Homework 3

SF2524, Group26

Jiacheng Xu jiacxu@kth.se Xinyi Ji jxinyi@kth.se

December 9, 2025

0

AI usage level: 2

1

(b) The error will be dominated by $|\lambda_i/\lambda_j|$ close to 1. The biggest λ_m is α^{m-1}, m is the dimension. The second λ_{m-1} is $\alpha^{m-1} \cdot (0.99 - \frac{1}{5\alpha})$ (due to alpha_example.jl). For large α, $\frac{\lambda_{m-1}}{\lambda_m}$ close to 0.99. In fact, The dominant convergence ratio is $\lambda_{19}/\lambda_{20} \approx 0.9898$.

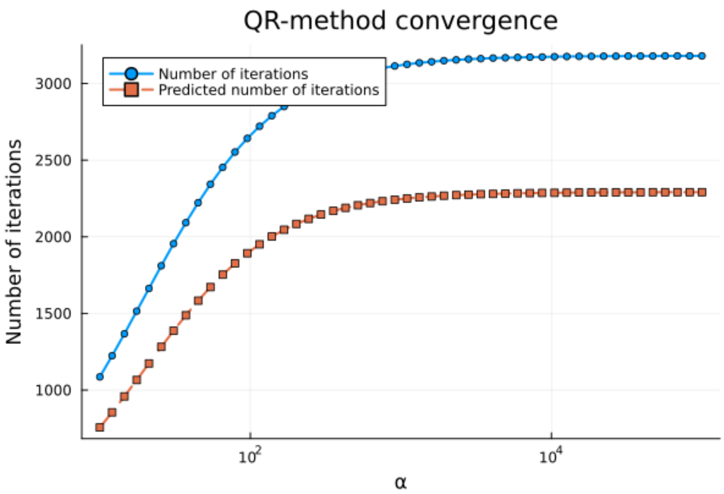

Figure 1: different alpha for predicted iterations and real iterations

(c) We saw the real iterations follows the same slope as the predicted iterations. confirming the theoretical understanding.

2

(a) Householder reflector P is Orthogonal Matrix, meaning it keeps vector norm unchanged.

$$||Px||_2 = ||x||_2 \tag{1}$$

Therefore, if $Px = \alpha y$

$$||\alpha y||_2 = ||x||_2 \tag{2}$$

$$\alpha = \pm\frac{||x||_2}{||y||_2} \tag{3}$$

The normal vector u of this hyperplane is defined as the difference between the starting vector and the target vector.

Thus, the normal vector u is:

$$u = x - \alpha y \tag{4}$$

The householder reflector is given by:

$$P = I - 2\frac{uu^T}{u^T u} \tag{5}$$

m	Algorithm 2	Naive
10	0.000008	0.000009
100	0.003856	0.010881
200	0.024893	0.067131
300	0.070165	0.320940
400	0.258229	0.867418

Table 1: Computational time between naive householder and Reduction to Hessenberg form

(c) The efficient Algorithm 2 is significantly faster, especially for larger matrices, as it avoids forming the full Householder matrices explicitly.

ε	$\lvert\bar{h}_{2,1}\rvert$ ($\sigma = 0$)	$\lvert\bar{h}_{2,1}\rvert$ ($\sigma = a_{2,2}$)
4.0×10^{-1}	9.6070×10^{-2}	7.6923×10^{-2}
1.0×10^{-1}	3.1077×10^{-2}	4.9875×10^{-3}
1.0×10^{-2}	3.3111×10^{-3}	4.9999×10^{-5}
1.0×10^{-3}	3.3311×10^{-4}	5.0000×10^{-7}
1.0×10^{-4}	3.3331×10^{-5}	5.0000×10^{-9}
1.0×10^{-5}	3.3333×10^{-6}	5.0000×10^{-11}
1.0×10^{-6}	3.3333×10^{-7}	5.0000×10^{-13}
1.0×10^{-7}	3.3333×10^{-8}	5.0000×10^{-15}
1.0×10^{-8}	3.3333×10^{-9}	5.0000×10^{-17}
1.0×10^{-9}	3.3333×10^{-10}	5.0000×10^{-19}
1.0×10^{-10}	3.3333×10^{-11}	5.0000×10^{-21}
0	0	0

Table 2: Comparison of shifted and non-shifted QR methods

(d) Implemented one-step shifted QR for the 2×2 matrix and generated the complete Table 2.

Unshifted $\sigma = 0$ shows linear convergence, convergence rate follows $|\lambda_2/\lambda_1|$ about 1/3.

