1.2 Solution

Figures 1 and 2 show $\|e^{tA}\|_2$ and the ratio $\|e^{tA}\|_2 / e^{t\alpha(A)}$ for 12 random 10×10 matrices $A =$ `randn`$(10) - 2I$.

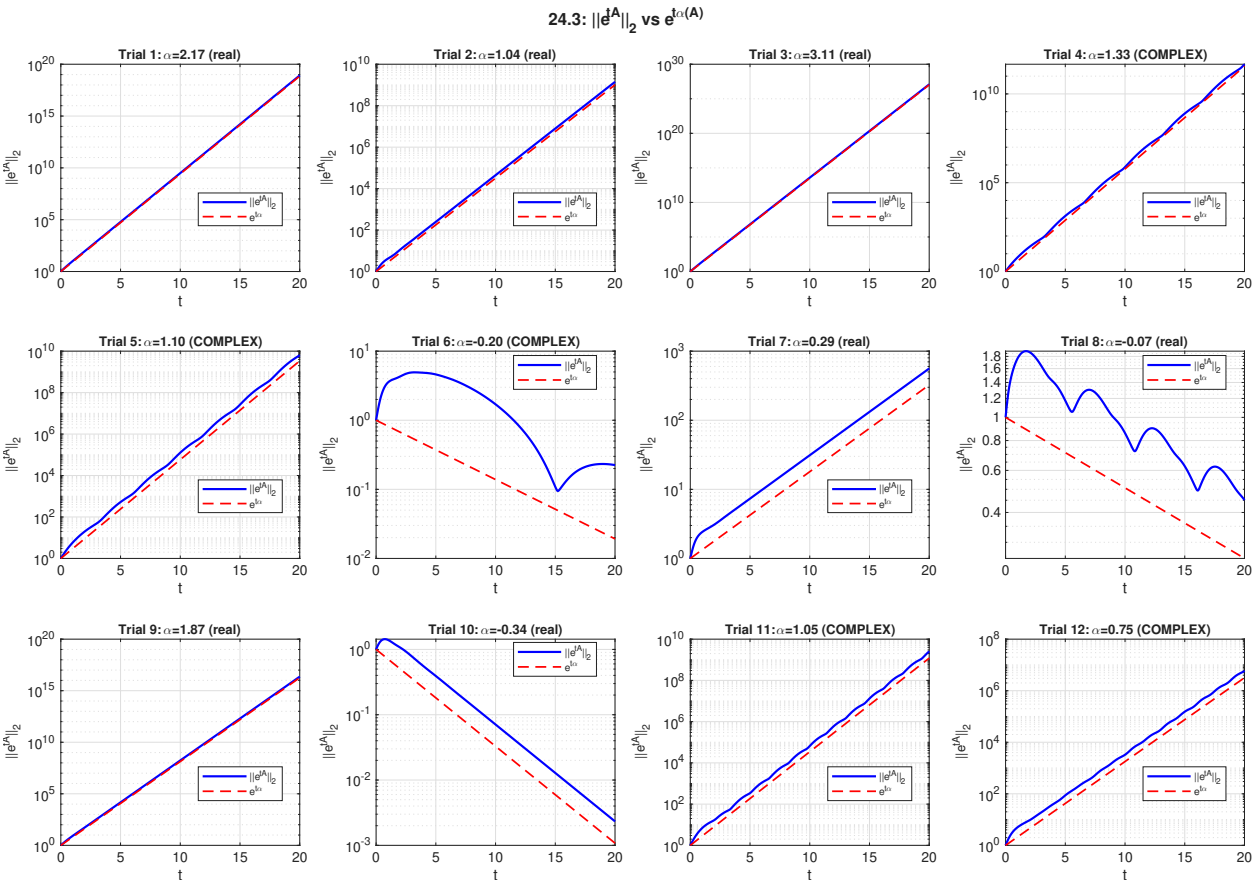

Figure 1: $\|e^{tA}\|_2$ (blue) vs. $e^{t\alpha(A)}$ (red dashed) on a log scale for 12 trials.

