Multiple Roots and Approximate GCDs

In numerical analysis, the computation of a root of high multiplicity is usually regarded as an ill-conditioned problem. However, we may recondition this problem by extracting the multiplicity information first via the computation of greatest common divisors (GCD), following the work of Zhonggang Zeng [3, 4, 5]. Approximate GCDs occur in image processing [2].

1 The Pejorative Manifold

The pejorative manifold was defined by Kahan in 1972. Although multiple roots of a polynomial are naturally ill conditioned, we have: if we know the multiplicities of the roots, then the roots are well conditioned.

Consider a polynomial p with k distinct roots z_i, the ith root has multiplicity m_i, for $i = 1, 2, \ldots, k$. The degree n of p is then the sum of multiplicities: $n = m_1 + m_2 + \cdots + m_k$. We derive the relationship between the roots and coefficients of p as follows:

$$\begin{aligned} p(x) &= (x - z_1)^{m_1}(x - z_2)^{m_2} \cdots (x - z_k)^{m_k} && (1)\\ &= x^n + g_1(z_1, z_2, \ldots, z_k)x^{n-1} + g_2(z_1, z_2, \ldots, z_k)x^{n-2} + \cdots + g_n(z_1, z_2, \ldots, z_k) && (2)\\ &= x^n + a_1x^{n-1} + a_2x^{n-2} + \cdots + a_n. && (3) \end{aligned}$$

The link between the roots z_i, $i = 1, 2, \ldots, k$ and the coefficients a_j is defined by the polynomials $g_j(\mathbf{z})$, $j = 1, 2, \ldots, n$ in the k roots $\mathbf{z} = (z_1, z_2, \ldots, z_k)$. For example,

$$\begin{aligned} p(x) &= (x - z_1)(x - z_2)^2 && (4)\\ &= x^3 + (-z_1 - 2z_2)x^2 + (z_2^2 + 2z_1z_2)x - z_1z_2^2 && (5)\\ &= x^3 + a_1x^2 + a_2x + a_3. && (6) \end{aligned}$$

Identifying the two distinct roots z_1, z_2 with the three coefficients a_1, a_2, and a_3 defines the polynomial system

$$G(\mathbf{z}) = \begin{cases} -z_1 - 2z_2 = a_1 \\ z_2^2 + 2z_1z_2 = a_2 \\ -z_1z_2^2 = a_3. \end{cases} \quad (7)$$

In general, for a coefficient vector $\mathbf{a} \in \mathbb{C}^n$ of a monic polynomial p of degree n with k distinct roots in $\mathbf{z} \in \mathbb{C}^k$ the system $G(\mathbf{z}) = \mathbf{a}$ is a system of n polynomial equations in the k unknown roots of the polynomial p. The system $G(\mathbf{z}) = \mathbf{a}$ was known already in the 16th century by Vieta.

Theorem 1.1 *Denote by J the Jacobian matrix of the Vieta system $G(\mathbf{z}) = \mathbf{a}$. The rank of J is k if and only if the k roots are distinct.*

Given $\mathbf{m} = (m_1, m_2, \ldots, m_k)$, a vector of positive natural numbers, the pejorative manifold $\mathcal{M}$ for $\mathbf{m}$ is

$$\begin{aligned} \mathcal{M} &= \{\ p(x) = (x - z_1)^{m_1}(x - z_2)^{m_2} \cdots (x - z_k)^{m_k} \mid \mathbf{z} \in \mathbb{C}^k,\ z_i \neq z_j, i \neq j\ \} && (8)\\ &= \{\ \mathbf{a} \in \mathbb{C}^n \mid \mathbf{a} = G(\mathbf{z}),\ \mathbf{z} \in \mathbb{C}^k,\ z_i \neq z_j, i \neq j\ \}. && (9) \end{aligned}$$

The system $G(\mathbf{z}) = \mathbf{a}$ defines the manifold $\mathcal{M}$ in $\mathbb{C}^n$ of all polynomials whose roots have a multiplicity structure prescribed by $\mathbf{m}$. The dimension of $\mathcal{M}$ equals k.

Once the multiplicities $\mathbf{m}$ of the roots of p are determined, the roots of p are solutions of a well-conditioned polynomial system given by the Vieta system $G(\mathbf{z}) = \mathbf{a}$. Before we compute the locations of the roots, we will first compute the multiplicities.

