The Chinese University of Hong Kong, Shenzhen
SDS · School of Data Science
Andre Milzarek · Fall Semester 2024-25

DDA 3005 — Numerical Methods

Solutions 4

Problem 1 (Power Iteration with Shift): (approx. *20* points)
Let the matrix $\boldsymbol{A} \in \mathbb{R}^{4\times 4}$ and the initial point $\boldsymbol{x}^0 \in \mathbb{R}^4$ be given via

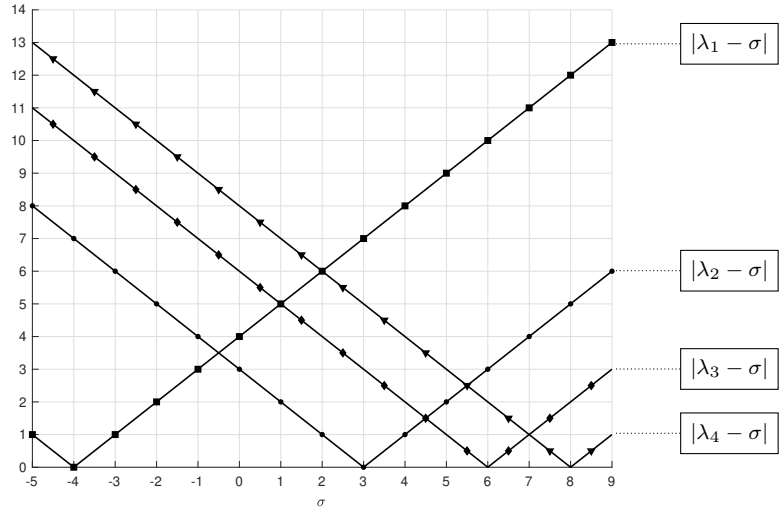
$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 5 & 0 \\ 0 & 3 & 0 & 0 \\ 5 & 0 & 1 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix}, \qquad \tilde{\boldsymbol{x}}^0 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \quad \boldsymbol{x}^0 = \frac{\tilde{\boldsymbol{x}}^0}{\|\tilde{\boldsymbol{x}}^0\|}. \tag{1}$$

In this problem, we want to apply the (normalized) power iteration with shift $\sigma \in \mathbb{R}$ to $\boldsymbol{A}$:

$$\tilde{\boldsymbol{x}}^k = (\boldsymbol{A} - \sigma \boldsymbol{I})\boldsymbol{x}^k, \quad \boldsymbol{x}^k = \tilde{\boldsymbol{x}}^k / \|\tilde{\boldsymbol{x}}^k\|, \quad \sigma_k = (\boldsymbol{x}^k)^\top \boldsymbol{A}\boldsymbol{x}^k, \quad k = 1, 2, \ldots$$

Let $(\lambda_i, \boldsymbol{v}_i)$, $i = 1, \ldots, 4$, further denote the corresponding eigenpairs of $\boldsymbol{A}$. It can be shown that $(\boldsymbol{x}^0)^\top \boldsymbol{v}_i \neq 0$ for all $i = 1, \ldots, 4$.

The plot below depicts the mappings $\sigma \mapsto |\lambda_i - \sigma|$, $i = 1, 2, 3, 4$ for the different eigenvalues of $\boldsymbol{A}$ and choices of σ.



a) State the eigenvalues $\lambda_1, \ldots, \lambda_4$ of $\boldsymbol{A}$.

b) Discuss which eigenpairs $(\lambda_i, \boldsymbol{v}_i)$ of $\boldsymbol{A}$ can be recovered by the (normalized) power iteration using suitable choices of the shift σ. Can all eigenpairs of $\boldsymbol{A}$ be recovered? Provide detailed explanations!

c) Derive the optimal choice of the shift σ for which the power iteration (1) converges with the fastest possible (optimal) convergence rate.