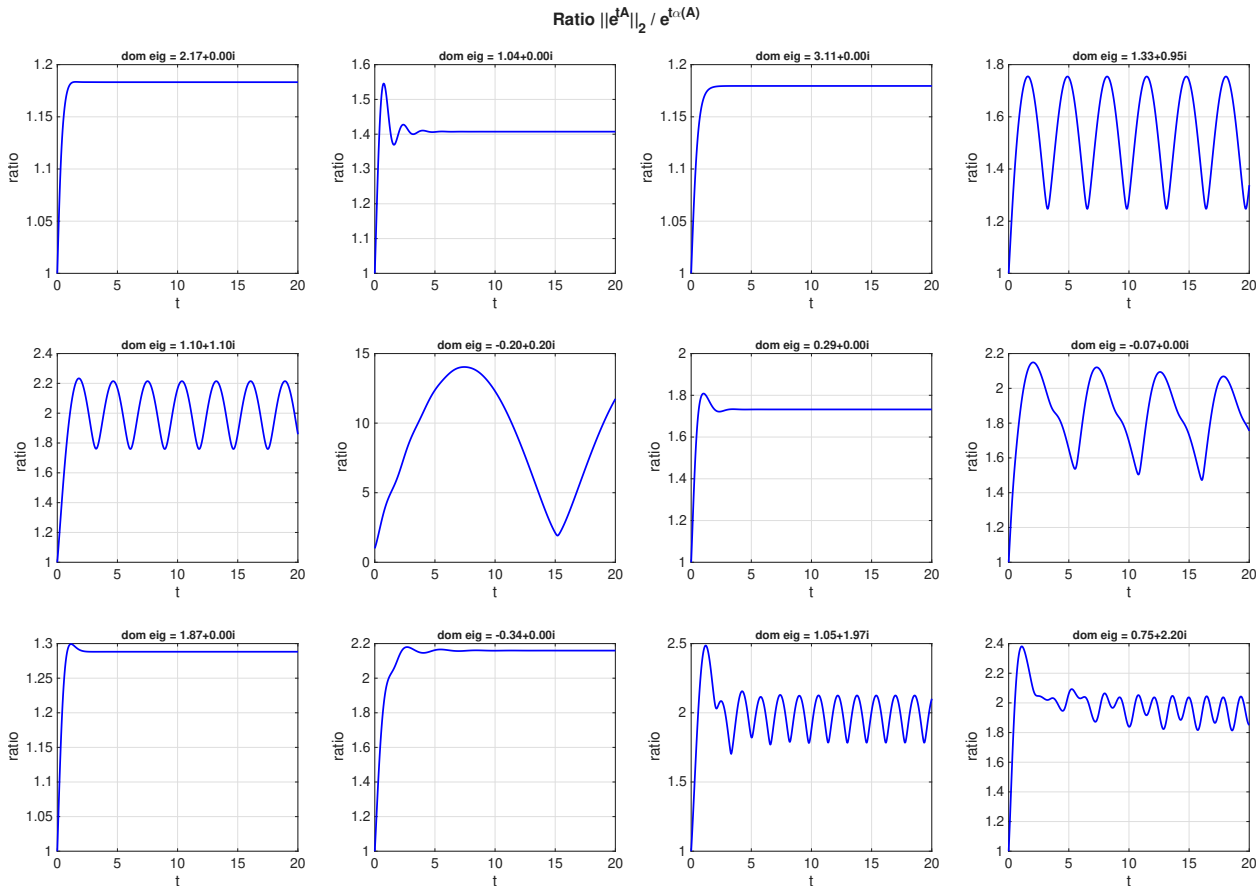

Figure 2: Ratio $\|e^{tA}\|_2 \,/\, e^{t\alpha(A)}$ for the same 12 trials. Titles show the dominant eigenvalue.

