POLYNOMIAL EQUATIONS IN GEOMETRIC MODELLING, THE CONTROL OF UNWANTED VIBRATIONS IN MANUFACTURING, AND THE DEBLURRING OF IMAGES

Joab Winkler
Department of Computer Science
The University of Sheffield
Sheffield
United Kingdom

j.winkler@dcs.shef.ac.uk
Contents

1 Applications
2 MATLAB
3 The computation of multiple roots
4 Image deblurring
5 Summary
Computer-aided geometric design:

Tangential intersections between curves and surfaces that are defined by polynomials are computed by solving a polynomial equation:

- These intersections are defined by *multiple roots* of the polynomial

Tangential intersections are required for:

- Aesthetic appeal
- Ease of handling of an object (the avoidance of sharp edges and corners)
- The formation of a blend between two surfaces
Applications

- **Manufacturing:**
  The vibrations of a cutter may excite one or more resonant frequencies of an object being machined, in which case the machining operation must be terminated.
  - Resonant frequencies of components must be calculated in order to design filters that damp out vibrations that may arise during their manufacture.

Figure: Two double eigenvalues (one at the top, and one at the bottom) of a component of a jet engine, and their frequency response functions
Difficulties of performing reliable computations on polynomials

Commonly used algorithms for computing the roots of a polynomial include:
- Bairstow, Graeffe, Jenkins-Traub, Laguerre, Müller and Newton

These methods yield satisfactory results if:
- The polynomial is of moderate degree
- The roots are simple and well-separated
- A good starting point in the iterative scheme is used
Example 1 Consider the polynomial

\[ y^4 - 4y^3 + 6y^2 - 4y + 1 = (y - 1)^4 \]

whose root is \( y = 1 \) with multiplicity 4. MATLAB returns the roots

1.0002, 1.0000 + 0.0002i, 1.0000 - 0.0002i, 0.9998

Example 2 The roots of the polynomial \((y - 1)^{100}\) were computed by MATLAB.

Figure: The computed roots of \((y - 1)^{100}\).
Example 3 The computation of multiple roots in the presence of noise.

Figure: The distribution of the roots of four perturbed polynomials.
Example 4

\[ \varepsilon_c = \frac{\text{componentwise noise amplitude}}{\text{componentwise signal amplitude}} = 10^{-8} \]

<table>
<thead>
<tr>
<th>multiplicity</th>
<th>exact root</th>
<th>computed root</th>
<th>relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-6.75470000000e-1</td>
<td>-6.7547000082e-1</td>
<td>1.2139725913e-9</td>
</tr>
<tr>
<td>6</td>
<td>5.73350000000e+0</td>
<td>5.7335000822e+0</td>
<td>1.4344694923e-8</td>
</tr>
<tr>
<td>7</td>
<td>2.17470000000e+0</td>
<td>2.17469999237e+0</td>
<td>3.5069931355e-9</td>
</tr>
<tr>
<td>10</td>
<td>-9.55680000000e+0</td>
<td>-9.556796740e+0</td>
<td>3.4111255034e-8</td>
</tr>
<tr>
<td>11</td>
<td>-6.55530000000e+0</td>
<td>-6.5553001701e+0</td>
<td>2.5954947075e-8</td>
</tr>
</tbody>
</table>

- The multiplicities of the exact roots are preserved, and the (small) errors are in the values of the roots.
Example 5

\[ \varepsilon_c = \frac{\text{componentwise noise amplitude}}{\text{componentwise signal amplitude}} = 10^{-8} \]

<table>
<thead>
<tr>
<th>multiplicity</th>
<th>exact root</th>
<th>computed root</th>
<th>relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-1.15390000000e+0</td>
<td>-1.1539000316e+0</td>
<td>2.7389266674e-8</td>
</tr>
<tr>
<td>5</td>
<td>4.08090000000e+0</td>
<td>4.0808998890e+0</td>
<td>2.7198581384e-8</td>
</tr>
<tr>
<td>6</td>
<td>-2.10590000000e+0</td>
<td>-2.1058999294e+0</td>
<td>3.2521823798e-8</td>
</tr>
<tr>
<td>7</td>
<td>3.66830000000e+0</td>
<td>3.6683000481e+0</td>
<td>1.3110114060e-8</td>
</tr>
<tr>
<td>13</td>
<td>-9.60840000000e+0</td>
<td>-9.6084001121e+0</td>
<td>1.1664214687e-8</td>
</tr>
</tbody>
</table>

The multiplicities of the exact roots are preserved, and the (small) errors are in the values of the roots.
MATLAB

MATLAB was used exclusively for the work in this presentation for several reasons:

- **Familiarity**
- It is used in many universities for pure and applied research
- Excellent references, including books with many examples from mathematics and engineering, and links on the internet
- Established record of high quality software, and continual updates and improvements
- An extensive library of functions in all branches of applied mathematics
- The support of a well-established company in scientific software development
- The pricing policy - users need only purchase the toolboxes they require
The computation of multiple roots

Why are the multiplicities of the multiple roots of the exact polynomial preserved in the computed roots of an inexact form of the given polynomial?

- Most polynomial root solvers use Newton’s iteration, or a variant of it, and successive estimates of the root $y_j$ of the polynomial $f(y)$ are given by

$$y_j^{(i+1)} = y_j^{(i)} - \frac{f\left(y_j^{(i)}\right)}{f'(y_j^{(i)})}, \quad i = 0, 1, \ldots,$$

which causes major problems at and near multiple roots of $f(y)$.

