Approximate Greatest Common Divisors and Polynomials Roots

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

This document provides abstracts for 6 lectures to be given by Joab Winkler on *Approximate Greatest Common Divisors and Polynomials Roots*. The lectures, which will take place at Tel Aviv University and The Technion in August 2008, follow the Summer School that was held at Oxford University, England in September 2007 (`http://www.dcs.shef.ac.uk/ml/summer_school/`).

Lectures 1 and 2: Numerical issues associated with computations on polynomials

This lecture will show by example some of the problems that occur when the roots of a polynomial are computed using a standard polynomial root solver. In particular, polynomials of high degree with a large number of multiple roots will be considered, and it will be shown that even roundoff error due to floating point arithmetic, in the absence of data errors, is sufficient to cause totally incorrect results. Since data errors are usually larger than roundoff errors (and fundamentally different in character), the errors encountered with real world data are significant and emphasise the need for a computationally robust polynomial root solver. These computational results will be quantified by considering the componentwise and normwise condition numbers, and the componentwise and normwise backward errors, of a root of a polynomial, and it will be shown that the forward error is not a practical error measure. The equation that links the forward error, backward error and condition number of a root of a polynomial will be considered.

The inability of all polynomial root solvers to compute high degree multiple roots correctly requires investigation of the cause of this failure. This leads naturally to a discussion of a *structured condition number* of a root of a polynomial, where *structure* refers to the form of the perturbations that are applied to the coefficients. It will be shown that this condition number differs significantly from the condition numbers considered above, which refer to random (unstructured) perturbations of the coefficients. Several examples will be given and it will be shown that the condition number of a multiple root of a polynomial due to a random perturbation in the coefficients is large, but the structured condition number of the same root is small. This large difference is typically several orders of magnitude.

A method developed by Gauss for computing the roots of a polynomial will be discussed. It is rarely used now, but it will be considered because it differs significantly from all other methods (Newton-Raphson, Bairstow, Laguerre, etc.) and is non-iterative. The computational implementation of this method raises, however, some non-trivial issues – the determination of the greatest common divisor of two polynomials and the estimation of the rank of a matrix in a floating point environment – and they will be discussed in the next set of lectures.

Lectures 3 and 4: Approximate Greatest Common Divisor Computations

The discussion of the structured condition number and Gauss' method for computing the roots of a polynomial shows that a sequence of *greatest common divisor* (GCD) computations of two polynomials must be performed. This is a classical problem in mathematics that is often solved by Euclid's algorithm. This algorithm is satisfactory for symbolic computations with exact polynomials, but the computation of the GCD of inexact polynomials is not trivial because it is ill-posed (not ill-conditioned) since the output (the GCD) is a discontinuous function of the input, that is, the polynomials. In particular, the polynomials $p(x)$ and $q(x)$ may have a non-constant GCD, but their perturbed forms $p(x) + \delta p(x)$ and $q(x) + \delta q(x)$ are, with probability almost one, coprime. Furthermore, the calculation of the degree of the GCD reduces to the determination of the rank of a matrix, which is a challenging problem in computational linear algebra. This topic will be addressed in Lecture 6.

Data from practical examples is always corrupted by noise, and it is therefore assumed that the given polynomials, the GCD of which it is required to compute, are inexact. It therefore follows that, with high probability, they are coprime, and thus only an *approximate greatest common divisor* (AGCD) can be computed. The term approximate greatest common divisor arises because the given inexact polynomials must be perturbed slightly in order to induce a non-constant GCD, and thus the computed GCD is approximate with respect to the given inexact polynomials. Several different methods can be used to compute an AGCD, and the technique based on the Sylvester resultant matrix $S(f,g)$ of the polynomials $f(x)$ and $g(x)$ will be described. This matrix, which arises in classical algebraic geometry, is convenient for computations because it has a well-defined linear structure that can be preserved when it is desired to compute a perturbed form of this matrix. As noted above, it is assumed that $f(x)$ and $g(x)$ are inexact and therefore coprime, and it is therefore necessary to calculate the perturbations $\delta f(x)$ and $\delta g(x)$ such that $f(x) + \delta f(x)$ and $g(x) + \delta g(x)$ have a non-constant GCD. This is achieved by applying *structured matrix methods* to $S(f+\delta f, g+\delta g)$ and its subresultants, which are matrices derived from $S(f+\delta f, g+\delta g)$ by deleting some rows and columns. More precisely, it will be shown that the computation of an AGCD yields a least squares equality problem, which can be solved by the QR decomposition.