Solution :

a) We can directly read the eigenvalues of $\boldsymbol{A}$ from the plot: $\lambda_1 = -4$, $\lambda_2 = 3$, $\lambda_3 = 6$, and $\lambda_4 = 8$.

b) The power iteration with shift σ "converges" to the eigenpair of $\boldsymbol{A} - \sigma\boldsymbol{I}$ corresponding to the largest eigenvalue of $\boldsymbol{A} - \sigma\boldsymbol{I}$ (in magnitude). As the eigenpairs of $\boldsymbol{A} - \sigma\boldsymbol{I}$ are given by $(\lambda_i - \sigma, \boldsymbol{v}_i)$, we can use the depicted plot to determine which eigenvalue has the largest magnitude. In particular, we are interested in finding

$$\max_{i=1,2,3,4} |\lambda_i - \sigma|$$

Clearly, for $\sigma < 2$, $\lambda_4 - \sigma = 8 - \sigma$ has the largest magnitude; for $\sigma > 2$, $\lambda_1 - \sigma = -4 - \sigma$ has the largest magnitude. For $\sigma = 2$, it holds that $|\lambda_4 - 2| = |\lambda_1 - 2|$, i.e., there is no gap between the first and second largest eigenvalue and the power method does not converge.

As $(\boldsymbol{x}^0)^\top \boldsymbol{v}_1 \neq 0$ and $(\boldsymbol{x}^0)^\top \boldsymbol{v}_4 \neq 0$, we can conclude that the power method with shift only allows to recover the eigenpairs $(-4, \boldsymbol{v}_1)$ and $(8, \boldsymbol{v}_4)$.

c) The rate of convergence depends on the ratio between the second and first largest eigenvalue of $\boldsymbol{A} - \sigma\boldsymbol{I}$ (in magnitude). Let us denote those two eigenvalues by $\tilde{\lambda}_1 - \sigma$ and $\tilde{\lambda}_2 - \sigma$ ($\tilde{\lambda}_1$ and $\tilde{\lambda}_2$ can change with the choice of σ). Then, the rate is given by:

$$\frac{|\tilde{\lambda}_2 - \sigma|}{|\tilde{\lambda}_1 - \sigma|} = 1 - \frac{|\tilde{\lambda}_1 - \sigma| - |\tilde{\lambda}_2 - \sigma|}{|\tilde{\lambda}_1 - \sigma|}$$

We see that the best rate can be obtained when $|\tilde{\lambda}_1 - \sigma| - |\tilde{\lambda}_2 - \sigma|$ is as large as possible and $|\tilde{\lambda}_1 - \sigma|$ is a small as possible. Using the plot, we can easily find $\tilde{\lambda}_1$ and $\tilde{\lambda}_2$. For $\sigma \leq 1$, we have

$$\frac{|\tilde{\lambda}_1 - \sigma| - |\tilde{\lambda}_2 - \sigma|}{|\tilde{\lambda}_1 - \sigma|} = \frac{2}{8 - \sigma} \quad \Longrightarrow \quad \text{best value at } \sigma = 1\text{: } \frac{2}{7}.$$

For $\sigma \geq 5.5$, we have

$$\frac{|\tilde{\lambda}_1 - \sigma| - |\tilde{\lambda}_2 - \sigma|}{|\tilde{\lambda}_1 - \sigma|} = \frac{7}{4 + \sigma} \quad \Longrightarrow \quad \text{best value at } \sigma = 5.5\text{: } \frac{14}{19}.$$

For $\sigma \in (1, 2]$, it holds that $\frac{|\tilde{\lambda}_1 - \sigma| - |\tilde{\lambda}_2 - \sigma|}{|\tilde{\lambda}_1 - \sigma|} = \frac{8 - \sigma - 4 - \sigma}{8 - \sigma} = \frac{4 - 2\sigma}{8 - \sigma}$ – this is largest for $\sigma = 1$ (again yielding $\frac{2}{7}$).

For $\sigma \in [2, 5.5)$, it holds that $\frac{|\tilde{\lambda}_1 - \sigma| - |\tilde{\lambda}_2 - \sigma|}{|\tilde{\lambda}_1 - \sigma|} = \frac{4 + \sigma - 8 + \sigma}{4 + \sigma} = \frac{2\sigma - 4}{4 + \sigma}$ – which is largest for $\sigma = 5.5$.

Overall, we can conclude that the optimal rate is attained for the shift $\sigma = 5.5$.