- This iteration is applied to all polynomials, without a prior test to determine if $f(y)$ has multiple roots.
  - Should the multiplicities of the roots be calculated before the values of the roots are computed?
A polynomial root solver that is explicitly designed to compute multiple roots has two stages:

Stage 1: Compute the multiplicities of the roots
Stage 2: Use the multiplicities of the roots to compute the values of the roots

But how can the multiplicities of the roots be computed?


The algorithm has numerous greatest common divisor (GCD) computations and deconvolutions, and structure-preserving matrix methods are therefore used extensively.
The geometry of multiple roots

How does a polynomial that has one or more of whose distinct roots differ from a polynomial that only has simple roots?

Let \( f(y) \) have \( p \) distinct roots \( y_i, i = 1, \ldots, p \), and let the multiplicity of the \( i \)th root be \( m_i \):

\[
f(y) = \sum_{i=0}^{m} a_i y^{m-i} = \prod_{i=1}^{p} (y - y_i)^{m_i}, \quad a_0 = 1, \quad \sum_{i=1}^{p} m_i = m
\]

- The roots \( y_i \) introduce \( \sum_{i=1}^{p} (m_i - 1) = m - p \) constraints on the coefficients of \( f(y) \).
- A monic polynomial of degree \( m \) has \( m \) degrees of freedom.
- The polynomial \( f(y) \) therefore lies on a manifold of dimension \( m - (m - p) = p \) in a space of dimension \( m \).
- This manifold is called a pejorative manifold because polynomials near this manifold are ill-conditioned.
Example of a pejorative manifold

Example 6: Consider a quartic polynomial $f(y)$ with real roots $y_1, y_2, y_3$ and $y_4$:

$$f(y) = y^4 - (y_1 + y_2 + y_3 + y_4)y^3 + (y_1y_2 + y_1y_3 + y_1y_4 + y_2y_3 + y_2y_4 + y_3y_4)y^2 - (y_1y_2y_3 + y_1y_2y_4 + y_1y_3y_4 + y_2y_3y_4)y + y_1y_2y_3y_4$$

- If $f(y)$ has a quartic root, then $y_1 = y_2 = y_3 = y_4$, then

$$f(y) = y^4 - 4y_1y^3 + 6y_1^2y^2 - 4y_1^3y + y_1^4$$

The pejorative manifold of this quartic polynomial is

$$\left( -4y_1, 6y_1^2, -4y_1^3, y_1^4 \right)$$
Example of a pejorative manifold cont’d

- If \( f(y) \) has one treble root and one simple root, then \( y_1 = y_2 = y_3 \neq y_4 \), and

\[
f(y) = y^4 - (3y_1 + y_4)y^3 + 3(y_1^2 + y_1y_4)y^2 - (y_1^3 + 3y_1^2y_4)y + y_1^3y_4
\]

The pejorative manifold of this quartic polynomial is

\[
\begin{pmatrix}
-(3y_1 + y_4) & 3(y_1^2 + y_1y_4) & -(y_1^3 + 3y_1^2y_4) & y_1^3y_4
\end{pmatrix}
\]

- If \( f(y) \) has two double roots, then \( y_1 = y_2 \) and \( y_3 = y_4 \neq y_1 \), and

\[
f(y) = y^4 - 2(y_1 + y_4)y^3 + (y_1^2 + 4y_1y_4 + y_4^2)y^2
- 2(y_1^2y_4 + y_1y_4^2)y + y_1^2y_4^2
\]

The pejorative manifold of this quartic polynomial is

\[
\begin{pmatrix}
-2(y_1 + y_4) & (y_1^2 + 4y_1y_4 + y_4^2) & -2(y_1^2y_4 + y_1y_4^2) & y_1^2y_4^2
\end{pmatrix}
\]
Example of a pejorative manifold cont’d

Figure: The pejorative manifolds of a cubic polynomial that has (a) a real cubic root, and (b) a real double root and a real simple root.
The computation of a pejorative manifold

- The \( \text{GCD} \) computations and polynomial divisions enable the multiplicities \( m_i \) of the roots of \( f(y) \) to be computed, and they therefore identify the pejorative manifold \( M \) on which \( f(y) \) lies.
- An iterative search is then performed on the manifold in order to identify the point on \( M \) that represents \( f(y) \).
- It can be shown that computations performed on a manifold are numerically stable because, by definition of a pejorative manifold, all points on it represent the set of polynomials whose roots have multiplicities \( m_i \). It follows that iterations that are restricted to \( M \) change the values of the roots, but not their multiplicities.

Mathematical techniques required for the computation of the roots of \( f(y) \):

- Computational linear algebra, non-linear structure-preserving matrix methods, linear programming, constrained optimisation
Image deblurring

- A blurred image is formed by the *convolution* of an exact image and a blurring function, called a point spread function.
- The multiplication of two polynomials reduces to the *convolution* of their coefficients.

It follows that:

- If an image is considered a bivariate polynomial, then polynomial methods (greatest common divisor, deconvolution, and more generally, algebraic geometry) can be used to deblur an image.
Two examples of image deblurring using polynomial methods

Original image

Distorted image

Restored image

Restored blurring function
Summary

- There are several applications in which it is required to compute roots of a polynomial.
- Multiple roots break up into simple roots in the presence of noise, and it is therefore necessary to ‘sew’ together the simple roots that belong to the same multiple roots.
- Structure-preserving matrix methods and computational linear algebra enable multiple roots of a polynomial to be computed.
- The geometry of polynomials that have one or more multiple roots can be considered from its pejorative manifolds.
- Polynomial computations can be used to deblur an image.
- The MATLAB code for the work on polynomials was easily adapted to image deblurring.