Multiple Roots & Approximate GCDs

1. The Pejorative Manifold
   - multiple roots with known multiplicities
   - distinct roots and the rank of the Vieta system
2. Blind Image Deconvolution
   - deblurring images
3. Gauss-Newton
   - Newton's method for overdetermined systems
4. Computing the Multiplicities
   - a succession of GCD computations to lower multiplities
   - the pejorative condition number

MCS 563 Lecture 22
Analytic Symbolic Computation
Jan Verschelde, 5 March 2014

Multiple Roots & Approximate GCDs

1. The Pejorative Manifold
   - multiple roots with known multiplicities
   - distinct roots and the rank of the Vieta system
2. Blind Image Deconvolution
   - deblurring images
3. Gauss-Newton
   - Newton's method for overdetermined systems
4. Computing the Multiplicities
   - a succession of GCD computations to lower multiplities
   - the pejorative condition number

multiple roots

Let $p \in \mathbb{C}[x]$, in one variable x, with multiple roots.

Although solving $p(x) = 0$ is numerically ill conditioned,
if we know the multiplicities, the problem is well conditioned.

Suppose p has k distinct roots z_i,
the ith root has multiplicity m_i, for $i = 1, 2, \ldots, k$, and:

$$\deg(p) = n = m_1 + m_2 + \cdots + m_k.$$

Roots and coefficients are related as follows:

$$\begin{aligned} p(x) &= (x - z_1)^{m_1}(x - z_2)^{m_2} \cdots (x - z_k)^{m_k} \\ &= x^n + g_1(z_1, z_2, \ldots, z_k)x^{n-1} + g_2(z_1, z_2, \ldots, z_k)x^{n-2} \\ &\quad + \cdots + g_n(z_1, z_2, \ldots, z_k) \\ &= x^n + a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_n. \end{aligned}$$

an example

Consider $z_1, z_2 \in \mathbb{C}$ with multiplicities 1 and 2:

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

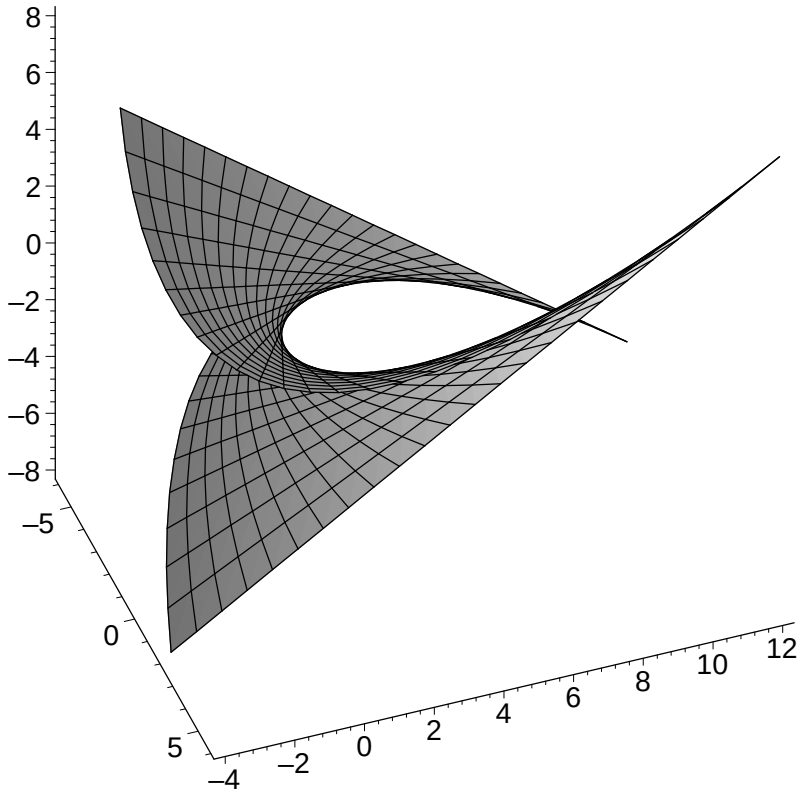
Identifying the two distinct roots z_1, z_2 with the three coefficients a_1, a_2, and a_3 defines the polynomial system

