Spectral Factorization—Computational Aspects

Wiener's optimum linear filtering problem [1], [2], and the problem of synthesizing the impedance matrix of linear n-port networks [3], [4], require for their solution the factorization of a matrix of rational functions in a complex variable s. More specifically, we are given an \( n \times n \) matrix \( \Phi(s) \) of rational transfer functions, with the properties:

\[
\Phi(j\omega) \geq 0 \quad \text{for all } \omega \quad (1)
\]

\[
\Phi(-s) = \Phi(s) \quad (2)
\]

The notation in (1) is shorthand for \( \Phi \) is nonnegative definite for imaginary values of \( s \). The superscript prime in (2) indicates matrix transposition.

We seek a matrix \( G(s) \) such that

\[
\Phi(s) = G'(-s) G(s) \quad (3)
\]

with \( G(s) \) satisfying two additional conditions. These are:

a) \( G(s) \) is analytic in the right half plane \( \text{Re } s > 0 \).

b) \( G(s) \) possesses a right inverse \( G^{-1}(s) \) analytic in the right half plane \( \text{Re } s > 0 \).

Condition (b) may be written equivalently as [5].

(\text{b'} \ ') \( G(s) \) has constant rank in \( \text{Re } s > 0 \) equal to \( r \), the rank of \( \Phi(s) \) almost everywhere.

The first full treatment of the problem of factoring \( \Phi \) appears to be in [5], where the author commented that the memory requirements for his factorization technique might exceed those of the available (1961) computers. Reference [6] introduced drastic simplification in an alternative factorization technique, and it is this technique which is the subject of our remarks.

We assume the reader has some familiarity with [6]. The factorization proceeds in several steps, given the names "pole removal phase," "determinant reduction phase," and "element order reduction phase," and concludes with a factorization of constant matrices.

In the pole removal phase, the poles of the elements of \( \Phi \) are eliminated by multiplying \( \Phi \) on the left and right by diagonal matrices whose elements are Hurwitz or non-Hurwitz polynomials, that is, polynomials with zeros all in the left or all in the right half plane. These polynomials may be determined by factoring denominator polynomials of the elements of \( \Phi \) into Hurwitz and non-Hurwitz parts, with \( j\omega \)-axis zeros being shared equally between the Hurwitz and non-Hurwitz parts.

In the determinant reduction phase, the object is to reduce the factorization problem from that requiring the factorization of a matrix \( \Phi(s) \) to that involving factorization of another matrix \( \Phi_2(s) \). Both \( \Phi(s) \) and \( \Phi_2(s) \) have elements with no poles in the finite \( s \)-plane, but \( \Phi_2 \) has a constant determinant, as opposed to the non-constant determinant of \( \Phi \). It is possible to show that the determinant of \( \Phi_2(s) \) is a polynomial which can be split into a Hurwitz and a non-Hurwitz part, \( j\omega \)-axis zeros being distributed evenly. Now although the reduction of \( \Phi_2 \) to \( \Phi_2 \) in [6] proceeds in steps by removing only two zeros of \( \det \Phi(s) \) at a time, it is easy to extend the technique of [6] to yield a reduction of \( \Phi_1 \) to \( \Phi_2 \) in one step. The only "difficult" part of this calculation is to initially factor \( \det \Phi(s) \) into its Hurwitz and non-Hurwitz parts. Note that if \( \Phi(s) \) is singular (which cannot normally be the case in the filtering problem, but can occur in the network synthesis problem), no determinantal reduction is necessary (det \( \Phi \) being zero and thus constant).

The remainder of the factorization procedure in [6], involving element order reduction and constant matrix factorization...
is straightforward, and readily amenable to computation.

The only awkward operations to be performed are those involving factorization of polynomials into Hurwitz and non-Hurwitz parts, in the pole removal and deterministic reduction phases. The problem of carrying out these operations is precisely the same as that occurring in the one-dimensional Wiener filtering problem, where $g(x)$ is a scalar function. In principle then, the multidimensional factorization problem is no more difficult than the one-dimensional problem.

The polynomial factorization problem may be stated as follows. Given a polynomial $f(x)$ such that $f(x)$ is real and nonnegative for all $x$, find a polynomial $g(x)$ whose zeros all lie on or to the left of the imaginary axis such that

\[ g(x) = g(-x)g(x). \] (4)

The polynomial $g(x)$ must actually be a polynomial in $x^s$ with real coefficients; some treatments express this fact by using $g(s^s)$ where $g(s)$ is the derivative of $g(x)$.