Problem 2 (Power Methods): (approx. *30* points)

In this exercise, we want to calculate eigen-pairs of the 5×5 matrix using power methods:

$$\boldsymbol{A} = \begin{bmatrix} 46 & 99 & 45 & 24 & -27 \\ -6 & -32 & -6 & 4 & 18 \\ 18 & 18 & 19 & 6 & 0 \\ -9 & -90 & -9 & -47 & 27 \\ 12 & 39 & 12 & 28 & 19 \end{bmatrix}.$$

a) Implement the power method and compute the largest eigenvalue and the corresponding eigenvector of $\boldsymbol{A}$, e.g., using the initial point $\boldsymbol{x}^0 = \boldsymbol{e}_1 = [1, 0, 0, 0, 0]^\top$.

Choose a proper stopping criterion for the power method and state how many iterations your algorithm requires to recover the largest eigenvalue (using modest or high accuracy). Explain the choice of your termination criterion.

b) Implement the inverse iteration (with shift σ) to calculate the remaining eigen-pairs of the matrix $\boldsymbol{A}$ (again using $\boldsymbol{x}^0 = \boldsymbol{e}_1$ as initial point). Possible choices for the shift σ are $\sigma \in \{0, 46, -32, 19, -47\}$.

c) Implement the Rayleigh Quotient iteration using the five initial points $\boldsymbol{x}^0 \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_4, \boldsymbol{e}_5\}$ and state the respective obtained eigen-pairs. Compare the performance of the Rayleigh Quotient iteration and the power iterations tested in part a) and b).

Solution :

a) An implementation of the power method is shown below:

```
% problem 2a: power method

A = [46,99,45,24,-27;-6,-32,-6,4,18;18,18,19,6,0;-9,-90,-9,-47,27;12,39,12,28,19];
maxit = 10000;

x = [1;0;0;0;0];
nx = norm(x);

for i = 1:maxit
    x       = A*x;
    nx      = norm(x);
    x       = x/nx;
    sig     = x'*(A*x);
    tol     = norm(A*x-sig*x);

    if tol <= 1e-12
        break
    end

    fprintf(1,'[%i] ; %1.4e ; %1.10f\n',i,tol,sig);
end
```

The method converges to the eigenvalue 55 within 848 iterations. We choose $\|\boldsymbol{A}\boldsymbol{x} - \sigma\boldsymbol{x}\| \leq$ `tol` as stopping criterion. In the case `tol` $= 0$, this implies that σ is an eigenvalue of $\boldsymbol{A}$ with eigenvector $\boldsymbol{x}$.

b) An implementation of the inverse iteration is shown below:

```
% problem 2b: inverse iteration

A = [46,99,45,24,-27;-6,-32,-6,4,18;18,18,19,6,0;-9,-90,-9,-47,27;12,39,12,28,19];
maxit = 10000;

for mu = [0,46,-32,19,-47]
    x   = [1;0;0;0;0];
    nx  = norm(x);

```

```
    fprintf(1,'\nInverse iteration with shift mu = %g\n',mu);

    for i = 1:maxit
        x       = (A-mu*eye(5))\x;
        nx      = norm(x);
        x       = x/nx;
        sig     = x'*(A*x);
        tol     = norm(A*x-sig*x);

        fprintf(1,'[%i] ; %1.4e ; %1.10f\n',i,tol,sig);
        if tol <= 1e-13
            break
        end
    end
end
```

As suggested in the problem description, we use the shifts $\sigma \in \{0, 46, -32, 19, -47\}$. The returned results are as follows:

σ	iterations	eigenvalue λ	eigenvector $\boldsymbol{v}$
0	11	1.0000	$\boldsymbol{v}^\top = [0.7071,\ -0.0000,\ -0.7071,\ 0.0000,\ -0.0000]$
46	47	55.0000	$\boldsymbol{v}^\top = [0.8165,\ 0.0000,\ 0.4082,\ -0.0000,\ 0.4082]$
-32	26	-26.0000	$\boldsymbol{v}^\top = [-0.0000,\ 0.3015,\ -0.0000,\ -0.9045,\ 0.3015]$
19	49	28.0000	$\boldsymbol{v}^\top = [0.3015,\ -0.3015,\ 0.0000,\ -0.0000,\ -0.9045]$
-47	26	-53.0000	$\boldsymbol{v}^\top = [-0.3015,\ 0.0000,\ 0.0000,\ 0.9045,\ -0.3015]$

