MACM 316 – Computing Assignment 5

- Read the *Guidelines for Assignments* first.
- Submit a one-page PDF report to Canvas and upload you Matlab scripts (as m-files). Do not use any other file formats.
- Keep in mind that Canvas discussions are open forums.
- You must acknowledge any collaborations/assistance from colleagues, TAs, instructors etc.

Polynomial root-finding continued

I have shown in class that roots of polynomials can be found, often very robustly, by converting the problem to one of computing eigenvalues of particular matrix. Specifically, in the demo *randpolyroots.m* we saw robust computation of the roots of high-degree polynomials whose coefficients were random variables. This was done using the `roots` command in Matlab, which computes eigenvalues of the *companion matrix* introduced in class.

Your goal in this assignment is to investigate several ways of computing polynomial roots as eigenvalues of different matrices. Specifically, you will compute and plot the roots of the polynomials $p_0, p_1, \ldots$ defined by the recursion

$$p_0(x) = 0, \qquad p_{n+1}(x) = x(p_n(x))^2 + 1, \ n = 0, 1, 2, \ldots.$$

The Matlab script *polycoeff.m* computes the first few of these polynomials in a for loop and the corresponding vectors of coefficients (denoted by `a` in the code).

(a) Using this code, compute the roots of the polynomials $p_0, p_1, \ldots, p_n$ up to some reasonable maximum value of n. As in the demo *randpolyroots.m* plot these roots in 2D, with the real and imaginary parts as the x and y axes respectively.

How robust does this computation appear to be? Discuss. (Hints: (i) it may help to complete part (b) of the assignment before answering this question. (ii) look at the coefficient vectors `a`.)

(b) In part (a), the roots of the polynomials were, as discussed, computed as eigenvalues of companion matrices. However, it transpires that the roots of these particular polynomials can be expressed as eigenvalues of a different sequence of matrices $M_2, M_3, \ldots$. Specifically, $M_2 = [-1]$ is the 1×1 matrix with entry -1, and for $n > 2$ we define

$$M_{n+1} = \begin{bmatrix} M_n & 0_{d_n \times 1} & -c_n r_n \\ -r_n & 0_{1\times 1} & 0_{1\times d_n} \\ 0_{d_n \times d_n} & -c_n & M_n \end{bmatrix}.$$

Here $d_n = 2^{n-1} - 1$, r_n and c_n are the $1 \times d_n$ and $d_n \times 1$ vectors

$$r_n = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \quad c_n = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

and the notation $0_{i\times j}$ means an $i \times j$ array of zeroes, i.e. `zeros(i,j)`. Note that M_n is a $d_n \times d_n$ matrix.

Write a Matlab script that computes these matrices and their eigenvalues (use the `eig` command) and, as in part (a), plots their real and imaginary parts.

Compare your results with those of part (a) and discuss. Which computation appears to be the more robust? If part (b) appears more robust than part (a), what could be the reason?

Bonus! Consider the polynomials $q_0, q_1, \ldots$ defined by the two-step recursion

$$q_0(z) = 0, \quad q_1(z) = 1, \qquad q_{n+1}(z) = zq_n(z)q_{n-1}(z) + 1, \ n = 2, 3, \ldots.$$

It can be shown that the roots of q_n are the eigenvalues of the matrices $M_3, M_4, \ldots$, given by $M_3 = [-1]$ and

$$M_{n+1} = \begin{bmatrix} M_n & 0_{d_n \times 1} & (-1)^{d_{n+1}} c_n r_{n-1} \\ -r_n & 0_{1\times 1} & 0_{1\times d_{n-1}} \\ 0_{d_{n-1}\times d_n} & -c_{n-1} & M_{n-1} \end{bmatrix}.$$

Here r_n and c_n are the $1 \times d_n$ and $d_n \times 1$ vectors

$$r_n = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \quad c_n = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

and d_n is the sequence of integers defined by

$$d_1 = 0, \quad d_2 = 0, \qquad d_{n+1} = d_n + d_{n-1} + 1, \ n = 2, 3, \ldots.$$

For the bonus part of the assignment, redo parts (a) and (b) with these new polynomials, and discuss your results. The bonus is worth **2 additional marks**.