For convenience we shall assume in the sequel that the $x_0$-axis zeros of $f$ have been removed. To assist in pinpointing such zeros, it is helpful to note that since these zeros must be even, they must be roots of the greatest common divisor of $f$ and the derivative of $f$.

The general factorization problem of finding $g(x)$ in (4) requires the use of iterative techniques, preferably implemented on a computer. A number of methods are available, with differing convergence properties. A "brute force" method is to find all the zeros of $f(x)$ and attribute to $g(x)$ those with negative real parts. It is sufficient to factor $\phi$, regarded as a polynomial in $x^s$, into a product of factors quadratic in $x^s$, possibly together with a factor linear in $x$. These factors have real coefficients, thus avoiding the use of complex numbers.

Lin [7], [8] devised an algorithm for factoring a polynomial into two polynomials of lower degree. When the process converges, the technique can then be applied successively to the reduced polynomials until a set of linear or quadratic factors are formed. Commonly one of the two polynomials at each stage of the factorization process is chosen to be quadratic.

Similar techniques, such as that proposed by Friedman [9], may converge faster, or for different polynomials, than Lin's method.

The Bairstow-Hitchcock method [10] employs a Newton-Raphson technique to accelerate convergence of Lin's method, and is commonly used on computers to split off quadratic factors. It is also possible [11] to generate several quadratic, or higher degree, factors at one time.

The above techniques are readily adaptable to computer operation. However, convergence is often very slow when zeros of almost equal modulus are present, and frequently convergence does not occur when multiple roots are present.

Since the zeros themselves are often of no interest, it is frequently convenient to factor $\phi(z)$ directly into Hurwitz and non-Hurwitz parts. As discussed in [12], Bayard proposes that $\phi(z)$ be written as

\[ \phi(z) = a_0 - a_1 z^b + a_2 z^b, \ldots + (-1)^m a_m z^m. \] (5)

Then if

\[ g(z) = b_1 + b_2 z + \ldots + b_m z^m \] (6)

the $b_i$ are positive roots satisfying the equations:

\[ b_0 = \sqrt{a_0}, \quad b_1 = \sqrt{a_1 + 2b_0}, \quad b_2 = \sqrt{a_2 + 2b_0 + b_1}, \ldots \] (7)

These equations cannot be solved in closed form except for $n \leq 4$. However, by choosing suitable positive values of the $b_i$ as a first approximation and substituting them into the right hand side of (7), a second set of approximations may be obtained. Successive approximations may lead to stable values of the $b_i$. Convergence can often be hastened by using the latest approximation to $b_i$ when calculating the approximation to a particular $b_i$. Tuttle [12] also mentions an extension of Newton's method for accelerating convergence. This method of calculating the expected increments in the $b_i$ for the next approximation involves the solution of $n$ simultaneous linear equations.

Bauer [13] solves (4) by first mapping the left half plane $R_s < 0$ into the unit circle $|z| < 1$. Then $g(z)$ is mapped into a reciprocal polynomial:

\[ h(z) = (s - 1)^m h_p \frac{z + 1}{s - 1}. \] (8a)

with

\[ h_{n-1} = h_{n-2} + \ldots + h_1. \] (8b)

Then $h(z)$ can be factored into two polynomials $u(z)$ and $v(z)$ where $u(z)$ has roots all inside the unit circle, and $v(z)$ has roots all outside the unit circle.

If

\[ u(z) = u_0 + \ldots + u_n z^n \] (9a)

then

\[ v(z) = v_0 + \ldots + v_n z^n \] (9b)

and the coefficients of $h(z)$ and $v(z)$ satisfy

\[ h_n = u_0 a_0 + u_1 a_1 + \ldots + u_n a_n. \] (10a)

\[ h_{n-1} = u_0 a_1 + u_1 a_2 + \ldots + u_{n-1} a_n. \] (10b)

Bauer expresses the factorization problem in matrix terms; he effectively solves the equations

\[ \begin{bmatrix} h_n \\ \vdots \\ h_0 \end{bmatrix} = \begin{bmatrix} u_0 & u_1 & \cdots & u_n \\ u_0 & u_1 & \cdots & u_n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}. \] (10c)

He uses the successive approximations using $u_i = 0$ for all $i$ as starting approximations.

This technique is self-correcting and convergent. Convergence is slow when $\phi(z)$ has zeros with small real parts. Convergence can often be hastened by noting that in generating an approximation to $u_i$, the matrix factorization technique does not use the latest available approximation for $u_j, j < i$.

ACKNOWLEDGMENT

Helpful discussions with Dr. R. E. Kalman are gratefully acknowledged.

A. C. RIDDLE

B. D. ANDERSON

Stanford Electronics Labs.
Stanford, Calif.

REFERENCES