c) An implementation of the Rayleigh Quotient iteration is shown below:

```
% problem 2c: Rayleight Quotient iteration

A = [46,99,45,24,-27;-6,-32,-6,4,18;18,18,19,6,0;-9,-90,-9,-47,27;12,39,12,28,19];
maxit = 10000;

for i = 1:5
    x    = zeros(5,1);
    x(i) = 1;

    fprintf(1,'\nRayleigh Quotient iteration with x0 = e%i\n',i);

    sig   = x'*(A*x)/norm(x);

    for j = 1:maxit
        x       = (A-sig*eye(5))\x;
        nx      = norm(x);
        x       = x/nx;
        sig     = x'*(A*x);
        tol     = norm(A*x-sig*x);

        fprintf(1,'[%i] ; %1.4e ; %1.10f\n',j,tol,sig);
        if tol <= 1e-13
            break
        end
    end
end
```

Running this code, we can obtain the following results:

$\boldsymbol{x}^0$	iterations	eigenvalue λ	eigenvector $\boldsymbol{v}$
$\boldsymbol{e}_1$	5	55.0000	$\boldsymbol{v}^\top = [-0.8165,\ 0.0000,\ -0.4082,\ 0.0000,\ -0.4082]$
$\boldsymbol{e}_2$	6	-26.0000	$\boldsymbol{v}^\top = [0,\ 0.3015,\ 0.0000,\ -0.9045,\ 0.3015]$
$\boldsymbol{e}_3$	8	1.0000	$\boldsymbol{v}^\top = [-0.7071,\ -0.0000,\ 0.7071,\ 0.0000,\ -0.0000]$
$\boldsymbol{e}_4$	5	-53.0000	$\boldsymbol{v}^\top = [0.3015,\ 0.0000,\ -0.0000,\ -0.9045,\ 0.3015]$
$\boldsymbol{e}_5$	5	28.0000	$\boldsymbol{v}^\top = [0.3015,\ -0.3015,\ -0.0000,\ -0.0000,\ -0.9045]$

Hence, in this example, the Rayleigh Quotient iteration also manages to find all eigenvalues and eigenvectors of $\boldsymbol{A}$. The method converges significantly faster than the inverse iteration even though $\boldsymbol{A}$ is not a symmetric matrix. However, we cannot necessarily ensure and observe fast cubic convergence in this case!

Problem 3 (Computing Roots of Polynomials): (approx. *50* points)

For given degree $n \in \mathbb{N}$ and coefficients $a_i \in \mathbb{R}$, $i = 0, \dots, n-1$, let us consider the univariate polynomial

$$p(x) := x^n + a_{n-1}x^{n-1} + \cdots + a_2x^2 + a_1x + a_0.$$

In this exercise, we want to develop an algorithm that allows computing all roots of the polynomial p using eigenvalues and QR iterations.

a) The so-called companion matrix associated with the polynomial p is given by

$$\boldsymbol{C}_p := \begin{bmatrix} 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & \cdots & 0 & -a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -a_{n-1} \end{bmatrix}.$$

Show that $\det(\boldsymbol{C}_p - \lambda \boldsymbol{I}) = (-1)^n p(\lambda)$ for all λ.

b) Implement the QR algorithm (with shift) discussed in the lecture to calculate the eigenvalues of the companion matrix $\boldsymbol{C}_p$.

In order to improve the performance of your algorithm, you can utilize the following deflation strategy: let $\boldsymbol{X}^k \in \mathbb{R}^{n\times n}$ be the current iterate of the QR iteration. We can terminate the iteration, if $\|\boldsymbol{X}^k(\texttt{n}, \texttt{1:n-1})\| \leq \texttt{tol}$, i.e., if the norm of the first $n-1$ entries of the last row of $\boldsymbol{X}^k$ is small. This ensures that the last row of $\boldsymbol{C}_p$ has been reduced to upper triangular form and $\boldsymbol{X}^k(\texttt{n},\texttt{n}) = \boldsymbol{X}^k_{nn}$ should be a good eigenvalue approximation. We can then continue the QR algorithm with the smaller (deflated) matrix $\boldsymbol{B} = \boldsymbol{X}^k(\texttt{1:n-1}, \texttt{1:n-1}) \in \mathbb{R}^{(n-1)\times(n-1)}$ (or $\mathbb{C}^{(n-1)\times(n-1)}$) as new initial point. This process continues until we have recovered all eigenvalues of $\boldsymbol{C}_p$. Suitable choices for `tol` are 10^{-12} or 10^{-13}. You can also use a shifted QR iteration to further enhance performance (e.g., the Rayleigh quotient or Wilkinson shift can be a good strategy).