- In every trial, $\|e^{tA}\|_2 \geq e^{t\alpha(A)}$, so the spectral abscissa acts as a lower bound on the growth/decay rate.
- The ratio $\|e^{tA}\|_2 / e^{t\alpha(A)}$ settles to a finite value as $t \to \infty$, meaning $e^{t\alpha(A)}$ captures the correct asymptotic rate.
- Some trials show transient growth even when $\alpha(A) < 0$ (e.g. Trials 6, 8, 10), due to the non-normality of A.

The curve $\|e^{tA}\|_2$ remains *oscillatory* as $t \to \infty$ when the dominant eigenvalue is **complex**. In the ratio plot, trials with a complex dominant eigenvalue (e.g. $1.33+0.95i$, $1.10+1.10i$, $1.05+1.97i$, $0.75 + 2.20i$) oscillate persistently, while those with a real dominant eigenvalue flatten out. This happens because $e^{t\lambda} = e^{t\,\mathrm{Re}(\lambda)} \cdot e^{it\,\mathrm{Im}(\lambda)}$: the first factor controls growth/decay, while the second oscillates with period $2\pi/|\,\mathrm{Im}(\lambda)|$.

2 Implement-and-Test – Householder Reduction to Hessenberg Form.

2.1 Problem

Submit: Code + Validation, show working 5×5 and 7×7 examples. Compare with a library call (e.g. `hess`) — for validation use a 9×9 example. Comment on similarities and differences.

2.2 Solution

The implementation follows Algorithm 26.1 (Trefethen & Bau). At each step k, a Householder reflector Q_k is chosen to leave the first k rows unchanged, introducing zeros below the subdiagonal in column k. The right-multiplication by Q_k only mixes columns $k{+}1, \ldots, m$, preserving the zeros already introduced. The result is a similarity transformation $H = Q^T A Q$ in upper Hessenberg form.

5×5 **matrix:**

$$H = \begin{bmatrix} 0.1609 & -1.3903 & 0.9019 & -0.8620 & 0.6595 \\ 1.2752 & -1.4534 & -1.5474 & 0.5078 & 1.6177 \\ 0 & -1.0510 & 0.8818 & -1.3579 & -1.3146 \\ 0 & 0 & 2.0099 & -0.5017 & 2.2336 \\ 0 & 0 & 0 & -0.2236 & -0.0437 \end{bmatrix}$$

$\|QHQ^T - A\| = 5.18 \times 10^{-15}, \quad \|Q^TQ - I\| = 1.55 \times 10^{-15}.$

7×7 **matrix:**

$$H = \begin{bmatrix} 1.3685 & -1.2520 & 0.6385 & 0.0816 & 0.6459 & -1.9721 & 0.2283 \\ 2.2339 & -0.1934 & 0.3656 & -0.1672 & -0.9138 & -0.1742 & -0.1961 \\ 0 & 2.6565 & 2.5680 & -0.7610 & -1.2410 & -0.0366 & -1.5187 \\ 0 & 0 & -1.5629 & 1.3058 & 1.4355 & -1.3195 & 0.6787 \\ 0 & 0 & 0 & 1.8152 & 0.7519 & -1.0083 & -0.6156 \\ 0 & 0 & 0 & 0 & -0.9047 & 0.1798 & -1.8903 \\ 0 & 0 & 0 & 0 & 0 & -1.4207 & -1.3685 \end{bmatrix}$$

$\|QHQ^T - A\| = 2.99 \times 10^{-15}, \quad \|Q^TQ - I\| = 9.45 \times 10^{-16}.$

9×9 **validation vs.** `hess()`**:**

	my code	`hess()`
$\|QHQ^T - A\|$	5.61×10^{-15}	2.50×10^{-15}
$\|Q^TQ - I\|$	1.47×10^{-15}	7.59×10^{-16}
max $\|H_{ij}\|$ below subdiag	0	0

The eigenvalues of A and H match to all displayed digits, confirming that the similarity transformation preserves the spectrum.