Proof of Theorem 1.1. $\Leftarrow$ The rank of the Jacobian matrix of $G(\mathbf{z}) = \mathbf{a}$ equals k if and only if for $c_i \in \mathbb{C}$, $i = 1, 2, \ldots, k$:

$$\begin{bmatrix} \frac{\partial g_1}{\partial z_1} & \frac{\partial g_1}{\partial z_2} & \cdots & \frac{\partial g_1}{\partial z_k} \\ \frac{\partial g_2}{\partial z_1} & \frac{\partial g_2}{\partial z_2} & \cdots & \frac{\partial g_2}{\partial z_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial z_1} & \frac{\partial g_n}{\partial z_2} & \cdots & \frac{\partial g_n}{\partial z_k} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{10}$$

implies that all coefficients $c_i = 0$. We have to show that all $c_i = 0$ if all roots of p are distinct.

We take the columns of the Jacobian matrix and multiply the ith row with x^{n-i}. For the first column this yields the polynomial

$$\begin{aligned} q_1(x) &= \frac{\partial g_1}{\partial z_1}x^{n-1} + \frac{\partial g_2}{\partial z_1}x^{n-2} + \cdots + \frac{\partial g_n}{\partial z_1} && (11) \\ &= \frac{\partial}{\partial z_1}\left(x^n + g_1(\mathbf{z})x^{n-1} + g_2(\mathbf{z})x^{n-2} + \cdots + g_n(\mathbf{z})\right), \text{ as } \frac{\partial}{\partial z_1}x^n = 0 && (12) \\ &= \frac{\partial}{\partial z_1}p(x) && (13) \\ &= \frac{\partial}{\partial z_1}\prod_{i=1}^{k}(x - z_i)^{m_i} && (14) \\ &= -m_1(x - z_1)^{m_1-1}\prod_{i=2}^{k}(x - z_i)^{m_i}. && (15) \end{aligned}$$

All elements of the first column are multiplied with c_1. As we did for the first column, we obtain the polynomials $q_i = \frac{\partial}{\partial z_i}p(x)$ and $Q(x) = c_1q_1 + c_2q_2 + \cdots + c_kq_k \equiv 0$ is the zero polynomial, or more explicitly:

$$\begin{aligned} Q(x) &= \sum_{j=1}^{k} c_j \left((-m_j)(x - z_j)^{m_j-1} \prod_{\substack{i=1 \\ i\neq j}}^{k}(x - z_i)^{m_i} \right) && (16) \\ &= -\left(\prod_{i=1}^{k}(x - z_i)^{m_i-1}\right)\left(\sum_{j=1}^{k} c_j m_j \prod_{\substack{i=1 \\ i\neq j}}^{k}(x - z_i)\right). && (17) \end{aligned}$$

For $Q(x) \equiv 0$, the second factor in $Q(x)$ must be zero. But, if all roots are distinct, then $\prod_{\substack{i=1 \\ i\neq j}}^{k}(z_j - z_i) \neq 0$, and $m_j > 0$. Therefore we must have $c_j = 0$ and the Jacobian matrix has rank k.

$\Rightarrow$ We proceed by contraposition and show that if not all roots are distinct, then the Jacobian matrix of $G(\mathbf{z}) = \mathbf{a}$ has rank less than k. Without loss of generality, we assume that $z_1 = z_2$, then the first two columns of the Jacobian matrix define the polynomials

$$q_1(x) = -m_1(x - z_1)^{m_1-1}(x - z_2)^{m_2}\prod_{i=3}^{k}(x - z_i)^{m_i} \tag{18}$$

and

$$q_2(x) = -m_2(x - z_1)^{m_1}(x - z_2)^{m_2-1}\prod_{i=3}^{k}(x - z_i)^{m_i} \tag{19}$$

respectively. If $z_1 = z_2$, then q_1 and q_2 differ by the multiples m_1 and m_2. Therefore, the first two columns of the Jacobian matrix are linearly dependent and the rank is less than k. $\square$

2 Blind Image Deconvolution

The greatest common divisor (abbreviated by GCD) appears in many application areas. The use of GCD to filter noise from images appears in [2]. The pictures below were captured from [2]:

(a) Blurred Image (Moving Truck on BQE; Early April 97)

(b) Image Magnitude After GCD Processing

Assume multiple versions of the same scene are available obtained via a Z-transform on approximate numbers:

$$\begin{aligned} F_1(z_1, z_2) &= P(z_1, z_2) D_1(z_1, z_2) \\ F_2(z_1, z_2) &= P(z_1, z_2) D_2(z_1, z_2) \end{aligned} \tag{20}$$

The F_1 and F_2 are the observed data. P is the original picture, blurred by D_1 and D_2 respectively. If $\text{GCD}(D_1, D_2) = 1$, then $P = \text{GCD}(F_1, F_2)$.