Test your implementation and find all roots of the polynomials:

- $p(x) = x^4 - 324x^3 + 7175x^2 + 8100x - 180000$.
- $q(x) = x^{10} - 167x^9 + 10081x^8 - 251447x^7 + 1676815x^6 + 17367175x^5 - 66421125x^4 - 352378125x^3 + 454612500x^2 + 1949062500x$.

Report all found roots of p and q (you can order the roots from smallest to largest).

c) What happens if the coefficients a_2, a_1, and a_0 in the polynomial p are changed to $a_2 = 7225$, $a_1 = -8100$, and $a_0 = 180000$? Can you explain your observations?

Hint: Use a Wilkinson shift or plot the adjusted polynomial p.

Acknowledgements: This question is motivated by a conversation and discussion with Xiuyuan Wang.

Solution :

a) We prove this result via induction. For $n = 1$, the polynomial p reduces to $p(x) = x + a_0$ and the associated companion matrix is given by

$$\boldsymbol{C}_p^1 = \boldsymbol{C}_p = -a_0 \quad \Longrightarrow \quad \det(\boldsymbol{C}_p^1 - \lambda \boldsymbol{I}) = -(a_0 + \lambda) = -p(\lambda).$$

Hence, the base case is obviously true. We now assume that the induction hypothesis holds for $\boldsymbol{C}_p^n$ and we consider $\boldsymbol{C}_p^{n+1} \in \mathbb{R}^{(n+1)\times(n+1)}$. Let us consider the specific form of $\boldsymbol{C}_p^{n+1} - \lambda \boldsymbol{I}$:

$$\boldsymbol{C}_p^{n+1} - \lambda \boldsymbol{I} = \begin{bmatrix} -\lambda & 0 & 0 & \cdots & 0 & -a_0 \\ 1 & -\lambda & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & -\lambda & \cdots & 0 & -a_2 \\ \vdots & \vdots & \ddots & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & -\lambda & -a_{n-1} \\ 0 & 0 & \cdots & 0 & 1 & -a_n - \lambda \end{bmatrix}.$$

Using the Laplace expansion of the determinant along the 1-st row, it follows:

$$\begin{aligned} \det(\boldsymbol{C}_p^{n+1} - \lambda \boldsymbol{I}) = (-1)^2 \cdot (-\lambda) \cdot \det([\boldsymbol{C}_p^{n+1} - \lambda \boldsymbol{I}]_{1,1}) \\ + (-1)^{n+2} \cdot (-a_0) \cdot \det([\boldsymbol{C}_p^{n+1} - \lambda \boldsymbol{I}]_{1,n+1}). \end{aligned}$$

Here, $[\boldsymbol{C}_p^{n+1} - \lambda \boldsymbol{I}]_{i,j}$ is the $n \times n$ submatrix of $\boldsymbol{C}_p^{n+1} - \lambda \boldsymbol{I}$ resulting from deleting the i-th row and j-th column. As $[\boldsymbol{C}_p^{n+1} - \lambda \boldsymbol{I}]_{1,1}$ is a companion matrix with coefficients $a_1, \ldots, a_{n-1}, a_n$, we can apply the induction hypothesis to obtain:

$$\det([\boldsymbol{C}_p^{n+1} - \lambda \boldsymbol{I}]_{1,1}) = (-1)^n(\lambda^n + a_n\lambda^{n-1} + a_{n-1}\lambda^{n-2} + \cdots + a_2\lambda + a_1).$$

Furthermore, it holds that

$$[\boldsymbol{C}_p^{n+1} - \lambda \boldsymbol{I}]_{1,n+1} = \begin{bmatrix} 1 & -\lambda & 0 & \cdots & 0 \\ 0 & 1 & -\lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & 0 & \cdots & 1 & -\lambda \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

This is an upper triangular matrix and we easily see $\det([\boldsymbol{C}_p^{n+1} - \lambda \boldsymbol{I}]_{1,n+1}) = 1$. Combining the results, we obtain

$$\det(\boldsymbol{C}_p^{n+1} - \lambda \boldsymbol{I}) = (-1)^{n+1}(\lambda^{n+1} + a_n\lambda^n + \cdots + a_1\lambda) + (-1)^{n+3}a_0 = (-1)^{n+1}p(\lambda).$$