Similarities: Both produce valid upper Hessenberg matrices with $\|QHQ^T - A\| = \mathcal{O}(\varepsilon_{\text{mach}})$, orthogonal Q, and identical eigenvalues.

Differences: The matrices H and Q are not identical between the two implementations.

3 Implement-and-Test – Rayleigh Quotient Iteration.

3.1 Problem

Submit: Code + Validation Minimum Validation: 11×11 matrix; (explicitly) show that at least one eigenvalue-eigenvector pair matches library call in MATLAB.

3.2 Solution

The implementation follows Algorithm 27.3 (Trefethen & Bau). The Rayleigh quotient $\lambda^{(k)} = v^{(k)T} A v^{(k)}$ gives a quadratically accurate eigenvalue estimate, and inverse iteration refines the eigenvector at each step. For real symmetric matrices, this yields cubic convergence (Theorem 27.3).

The algorithm was applied to a random 11×11 symmetric matrix from five starting vectors. In every case, RQI converged to an eigenvalue–eigenvector pair matching `eig()` to machine precision:

Start	Converged λ	Lib match #	$\lvert\lambda_{\text{RQI}} - \lambda_{\text{lib}}\rvert$
1	-1.244051896238503	5	1.78×10^{-15}
2	0.156283081516918	6	0
3	-1.244051896238503	5	1.11×10^{-15}
4	0.900730478209063	7	2.22×10^{-16}
5	0.156283081516918	6	1.11×10^{-16}

Eigenpair verification:

$$\begin{aligned} \lambda_{\text{RQI}} &= 2.828348903885657 \\ \lambda_{\text{lib}} &= 2.828348903885659 \\ |\lambda_{\text{RQI}} - \lambda_{\text{lib}}| &= 2.22 \times 10^{-15} \\ \|v_{\text{RQI}} - v_{\text{lib}}\|_2 &= 1.64 \times 10^{-15} \\ \|Av - \lambda v\|_2 &= 9.50 \times 10^{-16} \end{aligned}$$

Cubic convergence: The table below shows $|e_k| = |\lambda^{(k)} - \lambda_{\text{true}}|$ and the ratio $\log_{10}|e_k| / \log_{10}|e_{k-1}|$, which approaches 3 as predicted by Theorem 27.3:

k	$\lvert e_k\rvert$	$\log\lvert e_k\rvert / \log\lvert e_{k-1}\rvert$
1	1.5621×10^{-1}	3.09
2	3.2118×10^{-3}	3.09
3	2.0848×10^{-8}	3.08
4	1.3323×10^{-15}	1.94
5	2.2204×10^{-15}	0.99

The ratio is approximately 3 for iterations 1–3, confirming cubic convergence. At iteration 4 the error hits machine epsilon and the ratio breaks down, which is expected. Note that the near-singular warnings from MATLAB (suppressed in code) are expected and harmless: as $\lambda^{(k)} \to \lambda_J$, the matrix $(A - \lambda^{(k)} I)$ becomes singular by definition, but this does not affect the accuracy of inverse iteration (cf. Exercise 27.5).