The image problem concerns GCDs in two variables. But we may reduce this question to a univariate problem by fixing one of the variables to a random constant.

3 Gauss-Newton

Newton's method can be applied to systems with more equations than unknowns. For the system $G(\mathbf{z})-\mathbf{a}=\mathbf{0}$, where J denotes the Jacobian of G, the formula to produce a sequence of approximations $\mathbf{z}^{(i)}$ is

$$\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} - J^+(\mathbf{z}^{(i)}) \left(G(\mathbf{z}^{(i)}) - \mathbf{a} \right), \quad i = 0, 1, \ldots. \tag{21}$$

The J^+ denotes the Moore-Penrose inverse of the matrix J. We define J^+ via the singular value decomposition: $J = U\Sigma V^H$. U and V are orthogonal: $U^HU = I$ and $V^HV = I$, where ^{H} denotes the Hermitian transpose and I is the identity, so $U^{-1} = U^H$. Σ is diagonal. On the diagonal of Σ are the singular values, sorted in decreasing order. If the rank of J is r, then only the first r singular values are nonzero. The Moore-Penrose inverse Σ^+ is obtained by inverting the nonzero diagonal elements. Then $J^+ = V\Sigma^+U^H$.

Computing a singular value decomposition can be computationally expensive. Moreover, as in the usual Newton's method, we do not compute J^{-1}, but instead we rewrite (21) into

$$\underbrace{J(\mathbf{z}^{(i)})}_{Q \cdot R} \underbrace{\left(\mathbf{z}^{(i+1)} - \mathbf{z}^{(i)} \right)}_{\Delta \mathbf{z}} = \underbrace{\mathbf{a} - G(\mathbf{z}^{(i)})}_{\mathbf{b}}, \tag{22}$$

where $Q^HQ = I$ and R is an upper triangular matrix so that $Q \cdot R = J$. The solution $\Delta\mathbf{z}$ is then obtained as the solution of the upper triangular system $R\Delta\mathbf{z} = Q^H\mathbf{b}$. As this solution $\Delta\mathbf{z}$ minimizes the squares of the residual $||\mathbf{b} - J\Delta\mathbf{z}||_2^2$, it is called the least squares approximation to the solution of the linear system.

We will apply Gauss-Newton to the following problem. Given two polynomials $p, q \in \mathbb{C}[x]$. We want to compute their GCD: $u = \text{GCD}(p, q)$ and their cofactors v and w: $uv = p$ and $uw = p$, where $\text{GCD}(v, w) = 1$.

Consider for example $p = p_0x^3 + p_1x^2 + p_2x + p_3$ and $q = q_0x^4 + q_1x^3 + q_2x^2 + q_3x + q_4$. For $u = \text{GCD}(p, q)$, suppose $\deg(u) = 2$. So we want to compute the three coefficients of $u = u_0x^2 + u_1x + u_2$ and the coefficients of the cofactors. Because $uv = p$, $\deg(v) = 1$, so $v = v_0x + v_1$ and $uw = q$ implies $\deg(w) = 2$, so $w = w_0x^2 + w_1x + w_2$. Thus we have 8 unknowns and we identify the polynomials p, q, u, v, and w by their respective coefficient vectors $\mathbf{p}$, $\mathbf{q}$, $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$. As u is only determined up to a constant factor, we choose a random 3-vector $\mathbf{r} = [r_0\ r_1\ r_2]$ to scale the coefficients of u. The nonlinear system we have to solve is

$$F(\mathbf{u},\mathbf{v},\mathbf{w}) = \begin{cases} [r_0\ r_1\ r_2]^H \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} = 1 \\ \begin{bmatrix} v_0 & & \\ v_1 & v_0 & \\ & v_1 & v_0 \\ & & v_1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \\ \begin{bmatrix} w_0 & & \\ w_1 & w_0 & \\ w_2 & w_1 & w_0 \\ & w_2 & w_1 \\ & & w_2 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \end{cases}. \tag{23}$$