Numerous computations have been performed and excellent results on high degree polynomials with roots of multiplicity 20 and 30 have been obtained. Some of these results will be shown and it will be explained that the simple linear structure of the Sylvester matrix introduces a degree of freedom that can be exploited to obtain significantly improved results. The introduction of this degree of freedom is not required when computations are performed in a symbolic environment, but it is crucial when the computations are performed in a floating point environment. Its importance for obtaining good results on high degree polynomials with a large number of roots of high multiplicity will be shown, and a simple and fast algorithm to obtain its optimal value will be discussed.

Lecture 5: The method of non-linear least squares and a polynomial root solver

The theory and results presented in Lectures 3 and 4 are sufficient if the AGCD computation is to be performed once. The polynomial root solver that was introduced in Lecture 1 requires that this computation be performed several times, where the result of the ith AGCD computation forms the input to the $(i+1)$th AGCD computation. It is therefore necessary to obtain accurate answers to each computation, and this is achieved by obtaining initial estimates of the AGCD and coprime polynomials, which are then refined by the method of non-linear least squares. An initial estimate of the coprime polynomials is obtained by calculating the null space of the Sylvester resultant matrix, and an initial estimate of the AGCD can be obtained by either reducing the Sylvester resultant matrix to upper triangular form using the LU or QR decompositions, or by solving an over-determined linear algebraic equation. It is also necessary to perform a set of polynomial divisions, which can be implemented by the method of least squares.

The final section of the talk will show some results that are obtained when the theory presented in the previous lectures is combined to form a polynomial root solver for polynomials of high degree with a large number of multiple roots.

Lecture 6: The calculation of the rank of a matrix

The calculation of the rank of a matrix is a challenging problem that arises in several disciplines, including computational linear algebra, principal component analysis, and signal and image processing. The rank loss of the Sylvester matrix is equal to the degree of the GCD, and thus a robust method of calculating the rank of a matrix is required.

The rank of a matrix is usually estimated by placing a threshold on its small singular values, such that singular values that are smaller than the threshold are set equal to zero. There exist several criteria that can be used to define the threshold, and they all assume that the noise level is known. All these criteria are valid, but different criteria may yield different estimates of the rank. Moreover, problems arise when the noise level is not known, or only known approximately, in which case it is difficult to set a threshold.

The talk will discuss the application of the principle of maximum likelihood (ML) for the estimation of the rank of a matrix. This principle has the advantage that the noise level is estimated from the matrix, that is, the noise level is calculated from the data rather than specified *a priori*. The application of a threshold on the singular values in the standard rank estimators is replaced in the principle of ML by assumptions of the underlying probability distributions of the non-zero and zero singular values. These must, of necessity, be different because a two-sided distribution can be assigned to the non-zero singular values, but a one-sided distribution must be assigned to the zero singular values. These distributions allow an expression for the likelihood $L(r)$ as a function of the unknown rank r to be developed, and the value of r for which $L(r)$ achieves the maximum is the estimated rank.

Several computational examples will be included and the results compared with the rank estimator in MATLAB. It will be shown that if the correct noise level is specified, then the results from MATLAB and the principle of ML are very similar. When, however, the default threshold level in MATLAB is used, the principle of ML yields significantly better estimates of the rank of a matrix.