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

$G(\mathbf{z}) = \mathbf{a}$ is the system of Vieta relating the k distinct roots in $\mathbf{z} = (z_1, z_2, \ldots, z_k)$ to the n coefficients $\mathbf{a} = (a_1, a_2, \ldots, a_n)$.

a surface

```
> 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 pejorative manifold

Theorem

Denote by J the Jacobian matrix of the Vieta system $G(\mathbf{z}) = \mathbf{a}$. The rank of J is $k \Leftrightarrow$ 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

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

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.

Multiple Roots & Approximate GCDs

1. The Pejorative Manifold
   - multiple roots with known multiplicities
   - distinct roots and the rank of the Vieta system
2. Blind Image Deconvolution
   - deblurring images
3. Gauss-Newton
   - Newton's method for overdetermined systems
4. Computing the Multiplicities
   - a succession of GCD computations to lower multiplities
   - the pejorative condition number

proving the theorem

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

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

implies that all coefficients $c_i = 0$.

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

columns define polynomials

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

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

distinct roots imply full rank

All elements of the first column are multiplied with c_1.
For all columns, we obtain: $q_i = \frac{\partial}{\partial z_i} p(x)$
and $Q(x) = c_1 q_1 + c_2 q_2 + \cdots + c_k q_k \equiv 0$, or:

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

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 rank is k.

the $\Rightarrow$ part

By contraposition: if not all roots are distinct, then the Jacobian matrix of $G(\mathbf{z}) = \mathbf{a}$ has rank less than k.
Assume $z_1 = z_2$, then the first two columns of the Jacobian matrix define

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

and

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

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

Multiple Roots & Approximate GCDs

1. The Pejorative Manifold
   - multiple roots with known multiplicities
   - distinct roots and the rank of the Vieta system
2. Blind Image Deconvolution
   - deblurring images
3. Gauss-Newton
   - Newton's method for overdetermined systems
4. Computing the Multiplicities
   - a succession of GCD computations to lower multiplities
   - the pejorative condition number

blind image deconvolution

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

(b) Image Magnitude After GCD Processing

deblurring images

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

$$F_1(z_1, z_2) = P(z_1, z_2) D_1(z_1, z_2)$$
$$F_2(z_1, z_2) = P(z_1, z_2) D_2(z_1, z_2)$$

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

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.

Multiple Roots & Approximate GCDs

1. The Pejorative Manifold
   - multiple roots with known multiplicities
   - distinct roots and the rank of the Vieta system
2. Blind Image Deconvolution
   - deblurring images
3. Gauss-Newton
   - Newton's method for overdetermined systems
4. Computing the Multiplicities
   - a succession of GCD computations to lower multiplities
   - the pejorative condition number

Gauss-Newton with SVD

The system $G(\mathbf{z}) - \mathbf{a} = \mathbf{0}$ has more equations than unknowns. Let J denote the Jacobian matrix of G.
The method of Gauss-Newton produces a sequence $\mathbf{z}^{(i)}$ by

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

The J^{+} denotes the Moore-Penrose inverse of the matrix J.
We define J^{+} via the singular value decomposition (SVD):

$$J = U \Sigma V^{H}, \quad U^{H} U = I, V^{H} V = I,$$

and with singular values σ_i, $i = 1, 2, \ldots, n$:

$$\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n), \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n.$$

The Moore-Penrose inverse Σ^{+} is obtained by inverting the nonzero diagonal elements. Then $J^{+} = V \Sigma^{+} U^{H}$.

Gauss-Newton with QR

Because computing J^+ is expensive, instead of

$$\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} - J^+(\mathbf{z}^{(i)}) \left(G(\mathbf{z}^{(i)}) - \mathbf{a} \right),$$

we compute

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

where $Q^H Q = I$ and R is upper triangular 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 $\Delta \mathbf{z}$ minimizes the squares of the residual $||\mathbf{b} - J \Delta \mathbf{z}||_2^2$,
we call $\Delta \mathbf{z}$ the least squares approximation.

an example

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 $u = \mathrm{GCD}(p, q)$, suppose $\deg(u) = 2$.

We want to compute $u = u_0x^2 + u_1x + u_2$ and the coefficients of the cofactors v and w: $uv = p$, $uw = q$.