The matrices in the coefficients of $\mathbf{v}$ and $\mathbf{w}$ are convolution matrices arising from the multiplication with $\mathbf{u}$. Abbreviating these two convolution matrices respectively by $C_2(\mathbf{v})$ and $C_2(\mathbf{w})$, the system (23) then becomes

$$F(\mathbf{u},\mathbf{v},\mathbf{w}) = \begin{bmatrix} \mathbf{r}^H\mathbf{u} \\ C_2(\mathbf{v})\mathbf{u} \\ C_2(\mathbf{w})\mathbf{u} \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{p} \\ \mathbf{q} \end{bmatrix}. \tag{24}$$

Denoting the right hand side vector of (24) by $\mathbf{b}$, we have to solve $F(\mathbf{u},\mathbf{v},\mathbf{w}) = \mathbf{b}$. Those $(\mathbf{u},\mathbf{v},\mathbf{w})$ for which the residual $||\mathbf{b} - F(\mathbf{u},\mathbf{v},\mathbf{w})||_2$ is smaller than some ϵ defines the ϵ-GCD of the polynomials p and q, computed as the least squares approximation for the solution of $F(\mathbf{u},\mathbf{v},\mathbf{w}) = \mathbf{b}$.

4 Computing the Multiplicities

We explain how the GCD solves the problem of identifying the multiplicities of the roots of a univariate polynomial p. Recall that $\text{GCD}(p, p')$ gives a condition on the coefficients of p for p to have a multiple root. We will use the successive application of the GCD to lower the multiplicity.

For a given polynomial p, Consider the sequence

$$u_0 = p, \quad u_{i+1} = \text{GCD}(u_i, u_i'), \quad i = 0, 1, \ldots. \tag{25}$$

For the polynomial $p(x) = (x-1)^5(x-2)^3(x-3)$, the sequence (25) is

$$\begin{array}{rcccccc} u_0 & = & [(x-1)^4(x-2)^2] & (x-1) & (x-2) & (x-3) \\ u_1 & = & [(x-1)^3(x-2)] & (x-1) & (x-2) & \\ u_2 & = & [(x-1)^2] & (x-1) & (x-2) & \\ u_3 & = & [(x-1)] & (x-1) & & \\ u_4 & = & [1] & (x-1) & & \\ & & \text{GCD} & 5 & 3 & 1 \end{array} \tag{26}$$

We see that the sequence reveals the multiplicities of the three roots.

For the multiplicities, we need to compute a sequence of GCDs (25). To compute the GCD of f and f' we consider

$$\left\{ \begin{array}{rcll} u(x)v(x) & = & f(x) & u = \text{GCD}(f, f'), u \text{ is monic,} \\ u(x)w(x) & = & f'(x) & \text{GCD}(v, w) = 1, v, w \text{ are relative prime.} \end{array} \right. \tag{27}$$

There are four steps:

1. Find the degree of u. For $\deg(f) = n$, $u = \text{GCD}(f, f')$:

   $$\deg(u) = n - k \Leftrightarrow k\text{-th Sylvester discriminant matrix is the first rank deficient matrix.} \tag{28}$$

   The k-th Sylvester discriminant matrix for a polynomial f of degree n is defined as

   $$S_k(f) = [\ C_k(f') \mid C_{k-1}(f)\ ] \in \mathbb{C}^{(n+k)\times(2k+1)}. \tag{29}$$

2. Reformulate (27) using the convolution operator conv:

   $$\begin{pmatrix} u_0 \\ \text{conv}(\mathbf{u}, \mathbf{v}) \\ \text{conv}(\mathbf{u}, \mathbf{w}) \end{pmatrix} = \begin{pmatrix} 1 \\ \mathbf{f} \\ \mathbf{f}' \end{pmatrix} \quad \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{pmatrix} \in \mathbb{C}^{m+1} \times \mathbb{C}^{k+1} \times \mathbb{C}^k. \tag{30}$$

3. Find an initial approximation for $(\mathbf{u}, \mathbf{v}, \mathbf{w})$.

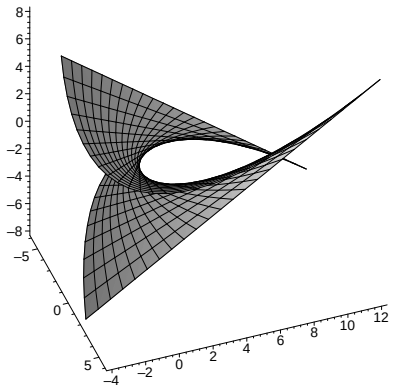
4. Apply Gauss-Newton to refine the approximations for $(\mathbf{u}, \mathbf{v}, \mathbf{w})$.