Appendix: MATLAB Code

```
%% MATH 543 HW 8
% by Parham Khodadi

clear; clc; close all;

%% Problem 1

rng(42);
m    = 10;
nPts = 500;
t    = linspace(0, 20, nPts);

% Pre-generate all 12 matrices so both figures use the same ones
matrices = cell(1,12);
for trial = 1:12
    matrices{trial} = randn(m) - 2*eye(m);
end

% Semilogy plots: ||e^{tA}||_2 vs e^{t*alpha(A)}
f1 = figure('Position', [100 100 1400 900]);
for trial = 1:12
    A       = matrices{trial};
    eigvals = eig(A);
    alphaA  = max(real(eigvals));
    [~,idx] = max(real(eigvals));
    domEig  = eigvals(idx);

    normEtA = arrayfun(@(s) norm(expm(s*A),2), t);
    refLine = exp(t * alphaA);

    if abs(imag(domEig)) > 1e-10, tag = 'COMPLEX'; else, tag = 'real'; end

    subplot(3,4,trial);
    semilogy(t, normEtA, 'b-', 'LineWidth',1.5); hold on;
    semilogy(t, refLine, 'r--','LineWidth',1.2); hold off; grid on;
    title(sprintf('Trial %d: \\alpha=%.2f (%s)', trial, alphaA, tag),'
        FontSize',9);
    xlabel('t'); ylabel('||e^{tA}||_2');
    legend('||e^{tA}||_2','e^{t\alpha}','Location','best','FontSize',7);
end
sgtitle('24.3: ||e^{tA}||_2 vs e^{t\alpha(A)}','FontSize',14,'FontWeight',
    'bold');
print(f1,'Figures/P1_main','-depsc2');

% Ratio plots: ||e^{tA}||_2 / e^{t*alpha(A)}
f2 = figure('Position', [100 100 1400 900]);
for trial = 1:12
    A       = matrices{trial};
    eigvals = eig(A);
    alphaA  = max(real(eigvals));
    [~,idx] = max(real(eigvals));
```

```
    domEig   = eigvals(idx);

    normEtA = arrayfun(@(s) norm(expm(s*A),2), t);
    ratio   = normEtA ./ exp(t*alphaA);

    subplot(3,4,trial);
    plot(t, ratio, 'b-', 'LineWidth',1.2); grid on;
    title(sprintf('dom eig = %.2f%+.2fi', real(domEig), imag(domEig)),'
       FontSize',8);
    xlabel('t'); ylabel('ratio');
end
sgtitle('Ratio ||e^{tA}||_2 / e^{t\alpha(A)}','FontSize',14,'FontWeight','
   bold');
print(f2,'Figures/P1_ratio','-depsc2');

%% Problem 2

fprintf('=== Problem 2: Householder Hessenberg ===\n\n');

% --- 5x5 ---
rng(100); A5 = randn(5);
[H5, Q5] = householder_hessenberg(A5);
fprintf('--- 5x5 ---\n');
fprintf('H =\n'); disp(H5);
fprintf('||Q*H*Q''-A|| = %.2e,  ||Q''Q-I|| = %.2e\n\n', ...
    norm(Q5*H5*Q5'-A5), norm(Q5'*Q5-eye(5)));

% --- 7x7 ---
rng(200); A7 = randn(7);
[H7, Q7] = householder_hessenberg(A7);
fprintf('--- 7x7 ---\n');
fprintf('H =\n'); disp(H7);
fprintf('||Q*H*Q''-A|| = %.2e,  ||Q''Q-I|| = %.2e\n\n', ...
    norm(Q7*H7*Q7'-A7), norm(Q7'*Q7-eye(7)));

% --- 9x9 validation vs hess() ---
rng(300); A9 = randn(9);
[H9, Q9]       = householder_hessenberg(A9);
[Q9lib, H9lib] = hess(A9);

fprintf('--- 9x9 Validation vs hess() ---\n');
fprintf('                Ours          hess()\n');
fprintf('||QHQ''-A||    %.2e    %.2e\n', norm(Q9*H9*Q9'-A9), norm(Q9lib*
   H9lib*Q9lib'-A9));
fprintf('||Q''Q-I||     %.2e    %.2e\n', norm(Q9'*Q9-eye(9)), norm(Q9lib'*
   Q9lib-eye(9)));