Because $uv = p$, $\deg(v) = 1$, so $v = v_0x + v_1$
and $uw = q$ implies $\deg(w) = 2$, so $w = w_0x^2 + w_1x + w_2$.

We identify the polynomials p, q, u, v, and w by their respective coefficient vectors **p**, **q**, **u**, **v**, and **w**.
In total we have 8 unknown coefficients.

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.

a nonlinear system

We solve

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

convolution matrices

The matrices in the coefficients of **v** and **w** are convolution matrices arising from the multiplication with **u**:

$$C_2(\mathbf{v}) = \begin{bmatrix} v_0 & & \\ v_1 & v_0 & \\ & v_1 & v_0 \\ & & v_1 \end{bmatrix} \quad C_2(\mathbf{w}) = \begin{bmatrix} w_0 & & \\ w_1 & w_0 & \\ w_2 & w_1 & w_0 \\ & w_2 & w_1 \\ & & w_2 \end{bmatrix}$$

Then in abbreviated form, the system becomes

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

Multiple Roots & Approximate GCDs

1. The Pejorative Manifold
   - multiple roots with known multiplicities
   - distinct roots and the rank of the Vieta system
2. Blind Image Deconvolution
   - deblurring images
3. Gauss-Newton
   - Newton's method for overdetermined systems
4. Computing the Multiplicities
   - a succession of GCD computations to lower multiplities
   - the pejorative condition number

computing the multiplicities

$\mathrm{GCD}(p, p')$ gives a condition on the coefficients of p for p to have a multiple root.
For a given polynomial p, Consider the sequence

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

For the polynomial $p(x) = (x-1)^5(x-2)^3(x-3)$, we compute:

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

The sequence reveals the multiplicities of the three roots.

GCD computations

For the multiplicities, we need to compute a sequence of GCDs of a polynomial and its derivative.

To compute the GCD of f and f' we consider

$$\begin{cases} u(x)v(x) &= f(x) \\ u(x)w(x) &= f'(x) \end{cases}$$

$u = \mathrm{GCD}(f, f'), u$ is monic,
$\mathrm{GCD}(v, w) = 1, v, w$
are relative prime.

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

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

Four Steps

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

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

2. Using the convolution operator conv:

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

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

Multiple Roots & Approximate GCDs

1. The Pejorative Manifold
   - multiple roots with known multiplicities
   - distinct roots and the rank of the Vieta system
2. Blind Image Deconvolution
   - deblurring images
3. Gauss-Newton
   - Newton's method for overdetermined systems
4. Computing the Multiplicities
   - a succession of GCD computations to lower multiplities
   - the pejorative condition number

the pejorative condition number

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:

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

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.

a bound on the error

Denote $\widetilde{\mathbf{a}} = \mathbf{a} + \Delta\mathbf{a}$ and $\widetilde{\mathbf{z}} = \mathbf{z} + \Delta\mathbf{z}$. We work with approximate coefficients $\bar{\mathbf{a}}$ and approximate roots are in $\bar{\mathbf{z}}$.

Then $\widetilde{\mathbf{a}}$ is the orthogonal projection of $\bar{\mathbf{a}}$ onto the pejorative manifold and $\widetilde{\mathbf{z}}$ is defined via $\widetilde{\mathbf{a}} = G(\widetilde{\mathbf{z}})$. Bounding the error:

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

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:

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

Maple experiments

Preparing the input, u = GCD(f,g):

```
[> 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
0.190905574 - 0.323567403 x  + 0.581067165 x
                   2
 + 0.797189083 x - 0.196140740 x
[> f := expand(p*u); g := expand(q*u);
```

computing an ϵ-GCD

```
[> d,e := SNAP[EpsilonGCD](f,g,x);
                                    2
HFloat(0.06250000000000001) x
- HFloat(0.1110393067635213) 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(1.049554549040763)
[> 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.

Summary + Exercises

Knowing multiplicities, computing multiple roots is well conditioned. We compute the multiplicities via GCD sequences with QR.

Exercises:

1. Use Maple or Sage to implement the GCD sequence. Show the outcome on $p(x) = (x-1)^4(x-2)^8(x-3)^7$.
2. Read the documentation of MATLAB on the $\backslash$ 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?

more exercises

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 Algorithm 835 of ACM TOMS or ApaTools and compute several examples with it. Does the MATLAB code run on Octave?