The determination of the rank of a matrix is done via QR decomposition in rank-revealing algorithms [1].

5 The Pejorative Condition Number

For the system in (7), the pejorative manifold is a surface in 3-space. The natural parameters of this surface are the roots z_1 and z_2. The sequence of Maple commands below produces the plot:

```
> p := (x-z[1])*(x-z[2])^2;
> g[1] := coeff(p,x,2);
> g[2] := coeff(p,x,1);
> g[3] := coeff(p,x,0);
> plot3d([g[1],g[2],g[3]],z[1]=-2..2,z[2]=-2..2,orientation=[-20,50],
    axes=framed,scaling=constrained);
```



The condition number of the Gauss-Newton method is given by the inverse of the smallest singular value $\sigma_{\min}$. We have the following relation:

$$||\Delta \mathbf{z}||_2 \leq \frac{1}{\sigma_{\min}} ||\Delta \mathbf{a}||_2 \tag{31}$$

between the exact $\mathbf{a} = G(\mathbf{z})$ and the perturbed zeros on the pejorative manifold: $\mathbf{a} + \Delta\mathbf{a} = G(\mathbf{z} + \Delta\mathbf{z})$. The number $\frac{1}{\sigma_{\min}}$ is the pejorative condition number. Denote $\widetilde{\mathbf{a}} = \mathbf{a} + \Delta\mathbf{a}$ and $\widetilde{\mathbf{z}} = \mathbf{z} + \Delta\mathbf{z}$.

In practice we do not have the exact coefficients $\mathbf{a}$ of the polynomial p, but we have to work with the approximate coefficient vector $\bar{\mathbf{a}}$. Then $\widetilde{\mathbf{a}}$ is obtained as the orthogonal projection of the given coefficient vector $\bar{\mathbf{a}}$ onto the pejorative manifold and $\widetilde{\mathbf{z}}$ is defined via $\widetilde{\mathbf{a}} = G(\widetilde{\mathbf{z}})$.

In that case, the approximate roots are in the vector $\bar{\mathbf{z}}$. We can derive the following bound on the error, starting from (31):

$$\begin{aligned} ||\mathbf{z} - \widetilde{\mathbf{z}}||_2 &\leq \frac{1}{\sigma_{\min}} ||\mathbf{a} - \widetilde{\mathbf{a}}||_2 && (32) \\ &\leq \frac{1}{\sigma_{\min}} ||(\mathbf{a} - \bar{\mathbf{a}}) + (\bar{\mathbf{a}} - \widetilde{\mathbf{a}})||_2 && (33) \\ &\leq \frac{1}{\sigma_{\min}} ||\mathbf{a} - \bar{\mathbf{a}}||_2 + ||\bar{\mathbf{a}} - \widetilde{\mathbf{a}}||_2 && (34) \end{aligned}$$

Since $\widetilde{\mathbf{a}}$ is the projection of $\bar{\mathbf{a}}$ on the manifold, we have $||\bar{\mathbf{a}} - \widetilde{\mathbf{a}}||_2 \leq ||\mathbf{a} - \bar{\mathbf{a}}||_2$, so we obtain:

$$||\mathbf{z} - \widetilde{\mathbf{z}}||_2 \leq \frac{2}{\sigma_{\min}} ||\mathbf{a} - \bar{\mathbf{a}}||_2. \tag{35}$$

We see that the error on the coefficients, multiplied by the pejorative condition number gives a bound on the error of the roots.