fprintf('max below      %.2e    %.2e\n', max(max(abs(tril(H9,-2)))), max(max
   (abs(tril(H9lib,-2)))));
fprintf('\neig(A):  '); fprintf('%.6f  ', sort(real(eig(A9)))); fprintf('\n'
   );
fprintf('eig(H):  '); fprintf('%.6f  ', sort(real(eig(H9)))); fprintf('\n\n'
   );

%% Problem 3
```

```
97
98  fprintf('=== Problem 3: Rayleigh Quotient Iteration (11x11) ===\n\n');
99
100 rng(543); m = 11;
101 Araw = randn(m); A = Araw + Araw';
102 [V_lib, D_lib] = eig(A);
103 eigvals_lib = diag(D_lib);
104
105 fprintf('Library eigenvalues:\n');
106 for j=1:m, fprintf('  lambda_%2d = %18.14f\n', j, eigvals_lib(j)); end
107
108 fprintf('\n--- RQI runs ---\n');
109 for s = 1:5
110     rng(1000+s);
111     v0 = randn(m,1); v0 = v0/norm(v0);
112     [lam, v, hist] = rayleigh_quotient_iteration(A, v0);
113     [err, idx] = min(abs(eigvals_lib - lam));
114     fprintf('Start %d: lambda=%.15f  matches #%d  |err|=%.2e  res=%.2e\n',
            ...
115         s, lam, idx, err, norm(A*v - lam*v));
116 end
117
118 % Verification
119 fprintf('\n--- Detailed eigenpair verification ---\n');
120 rng(2024);
121 v0 = randn(m,1); v0 = v0/norm(v0);
122 [lam, v, hist] = rayleigh_quotient_iteration(A, v0);
123 [~, idx] = min(abs(eigvals_lib - lam));
124 v_lib = V_lib(:,idx);
125 if dot(v,v_lib)<0, v_lib = -v_lib; end
126
127 fprintf('  RQI lambda   = %20.15f\n', lam);
128 fprintf('  Lib lambda   = %20.15f\n', eigvals_lib(idx));
129 fprintf('  |difference| = %.2e\n', abs(lam - eigvals_lib(idx)));
130 fprintf('  ||v_rqi - v_lib|| = %.2e\n', norm(v - v_lib));
131 fprintf('  ||Av - lam*v||    = %.2e\n\n', norm(A*v - lam*v));
132
133 fprintf('  Cubic convergence:\n');
134 true_lam = eigvals_lib(idx);
135 for k = 2:length(hist)
136     ep = abs(hist(k-1)-true_lam); ec = abs(hist(k)-true_lam);
137     if ep > 1e-15
138         fprintf('    k=%d: err=%.4e  ratio=%.2f\n', k-1, ec, log10(ec)/
              log10(ep));
139     end
140 end
141
142 %% Functions
143
144 function [H, Q] = householder_hessenberg(A)
145     m = size(A,1);
146     H = A;  Q = eye(m);
147     for k = 1:m-2
148         x  = H(k+1:m, k);
```

```
        s  = sign(x(1)); if s==0, s=1; end
        e1 = zeros(length(x),1); e1(1)=1;
        v  = s*norm(x)*e1 + x;
        v  = v/norm(v);
        H(k+1:m, k:m) = H(k+1:m, k:m) - 2*v*(v'*H(k+1:m, k:m));
        H(1:m, k+1:m) = H(1:m, k+1:m) - 2*(H(1:m, k+1:m)*v)*v';
        Q(:, k+1:m)   = Q(:, k+1:m)   - 2*(Q(:, k+1:m)*v)*v';
    end
    H = triu(H,-1);
end

function [lambda, v, hist] = rayleigh_quotient_iteration(A, v0, maxiter,
   tol)
    if nargin<3, maxiter=100; end
    if nargin<4, tol=1e-14; end
    m = size(A,1);
    v = v0/norm(v0);
    lambda = v'*A*v;
    hist = lambda;
    for k = 1:maxiter
        w = (A - lambda*eye(m)) \ v;
        v = w/norm(w);
        lambda = v'*A*v;
        hist(end+1) = lambda;
        if norm(A*v - lambda*v) < tol, break; end
    end
end
```