Shifted by $a_{2,2}$ shows quadratic convergence.

3

(a) We implement the Schur-Parlett method as described in the lecture notes. For the matrix:

$$A = \begin{pmatrix} 1 & 4 & 4 \\ 3 & -1 & 3 \\ -1 & 4 & 4 \end{pmatrix},$$

we compute:

$$\sin(A) \approx \begin{pmatrix} 0.8332 & 0.2203 & 0.2396 \\ 0.3967 & 0.5142 & -0.4111 \\ -0.3449 & -0.2320 & 0.4113 \end{pmatrix}.$$

(b) Following the instructions, we perform the calculations. To improve clarity, we also plot the moving average of computation times using a window size of 10. The results are shown in Figure 2.

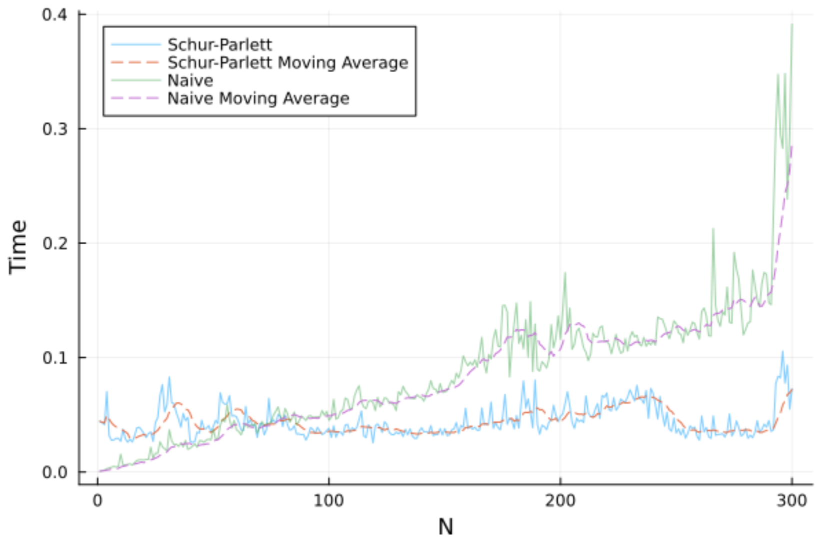

Figure 2: Computation times and moving average for the Schur-Parlett method.

The Schur-Parlett method becomes efficient for approximately $N > 80$.

(c) For the naive method, each matrix operation requires $\mathcal{O}(n^3)$ flops. Performing this N times results in a complexity of $\mathcal{O}(Nn^3)$.

For the Schur-Parlett method, the most computationally intensive step is the Schur factorization, which has a complexity of $\mathcal{O}(n^3)$. The cost of subsequent operations is negligible in comparison, giving an overall complexity of $\mathcal{O}(n^3)$. The dependence on N is constant, as supported by the steady computation times for fixed n shown in Figure 2.

4

(a)

$$A = \begin{pmatrix} \pi & 1 \\ 0 & \pi + \varepsilon \end{pmatrix}, \quad \varepsilon > 0.$$

Suppose polynomial p interpolates the given function g at the eigenvalues π and $\pi + \varepsilon$:

$$p(\pi) = g(\pi), \quad p(\pi + \varepsilon) = g(\pi + \varepsilon).$$

Let $p(z) = \alpha + \beta z$. From the interpolation conditions, we have:

$$\alpha + \beta\pi = g(\pi), \quad \alpha + \beta(\pi + \varepsilon) = g(\pi + \varepsilon).$$

Subtracting the first equation from the second gives:

$$\beta\varepsilon = g(\pi + \varepsilon) - g(\pi) \implies \beta = \frac{g(\pi + \varepsilon) - g(\pi)}{\varepsilon}.$$

Substitute β back into $\alpha + \beta\pi = g(\pi)$:

$$\alpha = g(\pi) - \beta\pi = g(\pi) - \frac{\pi\big(g(\pi + \varepsilon) - g(\pi)\big)}{\varepsilon}.$$

Thus,

$$\alpha = \frac{(\varepsilon + \pi)g(\pi) - \pi g(\pi + \varepsilon)}{\varepsilon}.$$

(b) Using the result from (a) for the exponential function $g(z) = e^z$, we have:

$$\beta = \frac{e^{\pi+\varepsilon} - e^{\pi}}{\varepsilon}, \quad \alpha = \frac{(\varepsilon + \pi)e^{\pi} - \pi e^{\pi+\varepsilon}}{\varepsilon}.$$