6 Using SNAP

The package SNAP in Maple provides the procedure `EpsilonGCD`, illustrated below (using Maple 14). We first prepare the input, generating at random a quadratic greatest common divisor `u` and two cofactors `p` and `q`, to form the input polynomials `f` = `up` and `g` = `uq`.

```
[> c := () -> RandomTools[Generate](float(range = -1..1)):
[> u := randpoly(x,coeffs=c,degree=2);
                              2
-0.438850937 + 0.418130698 x  - 0.742863086 x
```

```
[> p := randpoly(x,coeffs=c,degree=3);
                              3                2
0.431521350 + 0.454595796 x  - 0.775435781 x + 0.050666015 x
[> q := randpoly(x,coeffs=c,degree=4);
                              4                3                 2
0.190905574 - 0.323567403 x  + 0.581067165 x  + 0.797189083 x - 0.196140740 x
[> f := expand(p*u); g := expand(q*u);
```

Then we proceed with the computation of an ϵ-GCD:

```
[> d,e := SNAP[EpsilonGCD](f,g,x);
                                  2
HFloat(0.06250000000000001) x - HFloat(0.11103930676352133) x - HFloat(0.06559715343150478),
HFloat(6.389318726591376e-8)

[> sort(d/coeff(d,x,degree(d)));
              2
HFloat(1.0) x  - HFloat(1.776628908216341) x - HFloat(1.0495544549040763)
[> sort(u/coeff(u,x,degree(u)));
              2
1.000000000 x  - 1.776628909 x - 1.049554455
```

The default precision for Maple floats is 10 digits, but the `EpsilonGCD` works with hardware floats.

7 Three Strikes Principle

Zhonggang Zeng [6] formulates a three-strikes principle for formulating an approximate solution to an ill-posed problem:

(1) backward nearness: the approximate solution is the exact solution of a nearby problem.

(2) maximum codimension: the approximate solution is the exact solution of a problem on the nearby pejorative manifold of the highest codimension.

(3) minimum distance: the approximate solution is the exact solution of the nearest problem on the nearby pejorative manifold of the highest codimension.

Finding an approximate solution is then (likely) a well-posed problem. An approximate solution is a generalization of an exact solution.

8 Exercises

1. Use Maple or Sage to implement the sequence (25). Show the outcome on $p(x) = (x-1)^4(x-2)^8(x-3)^7$.
2. Read the documentation of MATLAB on the \ operator. Generate a random 3-by-2 matrix A and let $\mathbf{x}$ be a vector of ones. Then $\mathbf{b} = A\mathbf{x}$ defines a overdetermined linear system. Solve $A\mathbf{x} = \mathbf{b}$ via MATLAB (or Octave).

   If you implemented Newton's method in MATLAB or Octave – one of the exercises of Lecture 1 – will your implementation also work for overdetermined systems?
3. An alternative definition of the Moore-Penrose inverse, uses the normal equations, i.e.: $J^+ = \left[J^H J\right]^{-1} J^H$. Verify that $J^+ J = I$.

4. Use Maple or any other computer algebra package to write the algorithm to compute the GCD sequence to compute the multiplicities. In your implementation, you may assume that the coefficients of the polynomials are exact. Illustrate your code with some good examples.

5. Consider the polynomial $p(x) = (x-t)(x-1)(x-2)^2$, with the parameter $t \in [0,1]$. Make a plot of the pejorative condition number as t goes from 0 to 1.

6. Download the software from [4] or [6] and compute several examples with it.
   Does the MATLAB software in [4] run with Octave?

References

[1] T.Y. Li and Z. Zeng. A rank-revealing method with updating, downdating and applications. *SIAM J. Matrix Anal. Appl.*, 26(4):918–946, 2005.

[2] S.U. Pillai and B. Liang. Blind image deconvolution using a robust GCD approach. *IEEE Transactions on Image Processing*, 8(2):295–301, 1999.

[3] Z. Zeng. A method for computing multiple roots of inexact polynomials. In J.R. Sendra, editor, *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation (ISSAC 2003)*, pages 266–272. ACM, 2003.

[4] Z. Zeng. Algorithm 835: MultRoot – a Matlab package for computing polynomial roots and multiplicities. *ACM Transactions on Mathematical Software*, 30(2):218–236, 2004.

[5] Z. Zeng. Computing multiple roots of inexact polynomials. *Mathematics of Computation*, 74(250):869–903, 2005.

[6] Z. Zeng. ApaTools: a software toolbox for approximate polynomial algebra. In M.E. Stillman, N. Takayama, and J. Verschelde, editors, *Software for Algebraic Geometry*, volume 148 of *The IMA Volumes in Mathematics and Its Applications*, pages 149–167. Springer-Verlag, 2008.