This finishes our proof.

b) An implementation of the QR algorithm with deflation and Rayleigh shift is shown below:

```
% demo_proots

%-------------------------------------------------------------------------
% build companion matrices

A = [0,0,0,180000;
     1,0,0,-8100;
     0,1,0,-7175;
     0,0,1,+324];

% A =  [0,0,0,0,0,0,0,0,-1949062500 ; ...
%       1,0,0,0,0,0,0,0,-454612500; ...
%       0,1,0,0,0,0,0,0,352378125; ...
%       0,0,1,0,0,0,0,0,66421125; ...
%       0,0,0,1,0,0,0,0,-17367175; ...
%       0,0,0,0,1,0,0,0,-1676815; ...
%       0,0,0,0,0,1,0,0,251447; ...
%       0,0,0,0,0,0,1,0,-10081; ...
%       0,0,0,0,0,0,0,1,167];

%-------------------------------------------------------------------------
% setup QR algorithm

opts.maxit  = 100;
opts.tol    = 1e-13;
n           = size(A,1);

lambda      = zeros(n,1);
B           = A;

fprintf(1,'\nQR-Iteration with Deflation and Rayleigh Quotient Shift\n');
opts.shift  = 'Rayleigh';

for i = 1:n
    fprintf(1,'iteration: %i; matrix size [%i x %i]\n',i,n-i+1,n-i+1);
    [X,lam] = qr_iteration(B,opts);

    B       = X(1:end-1,1:end-1);
    lambda(n-i+1) = lam;
end

%-------------------------------------------------------------------------
% display

lambda  = sort(lambda,'ascend');

fprintf(1,'\n');
disp(lambda)

%-------------------------------------------------------------------------
% inner function

  function [X,lam] = qr_iteration(B,opts)
        X       = B;
        m       = size(B,1);

```

```
        for i = 1:opts.maxit
            switch opts.shift
                case 'pure'
                    sig     = 0;
                case 'Rayleigh'
                    sig     = X(m,m);
                case 'Wilkinson'
                    mu      = eig(X(max(m-1,1):m,max(m-1,1):m));
                    [~,ind] = min(abs(mu-X(m,m)));
                    sig     = mu(ind);
            end

            [Q,R]   = qr(X-sig*eye(m));
            X       = R*Q+sig*eye(m);

            fprintf(1,' --- [%i] ; %1.4e ; %1.4f\n',i,norm(X(m,1:m-1)),X(m,m));
            disp(X(m,m))

            if norm(X(m,1:m-1)) <= opts.tol
                break
            end
        end
        lam = X(end,end);
    end
```

The returned roots for p and q are given by:

$$\begin{aligned} p: &\quad x_1 = -5, \quad x_2 = 5, \quad x_3 = 24, \quad x_4 = 300, \\ q: &\quad x_1 = -5, \quad x_2 = -3, \quad x_3 = -3, \quad x_4 = 3, \quad x_5 = 5, \quad x_6 = 30, \quad x_7 = 35, \\ &\quad x_8 = 50, \quad x_9 = 55. \end{aligned}$$

Notice that the polynomial q has the additional root $x_0 = 0$ which was not explicitly included in the companion matrix. Overall, the method converges very quickly — each of the deflation steps only requires 1–10 iterations using a Rayleigh quotient shift. (Only the double root $x_2 = x_3 = -3$ for q needs more steps and convergence requires 23 iterations).

c) The QR algorithm with Rayleigh shift does not seem to converge in this case. Utilizing the Wilkinson shift, we can recover the roots

$$x_1 = -5i, \quad x_1 = 5i, \quad x_3 = 24, \quad x_4 = 300,$$

i.e., p has complex roots in this case! Hence, the behavior of the QR algorithm with Rayleigh shift is not surprising as this version of the algorithm does not introduce complex numbers (the initial matrix $\boldsymbol{A}$ is real, the QR factorization produce real matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, the Rayleigh shift remains real as well). The Wilkinson shift resolves this issue as the eigenvalues of the considered 2×2 submatrix can be complex. (There are also other ways to refine / adjust the QR algorithm to generally handle this case).

The `Python` code for problem 3 is similar to the presented `MATLAB` code can be found in the attachments.