Hence the matrix exponential is:

$$\exp(A) = \alpha I + \beta A = \frac{(\varepsilon + \pi)e^{\pi} - \pi e^{\pi+\varepsilon}}{\varepsilon} I + \frac{e^{\pi+\varepsilon} - e^{\pi}}{\varepsilon} A.$$

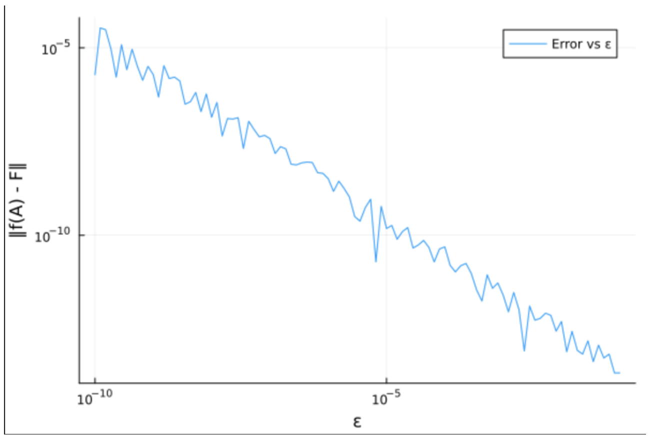

Figure 3: the error with ϵ

(c) As ϵ gets smaller, the computed matrix exponential gets closer to the exact result, so the error decreases.

5

Table 3: The total number of matrix-matrix products.

	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7
m=1	0	1	2	3	4	5	6	7
m=2	1	2	3	4	5	6	7	8
m=3	2	3	4	5	6	7	8	9
m=4	3	4	5	6	7	8	9	10
m=5	4	5	6	7	8	9	10	11
m=6	5	6	7	8	9	10	11	12
m=7	6	7	8	9	10	11	12	13

Table 4: log_{10} error

	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7
m=1	1.516	0.5117	-0.1337	-0.3723	-0.6585	-0.9545	-1.2534	-1.5535
m=2	1.5938	0.7375	-0.3206	-1.1107	-1.7883	-2.4243	-3.0424	-3.6523
m=3	1.5169	-0.1342	-1.1286	-2.1503	-3.1176	-4.0533	-4.9727	-5.8839
m=4	1.3283	-0.3969	-1.9238	-3.2678	-4.5404	-5.7786	-6.9998	-8.2124
m=5	1.052	-1.0446	-2.8219	-4.4678	-6.0432	-7.5834	-9.106	-10.6199
m=6	0.7033	-1.6867	-3.7853	-5.7354	-7.6132	-9.4552	-11.2793	-13.083
m=7	0.2931	-2.4079	-4.8082	-7.0614	-9.2413	-11.3849	-13.4566	-13.6518

Table 5: CPU-time.

	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7
m=1	0.000153	0.00038	0.000629	0.000819	0.004136	0.001417	0.002192	0.002951
m=2	0.000453	0.000772	0.000955	0.001386	0.00239	0.002901	0.005569	0.002816
m=3	0.000633	0.00084	0.001945	0.001112	0.001471	0.002232	0.001859	0.00377
m=4	0.000913	0.00109	0.001963	0.00139	0.001771	0.002374	0.002093	0.002838
m=5	0.001495	0.002324	0.001557	0.002477	0.001929	0.003152	0.002348	0.004732
m=6	0.001377	0.002026	0.001728	0.002613	0.002226	0.003047	0.004568	0.002887
m=7	0.002433	0.002066	0.002884	0.003344	0.003267	0.003398	0.003068	0.006568

6

(a) It proves the house holder reflector can map a vector (given vector x) to a first unit vector αe_1. So that $P := I - 2uu^T$ satisfy $Px = \alpha e_1$

(b) Givens rotations to perform QR factorization on unreduced Hessenberg matrices. It constructing upper triangular forms and achieves the final QR decomposition.

(c) The shift QR method is: $QR = H - uI$, u is one of the exact λ. If A is unreduced Hessenburg matrix and singular, The R matrix in QR factorization will have 0 at bottom right.

(d) Normalized simultaneous iteration is a generalization of the normalized power method. Equivalence between normalized and unnormalized holds if the R-matrices are invertible and the QR-factorization is chosen uniquely.

(e) The unnormalized simultaneous iteration converges to the eigenvector, have assumption on: A is diagonalizable with distinct and ordered eigenvalues. The initial matrix is lower triangular matrix with 1 one the diagonal.

(f)(g) Jiacheng is working on it.

7

Xinyi is working on it.