Appendix: MATLAB Output

```
=== Problem 2: Householder Hessenberg ===

--- 5x5 ---
H =
    0.1609   -1.3903    0.9019   -0.8620    0.6595
    1.2752   -1.4534   -1.5474    0.5078    1.6177
         0   -1.0510    0.8818   -1.3579   -1.3146
         0         0    2.0099   -0.5017    2.2336
         0         0         0   -0.2236   -0.0437

||Q*H*Q'-A|| = 5.18e-15,  ||Q'Q-I|| = 1.55e-15

--- 7x7 ---
H =
    1.3685   -1.2520    0.6385    0.0816    0.6459   -1.9721    0.2283
    2.2339   -0.1934    0.3656   -0.1672   -0.9138   -0.1742   -0.1961
         0    2.6565    2.5680   -0.7610   -1.2410   -0.0366   -1.5187
         0         0   -1.5629    1.3058    1.4355   -1.3195    0.6787
         0         0         0    1.8152    0.7519   -1.0083   -0.6156
         0         0         0         0   -0.9047    0.1798   -1.8903
         0         0         0         0         0   -1.4207   -1.3685

||Q*H*Q'-A|| = 2.99e-15,  ||Q'Q-I|| = 9.45e-16

--- 9x9 Validation vs hess() ---
              Ours          hess()
||QHQ'-A||   5.61e-15    2.50e-15
||Q'Q-I||    1.47e-15    7.59e-16
max below    0.00e+00    0.00e+00

eig(A): -1.930148 -1.930148 -0.942473 -0.066067 -0.066067 1.294533 1.294533 1.446307 2.698050
eig(H): -1.930148 -1.930148 -0.942473 -0.066067 -0.066067 1.294533 1.294533 1.446307 2.698050

=== Problem 3: Rayleigh Quotient Iteration (11x11) ===

Library eigenvalues:
  lambda_ 1 =  -8.07026331030929
  lambda_ 2 =  -6.35514062816386
  lambda_ 3 =  -4.24037246642239
  lambda_ 4 =  -3.73156424651871
  lambda_ 5 =  -1.24405189623850
  lambda_ 6 =   0.15628308151692
  lambda_ 7 =   0.90073047820906
  lambda_ 8 =   1.60010137723812
  lambda_ 9 =   2.82834890388566
  lambda_10 =   4.94499401678928
```

```
  lambda_11 =   6.00937039025832

--- RQI runs ---
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  1.02
> In HW8>rayleigh_quotient_iteration (line 168)
In HW8 (line 112)

Start 1: lambda=-1.244051896238503  matches #5  |err|=1.78e-15  res=8.03e-16
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  1.25
> In HW8>rayleigh_quotient_iteration (line 168)
In HW8 (line 112)

Start 2: lambda=0.156283081516918  matches #6  |err|=0.00e+00  res=7.31e-16
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  1.02
> In HW8>rayleigh_quotient_iteration (line 168)
In HW8 (line 112)

Start 3: lambda=-1.244051896238503  matches #5  |err|=1.11e-15  res=9.13e-16
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  4.21
> In HW8>rayleigh_quotient_iteration (line 168)
In HW8 (line 112)

Start 4: lambda=0.900730478209063  matches #7  |err|=2.22e-16  res=6.56e-16
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  3.13
> In HW8>rayleigh_quotient_iteration (line 168)
In HW8 (line 112)

Start 5: lambda=0.156283081516918  matches #6  |err|=1.11e-16  res=8.15e-16

--- Detailed eigenpair verification ---
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  1.80
> In HW8>rayleigh_quotient_iteration (line 168)
In HW8 (line 122)

  RQI lambda  =    2.828348903885657
  Lib lambda  =    2.828348903885659
  |difference|= 2.22e-15
  ||v_rqi - v_lib|| = 1.64e-15
  ||Av - lam*v||    = 9.50e-16

  Cubic convergence:
    k=1: err=1.5621e-01  ratio=3.09
    k=2: err=3.2118e-03  ratio=3.09
    k=3: err=2.0848e-08  ratio=3.08
    k=4: err=1.3323e-15  ratio=1.94
    k=5: err=2.2204e-15  ratio=0.99
```